



# **VNgen Reference Guide**

by XGASOFT



<b>1. Introduction</b>	<b>12</b>
<b>2. Buy Now</b>	<b>13</b>
<b>3. Download PDF</b>	<b>13</b>
<b>4. Changelog</b>	<b>13</b>
4.1. Compatibility Notes	20
4.2. Feature Comparison	28
<b>5. Getting Started</b>	<b>29</b>
5.1. Setup & Installation	30
5.1.1. Additional HTML5 Setup	33
5.1.2. Additional Language Setup	34
5.2. Intro to Q-script	37
5.3. Your First Visual Novel	41
5.3.1. Entities & Action Types	41
5.3.2. A Simple Dialog	42
5.3.3. A Simple Choice	46
5.3.4. A Simple Scene	50
5.3.5. A Simple Script	53
<b>6. Reference Guide</b>	<b>54</b>
6.1. Debug Functions	55
6.1.1. Intro to QCMD	57
6.1.2. Included Commands	58
6.1.3. vngen_do_debug	59
6.1.4. vngen_is_debug	60
6.2. Macros & Keywords	60
6.3. Animations	64
6.3.1. Creating Custom Animations	65
6.3.2. Included Animations	71
6.3.3. deform	72
6.3.4. keyframe	72
6.4. Effects	73
6.4.1. Creating Custom Effects	73
6.4.2. Included Effects	76
6.4.3. effect	77
6.5. Shaders	77
6.5.1. Included Shaders	77
6.6. Engine Functions	78
6.6.1. sys_vngen_config	78
6.6.2. sys_action_init	78



6.6.3. sys_action_skip .....	79
6.6.4. sys_action_term .....	79
6.6.5. sys_anim_init .....	79
6.6.6. sys_anim_perform .....	80
6.6.7. sys_anim_term .....	80
6.6.8. sys_anim_speech .....	81
6.6.9. sys_cmd_init .....	81
6.6.10. sys_cmd_add .....	83
6.6.11. sys_cmd_perform .....	83
6.6.12. sys_cmd_draw .....	83
6.6.13. sys_toggle_cmd .....	84
6.6.14. sys_deform_init .....	84
6.6.15. sys_deform_perform .....	85
6.6.16. sys_deform_draw .....	85
6.6.17. sys_deform_term .....	86
6.6.18. sys_effect_init .....	86
6.6.19. sys_effect_perform .....	87
6.6.20. sys_effect_term .....	87
6.6.21. sys_event_skip .....	88
6.6.22. sys_read_skip .....	88
6.6.23. sys_grid_delete .....	89
6.6.24. sys_grid_last .....	89
6.6.25. sys_layer_set_target .....	89
6.6.26. sys_layer_draw_scene .....	90
6.6.27. sys_layer_draw_char .....	90
6.6.28. sys_layer_draw_emote .....	90
6.6.29. sys_layer_draw_perspective .....	91
6.6.30. sys_layer_draw_effect .....	91
6.6.31. sys_layer_draw_textbox .....	91
6.6.32. sys_layer_draw_text .....	91
6.6.33. sys_layer_draw_label .....	92
6.6.34. sys_layer_draw_prompt .....	92
6.6.35. sys_layer_draw_button .....	92
6.6.36. sys_layer_draw_option .....	93
6.6.37. sys_layer_reset_target .....	93
6.6.38. sys_layer_log_set_target .....	93
6.6.39. sys_layer_draw_log .....	94
6.6.40. sys_layer_draw_log_button .....	94



6.6.41. sys_layer_log_reset_target .....	95
6.6.42. sys_log_init .....	95
6.6.43. sys_log_perform .....	95
6.6.44. sys_log_get_style .....	96
6.6.45. sys_log_get_xoffset .....	96
6.6.46. sys_mouse_hover .....	96
6.6.47. sys_option_init .....	97
6.6.48. sys_orig_init .....	97
6.6.49. sys_queue_enqueue .....	98
6.6.50. sys_queue_submit .....	98
6.6.51. sys_queue_destroy .....	98
6.6.52. sys_queue_empty .....	99
6.6.53. sys_scale_init .....	99
6.6.54. sys_shader_init .....	100
6.6.55. sys_shader_perform .....	100
6.6.56. sys_shader_exists .....	101
6.6.57. sys_shader_set_sampler .....	101
6.6.58. sys_text_init .....	103
6.6.59. sys_text_perform .....	103
6.6.60. sys_text_get_label .....	104
6.6.61. sys_text_get_xoffset .....	104
6.6.62. sys_text_style_init .....	104
6.6.63. sys_trans_init .....	105
6.6.64. sys_trans_perform .....	106
6.6.65. sys_vox_add .....	106
6.7. Global Functions .....	107
6.7.1. vngen_do_auto .....	107
6.7.2. vngen_is_auto .....	108
6.7.3. vngen_do_continue .....	108
6.7.4. vngen_do_pause .....	108
6.7.5. vngen_is_paused .....	109
6.7.6. vngen_do_ui_display .....	109
6.7.7. vngen_is_ui_displayed .....	110
6.7.8. vngen_count .....	110
6.7.9. vngen_exists .....	111
6.7.10. vngen_goto .....	111
6.7.11. vngen_goto_unread .....	112
6.7.12. vngen_instance_change .....	113



---

6.7.13. vngen_room_goto .....	113
6.7.14. vngen_set_auto_type .....	114
6.7.15. vngen_set_cursor .....	114
6.7.16. vngen_set_renderlevel .....	115
6.7.17. vngen_set_scale .....	115
6.7.18. vngen_set_shader_float .....	116
6.7.19. vngen_set_shader_matrix .....	117
6.7.20. vngen_set_shader_sampler .....	118
6.7.21. vngen_set_halign .....	118
6.7.22. vngen_get_halign .....	119
6.7.23. vngen_set_lineheight .....	120
6.7.24. vngen_get_lineheight .....	120
6.7.25. vngen_set_speed .....	121
6.7.26. vngen_get_speed .....	121
6.7.27. vngen_set_vol .....	122
6.7.28. vngen_get_vol .....	122
6.8. File Functions .....	123
6.8.1. vngen_file_save .....	123
6.8.2. vngen_file_save_map .....	123
6.8.3. vngen_file_load .....	124
6.8.4. vngen_file_load_map .....	124
6.8.5. vngen_file_delete .....	125
6.9. Language Functions .....	126
6.9.1. vngen_set_lang .....	126
6.9.2. vngen_get_lang .....	127
6.10. Property Functions .....	127
6.10.1. vngen_set_prop .....	128
6.10.2. vngen_get_prop .....	128
6.10.3. vngen_get_index .....	129
6.10.4. vngen_get_struct .....	129
6.10.5. vngen_get_width .....	130
6.10.6. vngen_get_height .....	130
6.10.7. vngen_get_x .....	131
6.10.8. vngen_get_y .....	131
6.10.9. vngen_get_xscale .....	132
6.10.10. vngen_get_yscale .....	132
6.10.11. vngen_get_rot .....	133
6.11. Backlog Functions .....	133

---

6.11.1. Buttons .....	134
6.11.1.1. vngen_log_button_create .....	134
6.11.1.2. vngen_log_button_create_ext .....	135
6.11.1.3. vngen_log_button_create_transformed .....	136
6.11.1.4. vngen_log_button_create_ext_transformed .....	138
6.11.1.5. vngen_log_button_destroy .....	140
6.11.1.6. vngen_log_button_clear .....	140
6.11.1.7. vngen_get_log_button .....	141
6.11.1.8. vngen_do_log_button_nav .....	141
6.11.1.9. vngen_is_log_button_hovered .....	142
6.11.1.10. vngen_do_log_button_select .....	142
6.11.1.11. vngen_is_log_button_selected .....	143
6.11.2. Input .....	143
6.11.2.1. vngen_do_log_display .....	143
6.11.2.2. vngen_is_log_displayed .....	144
6.11.2.3. vngen_do_log_nav .....	144
6.11.2.4. vngen_do_log_nav_touch .....	145
6.11.2.5. vngen_do_log_play .....	145
6.11.2.6. vngen_is_log_playing .....	146
6.11.3. vngen_log_init .....	146
6.11.4. vngen_log_add .....	147
6.11.5. vngen_log_draw .....	148
6.11.6. vngen_log_clear .....	149
6.11.7. vngen_log_count .....	149
6.11.8. vngen_log_get_index .....	149
6.12. Object Functions .....	150
6.12.1. vngen_object_init .....	150
6.12.2. vngen_object_draw .....	151
6.12.3. vngen_object_clear .....	151
6.13. Events .....	152
6.13.1. vngen_event_set_target .....	152
6.13.2. vngen_event .....	153
6.13.3. vngen_event_pause .....	154
6.13.4. vngen_event_reset_target .....	155
6.13.5. vngen_event_count .....	155
6.13.6. vngen_event_get_index .....	155
6.13.7. vngen_event_get_label .....	156
6.13.8. vngen_event_get_read .....	156



6.14. Actions .....	157
6.15. Perspective Actions .....	157
6.15.1. vngen_perspective_modify_pos .....	157
6.15.2. vngen_perspective_modify_direct .....	158
6.15.3. vngen_perspective_replace .....	159
6.15.4. vngen_perspective_anim_start .....	160
6.15.5. vngen_perspective_anim_stop .....	161
6.15.6. vngen_perspective_shader_start .....	161
6.15.7. vngen_perspective_shader_stop .....	161
6.16. Scene Actions .....	162
6.16.1. vngen_scene_create .....	162
6.16.2. vngen_scene_create_ext .....	163
6.16.3. vngen_scene_modify_style .....	164
6.16.4. vngen_scene_modify_pos .....	165
6.16.5. vngen_scene_modify_ext .....	166
6.16.6. vngen_scene_modify_direct .....	167
6.16.7. vngen_scene_replace .....	168
6.16.8. vngen_scene_replace_ext .....	168
6.16.9. vngen_scene_destroy .....	169
6.16.10. vngen_scene_anim_start .....	169
6.16.11. vngen_scene_anim_stop .....	170
6.16.12. vngen_scene_deform_start .....	171
6.16.13. vngen_scene_deform_stop .....	171
6.16.14. vngen_scene_shader_start .....	172
6.16.15. vngen_scene_shader_stop .....	172
6.17. Character Actions .....	173
6.17.1. vngen_char_create .....	173
6.17.2. vngen_char_create_ext .....	174
6.17.3. vngen_char_modify_style .....	175
6.17.4. vngen_char_modify_pos .....	176
6.17.5. vngen_char_modify_ext .....	177
6.17.6. vngen_char_modify_direct .....	178
6.17.7. vngen_char_replace .....	179
6.17.8. vngen_char_replace_ext .....	180
6.17.9. vngen_char_destroy .....	181
6.17.10. vngen_char_anim_start .....	181
6.17.11. vngen_char_anim_stop .....	182
6.17.12. vngen_char_deform_start .....	182



---

6.17.13. vnngen_char_deform_stop .....	183
6.17.14. vnngen_char_shader_start .....	183
6.17.15. vnngen_char_shader_stop .....	184
6.18. Character Attachment Actions .....	185
6.18.1. vnngen_attach_create .....	185
6.18.2. vnngen_attach_create_ext .....	186
6.18.3. vnngen_attach_modify_style .....	187
6.18.4. vnngen_attach_modify_pos .....	188
6.18.5. vnngen_attach_modify_ext .....	188
6.18.6. vnngen_attach_modify_direct .....	190
6.18.7. vnngen_attach_replace .....	190
6.18.8. vnngen_attach_replace_ext .....	191
6.18.9. vnngen_attach_destroy .....	192
6.18.10. vnngen_attach_anim_start .....	192
6.18.11. vnngen_attach_anim_stop .....	193
6.18.12. vnngen_attach_deform_start .....	194
6.18.13. vnngen_attach_deform_stop .....	194
6.18.14. vnngen_attach_shader_start .....	195
6.18.15. vnngen_attach_shader_stop .....	195
6.19. Emote Actions .....	196
6.19.1. vnngen_emote_create .....	196
6.19.2. vnngen_emote_create_ext .....	197
6.19.3. vnngen_emote_destroy .....	198
6.20. Effect Actions .....	198
6.20.1. vnngen_effect_start .....	199
6.20.2. vnngen_effect_stop .....	199
6.21. Textbox Actions .....	200
6.21.1. vnngen_textbox_create .....	200
6.21.2. vnngen_textbox_create_ext .....	200
6.21.3. vnngen_textbox_modify_style .....	201
6.21.4. vnngen_textbox_modify_pos .....	202
6.21.5. vnngen_textbox_modify_ext .....	203
6.21.6. vnngen_textbox_modify_direct .....	204
6.21.7. vnngen_textbox_replace .....	205
6.21.8. vnngen_textbox_replace_ext .....	205
6.21.9. vnngen_textbox_destroy .....	206
6.21.10. vnngen_textbox_anim_start .....	207
6.21.11. vnngen_textbox_anim_stop .....	207



---

6.21.12. vngen_textbox_deform_start .....	208
6.21.13. vngen_textbox_deform_stop .....	208
6.21.14. vngen_textbox_shader_start .....	209
6.21.15. vngen_textbox_shader_stop .....	209
6.22. Text Actions .....	210
6.22.1. Inline Markup .....	210
6.22.2. vngen_text_create .....	214
6.22.3. vngen_text_create_ext .....	215
6.22.4. vngen_text_modify_style .....	217
6.22.5. vngen_text_modify_pos .....	218
6.22.6. vngen_text_modify_ext .....	218
6.22.7. vngen_text_modify_direct .....	220
6.22.8. vngen_text_replace .....	221
6.22.9. vngen_text_replace_ext .....	221
6.22.10. vngen_text_destroy .....	223
6.22.11. vngen_text_anim_start .....	223
6.22.12. vngen_text_anim_stop .....	224
6.22.13. vngen_text_deform_start .....	224
6.22.14. vngen_text_deform_stop .....	225
6.22.15. vngen_text_shader_start .....	225
6.22.16. vngen_text_shader_stop .....	226
6.23. Label Actions .....	226
6.23.1. vngen_label_create .....	227
6.23.2. vngen_label_create_ext .....	228
6.23.3. vngen_label_modify_style .....	229
6.23.4. vngen_label_modify_pos .....	230
6.23.5. vngen_label_modify_ext .....	231
6.23.6. vngen_label_modify_direct .....	232
6.23.7. vngen_label_replace .....	233
6.23.8. vngen_label_replace_ext .....	234
6.23.9. vngen_label_destroy .....	235
6.23.10. vngen_label_anim_start .....	236
6.23.11. vngen_label_anim_stop .....	236
6.23.12. vngen_label_deform_start .....	237
6.23.13. vngen_label_deform_stop .....	237
6.23.14. vngen_label_shader_start .....	238
6.23.15. vngen_label_shader_stop .....	238
6.24. Prompt Actions .....	239

---

6.24.1. vngen_prompt_create .....	239
6.24.2. vngen_prompt_create_ext .....	240
6.24.3. vngen_prompt_modify_style .....	241
6.24.4. vngen_prompt_modify_pos .....	242
6.24.5. vngen_prompt_modify_ext .....	243
6.24.6. vngen_prompt_modify_direct .....	244
6.24.7. vngen_prompt_replace .....	245
6.24.8. vngen_prompt_replace_ext .....	245
6.24.9. vngen_prompt_destroy .....	246
6.24.10. vngen_prompt_anim_start .....	247
6.24.11. vngen_prompt_anim_stop .....	248
6.24.12. vngen_prompt_deform_start .....	248
6.24.13. vngen_prompt_deform_stop .....	249
6.24.14. vngen_prompt_shader_start .....	249
6.24.15. vngen_prompt_shader_stop .....	250
6.25. Button Actions .....	250
6.25.1. vngen_button_create .....	250
6.25.2. vngen_button_create_ext .....	252
6.25.3. vngen_button_create_transformed .....	253
6.25.4. vngen_button_create_ext_transformed .....	254
6.25.5. vngen_button_destroy .....	256
6.25.6. vngen_button_clear .....	257
6.25.7. vngen_get_button .....	257
6.25.8. vngen_do_button_nav .....	258
6.25.9. vngen_is_button_hovered .....	259
6.25.10. vngen_do_button_select .....	259
6.25.11. vngen_is_button_selected .....	259
6.26. Option Actions .....	260
6.26.1. vngen_option .....	260
6.26.2. vngen_option_create .....	261
6.26.3. vngen_option_create_ext .....	262
6.26.4. vngen_option_create_transformed .....	264
6.26.5. vngen_option_create_ext_transformed .....	266
6.26.6. vngen_option_clear .....	268
6.26.7. vngen_get_option .....	268
6.26.8. vngen_do_option_nav .....	269
6.26.9. vngen_is_option_hovered .....	270
6.26.10. vngen_do_option_select .....	270



---

6.26.11. vngen_is_option_selected .....	270
<b>6.27. Audio Actions .....</b>	<b>271</b>
6.27.1. vngen_audio_play_sound .....	271
6.27.2. vngen_audio_play_music .....	272
6.27.3. vngen_audio_play_voice .....	273
6.27.4. vngen_audio_modify .....	273
6.27.5. vngen_audio_replace .....	274
6.27.6. vngen_audio_pause .....	275
6.27.7. vngen_audio_resume .....	275
6.27.8. vngen_audio_stop .....	276
6.27.9. vngen_vox_play .....	276
6.27.10. vngen_vox_modify .....	277
6.27.11. vngen_vox_replace .....	278
6.27.12. vngen_vox_add .....	278
6.27.13. vngen_vox_remove .....	279
6.27.14. vngen_vox_pause .....	279
6.27.15. vngen_vox_resume .....	279
6.27.16. vngen_vox_stop .....	280
<b>6.28. Code Actions .....</b>	<b>280</b>
6.28.1. vngen_code_execute .....	281
6.28.2. vngen_code_execute_ext .....	281
6.28.3. vngen_script_execute .....	282
6.28.4. vngen_script_execute_ext .....	282
<b>7. Special Thanks .....</b>	<b>283</b>
<b>8. End-User License Agreement ("EULA") .....</b>	<b>284</b>

## 1. Welcome to VNgen - The Next-gen Visual Novel Engine



Easily create dynamic visual novel content and sequenced animations with VNgen by XGASOFT. Powered by [Quantum](#), VNgen uses a powerful scripting system within GameMaker Studio, breaking the limits of GML to deliver high-quality presentation both in the IDE and on target devices.

A complete rewrite of the popular [Edge VN](#), VNgen reimagines visual novel design from the ground up, providing access to advanced features previously only accessible to high-budget studios with custom engines. As a sequenced animation platform the possibilities are endless, offering a combination of built-in transitions and animations plus support for user-created animation scripts using simple keyframe logic. As a visual novel engine, VNgen supports virtually limitless entities of every kind, all part of a flexible, modular structure which can be customized to suit almost any user's needs.

**Just want something simple?** Good news: VNgen handles the complex stuff so you don't have to. Many functions have both simplified and extended counterparts, offering something for every level of programming experience.

**Building a full game?** Great! As an extension of GameMaker Studio, VNgen can be used entirely on its own or as a layer on top of your own custom code and game mechanics. You can even integrate your own code into VNgen without modifying anything!

### In addition, VNgen features...

- Fully resolution and framerate independent design
- Compatibility with desktop, mobile, and console platforms (HTML5 partially supported)
- Event/action structure with over 10 different types of entities to create, modify, animate, and destroy
- Advanced text generation with markup for style, speed, pausing, links, and more
- Four-color gradient blending for most elements
- Dynamic mesh support for most elements
- Advanced 2.5D camera-style perspective
- Custom depth sorting of elements within a single object
- Full-color backlog
- Composite character system with unlimited layers
- Built-in multi-language support
- Built-in debug mode with command console and a host of real-time statistics
- ... And much more! (Seriously.)



## In this reference guide, you'll learn...

- How to get started with your first project
- How to use VNgen's custom script syntax (Q-script)
- How to navigate events
- How to perform actions
- Advanced functions
- Individual script arguments, and what they mean

To get started, choose a topic from the menu to the left above to learn more.

## 2. Buy Now(<https://xgasoft.itch.io/vngen>)

## 3. Download PDF(<https://docs.xgasoft.com/wp-content/uploads/sites/2/vngen-reference-guide-14.pdf>)

## 4. Version History

### 1.0.8

- Added support for GameMaker Studio 2 individual sprite speeds
  - Global animation speed arguments have been removed from `vngen_object_draw`, `vngen_log_draw`, `vngen_emote_create`, and `vngen_emote_create_ext`
  - Other timing functions have also been improved as a result of this update
- Added `vngen_file_save_map` to facilitate adding VNgen data to fully custom save files
- Added `vngen_set_vol` and audio type macros to replace individual audio volume scripts
  - Volume control of UI/navigation sounds is now supported
  - Volume can now be set from the debug command console
- Added 'auto' keyword support to `vngen_log_add`, allowing custom content to be added to the backlog using the system queue
- Added looping animation support to emotes
- Added support for custom hotspots using rectangular collision masks (bounding boxes) on options and buttons
- Added new scanline and radial blur shaders
- Added new cubic bezier ease mode
- Added new 'output' functions to retrieve the state of many 'input' functions, such as whether or not the engine is paused, or a button or option is held down
- Added new 'get' functions for global audio volumes, text alignment, lineheight, text speed, and event read state
- Added new `vngen_code_execute` functions for running arbitrary code as VNgen actions without the use of scripts
- Updated `vngen_set_lineheight` to support per-entity lineheight multipliers
  - Save functions now also read and write global lineheight multipliers
- Updated input functions to use a common prefix and support explicit true/false settings in addition to toggles
  - **Important!** See compatibility notes for a list of syntax updates!
- Updated `vngen_option_clear` to support removing option results from history as well as memory

- Updated UI visibility toggle script to optionally allow progression while hidden
- Updated `vngen_room_goto` to optionally allow skipping to a VNgen event upon room change
- Updated highlight argument in `vngen_object_draw` and perform argument in `vngen_instance_change` to be optional (enabled by default)
- Updated auto prompts to match text scale and rotation, not just position
- Updated backlog audio to match global voice volume
- Updated `vngen_file_load` to support switching rooms upon loading files
- Updated `vngen_file_load_map` to accept `ds_maps` generated by `vngen_file_save_map`
- Fixed multiple crashes related to running file functions when files or save data do not exist
- Fixed room data missing from unencrypted save files loaded with `vngen_file_load_map`
- Fixed text alignment and speed reverting to global defaults after window is resized
- Fixed volume functions causing crashes when run outside of a VNgen object or after the running VNgen object has been cleared
- Fixed pause state not being preserved if `vngen_goto` is run while the engine is paused
- Fixed indefinite pauses applying even if auto mode is enabled
  - All negative values will now trigger indefinite pause, but if auto mode is enabled, the literal numeric value will be used instead. (e.g. `[pause=-2]` will pause for 2 seconds if auto mode is enabled, and pause indefinitely otherwise)
  - This behavior can be enabled or disabled with `vngen_set_auto_type`
  - This update applies to both events and text markup
- Fixed options being added to the backlog even when an empty string is used
  - Image-only options are now properly supported
- Fixed `vngen_get_option` clearing options prematurely if an option ID is provided in the same event as options are presented
- Miscellaneous fixes and improvements

## 1.0.7

- Replaced internal timing functions with a simpler, more versatile timing system
- Added automatic checks to preserve custom text alignment when drawing VNgen
  - Previously it was required to manually reset `draw_set_halign` or `draw_set_valign` before drawing VNgen. This is no longer necessary, as VNgen properly handles both functions.
  - (Requires GameMaker Studio v2.2.1.375 or newer)
- Added automatic option results logging. Setting an option block ID in `vngen_get_option` now returns user selections from any previous option!
  - Option selection data is now also saved/loaded with VNgen file functions
- Added `vngen_file_load_map` for restoring saved VNgen data to a user-defined `ds_map` for later usage
- Improved file functions
  - Fixed loading only restoring the VNgen object and not the event, in some cases
  - Save/load can now be performed in non-VNgen objects
  - Save files now also record the active room when saved
- Improved rendering on scenes and debug interface
  - Blending is now consistent between full/reduced and legacy render modes
  - Texture filtering is now disabled on debug interface for more legible text
- Improved handling of speech animations when text and voice are active simultaneously
- Fixed an issue where prematurely skipping text with an indefinite pause skipped the text, but not the pause
- Fixed incorrect backlog touch scrolling at non-standard framerates
- Fixed perspective shaders drawing a blank frame when initialized
- Fixed mouse interactions in HTML5
- Miscellaneous additional fixes and improvements

## 1.0.6



- Added support for setting the typewriter effect speed of individual text entities with `vngen_set_speed`
  - This complements the existing `[speed]` markup feature, which only allows setting a multiplier of the global speed rather than explicit CPS
- Added global `vngen_get_prop` and `vngen_set_prop` functions to modify properties not accessible through existing functions
  - Can also be used to modify a single property when other functions require modifying multiple at once, e.g. `vngen_text_modify_pos`
  - **WARNING:** This is an advanced feature, and improper usage can break things. Use at your own risk!
- Significantly improved performance of `vngen_get_*` functions when addressing multiple consecutive properties for the same entity
- Updated button entities to fully ignore the engine pause state
  - Buttons are now suitable for use in creating pause menus
- Renamed `vngen_perspective_modify` to `vngen_perspective_modify_pos` for consistency with other functions
- Fixed “out of range” errors when creating prompts while no corresponding text exists
- Fixed GUI not initializing to the correct scale when running `vngen_set_scale`
- Fixed `vngen_get_*` functions returning empty values on entities created in the same frame

## 1.0.5

- Added support for using variables in `[font]` markup
- Fixed button states not being cleared on destroy
- Fixed errors if `vngen_object_clear` is run when an object has already been cleared

## 1.0.4

- Added global volume controls for music, sound, voice, and vox
- Added support for built-in cursor states to `vngen_set_cursor`
- Added support for auto text positioning in extended option and button functions
- Fixed incorrect syntax guides for `vngen_scene_modify_style` and `vngen_label_replace_ext` (GMS2 only)
- Fixed overflow in wipe transitions
- Fixed `vngen_goto` performing certain skipped actions when performing skipped events is disabled
- Fixed `vngen_button_select` and `vngen_log_button_select` causing crashes if run when buttons do not exist
- Fixed typo causing errors in `vngen_label_create_ext`

## 1.0.3

- Added ‘auto’ keyword support to `vngen_text_create_*` actions to better facilitate NVL-style presentation
  - Using ‘auto’ positioning will create the new text entity directly below the previous one, eliminating the need to guess at unpredictable text surface dimensions
- Updated prompts and character speech animations to pause while `[pause]` tags are active
- Fixed a bug causing consecutive `[pause=-1]` tags to fail
- Fixed a bug causing incorrect vox playback
  - General vox behavior and text performance have been improved as a result of this fix
- Fixed a bug causing audio volume to be reset if fullscreen mode is changed
- Fixed outdated documentation examples in Getting Started guide

## 1.0.2



- Updated debug console to accept escaped commas as ^, instead of ,
  - This change was necessary to escape markup in GMS2, but was changed in both versions for interoperability
- Fixed `vnngen_goto` and `vnngen_room_goto` sometimes failing when executed in debug console (GMS 2 only)
- Fixed scene replacements sometimes failing to refresh scene on replace if dimensions are the same
- Fixed erroneous usage examples in some scripts

## 1.0.1

- Miscellaneous fixes and improvements

## 1.0.0

- Added native GameMaker Studio 2 support
  - GameMaker Studio 1 support now moved to critical updates and bug fixes only
- Added native 4K support to internal functions and debug interface
- Added `*_transformed` functions to VNgen options for added hover/select animation and stylization
  - This also replaced the extra color functionality previously added to `vnngen_option_create_ext`, resulting in a simpler `*_ext` function
- Added on-screen buttons as a new entity type
  - Log buttons have been rewritten to match the new button standard and can now be used to execute arbitrary code, not just scroll the log
  - The existing `vnngen_type_button` macro now refers to both log buttons and standard buttons and can be used to check both in property functions
- Added support for deformations of any number of subdivisions
  - Deformation columns and rows can now be set on a per-deformation basis
  - Updated the included `def_wave` deformation to display an actual sine wave
- Added new underwater-like wave shader
- Added `vnngen_count` to replace individual entity `*_count` functions with a universal function
- Added 'previous' keyword support to character face coordinates in character replace actions
- Added 'full' keyword support to `vnngen_audio_modify` loop clip settings
- Added `vnngen_event_get_label` to complement `vnngen_event_get_index`
- Added `vnngen_script_execute_ext` to perform scripts as VNgen actions, including when the running event is skipped
- Updated `vnngen_event_get_index` to optionally return the index of an event by label (rather than the current event)
- Replaced per-entity text speed with `vnngen_set_speed`, which sets speed for all text entities.
- Replaced `vnngen_instance_create` with `vnngen_instance_change`
  - The existing script was not really necessary and functioned more closely to the built-in `instance_change` function anyhow
- Replaced bracket escape character with ^. Markup can now be drawn literally as `^[` instead of `[`
  - This change was necessary to escape markup in GMS2, but was changed in both versions for interoperability
- Removed crop animations from replace transitions for consistency among all entities
  - Previously, only a few entities supported them, and created undesirable visual effects in some use-cases.
- Removed tiled scene deformations from renderlevel 1
  - It is technically not possible to deform tiled scenes at this renderlevel, therefore scenes will now fall back to non-deformed tiles rather than sacrifice tile mode for deformations
- Removed `vnngen_option_exists` as its functionality has been superseded by `vnngen_exists`
- Removed language macros as they have been superseded by `vnngen_type_*` macros
- Fixed text speeds of 0 or less (instant) causing crashes
- Fixed negative interpolation causing crashes if `ease_circ_*` modes are used



- Fixed opaque backgrounds appearing on highlighted objects if certain shaders or blend modes were used
- Fixed shaders being duplicated when a deform is also active
- Fixed auto mode being disabled when an indefinite event pause is encountered
- Fixed backlog override fonts/colors failing to apply
- Fixed missing backlog audio data if voice is paused in background while previous voice is played in backlog
- Fixed non-looped audio continuing to play when skipped while the engine is paused
- Fixed music losing position and clip settings if `vngen_goto` is run
- Fixed non-looped sounds failing to fade in on create
- Fixed mouse cursor getting stuck in hover state if `vngen_goto` is run while in this state
- Fixed scaling not updating when launched in fullscreen
- Miscellaneous additional fixes and improvements

### 0.9.9 (Early Access)

- Added `vngen_goto_unread` to skip to the nearest option block or unread event
- Added functions to set shader floats, samplers, and matrices
- Added functions to replace sounds and vox while synchronizing playback position and other properties
- Added multi-track and pitch range support to vox
  - Vox source can now be input as an array to create a list of vox which will be randomly selected from each time text increments
  - Additional sounds can be added or removed post-creation
  - Pitch is now randomized between two min/max values rather than simply enabled/disabled with a switch
- Added `trans_spin_in` and `trans_spin_out` transitions
- Added `[event]` markup to execute events inline directly (a la links, but without requiring a click)
- Updated deforms to use 2D point arrays instead of 1D fixed variables
  - This will allow for meshes of any number of columns and rows in a future update
- Updated effects to use custom variable array, allowing for infinite custom effect variables
- Raised `vngen_script_execute` argument limit to 32.
- Fixed compatibility issues with YYC. YYC is now fully supported
- Fixed audio fade transitions not being skipped if `vngen_goto` is run
- Improved HTML5 compatibility
  - Fixed broken mouse/touch hotspots when using scaling
  - Reduced renderlevel 2 surface usage to mitigate texture swapping glitches
- Miscellaneous additional fixes and improvements

### 0.9.8 (Early Access)

- Added per-entity shader support and a selection of included shaders
  - Support for custom uniforms coming in a future update
- Added `make_color_rgb_to_hex` to complement `make_color_hex`, which has now been renamed `make_color_hex_to_rgb` for consistency
- Added optional "perform" argument to `vngen_goto` to allow disabling performing skipped events
- Added optional "id" argument to `vngen_option_select` to allow selecting a specific option directly without navigation
- Fixed wrong colors being used when auto labels are replaced with the 'previous' color/font setting at the target event of `vngen_goto`
- Fixed style inheritance failing to record certain properties. Backlog now inherits label styles for speaker names
- Fixed incorrect scaling when launched directly in fullscreen
- Removed audio wrappers in favor of native functions
- Miscellaneous additional fixes and improvements

## 0.9.7 (Early Access)

- Added `vngen_room_goto` to switch rooms while properly cleaning up VNgen data and preserving log data
- Added support for multiple renderlevels to improve compatibility with some platforms such as Android, iOS, and HTML5
  - Also added command to set renderlevel in debug console
- Updated backlog with improved text processing and support for new rendering features such as text alignment
- Updated display scaling to function as a setting—running a script every step is no longer require
  - Scaling now supports HTML5
  - Also added command to set scaling view in debug console
- Unified label and text rendering to fix numerous bugs with labels
- Fixed a bug causing transitions to never be considered complete
- Standardized code to reduce redundancy and improve performance
- Miscellaneous additional fixes and improvements

## 0.9.6 (Early Access)

- Externalized transitions as a new category of keyframe animations
- Added support for performing animations, deformations, and effects in reverse
- Added optional ease override support to animations, deformations, and effects to match transition ease behavior
- Added `input_rot` as an available constant for keyframe animations (where applicable)
- Fixed a bug causing text links to crash
- Standardized code to reduce redundancy and improve performance
- Miscellaneous additional fixes and improvements

## 0.9.5 (Early Access)

- Added new tiled scenes system with support for rotation, gradient color blending, wipe transitions, and deforms
- Added paragraph alignment support to text (labels coming soon)
- Added animation blending support to transforms and deforms
- Added color gradient and wipe transition support to deforms
- Updated file functions to save/load text alignment and language settings
- Improved replace fade transitions to better support transparent entities
- Improved text auto linebreak accuracy
- Fixed character flipping in `vngen_char_replace_ext` being absolute instead of relative
- Fixed memory leaks in `vngen_object_clear`
- Removed scale from perspective calculations. Distance is now determined solely by z-index for better control and more predictable perspective behavior.
- Standardized code to reduce redundancy and improve performance
- Miscellaneous additional fixes and improvements

## 0.9.4 (Early Access)

- Updated syntax for consistency and ease of use
  - Added 'all' keyword support to modify, animate, effects, audio, and log button functions
  - Added 'idle' argument to `vngen_char_create_ext` for consistency with `vngen_char_replace_ext`
  - Added 'name' argument to `vngen_text_create_ext` and `vngen_text_replace_ext` for consistency with other functions



- Added 'any' keyword support to `vngen_exists` to check whether any entity of a given type exists
- Added optional easing support for all non-extended functions
- Added `vngen_option_create_ext` with support for origin, scaling, and variable text colors, and simplified `vngen_option_create`
- Added support for checking log buttons to `vngen_exists` and `vngen_get_index`
- Updated prompts to accurately reflect current state during replacement fade and accurately auto-position on rotated text
- Updated `vngen_get_option` to clear option results by default—setting the 'true' argument is no longer required
- Updated replace functions to fade the most recently viewed resource when combined with `vngen_goto` (rather than the most recent resource chronologically, which isn't always visually correct)
- Updated animations to rotate coordinates relative to perspective only, not the rotation of the animated entity
- Fixed option position being calculated incorrectly
- Fixed destroy actions never completing when 'all' keyword is used if no entities of the given type exist
- Fixed skip failure when skipping to an event label where the event is last in a series
- Fixed effects causing errors during some uses of `vngen_goto`
- Renamed audio functions for consistency with other functions and added wrappers for legacy syntax support
- Renamed `vngen_get_option_number` to `vngen_option_count` for consistency with other functions
- Renamed `vngen_get_option_active` to `vngen_option_exists` for consistency with other functions
- Standardized code to reduce redundancy and improve performance
- Miscellaneous additional fixes and improvements

### 0.9.3 (Early Access)

- Rewrote developer command console (now QCMD!)
  - Added support for keyboard cursor navigation (left/right/home/end)
  - Added support for repeat input if key is held down
  - Externalized commands – console is now extensible!
  - Added new language, log, and window commands (type 'help' in QCMD to learn more!)
- Added keyframe-based effect scripting system
  - Execute arbitrary code in VNgen keyframes!
  - Includes effect scripts for haptic feedback, screen flash, screen shake, and Dualshock 4 lightbar flash
- Added `vngen_get_index` function to retrieve internal numeric index for any given entity from the entity type and ID
- Added `vngen_event_count` function to retrieve the current total number of events
- Added support for 'all' keyword in `vngen_*_destroy` functions for destroying all entities of a given type
- Merged backgrounds and foregrounds into a single internal data structure for more efficient memory management
- Separated vox speech synthesis into its own data structure and added audio functions to independently pause, stop, and modify vox audio
- Renamed `vngen_log_get_size` to `vngen_log_count` for consistency with other functions
- Renamed `vngen_get_event` to `vngen_event_get_index` for consistency with other functions
- Updated skip functionality to allow skipping past the final event
- Updated `vngen_script_execute` to accept a full 16 arguments (as opposed to the previous 15-argument limitation)
- Fixed `vngen_file_load` failing due to outdated use of `vngen_goto` to restore object event
- Fixed `vngen_log_draw` generating errors if log was cleared without destroying the running object
- Fixed skipped event data being added to the backlog when jumping across objects
- Fixed text/label actions generating errors if empty string is input
- Standardized code to reduce redundancy and improve performance
- Miscellaneous additional fixes and improvements

## 0.9.2 (Early Access)

- Fixed an issue causing touch scrolling to sometimes scroll the backlog infinitely
- Fixed manual linebreaks being removed from backlog text
- Fixed backlog entries being listed in literal order rather than historical order.
- Added `vngen_log_get_index` script to return historical log entry index from an entry's on-screen order.
- Added new proportion scaling modes which scale relatively to changes in display scale
- Added new properties functions (`vngen_get_*`) to return the calculated width, height, x, y, xscale, yscale, and rotation of VNgen entities, factoring in animations and modifications
- Updated `vngen_exists` to optionally check a specific character for attachments, rather than check all characters
- Enhanced all built-in animations for use with perspective
- Reduced linebreak strictness so that text surface width can expand or shrink to fit text automatically
- Renamed `vngen_pragma` to `sys_config` for consistency with other new engine functions
- Standardized code to reduce redundancy and improve performance
- Miscellaneous additional fixes and improvements

## 0.9.1 (Early Access)

- Added online and offline documentation
- Added additional macros for entity types, which can now be tested with `vngen_exists`
- Added new zoom transitions which replace the old scale transition
- Rewrote skip functionality to fix bugs with labels and some jumps taking more than a frame to complete
- Fixed backlog data being duplicated when the window is resized, forcing an event restart—without also disabling duplicate data for intentional restarts. This required creating a new backlog queue system to hold all new log data until rescaling is complete.
- Fixed attachments scaling to the display rather than the parent character with `vngen_attach_replace_ext`
- Fixed option button sprites not respecting sprite alignment
- Renamed existing `vngen_emote_create` to `vngen_emote_create_ext` and added a simplified `vngen_emote_create` in its place.
- Standardized code to reduce redundancy and improve performance
- Miscellaneous additional fixes and improvements

## 0.9.0 (Early Access)

- Initial release

## 4.1. Compatibility Notes

Some updates include certain changes which require existing projects to be modified to retain compatibility with updated versions. This section documents those changes as well as the remedies to any incompatibilities they create.

### 1.0.8

- **Added support for rectangular collision masks (bounding boxes) on options and buttons**
  - Existing button/option sprites may need to have their bounding boxes manually adjusted if hotspots seem inaccurate
  - Transparent sprites will not be clickable at all without manual bounding box adjustment!

- **Updated `vngen_log_add` to include support for per-entity lineheight**
  - Existing instances of this function must be updated to match the new syntax
  - If no custom lineheight is desired, use `vngen_get_lineheight(all, vngen_type_text)`
  - (**Recommended:** use global search & replace)
- **Updated input functions to use clearer nomenclature and support explicit true/false settings, not just toggles**
  - A table of old functions and their new names is listed below. Existing instances of these functions must be renamed to match the new nomenclature and have arguments added to match the new syntax

**Old Syntax**`show_debug_vngen([enable])`**New Syntax**`vngen_do_debug(toggle, [sound])``vngen_continue([sound])``vngen_do_continue([sound])``vngen_toggle_auto(delay,  
[sound])``vngen_do_auto(toggle, [delay], [sound])``vngen_toggle_pause([sound])``vngen_do_pause(toggle, [sound])``vngen_toggle_visible([sound])``vngen_do_ui_display(toggle, [stop], [sound])``vngen_toggle_log([sound])``vngen_do_log_display(toggle, [sound])``vngen_log_nav(amount)``vngen_do_log_nav(amount)``vngen_log_nav_touch()``vngen_do_log_nav_touch()``vngen_log_play_audio()``vngen_do_log_play()``vngen_log_button_nav(amount)``vngen_do_log_button_nav(amount)``vngen_log_button_select([id])``vngen_do_log_button_select([id])``vngen_button_nav(amount)``vngen_do_button_nav(amount)`

`vngen_button_select([id])``vngen_do_button_select([id])``vngen_option_nav(amount)``vngen_do_option_nav(amount)``vngen_option_select([id])``vngen_do_option_select([id])`

- (Recommended: use global search & replace)
- **Updated `vngen_set_lineheight` to support per-entity lineheight multipliers**
  - This required adding two new arguments to `vngen_set_lineheight` for inputting an entity ID and type
  - Existing instances of this function must be updated to match the new syntax
  - (Recommended: use global search & replace)
- **Updated negative pause values to be interpreted as timed delay while auto mode is enabled (rather than indefinite)**
  - While no change is generally required, it is important to note that existing events and `[pause]` markup will now exhibit different behavior while auto mode is enabled
  - To revert to the old behavior, use `vngen_set_auto_type`
  - There is no change in behavior when auto mode is disabled
- **Updated easing macros for simplicity**
  - All `*_in_out` macros have been renamed to their plain counterparts, e.g. `ease_quad_in_out` is now just `ease_quad`
  - Existing instances of these macros must be renamed to match the new syntax
  - (Recommended: use global search & replace)
- **Updated auto prompts to match text scale and rotation, not just position**
  - Any prompt modification actions made to match these properties manually can and should be removed
  - (Recommended: use global search & replace)
- **Renamed `vngen_get_structure` to `vngen_get_struct` for brevity and consistency**
  - Existing instances of this script must be renamed to match the new syntax
  - (Recommended: use global search & replace)
- **Removed legacy animation speed multipliers in favor of GameMaker Studio 2 individual sprite speeds**
  - All sprite assets must be updated to use the intended framerate, either using the GUI or `sprite_set_speed`
  - Existing instances of `vngen_object_draw`, `vngen_log_draw`, `vngen_emote_create`, and `vngen_emote_create_ext` must be updated to remove speed arguments
  - (Recommended: use global search & replace)
- **Replaced individual `vngen_set_vol_*` scripts with a single `vngen_set_vol` script and audio type macros**
  - Existing instances of the old scripts must be updated to use the new syntax
  - See [Macros & Keywords](#) for details
  - (Recommended: use global search & replace)

## 1.0.7

- The `vnngen_option` script now requires an ID to be assigned for the entire option block
  - This is required so that user choices can be saved and restored later
  - Existing instances of this script must be updated to include a real or string ID
  - (**Recommended:** use global search & replace)

## 1.0.6

- Renamed `vnngen_perspective_modify` to `vnngen_perspective_modify_pos` for consistency with other functions
  - Existing instances of this function must be updated to match the new nomenclature
  - (**Recommended:** use global search & replace)
- Removed `vnngen_get_renderlevel` for consistency with other `vnngen_set_*` functions which have no comparable `vnngen_get_*` function
  - Retrieving the values of these properties is only necessary for advanced usage, and can be handled by addressing the property variable directly; in this case, `vnngen_get_renderlevel()` can be replaced with `global.vnngen_renderlevel`
  - (**Recommended:** use global search & replace)

## 1.0.4

- Added support for built-in cursor states to `vnngen_set_cursor`
  - Previously, -1 could be used to reset cursors to their defaults. This value has now been changed to -4 (or keyword 'none') to avoid conflicts with the cursor state `cr_none`, which is a valid cursor
  - Existing instances of this function using -1 to reset cursors must be updated to keyword 'none' to match the new standard
  - (**Recommended:** use global search & replace)
- Added support for auto text positioning in extended option and button functions
  - As 'auto' is also interpreted as -1, text offsets previously at an X or Y coordinate of -1 will need to be changed to other values (e.g. 0 or -2) to avoid enabling auto positioning
  - (**Recommended:** use global search & replace)

## 1.0.3

- Added 'auto' keyword support to `vnngen_text_create_*` actions to better facilitate NVL-style presentation
  - As 'auto' is also interpreted as -1, text entities previously created at an X or Y coordinate of -1 will need to be changed to other values (e.g. 0 or -2) to avoid enabling auto positioning
  - (**Recommended:** use global search & replace)

## 1.0.0

- Added `vnngen_count` to replace individual entity `*_count` functions with a universal function
  - Existing instances of `*_count` functions (e.g. `vnngen_option_count`) must be replaced with the



- new function (e.g. `vnngen_count(vnngen_type_option)`)
- This does NOT include events, which are not entities and therefore still have their own `vnngen_event_count` function
- (Recommended: use global search & replace)
- **Added 'previous' keyword support to character face coordinates in character replace actions**
  - As 'previous' is equivalent to -1, faces positioned at coordinates -1, -1 must be repositioned.
  - (Recommended: adjust sprite offsets to compensate and use global search & replace)
- **Added 'full' keyword support to `vnngen_audio_modify` loop clip settings**
  - Existing instances of `vnngen_audio_modify` using clip settings must add a new argument to enable or disable playing from the current position before clipping
  - This does NOT include instances where clip settings are not used, as clip arguments are optional
  - (Recommended: use global search & replace)
- **Added `*_transformed` functions to VNgen options, replacing the extra color functionality previously added to `vnngen_option_create_ext`**
  - Existing instances of `vnngen_option_create_ext` must have hover and select color arguments removed
  - Alternatively, instances of this script may be replaced with either `vnngen_option_create_transformed` or `vnngen_option_create_ext_transformed`
  - (Recommended: use global search & replace)
- **Updated log buttons to match the new button standard**
  - Existing instances of `vnngen_button_create` must be rewritten to match the updated syntax
  - In order for buttons to scroll the log as before, `vnngen_get_log_button` must be run in the log object Step event to execute `vnngen_log_nav` when buttons are selected (see included demo objects for examples)
- **Replaced per-entity text speed with `vnngen_set_speed`, which sets speed for all text entities.**
  - Existing instances of `vnngen_text_create`, `vnngen_text_create_ext`, and `vnngen_text_replace_ext` must be modified to remove the "speed" argument.
  - Text speed should now be set with `vnngen_set_speed` in the Create Event, or in VNgen events with `vnngen_script_execute_ext`.
  - (Recommended: use global search & replace)
- **Replaced `vnngen_instance_create` with `vnngen_instance_change`**
  - Existing instances of `vnngen_instance_create` must be replaced with either the new function or the standard `instance_create` function coupled with `vnngen_object_clear` (if destroying the current object is desired)
  - (Recommended: use global search & replace)
- **Replaced bracket escape character with `^`**
  - Existing instances of escaped markup must be replaced with the new syntax, i.e. `[` becomes `^[`
  - This does NOT apply to the newline character, which is still `n`
  - (Recommended: use global search & replace)
- **Removed `vnngen_option_exists` as its functionality has been superseded by `vnngen_exists`**
  - Existing instances of `vnngen_option_exists` must be replaced with `vnngen_exists(any, vnngen_type_option)`
  - (Recommended: use global search & replace)
- **Removed language macros as they have been superseded by `vnngen_type_*` macros**
  - Existing instances of `lang_text` and `lang_audio` must be replaced with `vnngen_type_text` and `vnngen_type_audio`
  - The `lang_all` macro has no replacement, and the 'type' argument in `vnngen_set_lang` can now be omitted to set both types simultaneously instead
  - Instances of `vnngen_set_lang` and `vnngen_get_lang` using integers instead of macros to set type do not need to be changed
  - (Recommended: use global search & replace)
- **Removed `ef_hap_ramp_down` and renamed `ef_hap_ramp_up` to `ef_hap_ramp`**
  - As effects and animations can now be reversed, bidirectional ramp scripts were no longer necessary



- Existing instances of the old effects must be renamed to `ef_hap_ramp` and reversed if desired
- **(Recommended):** use global search & replace)

## 0.9.9 (Early Access)

- **Added min/max pitch arguments to vox actions. Pitch is now randomized between these two values, rather than being a switch enabled by setting pitch to -1**
  - Existing instances of `vngen_vox_play` and `vngen_vox_modify` must be updated to include a second pitch argument.
  - Pitch must now be written as min/max values. Using -1 or keyword 'auto' for random pitch is now invalid.
  - **(Recommended):** use global search & replace)
- **Rearranged `vngen_vox_play` arguments so that pitch comes before volume for consistency with other audio functions.**
  - Existing instances of this script must be updated to have their argument order switched to match this change.
  - **(Recommended):** use global search & replace)
- **Updated effects to use custom variable array**
  - Existing effect scripts must be updated to reflect this change. Effect variables previously named "`ef_var0`", "`ef_var1`", and so forth must now be written as "`ef_var[0]`" and "`ef_var[1]`".
  - More than 16 variables can now be used with this method. Any previous workarounds may be replaced with the built-in system.
  - **(Recommended):** use global search & replace)
- **Updated deforms to use 2D point arrays**
  - Existing deform scripts must be updated to reflect this change. Deformation variables previously named "`def_xpoint0`", "`def_xpoint1`", and so forth must now be written as "`def_xpoint[0, 0]`" and "`def_xpoint[1, 0]`".
  - Points are now properly treated as a grid of columns and rows ranging from [0, 0] to [1, 3], not a linear range of 0-7.
  - **(Recommended):** use global search & replace)

## 0.9.8 (Early Access)

- **The `make_color_hex` function has been renamed `make_color_hex_to_rgb` for consistency with its new opposite function, `make_color_rgb_to_hex`.**
  - Existing instances of `make_color_hex` must be renamed.
  - **(Recommended):** use global search & replace)
- **Audio wrappers have been removed in preparation of v1.0. If you have not updated to the new audio functions yet, you must do so now.**
  - Existing instances of old audio and vox functions must be renamed
  - **(Recommended):** use global search & replace)
- **Fixed incorrect scaling when launched directly in fullscreen by sampling base resolution when `vngen_set_scale` is run.**
  - This means unlike previous versions, `vngen_set_scale` should NOT be run in the Step event or any event where it will be executed repeatedly. Ideally, `vngen_set_scale` should ONLY be run in the Create event instead.

## 0.9.7 (Early Access)

- **Updated `vnngen_log_draw` to allow changing separation between entries.**
  - Existing instances of this script must be updated to include a separation value (default 64).
  - **(Recommended:** use global search & replace)
- **Replaced `vnngen_set_legacy` with `vnngen_set_renderlevel`.**
  - Existing instances of the old function must be replaced with the new one.
  - While it is not required to change the values set in the old function, optional new values are also available. It is recommended to see documentation on the new function before updating.
  - **(Recommended:** use global search & replace)

## 0.9.6 (Early Access)

- **Externalized transitions as a new category of keyframe animations.**
  - Transitions are now referenced as scripts, not macros, and any instances of their old numeric equivalents will need to be updated to match the new script names.
  - **No change is necessary if macros were used as intended.**
  - The `trans_none` macro is still valid and can still be used to disable transitions. This is NOT the same as the numeral 0, which can no longer be used to disable transitions.
  - Transitions out are now reversed from transitions in. For transitions with directional counterparts, this may require switching directions to retain previous behavior.
  - **(Recommended:** use global search & replace)
- **Added support for performing animations, deformations, and effects in reverse.**
  - Existing instances of `vnngen_*_anim_start`, `vnngen_*_deform_start`, and `vnngen_effect_start` must be updated to include a 'reverse' argument.
  - **(Recommended:** use global search & replace)

## 0.9.5 (Early Access)

- **Updated tiled scenes to behave like standard scenes.**
  - Adjustments being made to compensate for old tiled scenes can be removed
  - Alternatively, the old style of tiled scenes can be kept by running `vnngen_set_legacy(true)`.
- **Removed scale from perspective calculations.**
  - Elements relying on scale for changes to parallax strength must have z-index adjusted to compensate (ratio of z-index to scale is 100:1)
- **Fixed character flipping in `vnngen_char_replace_ext` being absolute instead of relative**
  - Existing instances of `vnngen_char_replace_ext` must be updated to set the intended character orientation
  - **(Recommended:** use global search & replace)

## 0.9.4 (Early Access)

- **Added optional easing support to non-extended actions.**
  - Text and label actions using real numbers as language tags must either have a separate ease argument supplied prior to the language tag, or the language tag must be converted to a string
  - All other actions, and text and label actions using string language tags DO NOT require modification
- **Added 'idle' argument to `vnngen_char_create_ext` for consistency with `vnngen_char_replace_ext`.**

- This argument was originally omitted due to GameMaker Studio's 16-argument limitation, which has now been removed.
- Existing instances of `vngen_char_create_ext` must be updated to add an 'idle' argument.
- **(Recommended:** use global search)
- **Added 'name' argument to `vngen_text_create_ext` and `vngen_text_replace_ext` for consistency with other functions.**
  - This argument was originally omitted due to GameMaker Studio's 16-argument limitation, which has now been removed.
  - Existing instances of `vngen_text_create_ext` and `vngen_text_replace_ext` must be updated to add a name argument, or "" for none or if name markup is used instead.
  - **(Recommended:** use global search & replace)
- **Fixed option position being calculated incorrectly.**
  - Existing options using sprite offsets will need to be repositioned to stop compensating for sprite offset, as this is now properly factored into position automatically.
  - **(Recommended:** use global search & replace)
- **Simplified `vngen_option_create` in lieu of `vngen_option_create_ext`.**
  - Existing instances of `vngen_option_create` must be updated to use the new simplified syntax or be replaced with `vngen_option_create_ext`.
  - **(Recommended:** use global search)
- **Renamed audio functions for consistency with other VNgen functions.**
  - Wrappers have been added which allow using the old syntax, which is consistent with standard GameMaker Studio syntax. However, users are encouraged to switch to the new VNgen naming scheme, as wrappers do incur a small performance penalty.
  - **Exception:** `vngen_sound_modify` is wrapped as `vngen_modify_sound`. Existing instances of `vngen_sound_modify` must be replaced with either the new VNgen syntax or the GameMaker syntax wrapper.
  - **(Recommended:** use global search & replace)
- **Renamed `vngen_get_option_number` to `vngen_option_count` for consistency with other functions.**
  - Existing instances of `vngen_get_option_number` must be renamed to `vngen_option_count`.
  - **(Recommended:** use global search & replace)
- **Renamed `vngen_get_option_active` to `vngen_option_exists` for consistency with other functions.**
  - Existing instances of `vngen_get_option_active` must be renamed to `vngen_option_exists`.
  - **(Recommended:** use global search & replace)

### 0.9.3 (Early Access)

- **Merged style inheritance data for text and labels, which are now saved and loaded as a single string.**
  - Save files generated with previous versions of `vngen_file_save` will need to be regenerated for style data to persist. For projects spanning multiple VNgen objects, this may require using `vngen_goto` to revisit previous objects in order to rebuild style data.
- **Renamed `vngen_log_get_size` to `vngen_log_count` for consistency with other functions.**
  - Existing instances of `vngen_log_get_size` must be renamed to `vngen_log_count`
  - **(Recommended:** use global search & replace)
- **Renamed `vngen_get_event` to `vngen_event_get_index` for consistency with other functions.**
  - Existing instances of `vngen_get_event` must be renamed to `vngen_event_get_index`
  - **(Recommended:** use global search & replace)
- **Migrated vox audio to its own data structure.**
  - **No change** is required to existing instances of `vngen_play_vox`.

- Existing instances of `vngen_stop_sound` to end vox must be renamed to `vngen_stop_vox`.
- (**Recommended:** use global search & replace)
- **Skipping to event IDs greater than or equal to the total number of events now skips past the final event rather than ending the skip operation when the final event begins.**
  - Any instances of `vngen_goto` using large values to brute-force skip to the final event must be edited to skip precisely to the event itself.
  - (**Recommended:** use the new `vngen_event_count` function to navigate precisely to the final event, minus one)

### 0.9.1 (Early Access)

- **Added `vngen_emote_create_ext`.**
  - Existing instances of `vngen_emote_create` must be renamed to `vngen_emote_create_ext`.
  - (**Recommended:** use global search & replace)
- **Added `trans_zoom_out` transition and replaced `trans_scale` with `trans_zoom_in`.**
  - Existing instances of `trans_scale` must be renamed to `trans_zoom_in` to complement the new transition.
  - (**Recommended:** use global search & replace)
- **Reversed argument order in `vngen_set_lang` for consistency with other functions.**
  - Existing instances of `vngen_set_lang` must have their arguments ordered 'language' first, then 'type'.
  - (**Recommended:** use global search & replace)

## 4.2. Features & Changes From Edge VN

```
?table { border: 1px solid #555; } table td { width: 50%; } table tr:nth-child(odd) td { background-color: #555; } table tr td::before { content: "- "; color: #F00; } table tr > td:first-of-type::before { content: "+ "; color: #0F0; } table tr:first-of-type td::before { content: ""; } @media screen and (max-width: 767px) { table tr td { width: 100%; display: inline-block; } }
```

- Fully rewritten from the ground up to utilize a powerful new structure and scripting syntax
- New timeline structure allows any functions to execute simultaneously or in sequence, not just relative to text
- Timeline uses automatic IDs, making editing large scripts as easy as cut and paste. Other element IDs are arbitrary—name them whatever you like!
- Separate functions for create, replace, modify, destroy, and more, each with their own transitions
- Over 50 built-in animations and animation modes, plus extensible scripted animation systems for virtually limitless possibilities!
- Systematic design gives uniform feature sets across all major elements and makes future add-ons possible
- Global parallax camera perspective controls give a sense of depth without being tied to a particular scene
- Custom depth sorting for each VNgen element
- Enhanced backlog with support for multiple simultaneous strings and related audio, as well as a new display format with better customization and more fluid touch support
- Built-in debug mode and command console saves time creating and debugging projects

## Comparison:



Nonlinear event-based timeline	Linear text-based timeline
Data structure-based memory management	Array-based memory management
Adapts to delta time	Adapts to framerate
Unified animation systems for most entities	Segregated animation systems for limited entities
Full 30-mode ease support for all animations	Limited easing on limited animations
Separate transitions for in/out	Forced bidirectional transitions
Dynamic mesh support for most entities	Fixed mesh support for characters only
Six sprite scaling modes for most entities	Two sprite scaling modes for backgrounds only
Advanced “3D” camera-style perspective	Basic parallax perspective
Semi-automatic depth sorting	Manual depth sorting
Full color backlog	Plain text backlog
Integrated per-entity GLSL shader support	No integrated shader support
Advanced custom effects system	N/A
Composite character attachments	N/A
Four-color gradient blending on most elements	N/A
Basic save data system	N/A
Built-in debug mode with visual stats and helpers, plus extensible command console	N/A
Text markup support for font (includes bold, italic, family, and size), up to four color gradient, shadow color, outline color, event, link, speed, and pause	Text markup support for bold, italic, color, and link
Integrate your own custom code without modifying anything!	Requires modification for custom code
Full GameMaker Studio 2 support	Partial GameMaker Studio 2 support

## 5. Getting Started

XGASOFT VNgen is an extension of [GameMaker Studio by YoYoGames](#). To use it, you must have a registered license of:

- GameMaker Studio 1.4 or newer
- GameMaker Studio 2.1 or newer

GameMaker Studio is a product of YoYoGames, and XGASOFT does not provide support for GameMaker Studio itself or its related products and services. Should you encounter any issues obtaining, installing, or using GameMaker Studio, contact [YoYoGames support](#) for assistance.

In this section, we'll examine how to set up and install VNgen in GameMaker Studio, as well as explore VNgen's core structure and how to create your first visual novel.

## 5.1. Setup & Installation

### 1. Obtaining VNgen

VNgen is available for purchase in two forms:

1. As a [GameMaker Marketplace asset](#)
2. As a [standalone extension](#)

Both versions are functionally equal, however the setup process differs slightly depending on which version of VNgen you have purchased.

### 2a. Installation – Marketplace

Installing an asset purchased from the GameMaker Marketplace is easy.

1. First, open GameMaker Studio and create a new project, or open an existing project you wish to add VNgen into.
2. Choose the **Marketplace** tab at the top of the screen and select **My Library**. If you are not currently logged in to your YoYo Games account, you may be prompted to do so before accessing your asset library.
3. Once you have purchased VNgen, it will appear in your asset library window. You may download it now by clicking the **download** icon.
4. Once VNgen is downloaded, click the **import** icon and choose **Add All**, then **Import**.
5. VNgen will be added to the current project.

To update an existing project, simply delete all VNgen folders and follow the steps above again. (Make sure to back up any custom code and assets you may have added to the VNgen folders beforehand.)

### 2b. Installation – Extension

Installing an asset from a standalone extension follows a similar process as a Marketplace asset, however the extension will not be added to your Marketplace library.

#### GameMaker Studio 1.x

##### To Install:

1. First, download the extension anywhere on your hard drive and extract it, if necessary.
2. Next, open GameMaker Studio and create a new project, or open an existing project you wish to add VNgen into.
3. In your project's Resources panel, right-click on the Extensions folder and choose **Import Extension**.
4. Locate the extension (\*.gmez) on your hard drive and import it.
5. Right-click on the imported extension and choose **Import Resources**.
6. Choose **Import All**, then **OK**.
7. VNgen will be added to the current project.

##### To Update:

1. First, download the updated extension anywhere on your hard drive and extract it, if necessary.
2. Next, open your existing GameMaker Studio project.
3. In your project's Resources panel, **delete all VNgen folders and files**, aside from any custom assets you have created.
4. Right-click on the Extensions folder and choose **Import Extension**.
5. Locate the extension (\*.gmez) on your hard drive and import it.
6. Right-click on the imported extension and choose **Import Resources**.
7. Choose **Import All**, then **OK**.
8. The latest version of VNgen will be added to the current project.

## GameMaker Studio 2.x

### To Install:

1. First, download the extension anywhere on your hard drive and extract it, if necessary.
2. Next, open GameMaker Studio and create a new project, or open an existing project you wish to add VNgen into.
3. Locate the extension (\*.yyp) on your hard drive and drag-and-drop it into your GameMaker Studio window.
4. A message will appear indicating a marketplace package has been detected. Choose **Yes** to import the package.
5. Choose **Add All**, then **Import**.
6. VNgen will be added to the current project.

### To Update:

1. First, download the extension anywhere on your hard drive and extract it, if necessary.
2. Next, open your existing GameMaker Studio project.
3. In your project's Resources panel, **delete all VNgen folders and files**, aside from any custom assets you have created.
4. Locate the extension (\*.yyp) on your hard drive and drag-and-drop it into your GameMaker Studio window.
5. A message will appear indicating a marketplace package has been detected. Choose **Yes** to import the package.
6. Choose **Add All**, then **Import**.
7. The latest version of VNgen will be added to the project.

## 3. Project Setup

Once VNgen is added to your project, you may proceed to create content with it! Demo objects are included to get you started, but there are times when you may wish to start from scratch as well.

**Important note 1:** GameMaker Studio 1.x users must import the included **macros.txt** file located in the **Included Files** folder before using VNgen. This can be done from the **Resources > Define Macros** menu and selecting **Load**.

**Important note 2:** The included mascot demo runs at 4K resolution and requires textures pages at least **4096x4096** in size. This option can be set via **Game Options > Platform Settings > [Your Platform] > Graphics > Texture Page Size**

To start, you'll need to create two objects: these will be used as a **backlog** and **script**, respectively. (Using a backlog is optional, but recommended.) VNgen is highly flexible and can operate in a wide variety of configurations, but for standard functionality you should set up your objects to run the following functions in the

corresponding events:

## Backlog Object

### ▪ Create Event

- `vngen_log_init(50);`

### ▪ Step Event

- `vngen_do_log_nav_touch();`

### ▪ Draw GUI Event

- `vngen_log_draw(spr_log_bg, spr_divider, spr_audio_off, spr_audio_on, 64, display_get_gui_width()*0.75, inherit, inherit);`

### ▪ Press Up Event

- `vngen_do_log_nav(-1);`

### ▪ Press Down Event

- `vngen_do_log_nav(1);`

### ▪ Press Space Event

- `vngen_do_log_play();`

### ▪ Press L Event

- `vngen_do_log_display();`

## Script Object

### ▪ Create Event

- `vngen_object_init();`

### ▪ Step Event

- `vngen_event_set_target();`
- `if (vngen_event()) {`  
    `//Actions`  
    `}`
- `vngen_event_reset_target();`

### ▪ Global Mouse Left Released Event

- `vngen_do_continue(snd_continue);`



- **Draw Event**

- `vngen_object_draw(0, 0, true);`

- **Release Enter Event**

- `vngen_do_option_select();`

- **Press Up Event**

- `vngen_do_option_nav(-1);`

- **Press Down Event**

- `vngen_do_option_nav(1);`

- **Press A Event**

- `vngen_do_auto(toggle, 1.5);`

- **Press V Event**

- `vngen_do_ui_display(toggle);`

- **Press Escape Event**

- `vngen_do_pause(toggle);`

Each of these functions will be covered in detail later, so for now just think of this as a standard VNgen configuration. Especially take note of the script object's Step Event, as this is where most of the VNgen magic will take place.

Once your objects are set up and placed in the room of your choice, you're ready to move on to the next section: [learning to create with Q-script](#).

### 5.1.1. HTML5 Setup & Best Practices

VNgen partially supports HTML5 as a target platform. Many advanced features available to other platforms will not work in HTML5, however it is still possible to create basic visual novels with VNgen which can be enjoyed right within your web browser.

Before exporting, it is recommended to disable the following two settings in order for VNgen to function best in HTML5:

1. Global Game Settings > HTML5 > Graphics > WebGL – **disabled**
2. Global Game Settings > HTML5 > Graphics > Center the game in the browser – **disabled**

These settings are optional, and may be left enabled depending on your intended usage of VNgen in HTML5. However, some features may not work as you expect, and others may not work at all. That being said, please

keep these additional points in mind:

- Centering the game in the browser should only be left enabled if view scaling with `vnngen_set_scale` is disabled. However, due to the varying dimensions of browser windows, it is highly recommended to enable view scaling with HTML5.
- HTML5 will render VNgen at renderlevel 2 by default. This is the simplest and most compatible rendering method, but lacks many advanced features like shaders and deformations. To enable these features again, renderlevel can be overridden with `vnngen_set_renderlevel`, where renderlevel 0 is the highest method and renders all features, and renderlevel 1 renders most features, but with some compromises to visuals for compatibility. Many of these features require WebGL to work properly, and some will not work with HTML5 even if the renderlevel setting is overridden.
- **Build HTML5 games for HTML5 first.** It is much easier to work within the constraints of HTML5 and then add additional effects for other platforms, rather than create a highly advanced visual novel first and then port it down to HTML5.

Note that these recommendations are subject to change with further updates to VNgen and the HTML5 export module for GameMaker Studio itself.

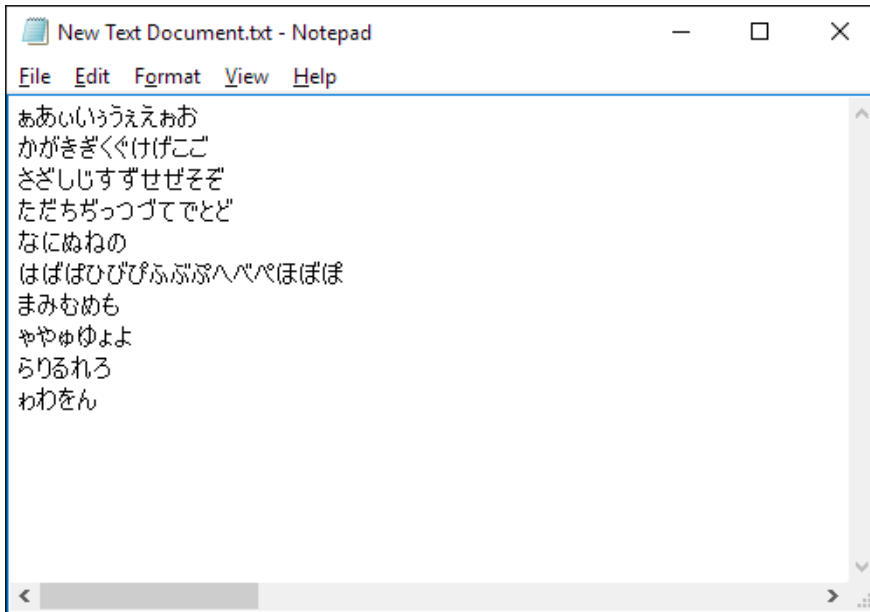
## 5.1.2. Using Additional Languages in VNgen

While GameMaker Studio (and by consequence, VNgen) is designed with Latin text drawing in mind, it is also entirely possible to tell your stories in completely different languages and alphabets, provided you import the appropriate font and special characters to your project. **Special characters** refers to any non-Latin font elements beyond the scope of English text, numbers, and symbols. While a non-Latin font may contain all the characters it needs to display text in a different language, simply importing the font into Game Maker Studio will not import these characters by default. Instead, GameMaker Studio must be instructed to look for the characters in the font file specified so that more than the standard range of English characters is actually added to the game.

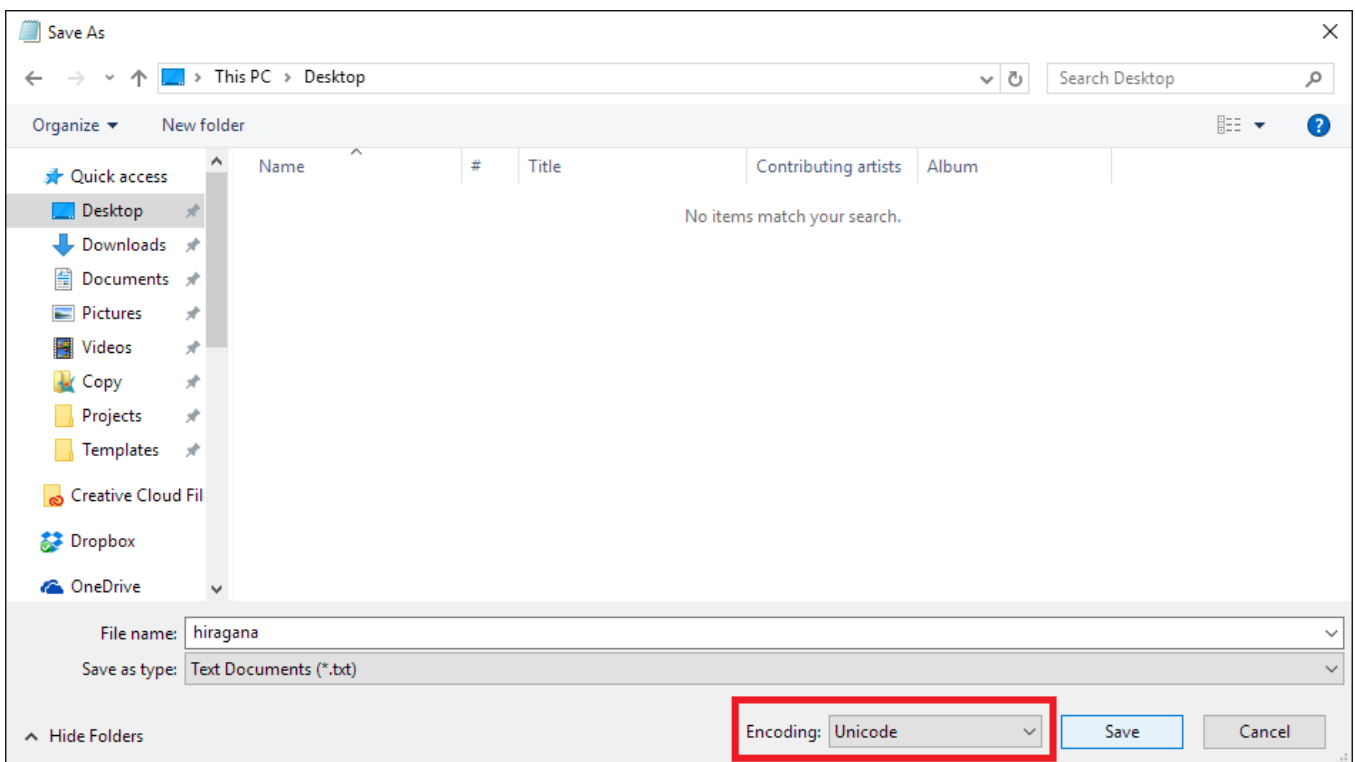
To do this, first create a new font asset in your project and select a non-English font from the dropdown list in the font window. As you will see, by default only the character range 32 to 127 will be imported to GameMaker Studio.



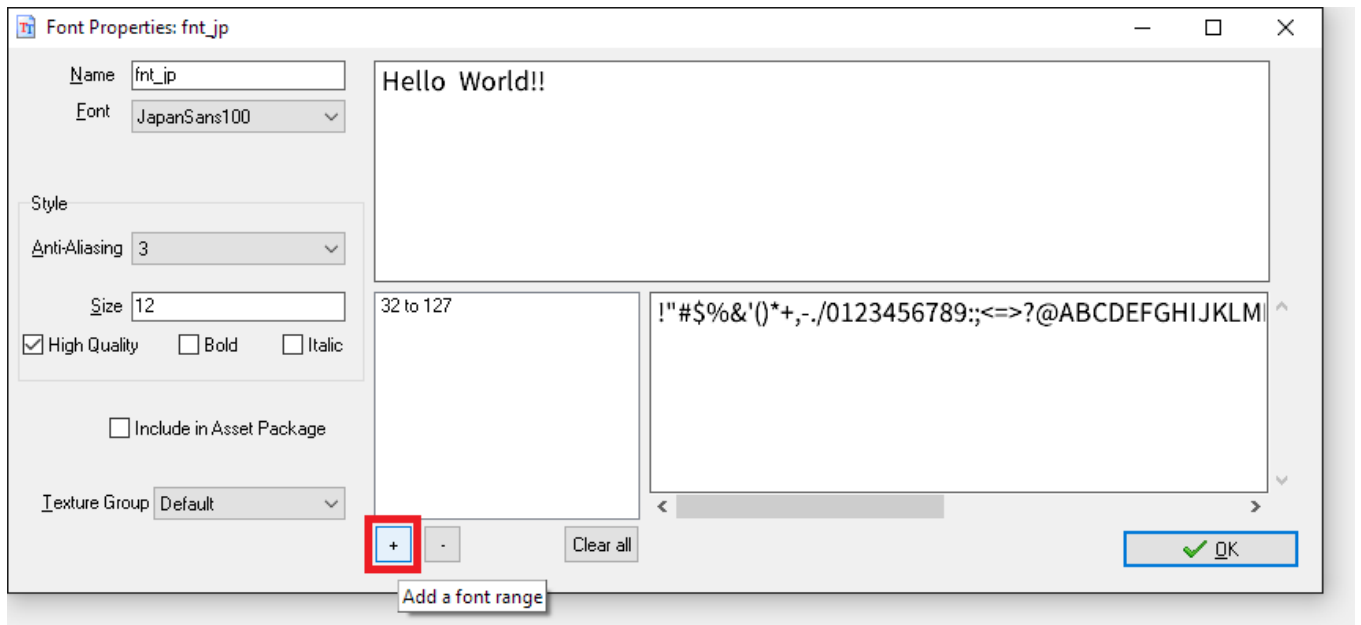
To add additional characters from the font file into the font asset, first open your text editor of choice and begin typing all the basic special characters you will need. This will be used to tell GameMaker Studio which special characters to look for in the font file. Don't worry about missing one or two—you can always adjust the imported character range later.



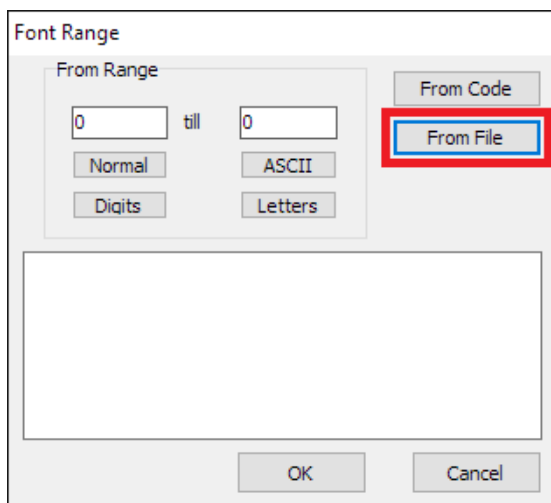
With your text file prepared, save it to your local hard drive as a .txt file with encoding set to 'Unicode'. **This step is important**, as your special characters will be lost if you attempt to save with different encoding.



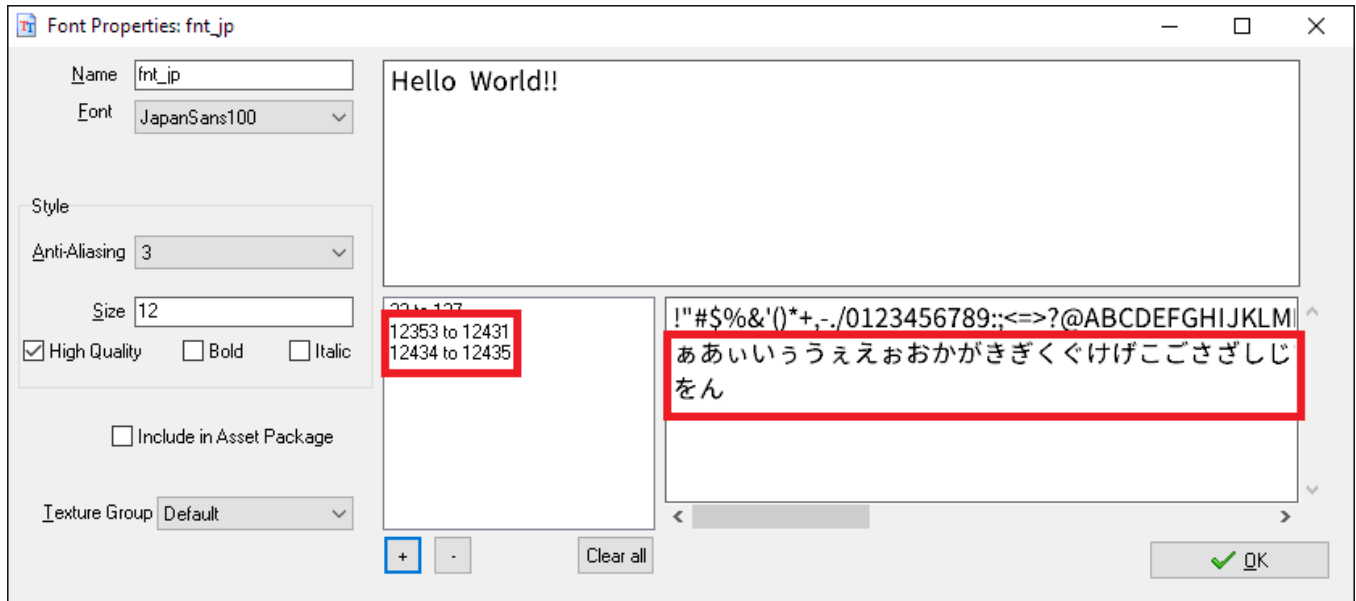
Next, return to your font asset in Game Maker Studio and select the '+' icon beneath the font range indicator.



This will open a new window where you can set the range of characters to be imported from the font file. Rather than guess at these values, you can now instead import your text file by clicking the 'From File' button and GameMaker Studio will detect them automatically.



The font range will now have one or more additional entries in it, indicating the range of characters represented in the imported text file.



If there is more than one new font range, it is likely that some important characters were missed. To avoid encountering missing characters in your game text, select the '+' icon again and simply edit the values manually so that the range spans from the smallest to the largest value available.

With that, you are now ready to use your new language in VNgen!

## 5.2. Introduction to Q-script

**Q-script** is an abbreviation for **Quantum script**, or code which runs in **Quantum**, the core framework products such as [VNgen](#) are built on. [Quantum](#) is a framework by XGASOFT designed to assign numeric IDs to blocks of code itself and provide a set of common practical functions to manipulate them, allowing for the creation of entirely custom programming languages within the **GameMaker Language** (or **GML**).

### Structure

Q-script is structured in a hierarchy of **events** and **actions**, with some actions occasionally functioning as sub-events. Each event is self-contained, and will be executed in sequence. Individual actions within an event are executed simultaneously when the containing event is active. Once all of an event's actions are complete, the current event will deactivate and the next event will be activated in its place. This process repeats itself until all events have been executed or the Q-script is terminated prematurely by the user.

Although Q-script itself does not require that events and actions be triggered by logical operators, for the sake of performance Quantum events are usually designed to be preceded by an 'if' statement, with actions following between brackets.

#### Example:

```
if (vnngen_event()) {  
  
    //Action 1  
  
    //Action 2
```

```
//Action 3  
  
}
```

This sequence creates a single event, and should always be run in the Step Event of the running object. While it may appear that having many 'if' statements testing the same function will cause all of them to return the same result (by being either active or inactive simultaneously) Quantum enables each 'if' statement to return a unique result based on the corresponding event's internal ID, which is generated automatically. Like quantum physics dealing with properties in indeterminate states, this unique and powerful functionality is what earned this framework the name 'Quantum'.

## Initialization

Q-script is initialized in three phases:

1. The object **Create Event**
2. The **\*\_event\_set\_target** operation
3. The **first frame** of a **new, active** Quantum event

If the last two sound unfamiliar, don't worry: these initialization phases are unique to Quantum and only require setting up once.

But first, the object **Create Event**. Before Q-script can run, the object it's written in must be set up to run it. This is done with a simple **\*\_init** script, which can be customized to suit the needs of the individual product built on Quantum. In the case of VNgen, for example, this script is named 'vngen\_object\_init' and is customized with values that support both VNgen and Quantum itself out of the box.

### Example:

```
vngen_object_init();
```

The other two initialization phases occur in the same object's **Step Event**. Typically, all code placed in GameMaker Studio's Step Event is executed every frame, but Q-script prevents this by only executing one Quantum event at a time. Again, this is due to its ability to assign numeric IDs to individual blocks of code—a process which itself must be initialized before Q-script can be used. This is done with the **\*\_event\_set\_target** script, which must be run *before* any Q-script in the Step Event. Likewise, it's just as imperative that the event target be *reset* once all Quantum event code is complete, which is handled by the **\*\_event\_reset\_target** script run *after* any Q-script in the code editor. Together, these two functions signal GameMaker Studio when Q-script begins and ends.

### Example:

```
vngen_event_set_target();  
  
if (vngen_event()) {  
  
    //Action 1
```

```
//Action 2

}

if (vnngen_event()) {

    //Action 1

    //Action 2

}

vnngen_event_reset_target();
```

The third and final initialization phase is performed within actions themselves, and can serve a variety of purposes depending on the action. Unlike the previous two phases, actions initialize automatically. However, it is important to be aware that any actions written in Q-script will not initialize until their containing event is active. When that occurs, there is a single frame in which all actions may perform any initialization necessary before proceeding onto any time-based activity, such as animations. There is no limit to how brief or long an individual action may be, but until all actions are complete or skipped by the user, the next event will not be activated.

## Manipulating Events

As Quantum events are essentially nodes on a timeline, there will often come occasions when it is desirable to alter individual events' behavior in a few key ways, such as:

- **Timing**
- **Persistence**
- **Execution order**

These pertain to three optional arguments which can be supplied when creating Quantum events: **pause**, **noskip**, and **label**.

First, setting an event **pause** will delay execution of all further actions by the number of seconds supplied. Since events progress automatically, there may be times when stringing one event's actions to the next feels too abrupt, or occurs too quickly for the user to keep up. In cases such as these, an event pause can work wonders for creating a smooth sense of flow, and in the case of sequenced animations, good timing is critical.

**Tip:** You can also add pauses between individual actions with an `*_event_pause` script

Second, an event's **persistence** refers to whether or not its actions can be skipped by the user. If the 'noskip' argument is set to 'true', all actions will play out automatically for their full duration regardless of user input. However, be aware that some actions may override this behavior to prevent scenarios where events become impossible to complete due to user input being required.

The final property, the **label**, assigns a string to identify the event in addition to the automatically-generated numeric ID. Though not required, labels are inherently more memorable and become quite useful when used in combination with `*_goto`. As the name implies, running `*_goto` (e.g. `vnngen_goto("my_event");`) will *go to* the Quantum event of your choice, effectively jumping anywhere in the timeline, forwards or backwards or even to other objects! This gives full control over the order events are executed.

These three arguments can be supplied in any order and any combination, but be aware that numeric values will always be interpreted as 'pause' first, then 'noskip'.

**Example:**

```
vngen_event_set_target();

if (vngen_event("first_event")) {

    //Action

}

if (vngen_event(2)) {

    //Delayed action

}

if (vngen_event(0, true, "last_event")) {

    //Action

    vngen_event_pause(2);

    //Delayed action

}

vngen_event_reset_target();
```

## Manipulating Actions

While Quantum events perform a fairly straightforward function, Quantum *actions* are much more difficult to define. There are very few limits on what actions can achieve, and some can even behave as sub-events themselves! However, actions are **not** plain code, nor should plain code be executed within a Quantum event. While events' use of 'if' statements means this is possible to some extent, plain code will inherit almost none of the benefits offered by Q-script. It will have no third initialization phase, no ability to be skipped or prevented from skipping, and no protection from being executed repeatedly for as long as the event is active.

Therefore, Quantum actions, in the most basic sense, become Quantum actions when they are designed to follow a template which *does* inherit the benefits of Q-script: first they are initialized, then they are executed, then they perform any conditions for testing when the action is complete. Many actions also provide special conditions for how to behave when skipped.

With these basic behaviors in place, each Quantum-based product can use actions to achieve almost any effects desired. For more information, see related documentation for the action scripts you're most interested in.

**Tip:** While you may not be able to run plain code in Quantum events, VNgen includes an action which 'wraps' other scripts to be executed as if they were Q-script, no modification required! See: `vngen_script_execute()`.



## 5.3. Creating Your First Visual Novel

At this point you should have a basic understanding of how to set up and run VNgen, as well as a basic understanding of Q-script and its role in your projects.

Armed with this knowledge, you are now ready to begin creating your first visual novel in VNgen!

### 5.3.1. Entities & Action Types

As you already know, VNgen is built upon the Quantum framework by XGASOFT. In essence, this makes VNgen a collection of **actions** to be run in Quantum's **events**.

VNgen's actions can be categorized into several groups where each group relates to a separate **entity**. There are 14 main entity types in VNgen:

- The perspective camera
- Scenes
- Characters
- Character attachments
- Emotes
- Textboxes
- Text
- Labels
- Prompts
- Buttons
- Options
- Audio
- Vox
- Effects

Each entity may have as many as 15 functions or even more, but don't worry: while it might seem like a lot to learn at first glance, most of these functions are standardized so you only need to familiarize yourself with one entity to have a basic grasp on all of them. There are exceptions, however, so be sure to read the full reference guide to get the most out of VNgen.

In general, each entity possesses functions to...

- Create a new instance of an entity
- Modify an existing entity's position, scale, rotation, and color blending
- Replace an existing entity with a new one
- Destroy an existing entity
- Animate an existing entity
- Deform an existing entity
- Apply a shader on an existing entity

Some of these functions are provided in multiple variants which, again, are typically standardized across other entities. The best example of this is **extended** functions, which provide the same functionality as their simpler counterparts while *extending* the number of available options to customize the entity being modified.

#### Example:

```
vnngen_textbox_create("textbox", spr_textbox, 0, 1080, 0, trans_fade, 3);
```

```
vnngen_textbox_create_ext("textbox", spr_textbox, orig_left, orig_bottom, 0, 1080, 0,  
    scale_stretch_x, trans_fade, 3, ease_cubic_out);
```

**Note:** Because both scripts serve the same purpose, they should **never** be used to create the same entity twice as shown in the example above. This is merely to demonstrate what the two types of functions look like side-by-side.

In the example above, both scripts will create a new textbox named “textbox”, display it at the bottom of the screen, and fade it in with a duration of 3 seconds. However, the first script assumes the sprite named ‘spr\_textbox’ is already aligned to the bottom left corner, while the second script allows us to determine its **origin point** on the fly. The extended function also allows us to **scale** our textbox, in this case stretching it horizontally to fill the screen regardless of the sprite’s dimensions and the screen’s display resolution. Last, but not least, we can even specify an **ease** mode for the fade transition in the more advanced function, while the simpler version does not require easing.

Each of these properties will be covered in detail later, so don’t worry about all the extended options for now. This just goes to show why multiple versions of the same function exist: in the end, they’re all just different ways to perform the same handful of fundamental operations as listed above: **create, modify, replace, animate, destroy**.

### 5.3.2. A Simple Dialog

In any case, the best way to learn is by doing. With your project prepared as shown in the setup guide, your script object’s Step Event should currently look something like this:

**Example:**

```
vnngen_event_set_target();  
  
if (vnngen_event()) {  
    //Actions  
}  
  
vnngen_event_reset_target();
```

Now it’s time to start filling those events with actions!

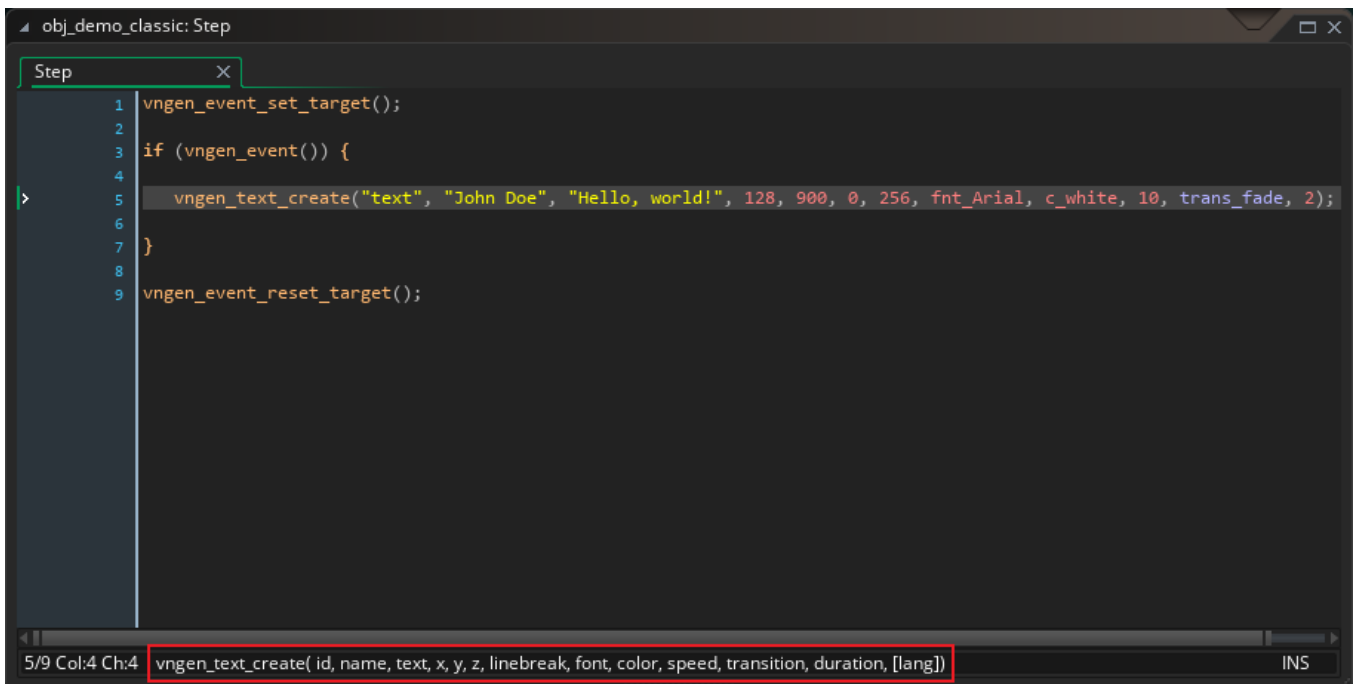
Let’s start with some text:

**Example:**

```
vnngen_event_set_target();  
  
if (vnngen_event()) {  
    vnngen_text_create("text", "John Doe", "Hello, world!", 128, 900, 0, 256, fnt_Aria  
1, c_white, trans_fade, 2, true);  
}
```

```
vnngen_event_reset_target();
```

The simplified `vnngen_text_create` script accepts arguments in the following order: *ID*, *speaker name*, *text*, *x*, *y*, *z*, *linebreak* (or *width*), *font*, *text color*, *transition*, *transition duration*, and *ease mode*. In practice, you won't need to remember this order since syntax guides will always be provided at the bottom of your code editor window.



In this example, the text “Hello, world!” is assigned to the character “John Doe” and displayed 128 pixels from the left and 900 pixels from the top of the screen. At 256 pixels wide, the text will wrap into a new line, but this should be plenty to fit our text with the chosen font, ‘fnt\_Arial’. Furthermore, the text faded onto the screen over a period of two seconds.

Quite a lot of information conveyed in a single, brief line!

But what if we want to wipe the screen clean and start a new line of text from scratch? Do we have to destroy the first one and create it again?

While that is certainly possible, there's a much better way: *replacing* the text we already created.

#### Example:

```
vnngen_event_set_target();

if (vnngen_event()) {

    vnngen_text_create("text", "John Doe", "Hello, world!", 128, 900, 0, 256, fnt_Aria
1, c_white, trans_fade, 2, true);

}

if (vnngen_event()) {
```

```
    vngen_text_replace("text", "John Doe", "How are you?", fnt_Arial, c_white, 2);  
}  
  
vngen_event_reset_target();
```

First, we create a new event. Remember: events are like nodes on a timeline, where everything contained inside an event occurs simultaneously, while events themselves play out in sequence. Since we don't want our original text to get replaced until the user has read it, that means the replace operation must occur in a separate event.

In the first event, we created a line of text with the ID "text". Now we can create a new event and manipulate the same line of text by addressing its ID. In the example above, only the line of text being spoken has changed, but if we wanted, we could also change who's speaking, what font and color the text appears in, and how long to fade from the old text to the new text.

Of course, it's a bit of a waste to specify all these values again and again if only one property is changing. Fortunately, there's a way we can simplify our example even more:

**Example:**

```
vngen_event_set_target();  
  
if (vngen_event()) {  
    vngen_text_create("text", "John Doe", "Hello, world!", 128, 900, 0, 256, fnt_Arial,  
        c_white, trans_fade, 2, true);  
}  
  
if (vngen_event()) {  
    vngen_text_replace("text", previous, "How are you?", inherit, inherit, 2);  
}  
  
vngen_event_reset_target();
```

Notice how in this version of the example, the character name, font, and text color aren't specified at all, but instead use keywords to save space and time. **Keywords** are macros which trigger special behaviors in certain scripts.

**Note:** Not all properties support keywords. Refer to the full reference guide to learn which properties support which keywords.

In this case, 'previous' is used to set the speaker name to the name used in the *previous* instance of the text ID "text". 'Inherit' is even more special: each time you specify a new text font or color, it is assigned to the speaker name and stored in memory for the duration of the entire visual novel. And the 'inherit' keyword restores that data. This means any custom text styling you've done once *never has to be done again* for that speaker's text.

In fact, let's add some style to our character's text now by *modifying* the original string:

**Example:**

```
vnngen_event_set_target();

if (vnngen_event()) {

    vnngen_text_create("text", "John Doe", "Hello, world!", 128, 900, 0, 256, fnt_Aria
1, c_white, trans_fade, 2, true);

    vnngen_text_modify_style("text", c_white, c_white, c_gray, c_gray, c_black, c_blac
k, 1, 2);

}

if (vnngen_event()) {

    vnngen_text_replace("text", previous, "How are you?", inherit, inherit, 2);

}

vnngen_event_reset_target();
```

Our text will now be displayed with a four-color gradient fading vertically from white to gray, along with a black shadow and outline effect. The two-second style change duration matches up with our initial fade transition to ensure text appears seamlessly. From now on, so long as we use the 'inherit' keyword for the 'color' property on any lines where "John Doe" is the speaker, this gradient will be restored without having to run another `vnngen_text_modify_style` operation again!

Now, let's take this concept a step further to bring in a second character to interact with:

**Example:**

```
vnngen_event_set_target();

if (vnngen_event()) {

    vnngen_text_create("text", "John Doe", "Hello, world!", 128, 900, 0, 256, fnt_Aria
1, c_white, trans_fade, 2, true);

    vnngen_text_modify_style("text", c_white, c_white, c_gray, c_gray, c_black, c_blac
k, 1, 2);

}

if (vnngen_event()) {

    vnngen_text_replace("text", previous, "How are you?", inherit, inherit, 2);

}

if (vnngen_event()) {

    vnngen_text_replace("text", "Jane Doe", "I'm good, thanks! How about you?", previo
```

```
us, previous, 2);  
  
}  
  
vnngen_event_reset_target();
```

As you can see, in a single new line, we've created a new string of text, assigned it to a new speaker, and passed our first character's text stylization into the second, effectively combining three actions into one! Just like the 'previous' keyword can be used to avoid changing speaker names, here it avoids setting a new font or text color, instead using the gradient set in the first event. Now "John Doe" and "Jane Doe" each have their own copy of the same style, which can be modified later without affecting the other character.

In this way, effective use of available actions makes displaying text both easy and powerful!

### 5.3.3. A Simple Choice

But what if we want to give our players a voice of their own?

VNgen includes actions for creating basic menus, aptly named **options**. On their own, options only provide an *interface* for the user to select from a list of items, but by combining options with regular functions such as `vnngen_goto` and a little custom logic, it is possible to use options to jump to different portions of Q-script and create branching narratives across one or more objects.

Put it all together, and a dialog choice might look something like this:

**Example:**

```
if (vnngen_event()) {  
  
    if (vnngen_option("options", 960, 540, 1, 1, snd_hover, snd_select)) {  
  
        vnngen_option_create("option_good", "I'm doing great!", spr_option, spr_option_  
hover, spr_option_select, 0, 0, -1, fnt_Arial, c_white, trans_slide_right, 1);  
  
        vnngen_option_create("option_bad", "Well...", spr_option, spr_option_hover, spr_  
_option_select, 0, 100, -2, fnt_Arial, c_white, trans_slide_right, 1);  
  
    }  
  
    switch (vnngen_get_option()) {  
  
        case "option_good": vnngen_goto("event_good"); break;  
  
        case "option_bad": vnngen_goto("event_bad"); break;  
  
    }  
  
}  
  
if (vnngen_event("event_good")) {  
  
    vnngen_text_replace("text", "John Doe", "I'm doing great!", inherit, inherit, 2);
```

```
}

if (vnngen_event()) {

    vnngen_script_execute(vnngen_goto, "event_complete");

}

if (vnngen_event("event_bad")) {

    vnngen_text_replace("text", "John Doe", "Not so great, I'm afraid...", inherit, inherit, 2);

}

if (vnngen_event("event_complete")) {

    vnngen_text_replace("text", "Jane Doe", "I see.", inherit, inherit, 2);

}
```

Now, such a setup might seem complex at first, but in actuality it relies on all the same processes we've already learned.

Creating a complete dialog option sequence consists of three main parts:

1. Obtaining user input (the **choice**)
2. Reacting to user input (the **condition**)
3. Providing user feedback (the **result**)

Let's break down each step in detail.

## 1. The Choice

The `vnngen_option` script is an action which behaves like an event. It does **not** replace `vnngen_event` and must be nested inside an event just like any other action. At the same time, however, it possesses its own sub-actions which cannot be run outside the context of `vnngen_option`. Think of a group of options as a single unit?because that's exactly how VNgen sees it!

### Example:

```
if (vnngen_event()) {

    if (vnngen_option("options", 960, 540, 1, 1, snd_hover, snd_select)) {

        vnngen_option_create("option_good", "I'm doing great!", spr_option, spr_option_hover, spr_option_select, 0, 0, -1, fnt_Arial, c_white, trans_slide_right, 1);

        vnngen_option_create("option_bad", "Well...", spr_option, spr_option_hover, spr_option_select, 0, 100, -2, fnt_Arial, c_white, trans_slide_right, 1);

    }

}
```

```
}
```

Here we have an options menu with two options to choose from. To simply display a menu, this is all the code we need: one instance of `vngen_option` as the parent, and one or more instances of `vngen_option_create` as the child. Although each has its own unique properties, some are also shared. The first two parameters in `vngen_option` specify the X and Y position to display the menu on the screen, with the X and Y values in `vngen_option_create` being relative to the parent's coordinates. The next two values of `vngen_option` set delays before and after options are created? a pair of values which corresponds to the two transitions set in `vngen_option_create`. And finally, the two sound effects supplied are only played when an individual option is hovered or selected. The two scripts really are inseparably linked!

For full details on these scripts' syntax, check out the full reference guide, but for now there's one other property worth taking note of: the **option ID**. Not only does this value distinguish different options from one another, but is also the value that will be stored in memory once a selection has been made.

## 2. The Condition

Once the user has input their choice, the selected option's ID will be stored in memory for future use. No other action is taken automatically? which is by design? meaning it's up to us to make use of that information however we see fit. The possibilities are practically endless, but the most common way to utilize selected option data is through conditional statements such as 'if' or 'switch'.

### Example:

```
switch (vngen_get_option()) {  
  
    case "option_good": vngen_goto("event_good"); break;  
  
    case "option_bad": vngen_goto("event_bad"); break;  
  
}
```

For those not familiar, a 'switch' statement is like a series of 'if' statements strung together in a single group. One **condition** is tested against multiple **cases**, and whichever case is found to be true is executed until the 'break' statement is reached.

In this scenario, we use the `vngen_get_option` script to retrieve the ID of the previously-selected option and jump to different Quantum events depending on which option was selected (note how the cases here match the IDs of the options created in the first step). It's important to note that this value will remain in memory until another options menu is created, which may or may not be desirable depending on how our code is set up. For example, if the selected option remains in memory forever, whatever code we include in our 'switch' statement may get executed endlessly as well! For this reason, `vngen_get_option` will also **clear** the selected option from memory after it has been retrieved, unless this has been disabled by setting the 'clear' argument to **false**. This guarantees that whatever the result, it will only be triggered once.

**Note:** `vngen_get_option` is a very special script. While it is not an action, it does share some behaviors so that it can be used like one. Unlike other VNgen functions, `vngen_get_option` can be run anywhere in your game? both inside and outside of events!

For our 'switch' statement, there are only two possibilities: either the user selected "option\_good" or "option\_bad". These options will run `vngen_goto` and jump to the event labeled "event\_good" or "event\_bad",



respectively.

### 3. The Result

Of course, using options to trigger `vnngen_goto` means we must have somewhere for the user to go to! As previously mentioned, we can place our “event\_good” and “event\_bad” labels somewhere in the current object or an entirely different one, but both possibilities can often run into a common problem: how do we keep one option result from running straight into another?

Consider the dialog branches shown in the original example:

**Example:**

```
if (vnngen_event("event_good")) {  
  
    vnngen_text_replace("text", "John Doe", "I'm doing great!", inherit, inherit, 2);  
  
}  
  
if (vnngen_event("event_bad")) {  
  
    vnngen_text_replace("text", "John Doe", "Not so great, I'm afraid...", inherit, inherit, 2);  
  
}  
  
if (vnngen_event()) {  
  
    vnngen_text_replace("text", "Jane Doe", "I see.", inherit, inherit, 2);  
  
}
```

If the user selects “option\_good”, they will respond that they’re feeling well, whereas if they select “option\_bad”, they will respond that they’re not feeling so well. These are our two dialog branches, and the intention is for them to converge back onto a single path when the character “Jane Doe” acknowledges their response. Now, this will work just fine if we jump to “event\_bad” since the branches converge in the next event, but what about “event\_good”?

In this case, we must put a second event between “event\_good” and “event\_bad” which causes “event\_bad” to be skipped if “option\_good” is chosen. However, we can’t simply run `vnngen_goto` in an event by itself since *it is not an action*. Instead, we must wrap it in `vnngen_script_execute`, which takes non-action scripts and executes them in the context of Quantum events. We can also make things easier by assigning a label to the event where our options converge, so we know exactly where to tell `vnngen_goto` to go.

**Example:**

```
if (vnngen_event("event_good")) {  
  
    vnngen_text_replace("text", "John Doe", "I'm doing great!", inherit, inherit, 2);  
  
}  
  
if (vnngen_event()) {
```

```
vngen_script_execute(vngen_goto, "event_complete");  
  
}  
  
if (vngen_event("event_bad")) {  
  
    vngen_text_replace("text", "John Doe", "Not so great, I'm afraid...", inherit, inherit, 2);  
  
}  
  
if (vngen_event("event_complete")) {  
  
    vngen_text_replace("text", "Jane Doe", "I see.", inherit, inherit, 2);  
  
}
```

This will cause “event\_bad” to be skipped as soon as the player has completed “event\_good” and end the options sequence at “event\_complete”.

And of course, this is only scratching the surface. By following these same techniques it is possible to create deep dialog sequences and wildly branching narratives, plus entire game menu interfaces and more!

### 5.3.4. A Simple Scene

At this point we’ve set up VNgen and created a simple dialog sequence complete with interactive responses for the user to choose from. However, if we were to run our game right now, it would look a little bland, having just text. Who are the characters we’ve created, and what do they look like? Where are they? Visual novels being visual, details like these are most often left to context, so it’s important that once the script is in place, other elements are soon to follow.

Fortunately, as previously mentioned, VNgen actions are highly standardized. Now that you have a grasp on creating text, creating other kinds of entities should come naturally. Once you know what entity types are available and what functions they all have, exploring new possibilities is a simple matter of combining your knowledge of the two.

Let’s begin with a scene. VNgen makes it possible to create highly dynamic environments with multiple parallaxing layers, but for now we’ll start with a simple backdrop.

#### **Example:**

```
vngen_event_set_target();  
  
if (vngen_event()) {  
  
    vngen_scene_create("bg", spr_background, 0, 0, 0, false, false, trans_fade, 2);  
  
    vngen_text_create("text", "John Doe", "Hello, world!", 128, 900, 0, 256, fnt_Aria  
1, c_white, trans_fade, 2, true);  
  
    vngen_text_modify_style("text", c_white, c_white, c_gray, c_gray, c_black, c_blac
```



```
k, 1, 2);  
  
}  
  
...
```

Going back to our original example, here we've added a background, or **scene**, to our first event so that our text and environment appear synchronously. Scenes can be created as either backgrounds or foregrounds and also set to endlessly repeat, but aside from these unique properties you'll notice that, for the most part, creating a scene is quite like creating text. We even have the same selection of transitions to choose from, and can set the transition duration to match that of our text and text style modifications. The only difference is that we assign a **sprite** to an ID rather than strings.

Creating characters is a similar matter, only we'll also need an extra sprite or two just for facial animations when idle and talking, as seen in the example below.

**Example:**

```
vnngen_event_set_target();  
  
if (vnngen_event()) {  
  
    vnngen_scene_create("bg", spr_background, 0, 0, 0, false, false, trans_fade, 2);  
  
    vnngen_char_create("John Doe", spr_body, spr_face_idle, spr_face_talk, 0, 1080, 0,  
180, 200, false, trans_slide_right, 2);  
  
    vnngen_text_create("text", "John Doe", "Hello, world!", 128, 900, 0, 256, fnt_Aria  
1, c_white, trans_fade, 2, true);  
  
    vnngen_text_modify_style("text", c_white, c_white, c_gray, c_gray, c_black, c_blac  
k, 1, 2);  
  
}  
  
...
```

Note how the character created in this example has the same ID as the speaker name assigned in the text action. In order for automatic animations to function, both names must match exactly. If they don't, both the text and character will still display properly, but facial animations will not be played in-sync with text.

Typically, faces should be rendered separate from the body into their own sprites, which are then composited onto the body in real-time in VNgen. This avoids unnecessarily drawing the body over and over while only the face is changing, constituting wasted performance and storage space. Face sprites need not even match body sprite dimensions, since `vnngen_char_create` includes a second set of X/Y coordinates just for where to position the face relative to the body sprite.

And that's not all. We can save even more performance when creating our other character as well:

---

**Example:**

```
...

if (vnngen_event()) {

    vnngen_char_create("Jane Doe", spr_body, spr_face_idle, spr_face_talk, 0, 1080, 0,
180, 200, true, trans_slide_left, 2);

    vnngen_text_replace("text", "Jane Doe", "I'm good, thanks! How about you?", previo
us, previous, 2);

}

...
```

Notice anything different? Aside from the name, another subtle change here makes a big difference in how our second character is displayed. Note the 'true' value here?as explained in the full reference guide, this value enables **flipping** the character horizontally so that only one version of a character needs to be created, facing either direction, since VNgen can mirror the character in real-time later.

At this point our scene is beginning to look a bit more alive. Perhaps even a bit *too* much. Busy backdrops can often lack contrast with text, making it difficult or unpleasant to read. Fortunately, the solution is simple:

**Example:**

```
vnngen_event_set_target();

if (vnngen_event()) {

    vnngen_scene_create("bg", spr_background, 0, 0, 0, false, false, trans_fade, 2);

    vnngen_char_create("John Doe", spr_body, spr_face_idle, spr_face_talk, 0, 1080, 0,
180, 200, false, trans_slide_right, 2);

    vnngen_textbox_create("textbox", spr_textbox, 0, 1080, 0, trans_wipe_right, 2);

    vnngen_text_create("text", "John Doe", "Hello, world!", 128, 900, 0, 256, fnt_Aria
1, c_white, trans_fade, 2, true);

    vnngen_text_modify_style("text", c_white, c_white, c_gray, c_gray, c_black, c_blac
k, 1, 2);

}

...
```

That's right: a textbox. As textboxes are purely decorative, they are one of the simplest and most standard of all VNgen entities. Text is not physically confined to them in any way, but they still serve an important purpose in

making text readable, and even play a role in setting the mood for your particular story and setting. On one hand, it's a finishing touch, and on the other, it's just the beginning of a much bigger design language encompassing the entire UI.

And indeed, from here, the possibilities are endless. From further UI elements like **prompts** to advanced composite character **attachments**, with an understanding of the fundamental principles of VNgen content creation the power is yours to build engaging, interactive narratives full of animated characters and environments. Take the knowledge you've gained from creating your first visual novel and you'll be a master storyteller in no time!

### 5.3.5. A Simple Script

Below you'll find the final script from this tutorial. However, don't just take it at face value: customize it and make it your own! To learn more, refer to the full reference guide for details on each and every event, action, and function of VNgen.

Happy scripting!

**Example:**

```
vnngen_event_set_target();

if (vnngen_event()) {

    vnngen_scene_create("bg", spr_background, 0, 0, 0, false, false, trans_fade, 2);

    vnngen_char_create("John Doe", spr_body, spr_face_idle, spr_face_talk, 0, 1080, 0, 180, 200, false, trans_slide_right, 2);

    vnngen_textbox_create("textbox", spr_textbox, 0, 1080, 0, trans_wipe_right, 2);

    vnngen_text_create("text", "John Doe", "Hello, world!", 128, 900, 0, 256, fnt_Arial, c_white, trans_fade, 2, true);

    vnngen_text_modify_style("text", c_white, c_white, c_gray, c_gray, c_black, c_black, 1, 2);

}

if (vnngen_event()) {

    vnngen_text_replace("text", "John Doe", "How are you?", fnt_Arial, c_white, 2);

}

if (vnngen_event()) {

    vnngen_char_create("Jane Doe", spr_body, spr_face_idle, spr_face_talk, 0, 1080, 0, 180, 200, true, trans_slide_left, 2);

    vnngen_text_replace("text", "Jane Doe", "I'm good, thanks! How about you?", previous, 2);

}
```

```
if (vnngen_event()) {

    if (vnngen_option("options", 960, 540, 1, 1, snd_hover, snd_select)) {

        vnngen_option_create("option_good", "I'm doing great!", spr_option, spr_option_
        hover, spr_option_select, 0, 0, -1, fnt_Arial, c_white, trans_slide_right, 1);

        vnngen_option_create("option_bad", "Well...", spr_option, spr_option_hover, spr
        _option_select, 0, 100, -2, fnt_Arial, c_white, trans_slide_right, 1);

    }

    switch (vnngen_get_option()) {

        case "option_good": vnngen_goto("event_good"); break;

        case "option_bad": vnngen_goto("event_bad"); break;

    }

}

if (vnngen_event("event_good")) {

    vnngen_text_replace("text", "John Doe", "I'm doing great!", inherit, inherit, 2);

}

if (vnngen_event()) {

    vnngen_script_execute(vnngen_goto, "event_complete");

}

if (vnngen_event("event_bad")) {

    vnngen_text_replace("text", "John Doe", "Not so great, I'm afraid...", inherit, in
    herit, 2);

}

if (vnngen_event("event_complete")) {

    vnngen_text_replace("text", "Jane Doe", "I see.", inherit, inherit, 2);

}

vnngen_event_reset_target();
```

## 6. VNgen Reference Guide

VNgen uses an event/action system built on the [Quantum framework](#) by XGASOFT to provide both a great

developer and user experience. In this section, each action is examined in detail along with every other VNgen function to provide a comprehensive reference for VNgen syntax and functionality.

## 6.1. The Debug Console

VNgen features a built-in debug mode which can be used to view many different types of statistics and easily test different parts of your projects in real-time.

Debug mode is disabled by default, but can be enabled by running the `show_debug_vngen` command. When enabled, `DEBUG MODE` will appear in the top-left corner of the display as an indicator. At this point, it is now possible to operate the command console.



The built-in **command console** is the heart of VNgen's debug functions, and can be opened and closed with the tilde (~) key whenever debug mode is active. As its name implies, the command console allows the user to manually input and execute commands in real-time without setting up custom code and recompiling (as would normally be the case).

For a list of all supported commands, type 'help' in the console and press Enter. This will open a command reference in the default web browser, which you can keep open while continuing to work.

**Tip 1:** On systems without a compatible browser, entering 'localhelp' will display the same reference directly within VNgen.

**Tip 2:** Tired of entering the same commands over and over? The command console keeps a history of up to 32 recently-entered commands which can be accessed by simply pressing the up and down arrow keys on your keyboard!

A recommended command to start with is:

**Example:**

```
show_debug_stats(true)
```

This will enable drawing a host of stats specific to VNgen, and is the first place you should check to get an idea of

what's happening under-the-hood as your script progresses.



In the column on the left are global stats covering the backlog and the current script object's events and actions. In the right column is a list of active entities, showing how many of each type currently exist.

While this information alone can be incredibly helpful, you might notice a few other visual changes to the scene as well. These colored wireframe indicators are called **helpers**, and are displayed on the perspective, entities with active deformations, text links, and more. They are enabled by default, but can be disabled with a simple console command:

**Example:**

```
show_debug_helpers(false)
```

Not only does disabling helpers unclutter the scene when you want to get a good look at it, but saves a bit of performance as well. In fact, both debug stats and debug helpers will dramatically impact performance when enabled. Normally this isn't a problem, as these tools exist for developer use only, but it's important to note that these features should be disabled before doing any performance testing, which, of course, is also possible through the command console:

**Example:**

```
show_debug_overlay(true)
```

This will enable live performance reports across the top of the screen, providing frames per-second and visual indicators for individual time spent on a number of different tasks.



VNgen is designed from the ground up for optimal performance, so ideally you shouldn't see each and every indicator visually represented on the debug overlay. If the colored bar on the left only occupies a small portion of



the screen, you are experiencing excellent performance. Regardless, it is important to test your projects on as many hardware configurations as possible. On low-end machines, knowing what each color in the performance bar represents is critical:

Color	Description
Green	I/O (keyboard, mouse, or gamepad input, network communications, etc.)
Red	Frame update time
Yellow	Draw Event processing time
Orange	Debug overlay update time
White	Time spent waiting for the GPU to finish rendering the current frame
Cyan	Text rendering time
Gray	Screen clear time
Dark Red	GPU clear time

If any of these colors appears to occupy too large a portion on the debug overlay, it may be time to seek ways to reduce overhead in the particular category it refers to. It's also important to note the orange portion of the performance bar and disregard factoring it into performance as a whole, since the debug overlay itself consumes a certain amount of resources which will be available in the finished product.

### 6.1.1. Intro to QCMD

**QCMD** is an abbreviation for **Quantum Command**, the modular command console which comes hand-in-hand with **Quantum**, the core framework products such as [VNgen](#) are built on. [Quantum](#) is a framework by XGASOFT designed to assign numeric IDs to blocks of code itself and provide a set of common practical functions to manipulate them, allowing for the creation of entirely custom programming languages within the **GameMaker Language** (or **GML**).

## Structure

QCMD is comprised of two parts: the **console** itself and one or more **commands** which are written and loaded into the console externally. Each command is a miniature program of its own, defining its own syntax and instructions. When complete, a command may also return a custom string to the console as feedback for the user.

As a development tool, QCMD is cleverly designed to remain outside of memory until called. This means keeping it in commercial projects won't occupy unnecessary resources on consumer devices, but it will still be there to debug compiled builds when needed.

## Creating Custom Commands

Using the built-in `sys_cmd_add` function, any script can be turned into a QCMD command. Simply specify a command as a string to listen for and the script resource to execute when that string is input to the QCMD console.

### Example:

```
sys_cmd_add("q_goto", q_goto);
```



It is not necessary to define arguments when adding a new command to QCMD, as arguments are processed in real-time.

**Note:** QCMD stores commands in a database which occupies a small amount of memory once initialized.

Although this function is all that is necessary to add new commands to QCMD, when designing functions specifically to operate as commands there are a few caveats and best practices to keep in mind:

1. All arguments are passed into scripts as strings (because user input is obtained as a string). Other types of data must be converted, e.g. `real(argument0)` or `asset_get_index(argument0)` to convert string input to a real number or asset index, respectively.
2. Commands are only executed once and will not continue being processed over time. They can, however, be used to *trigger* longer processes.
3. All erroneous forms of input should return a string alerting the user of what went wrong. This is done simply by using the `return` command to return a string to the console, e.g. `if (argument_count == 0) { return "Error: this command requires X arguments."; }`
4. If a command is processed successfully, say so! Alert the user of what the command has done by using the `return` command, e.g. `if (argument0 == true) { window_set_fullscreen(true); return "Successfully set window mode to fullscreen."; }`

While Quantum-based products include a host of built-in commands, effective use of custom commands in QCMD can radically improve workflow when testing and debugging projects in real-time.

## 6.1.2. Included QCMD Commands

VNgen includes a number of console commands out of the box which can be used to test and debug your projects in real-time. QCMD commands are written externally and defined in a command database as string/script pairs.

Commands can be performed in QCMD by inputting the command **string**, which will in turn execute the command **script** according to the table below.

### Console Commands

Command	Script	Description
!!	<code>cmd_firework()</code>	Celebrates the little victories with you
exit	<code>cmd_exit()</code>	Closes the QCMD console
game_end	<code>cmd_game_end()</code>	Closes the game itself
game_restart	<code>cmd_game_restart()</code>	Restarts the game itself
help	<code>cmd_help()</code>	Displays a sortable list of console commands in the default web browser
localhelp	<code>cmd_localhelp()</code>	Displays a list of console commands in a dialog window
room_goto	<code>cmd_vngen_room_goto(room)</code>	Same as <code>vngen_room_goto</code> , but in standard GameMaker syntax
show_debug_helpers	<code>cmd_show_debug_helpers([enable])</code>	Enables or disables displaying perspective statistics and wireframes around active deforms, text links, and more
show_debug_overlay	<code>cmd_show_debug_overlay(enable)</code>	Enables or disables displaying live performance statistics



show_debug_stats	cmd_show_debug_stats([enable	Enables or disables displaying VNgen-specific statistics
vnngen_file_save	cmd_vnngen_file_save(filename	Saves the current VNgen state to a file on the hard drive with optional encryption (enabled by default)
vnngen_file_load	cmd_vnngen_file_load(filename	Restores the VNgen state from a previous save file
vnngen_file_delete	cmd_vnngen_file_delete(filename)	Permanently erases a previous VNgen save file
vnngen_goto	cmd_vnngen_goto(event,	Jumps to the specified VNgen event, optionally switching to the target object, if specified.
vnngen_goto_unread	cmd_vnngen_goto_unread([perform])	Jumps to the next option block or unread event
vnngen_log_clear	cmd_vnngen_log_clear([destroy])	Clears all entries from the backlog, optionally destroying the log object (disabled by default)
vnngen_room_goto	cmd_vnngen_room_goto(room,	Jumps to the specified room, clearing VNgen data from memory
vnngen_set_lang	cmd_vnngen_set_lang(lang,	Sets the text or audio language to the specified language flag
vnngen_get_lang	cmd_vnngen_get_lang(type)	Displays the current text or audio language flag
vnngen_set_renderlevel	cmd_vnngen_set_renderlevel(level)	Sets renderlevel, where 0 is default
vnngen_set_scale	cmd_vnngen_set_scale(view)	Sets a viewport to scale to window dimensions
vnngen_set_speed	cmd_vnngen_set_speed(speed)	Sets the text typewriter effect speed
vnngen_set_vol	cmd_vnngen_set_vol(type, vol,	Sets the global volume offset for a particular type of sound ( <i>use keyword 'all' for all types</i> )
vnngen_version	cmd_vnngen_version	Displays the current VNgen version and build date
window_set_fullscreen	cmd_window_set_fullscreen(full)	Enables or disables fullscreen mode

### 6.1.3. The "vnngen\_do\_debug" Function

#### Syntax:

```
vnngen_do_debug(toggle, [sound]);
```

Argument	Type	Description
toggle	boolean/macro	Enables, disables, or toggles debug mode
[sound]	sound	<i>Optional:</i> A sound to be played when the script is run

#### Description:

Toggles or explicitly enables/disables VNgen's built-in debug mode. Debug mode is disabled by default. When enabled, "DEBUG MODE" will be printed across the top-left corner of the display. While this notice is visible, pressing the tilde (~) key will open and close VNgen's command console. Typing 'help' in the command console

will display a list of available debug commands.

**Example:**

```
vngen_do_debug(toggle);  
vngen_do_debug(true, snd_input);
```

## 6.1.4. The "vngen\_is\_debug" Function

**Syntax:**

```
vngen_is_debug();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Checks whether VNgen's debug mode is enabled and returns true or false.

**Example:**

```
if (vngen_is_debug()) {  
    keyboard_virtual_show(kbv_type_default, kbv_returnkey_default, kbv_autocapitaliz  
e_none, false);  
}
```

## 6.2. Macros & Keywords

**Important note:** GameMaker Studio 1.x users must import the included **macros.txt** file located in the **Included Files** folder before using VNgen. This can be done from the **Resources > Define Macros** menu and selecting **Load**.

For the sake of memorability and forwards-compatibility, many VNgen constants use keyword counterparts (or macros) interchangeably with their numeric values. This section covers both, listing available keywords and their numeric counterparts for you to better understand their relation to each other and to the functions in which they are used. However, it is strongly recommended to always use keywords when available, as their numeric values may change in future updates.

It is important to note that not every macro is supported by every script, and as such using macros in undocumented ways may not have the intended effect, or no effect at all. To learn which macros are supported by which arguments and functions, refer to the full reference guide. However, most macros should be self-explanatory.

**Tip:** Many keywords are prefixed, meaning you don't have to memorize every keyword to use them. Simply begin typing the first few letters of a keyword in your code editor and you'll be shown auto-complete options to choose the desired keyword from a list.

## Miscellaneous



Keyword/Macro	Value	Description
auto	-1	Defers to the engine to determine a value automatically
previous	-1	Does not change the current value from its previous definition
inherit	-2	Retrieves an external value stored in memory
toggle	-2	Alternates between enabled and disabled based on the previous value
any	-3	A general selector to apply a modification to all qualifying entities
none	-4	Disables a value from being used

## Animations

Keyword/Macro	Value	Description
anim_zoom	anim_xscale	Used in animations designed for the perspective. Same as anim_xscale to other entities.
anim_strength	anim_yscale	Used in animations designed for the perspective. Same as anim_yscale to other entities.
input_zoom	input_xscale	Used in animations designed for the perspective. Same as input_xscale to other entities.
input_strength	input_yscale	Used in animations designed for the perspective. Same as input_yscale to other entities.

## Easing/Tweening

**Tip:** To get a better feel for how different ease modes work, check out <https://easings.net/> and <https://cubic-bezier.com/>

Keyword/Macro	Value	Description
ease_none	-4	Sets the animation ease mode to linear math, or no easing
ease_sin_in_out	1	Sets the animation ease mode to sine math at the start and end
ease_sin_in	2	Sets the animation ease mode to sine math at the start only
ease_sin_out	3	Sets the animation ease mode to sine math at the end only
ease_quad_in_out	4	Sets the animation ease mode to quadratic math at the start and end
ease_quad_in	5	Sets the animation ease mode to quadratic math at the start only
ease_quad_out	6	Sets the animation ease mode to quadratic math at the end only
ease_cubic_in_out	7	Sets the animation ease mode to



<code>ease_cubic_in</code>	8	cubic math at the start and end Sets the animation ease mode to cubic math at the start only
<code>ease_cubic_out</code>	9	Sets the animation ease mode to cubic math at the end only
<code>ease_quart_in_out</code>	10	Sets the animation ease mode to quartic math at the start and end
<code>ease_quart_in</code>	11	Sets the animation ease mode to quartic math at the start only
<code>ease_quart_out</code>	12	Sets the animation ease mode to quartic math at the end only
<code>ease_quint_in_out</code>	13	Sets the animation ease mode to quintic math at the start and end
<code>ease_quint_in</code>	14	Sets the animation ease mode to quintic math at the start only
<code>ease_quint_out</code>	15	Sets the animation ease mode to quintic math at the end only
<code>ease_expo_in_out</code>	16	Sets the animation ease mode to exponential math at the start and end
<code>ease_expo_in</code>	17	Sets the animation ease mode to exponential math at the start only
<code>ease_expo_out</code>	18	Sets the animation ease mode to exponential math at the end only
<code>ease_circ_in_out</code>	19	Sets the animation ease mode to circular math at the start and end
<code>ease_circ_in</code>	20	Sets the animation ease mode to circular math at the start only
<code>ease_circ_out</code>	21	Sets the animation ease mode to circular math at the end only
<code>ease_rubber_in_out</code>	22	Sets the animation ease mode to exceed values and smoothly fall back at the start and end
<code>ease_rubber_in</code>	23	Sets the animation ease mode to exceed values and smoothly fall back at the start only
<code>ease_rubber_out</code>	24	Sets the animation ease mode to exceed values and smoothly fall back at the end only
<code>ease_elastic_in_out</code>	25	Sets the animation ease mode to exceed values and roughly fall back at the start and end
<code>ease_elastic_in</code>	26	Sets the animation ease mode to exceed values and roughly fall back at the start only
<code>ease_elastic_out</code>	27	Sets the animation ease mode to exceed values and roughly fall back at the end only
<code>ease_bounce_in_out</code>	28	Sets the animation ease mode to bounce between values at the start and end
<code>ease_bounce_in</code>	29	Sets the animation ease mode to bounce between values at the start only
<code>ease_bounce_out</code>	30	Sets the animation ease mode to bounce between values at the end



ease_bezier	31	only Sets the animation ease mode to a custom curve based on two pairs of X, Y coordinates
-------------	----	---

## Origin

Keyword/Macro	Value	Description
orig_top	0	Triggers vertically aligning the current entity by the top edge
orig_left	0	Triggers horizontally aligning the current entity by the left edge
orig_center	-1	Triggers horizontally or vertically aligning the current entity by the center
orig_bottom	-2	Triggers vertically aligning the current entity by the bottom edge
orig_right	-2	Triggers horizontally aligning the current entity by the right edge

## Scaling

Keyword/Macro	Value	Description
scale_none	-4	Disables automatic scaling
scale_x_y	1	Triggers automatic scaling to fill the screen in both directions, maintaining aspect ratio
scale_x	2	Triggers automatic scaling to fill the screen horizontally, maintaining aspect ratio
scale_y	3	Triggers automatic scaling to fill the screen vertically, maintaining aspect ratio
scale_stretch_x_y	4	Triggers automatic scaling to fill the screen in both directions, disregarding aspect ratio
scale_stretch_x	5	Triggers automatic scaling to fill the screen horizontally, disregarding aspect ratio
scale_stretch_y	6	Triggers automatic scaling to fill the screen vertically, disregarding aspect ratio
scale_prop_x_y	7	Triggers automatic scaling relative to changes in screen resolution in both directions
scale_prop_x	8	Triggers automatic scaling relative to changes in horizontal screen resolution only
scale_prop_y	9	Triggers automatic scaling relative to changes in vertical screen resolution only



## Transitions

Keyword/Macro	Value	Description
trans_none	-4	Disables scripted transition animations

## Entity Types

Keyword/Macro	Value	Description
vngen_type_perspective	0	Points to the global perspective camera
vngen_type_scene	1	Points to scene entities
vngen_type_char	2	Points to character entities
vngen_type_attach	3	Points to character attachment entities
vngen_type_emote	4	Points to emote entities
vngen_type_textbox	5	Points to textbox entities
vngen_type_text	6	Points to text entities
vngen_type_label	7	Points to label entities
vngen_type_prompt	8	Points to prompt entities
vngen_type_option	9	Points to option entities
vngen_type_audio	10	Points to regular audio entities
vngen_type_vox	11	Points to speech synthesis audio entities
vngen_type_effect	12	Points to scripted effect entities
vngen_type_button	13	Points to button entities
vngen_type_speaker	14	Points to the current active speaker(s) ( <b>Exception:</b> not an entity)

## Audio Types

Keyword/Macro	Value	Description
audio_type_sound	0	Points to sound effects or looped sound effects
audio_type_voice	1	Points to spoken dialog
audio_type_music	2	Points to music
audio_type_vox	3	Points to vox (speech 'blips')
audio_type_ui	4	Points to UI sound effects

## 6.3. Animations

VNgen features a unified scripted animation system which can perform custom keyframe animation sequences on almost every entity in the engine using `vngen_*_anim_start` and `vngen_*_deform_start` functions, as well as `vngen_*_create` and `vngen_*_destroy` functions for transitions.

Animations in VNgen come in three categories: **transitions**, **transforms**, and **deforms**. Each modifies different properties, but the scripting syntax is the same. It is also worth noting that while two animations of the same type



cannot be performed simultaneously, a single transition, transform, and deform animation can be performed on any entity at the same time.

In this section we will examine included animations for all three types as well as how to write your own.

### 6.3.1. Creating Custom Animations

At its core, VNgen is a sequenced animation system with visual novel functions layered on top. Events and actions form a timeline of visual changes, often enhanced by built-in transitions and animation effects. But what if you've dreamed up an animation completely unlike the built-in ones? What about animating entities after their transitions have already completed? What about performing entire animations in a single event?

With VNgen's **keyframe animation scripts**, all of these things are possible.

## Intro to Keyframes

In animation (both digital and otherwise), a **keyframe** represents a significant change in motion. Rather than produce an animation one subtle frame at a time, creators focus only on the start and end of each motion, resulting in a sequence that roughly portrays how the final animation will play out. Once all the keyframes are in place, it's much easier to fill in the gaps with frames in between, smoothly transitioning from one keyframe to the next.

In VNgen, these "in-betweens" are generated automatically, meaning all you have to worry about are the keyframes themselves. Set any properties you want to change in a keyframe and they'll smoothly interpolate to their new values.

## Animatable Properties

Keyframe animations come in three categories: **transitions**, **transforms**, and **deforms**. The syntax for writing all three is the same; fundamentally, the only difference is which properties they animate.

As animations represent *temporary* changes to the entity being animated, VNgen does not modify animated properties directly. Instead, it passes certain values into animation scripts, checks keyframes for changes, and reads the results back into the animated entity in real-time. Not only does this preserve the entity's original values, but provides a common set of input and output values just for writing keyframe animations.

**Note:** Another consequence of animations being temporary is that any animatable properties *not* specified in a keyframe will be interpreted as their default values. For this reason, even if a modified property isn't meant to change for several keyframes, it must be declared in every keyframe for the modification to persist.

First, the *output*, or *modifiable* values:

### Transitionable Properties

General properties	Type	Description
trans_left	real	Relative left crop
trans_top	real	Relative top crop
trans_width	real	Relative width crop
trans_height	real	Relative height crop
trans_x	real	Relative horizontal position
trans_y	real	Relative vertical position
trans_xscale	real	Relative horizontal scale multiplier
trans_yscale	real	Relative vertical scale multiplier



trans_rot	real	Relative rotation, in degrees
trans_alpha	real (0-1)	Relative alpha value
trans_ease	integer/macro	Ease mode for the current keyframe

## Transformable Properties

General properties	Type	Description
anim_x	real	Relative horizontal position
anim_y	real	Relative vertical position
anim_xscale	real	Relative horizontal scale multiplier
anim_yscale	real	Relative vertical scale multiplier
anim_rot	real	Relative rotation, in degrees
anim_col1	color	Top-left gradient color
anim_col2	color	Top-right gradient color
anim_col3	color	Bottom-right gradient color
anim_col4	color	Bottom-left gradient color
anim_alpha	real (0-1)	Relative alpha value
anim_ease	integer/macro	Ease mode for the current keyframe

### Perspective Only

anim_xoffset	real	Relative perspective horizontal 'angle'
anim_yoffset	real	Relative perspective vertical 'angle'
anim_zoom	real	Relative perspective zoom multiplier
anim_strength	real	Relative perspective parallax strength multiplier

**Note:** Some perspective animation properties are mapped to general properties and can be used interchangeably: `anim_zoom = anim_xscale` and `anim_strength = anim_yscale`. Only `anim_xoffset` and `anim_yoffset` are unique to perspective.

## Deformable Properties

Properties	Type	Description
def_width	integer	The number of columns in the deform mesh
def_height	integer	The number of rows in the deform mesh
def_xpoint[0, 0] ... def_xpoint[1, 3]	real	Mesh point horizontal offsets
def_ypoint[0, 0] ... def_ypoint[1, 3]	real	Mesh point vertical offsets
def_ease	real	Ease mode for the current keyframe

**Note:** Deformations cannot be performed on the perspective, therefore there are no special deformable properties.

As you've probably noticed, most animatable properties are not only temporary, but *relative* to the entity's current properties. This means that setting `anim_x` to 50, for example, would *offset* the animated entity by 50 pixels to the right of its *current position*, rather than to the 50th horizontal pixel in the room. This is especially important for deformations, which would otherwise constrain all animated entities to the same shape!

Even so, there are still times when it is useful to know some of the animated entity's original properties. This way, animations can be made that adapt to a variety of shapes and sizes without modification. As such, VNgen also

passes certain *input* properties to animation scripts.

General Properties	Type	Description
<code>input_width</code>	real	The current entity width
<code>input_height</code>	real	The current entity height
<code>input_x</code>	real	The current entity horizontal room position
<code>input_y</code>	real	The current entity vertical room position
<code>input_xscale</code>	real	The current entity horizontal scale multiplier
<code>input_yscale</code>	real	The current entity vertical scale multiplier
<code>input_rot</code>	real	The current entity rotation, in degrees
<b>Perspective Only</b>		
<code>input_zoom</code>	real	The current perspective zoom
<code>input_strength</code>	real	The current perspective parallax strength multiplier

**Note:** As with the corresponding output properties, perspective input properties are mapped to general properties and can be used interchangeably: `input_zoom = input_xscale` and `input_strength = input_yscale`

These values cannot be modified, and represent the properties of the entity currently being animated. Transitions, transforms, and deforms all have access to them, so don't hesitate to include them in your scripts!

## A Simple Transform

Now that you're familiar with the concept of keyframes and which properties can be animated with them, it's time to start creating keyframe animation scripts of your own!

To begin, create a new script in your project's asset browser and add your first keyframe:

**Example:**

```
if (keyframe()) {  
    //Properties  
}
```

If you're already comfortable with Q-script, this syntax should be quite familiar to you. In a sense, VNgen keyframes behave like stripped-down Quantum events, only instead of populating them with actions, we'll need to populate them with animation properties instead. In keeping with our earlier example, a simple keyframe would look something like this:

**Example:**

```
if (keyframe()) {  
    anim_x = 50;  
}
```

At this point, we already have a complete animation: the animated entity will move 50 pixels to the right, then move back *even though we haven't told it to*. When an animation ends, all modified properties are smoothly returned to their default values over the same duration as other keyframes. You can think of it as an 'invisible' keyframe automatically placed at the end of every script.

Of course, this all assumes the animation isn't looped. If it is, the entity will appear to get stuck with a 50-pixel offset since our one keyframe will be repeated over and over without end. To create a meaningful loop, we'll need to add a second keyframe as well:

**Example:**

```
if (keyframe()) {  
    anim_x = 50;  
}  
  
if (keyframe()) {  
    anim_x = -50;  
}
```

Now the entity will shift 50 pixels to the right, then 100 pixels to the left, ending up 50 pixels left of its original position. If the animation is looped, it will continue bouncing 100 pixels back and forth for as long as we allow the animation to run. And of course, there's no limit to how many keyframes we include in our animation, or how many properties we modify in a single keyframe. Perhaps we should add a little rotation too? How about an ease effect for impact?

**Example:**

```
if (keyframe()) {  
    anim_x = 50;  
    anim_rot = -5;  
    anim_ease = ease_bounce_out;  
}  
  
if (keyframe()) {  
    anim_x = -50;  
    anim_rot = 5;  
    anim_ease = ease_bounce_out;  
}
```

Once your animation is complete, save the script and execute it with a `vnngen_*_anim_start` action. It really is that simple!

## A Simple Transition

The same principles of regular animations apply to transitions as well, but with a few key differences:

First, transitions can only be applied with `vnngen_*_create` and `vnngen_*_destroy` scripts.

Second, on create, an entity jumps directly to the values specified in the first keyframe of a transition animation (instead of interpolating there from its default values). This makes single-keyframe transitions much more useful than regular animations, since a second keyframe is always implied. For example, the transition:

```
if (keyframe()) {  
    trans_alpha = 0;  
}
```

... would cause the entity to fade in from an alpha of 0 to a default alpha of 1 at the time of create, or fade out from the current alpha to 0 at the time of destroy.

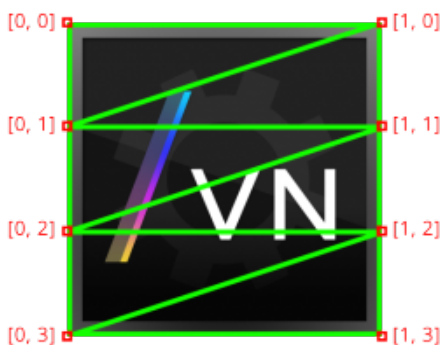
Third, while other animations can be played forwards or backwards at will, a transition's direction is determined by when it occurs. Create transitions always perform forwards, while destroy transitions always perform in reverse.

Keep these simple differences in mind and you'll be making your own transitions in no time!

## A Simple Deform

What might not be so simple is applying the same logic to deforms. It's relatively easy to form a mental image of what changing an entity's position and rotation will look like, but deforms can be much harder to imagine. However, in principle the two types of animations are the same.

Consider the following example:



**Tip:** Enabling VNgen's built-in [debug mode](#) will display overlays like the example above on all active deform animations in real-time

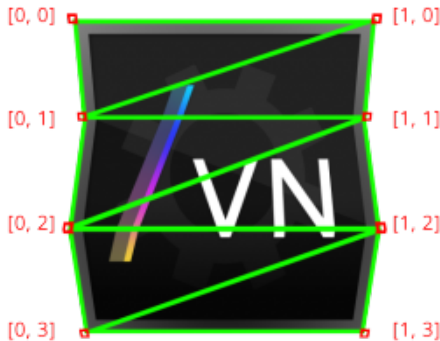
When a deform is applied to a VNgen entity, it switches from the standard drawing process to being drawn as a mesh of triangles. These triangles are mapped to a grid of points forming **columns** (the X axis) and **rows** (the Y axis). By default, deforms are assumed to be two columns wide and four rows tall, giving a range of points from [0, 0] to [1, 3]. Other subdivisions can optionally be specified inside a special keyframe labeled simply 'deform' and placed at the top of your deformation script.

```
if (deform()) {  
    def_width = 2;  
    def_height = 4;  
}
```

In the above example, the deform width (columns) and height (rows) are set to the default values, in which case including this section is not necessary and can be left out of the deformation script. For the purposes of this

documentation, these defaults will be assumed.

Rather than modify global properties like position and rotation, deformation animations manipulate each point's X and Y coordinates to stretch the final image into different shapes.



**Note:** Simulated image

In this example, all we've done is modify the X position of points [0, 1], [1, 1], [0, 3], and [1, 3], much like in the example of a transform animation. The similarity becomes even more apparent when written as a keyframe:

**Example:**

```
if (keyframe()) {  
    def_xpoint[0, 1] = 50; def_xpoint[1, 1] = -50;  
    def_xpoint[0, 3] = 50; def_xpoint[1, 3] = -50;  
}
```

And again, if we wanted to reverse the effect for an endless loop, we could simply create a second keyframe modifying points [0, 0], [1, 0], [0, 2], and [1, 2] instead:

**Example:**

```
if (keyframe()) {  
    def_xpoint[0, 1] = 50; def_xpoint[1, 1] = -50;  
    def_xpoint[0, 3] = 50; def_xpoint[1, 3] = -50;  
}  
  
if (keyframe()) {  
    def_xpoint[0, 0] = 50; def_xpoint[1, 0] = -50;  
    def_xpoint[0, 2] = 50; def_xpoint[1, 2] = -50;  
}
```

Note that the second keyframe does not define X points [0, 1], [1, 1], [0, 3], and [1, 3]. This would only be necessary if we wanted them to maintain their positions from the first keyframe or shift positions again. In VNgen animation scripts, any undefined properties in a given keyframe will be returned to their default values, therefore each point from the first keyframe will be treated as 0.

Also note that in this example, point values are arranged in a grid imitating each point's physical position relative to one another. This is a great way to visualize your keyframes as you write them.

With keyframes such as these in place, we can now save the script and execute it with a `vngen_*_deform_start` action. Don't let the 2D structure intimidate you: mesh deforms are really quite simple and just as easy to create as transform animations.

## Creating 'Living' Characters

While many entities in VNgen support deform animations, a prime use-case is **characters**. Characters are drawn as a composite texture of one or more sprites comprising the body, face, and any other elements you may desire. Using character **attachments**, it is possible to draw separate sprites for hair, clothing, even individual limbs and beyond! And as unique entities themselves, each attachment, or 'layer', can have its own animations of any kind. Try creating custom deform scripts for each attachment and watch your characters come to life!

For full details, see the [Characters & Attachments](#) section.

### 6.3.2. Included Animations

VNgen includes a number of transition, transform, and deform animations out of the box which can be used in your own projects or as templates for custom animation scripts. These animations will scale to any size sprite without modification.

Animations and deformations can be applied to VNgen entities via `vngen_*_anim_start` and `vngen_*_deform_start` scripts, respectively, while transitions are applied with `vngen_*_create` and `vngen_*_destroy` scripts instead.

## Transitions

### Animation

`trans_fade`  
`trans_zoom_in`  
`trans_zoom_out`  
`trans_slide_up`  
  
`trans_slide_down`  
  
`trans_slide_left`  
  
`trans_slide_right`  
  
`trans_spin_in`  
`trans_spin_out`  
`trans_wipe_up`  
`trans_wipe_down`  
`trans_wipe_left`  
`trans_wipe_right`

### Description

A simple fade in  
A zoom in from 0x original size  
A zoom out from 2x original size  
A fade/slide up based on the height of the animated entity  
A fade/slide down based on the height of the animated entity  
A fade/slide left based on the width of the animated entity  
A fade/slide right based on the width of the animated entity  
A spin/scale in from 0x original size  
A spin/scale out from 2x original size  
A linear wipe up, cropping the entity until fully visible  
A linear wipe down, cropping the entity until fully visible  
A linear wipe left, cropping the entity until fully visible  
A linear wipe right, cropping the entity until fully visible

## Transforms

### Animation

### Description



anim_bob	A subtle back-and-forth rotation
anim_bounce	A quick jump up and down
anim_shake	A simple slide left and right
anim_quake	A simple slide up and down
anim_wiggle	A repeated slide left and right of increasing, then decreasing intensity
anim_wobble	A repeated slide up and down of increasing, then decreasing intensity
anim_impact	A quick zoom-in coupled with horizontal and vertical slides while zooming back out
anim_squish	A cartoonish vertical stretch followed by a horizontal stretch before returning to normal
anim_flash	A quick fade out and back in
anim_siren	A red, blue color cycle simulating spinning police sirens
anim_alarm	A circular red color cycle simulating spinning indoor alarm lights

## Deforms

### Animation

def\_breathe  
def\_sweep

def\_wave

### Description

A subtle top-right corner tilt to simulate breathing

A subtle bottom-right corner slant to simulate soft body movement

A sine wave distortion to simulate liquid

## 6.3.3. The "deform" Function

### Syntax:

```
deform( );
```

### Argument

N/A

### Type

N/A

### Description

No arguments

### Description:

Initializes a segment of code containing definitions for a VNgen deformation script. Deformations are defined as being 2 columns wide and 4 rows tall by default. With this script, different dimensions can be defined at the start of a deformation script to create meshes of other sizes. Cannot be used in the regular Q-script loop, but rather is intended for use in dedicated animation scripts. See [Creating Custom Animations](#) for full details.

### Example:

```
if deform() {  
    def_width = 5;  
    def_height = 5;  
}
```

## 6.3.4. The "keyframe" Function



## Syntax:

```
keyframe();
```

Argument	Type	Description
N/A	N/A	No arguments

## Description:

Initializes a segment of code containing VNgen animation script. Cannot be used in the regular Q-script loop, but rather is intended for use in dedicated animation scripts. See [Creating Custom Animations](#) for full details.

### Example:

```
if keyframe() {  
    anim_x = 50;  
}
```

## 6.4. Effects

VNgen features a scripted effects system which can execute arbitrary code within custom keyframe sequences, similar to animations. Unlike animations, however, effects are independent and do not apply to any particular entity. Rather, effects are treated as entities themselves.

Although effects and animations operate on fundamentally different properties, the scripting syntax is nearly identical. It is also worth noting that while two animations of the same type cannot be performed simultaneously, as many effects as desired can be performed at once.

In this section we will examine included effects as well as how to write your own.

### 6.4.1. Creating Custom Effects

At its core, VNgen is a sequenced animation system with visual novel functions layered on top. Events and actions form a timeline of visual changes, often enhanced by built-in transition effects and keyframe animations. But what if you've dreamed up an action or animation completely unlike those offered by other systems? What about animating game properties that aren't part of VNgen at all? What about creating, drawing, and removing custom elements in a single action?

With VNgen's **keyframe effect scripts**, all of these things are possible.

## Intro to Keyframes

In animation (both digital and otherwise), a **keyframe** represents a significant change in motion. Rather than produce an animation one subtle frame at a time, creators focus only on the start and end of each motion, resulting in a sequence that roughly portrays how the final animation will play out. Once all the keyframes are in place, it's much easier to fill in the gaps with frames in between, smoothly transitioning from one keyframe to the next.

In VNgen, these "in-betweens" are generated automatically, meaning all you have to worry about are the keyframes themselves. Set any properties you want to change in a keyframe and they'll smoothly transition to their new values.



## Modifiable Properties

Effects are written using two separate categories of keyframes: **properties** (the 'keyframe') and **code** (the 'effect'). The syntax for writing standard keyframes is the same as other animations; fundamentally, the only difference lies in the way these properties are used in custom code.

VNgen uses an array of special variables which can be defined as virtually anything for the duration of the effect. However, it is important to note that these variables are *temporary*, and their values will be reset at the end of each frame (or when the next effect is executed, if any). As such, effect variables should never be used as references to 'volatile' data such as surfaces or data structures, as doing so will result in a memory leak.

**Note:** Another consequence of effect variables being temporary is that any properties *not* specified in a keyframe will be interpreted as their default values. For this reason, even if a modified property isn't meant to change for several keyframes, it must be declared in every keyframe for the modification to persist.

Property	Type	Description
ef_var[0] ... ef_var[#]	real/string	Custom variable, default value = 0
ef_ease	integer/macro	Ease mode for the current keyframe

Although effect values are custom and can serve a wide variety of purposes, for many effects it is helpful to have access to certain outside variables as well. This way, effects can be made that adapt to a variety of scenarios without modification. As such, VNgen also passes certain *input* properties to effect scripts.

General Properties	Type	Description
input_rate	real	The current game speed multiplier, adjusted for FPS and delta time
input_width	real	The current viewport width
input_height	real	The current viewport height
input_x	real	The current global horizontal offset defined in <code>vngen_object_draw</code>
input_y	real	The current global vertical offset defined in <code>vngen_object_draw</code>

These values cannot be modified, and represent properties of the engine itself.

## A Simple Effect

Now that you're familiar with the concept of keyframes and which properties can be animated with them, it's time to start creating keyframe effect scripts of your own!

To begin, create a new script in your project's asset browser. Before we begin adding keyframes, however, we must first decide what code our effect will execute, which will also determine what properties our variables will behave as.

All effect code is executed under a single keyframe labeled simply 'effect':

### Example:

```
if (effect()) {  
    //Code  
}
```

While there's no limit to what code an effect can contain, it's important to note that all effects will be executed in the Draw event, and as such are unsuitable for performing regular programming tasks. Instead, let's begin by drawing a colored rectangle which will flash on and off the screen.

**Example:**

```
if (effect()) {
    draw_set_alpha(ef_var[0]);
    draw_rectangle(input_x, input_y, input_x + input_width, input_y + input_height, f
else);
    draw_set_alpha(1);
}
```

Note how in the example above, `ef_var[0]` is input as the alpha value even though it hasn't been declared yet. It's good practice to plan out your use of available effect variables in this way before beginning to create keyframes. Of course, it would also be possible to use `ef_var[1]`, `ef_var[2]`, `ef_var[3]`, and `ef_var[4]` as the rectangle x, y, width, height, and so on if we desired, but for the sake of this example we'll stick to using `ef_var[0]` as alpha only.

With our code set up and our variables decided, we can now add our first keyframe:

**Note:** Keyframes may appear before or after effect code. Because execution order is predetermined, where each type of keyframe appears in the script is inconsequential.

**Example:**

```
if (keyframe()) {
    //Variables
}
```

If you're already comfortable with Q-script, this syntax should be quite familiar to you. In a sense, VNgen keyframes behave like stripped-down Quantum events, only instead of populating them with actions, we'll need to populate them with effect variables instead. In keeping with our earlier example, a simple keyframe would look something like this:

**Example:**

```
if (keyframe()) {
    ef_var[0] = 1;
}
```

At this point, we already have a complete effect: a white rectangle with the position and size of the perspective camera will fade from fully transparent to fully opaque, then fade back out *even though we haven't told it to*. When an effect ends, all modified properties are smoothly returned to their default values over the same duration as other keyframes. You can think of it as an 'invisible' keyframe automatically placed at the end of every script.

Of course, this all assumes the effect isn't looped. If it is, the screen will remain white forever since our one

keyframe will be repeated over and over without end. To create a meaningful loop, we'll need to add a second keyframe as well:

**Example:**

```
if (keyframe()) {  
    ef_var[0] = 1;  
}  
  
if (keyframe()) {  
    ef_var[0] = 0;  
}
```

Now the entity will fade in and out regardless of whether loop is enabled or not. If so, the screen will continue to flash for as long as we allow the effect to run. And of course, there's no limit to how many keyframes we include in our effect, or how many variables we create or modify in a single keyframe.

Once your effect is complete, save the script and execute it with the `vnngen_effect_start` action. It really is that simple!

## 6.4.2. Included Effects

VNgen includes a number of effects out of the box which can be used in your own projects or as templates for custom effect scripts. Where applicable, these effects will scale to any resolution and framerate without modification.

Effects can be performed with `vnngen_effect_start`.

### Dualshock

**Effect**

`ef_lightbar_flash`

**Description**

Flashes the Dualshock4 lightbar from bright to dim (Playstation 4 only)

### Haptics

**Effect**

`ef_hap_heartbeat`

`ef_hap_phone`

`ef_hap_pulse`

`ef_hap_ramp`

`ef_hap_sustain`

**Description**

Vibrates compatible gamepads with a heartbeat pattern

Vibrates compatible gamepads with a phone ringer pattern

Vibrates compatible gamepads with a brief pulse

Vibrates compatible gamepads from subtle to strong

Vibrates compatible gamepads persistently until stopped

### Visual

**Effect**

`ef_scrn_flash`

`ef_scrn_shake`

**Description**

Flashes the screen with the current color set by `draw_set_color`

Shakes the game window itself (does not apply in fullscreen)

### 6.4.3. The "effect" Function

#### Syntax:

```
effect();
```

Argument	Type	Description
N/A	N/A	No arguments

#### Description:

Initializes a segment of code containing VNgen effect instructions only (not regular keyframe data). Cannot be used in the regular Q-script loop, but rather is intended for use in dedicated effect scripts. See [Creating Custom Effects](#) for full details.

#### Example:

```
if effect() {  
    draw_set_alpha(ef_var[0]);  
    draw_rectangle(input_x, input_y, input_x + input_width, input_y + input_height, f  
else);  
    draw_set_alpha(1);  
}
```

## 6.5. Shaders

VNgen features integrated shader support for each individual entity. While using shaders with VNgen is similar to other animations and effects, shaders are written externally in a separate programming language (typically GLSL ES).

In a nutshell, shaders modify the position and/or color of vertices and/or pixels. While simple in principle, shaders can be used to create advanced visual effects like depth of field, light bloom, motion blur, and even entire 3D environments.

While instructions on how to write shaders of your own is beyond the scope of this reference guide, in this section we will examine shaders which are included with VNgen.

### 6.5.1. Included Shaders

VNgen includes a number of shaders out of the box which can be used in your own projects or as templates for custom shaders. Where applicable, these shaders will scale to any resolution and framerate without modification.

Shaders can be performed with `vngen_*_shader_start` scripts.

## Shaders

Shader	Description
sh_bloom	A soft, blur-based light bloom filter
sh_blur	An approximate gaussian blur filter
sh_chroma	An edge chromatic aberration filter



sh_gray	A color-averaged grayscale filter
sh_invert	An RGB-based color inversion filter
sh_radial	A radial zoom blur filter
sh_scanline	An old display simulation filter
sh_sepia	A color-averaged sepia tone filter
sh_wave	A bidirectional sine wave distortion filter

## 6.6. Engine Functions

VNgen was designed with customizability in mind. This means exposing every possible option to the developer to modify as they see fit, tailoring VNgen to suit any project. However, not all functions are meant to be used by developers. VNgen is also highly modular, and so certain engine functions are themselves written separately from the code that utilizes them.

In this section we'll examine available engine functions and their roles in VNgen. These functions do not require any action from the developer and typically should not be executed manually, however advanced users may find ways to utilize them for low-level customizations that wouldn't be possible otherwise.

**Important note:** Making undocumented changes to engine functions is unsupported. Users working from modified engine functions will be ineligible to receive support for bugs and other issues which occur as a result of these modifications.

### 6.6.1. The "sys\_vngen\_config" Function

#### Syntax:

```
sys_vngen_config();
```

Argument	Type	Description
N/A	N/A	Self-executing script

#### Description:

Automatically executes once at runtime to initialize all global variables and enumerators required for VNgen to operate.

It is never necessary to run this script manually.

### 6.6.2. The "sys\_action\_init" Function

#### Syntax:

```
sys_anim_init();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Automatically run by action scripts. Assigns an ID to the running action and tests whether the action should be performed, returning true or false based on the result.

As this script is run automatically by action scripts, it is never necessary to run `sys_action_init` manually.

### 6.6.3. The "sys\_action\_skip" Function

**Syntax:**

```
sys_action_skip([enable]);
```

Argument	Type	Description
[enable]	boolean	<i>Optional:</i> Enables or disables skipping actions in the current event

**Description:**

Checks whether action skipping is currently enabled and returns the result, or if a boolean value is supplied, enables or disables skipping actions in the current event.

As this script is run automatically by other engine functions, it is almost never necessary to run `sys_action_skip` manually.

### 6.6.4. The "sys\_action\_term" Function

**Syntax:**

```
sys_action_term();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Terminates the running action so that it is considered complete and will no longer be performed.

As this script is run automatically by all action scripts, it is never necessary to run `sys_action_term` manually.

### 6.6.5. The "sys\_anim\_init" Function

**Syntax:**

```
sys_anim_init(entity, index, anim, duration, loop, reverse, ease);
```

Argument	Type	Description
entity	integer	The data structure of the entity to be

index	integer	animated The index of the row containing the target entity
anim	script	The animation script to be performed
duration	real	Sets the duration of the entire animation, in seconds
loop	boolean	Enables or disables endlessly looping the animation
reverse	boolean	Enables or disables reversing the animation
ease	integer/macro	Sets the animation easing override

### Description:

Initializes a given scripted animation for the input entity which can be performed with `sys_anim_perform`.

As this script is run automatically by any animation scripts tailored to specific entity types, it is almost never necessary to run `sys_anim_init` manually.

See [Animations](#) for a list of included animation scripts and how to make your own.

## 6.6.6. The "sys\_anim\_perform" Function

### Syntax:

```
sys_anim_perform(entity, index);
```

Argument	Type	Description
entity	integer	The data structure of the entity to be animated
index	integer	The index of the row containing the target entity

### Description:

Performs a scripted animation on the input entity which has been initialized with `sys_anim_init`.

As this script is run automatically by VNgen draw functions, it is almost never necessary to run `sys_anim_perform` manually.

See [Animations](#) for a list of included animation scripts and how to make your own.

## 6.6.7. The "sys\_anim\_term" Function

### Syntax:

```
sys_anim_term(entity, index);
```

Argument	Type	Description
entity	integer	The data structure of the entity to end animation
index	integer	The index of the row containing the



target entity

**Description:**

Terminates a scripted animation which has been performed on the input entity with `sys_anim_perform`. By nature, only looped animations need to be terminated.

As this script is run automatically by any animation scripts tailored to specific entity types, it is almost never necessary to run `sys_anim_term` manually.

See [Animations](#) for a list of included animation scripts and how to make your own.

### 6.6.8. The "sys\_anim\_speech" Function

**Syntax:**

```
sys_anim_speech(name, speech, type, [highlight]);
```

Argument	Type	Description
name	string	The name of the character to apply animation to
speech	boolean	Enables or disables speech animations
type	integer/macro	Sets whether to base speech animations on text or audio
[highlight]	boolean	<i>Optional:</i> Enables or disables highlight animations

**Description:**

Enables or disables character speech animations and optional highlighting features.

As this script is run automatically by text and voice functions, it is never necessary to run `sys_anim_speech` manually.

### 6.6.9. The "sys\_cmd\_init" Function

**Syntax:**

```
sys_cmd_init();
```

Argument	Type	Description
N/A	N/A	Self-executing script

**Description:**

Automatically executes once at runtime to initialize all global variables required for QCMD to operate. Note, however, that the actual console database is not built until the console is opened so as to prevent memory waste in shipping applications.



It is never necessary to run this script manually.

## 6.6.10. The "sys\_cmd\_add" Function

### Syntax:

```
sys_cmd_add(command, script);
```

Argument	Type	Description
command	string	The command to listen for as a string, minus arguments
script	script	The script to execute when the command is input

### Description:

Adds a command to the command console which will execute the specified script when input. Note that the command here should only be written as the command itself; any additional arguments will be passed into the script automatically.

This script is run automatically by other engine functions to add included VNgen commands to the console database when it is opened. It is only necessary to run this script manually when adding custom commands to the console. As the command console is a global entity, custom commands can be added from anywhere in the project.

## 6.6.11. The "sys\_cmd\_perform" Function

### Syntax:

```
sys_cmd_perform();
```

Argument	Type	Description
N/A	N/A	Self-executing script

### Description:

Manages and receives command console user input and executes commands when the console is visible.

As this script is run automatically by other engine functions, it is almost never necessary to run `sys_cmd_perform` manually.

## 6.6.12. The "sys\_cmd\_draw" Function

### Syntax:

```
sys_cmd_draw(x, y, width, height, font, col1, col2);
```

Argument	Type	Description
x	real	The horizontal position to draw the console
y	real	The vertical position to draw the



width	real	console
height	real	sets the console width in pixels
font	font	sets the console height in pixels
		sets the console output font
col1	color	sets the console background color
col2	color	sets the console foreground color

### Description:

Draws the command console with the specified position, dimensions, and style when visible. The console is positioned from the top-left corner.

Note that any input longer than the width of the console will be scrolled to the position of the cursor, not line-wrapped.

As this script is run automatically by other engine functions, it is almost never necessary to run `sys_cmd_draw` manually.

## 6.6.13. The "sys\_toggle\_cmd" Function

### Syntax:

```
sys_toggle_cmd([sound]);
```

Argument	Type	Description
[sound]	sound	<i>Optional:</i> A sound to be played when the script is run

### Description:

Hides the command console if visible, or unhides it if invisible. Also blocks most engine input functions while the console is displayed.

As this script is run automatically by other engine functions, it is almost never necessary to run `sys_toggle_cmd` manually. By default, the console can be opened and closed using the tilde (~) key when debug mode is enabled.

## 6.6.14. The "sys\_deform\_init" Function

### Syntax:

```
sys_deform_init(entity, index, def, duration, loop, reverse, ease);
```

Argument	Type	Description
entity	integer	The data structure of the entity to be animated
index	integer	The index of the row containing the target entity
def	script	The deformation script to be performed



duration	real	Sets the duration of the entire deformation, in seconds
loop	boolean	Enables or disables endlessly looping the deformation
reverse	boolean	Enables or disables reversing the deformation
ease	integer/macro	Sets the deformation easing override

### Description:

Initializes a given scripted deformation for the input entity which can be performed with `sys_deform_perform`.

As this script is run automatically by any deformation scripts tailored to specific entity types, it is almost never necessary to run `sys_deform_init` manually.

See [Animations](#) for a list of included deformation scripts and how to make your own.

## 6.6.15. The "sys\_deform\_perform" Function

### Syntax:

```
sys_deform_perform(entity, index);
```

Argument	Type	Description
entity	integer	The data structure of the entity to be deformed
index	integer	The index of the row containing the target entity

### Description:

Performs a scripted deformation on the input entity which has been initialized with `sys_deform_init`.

As this script is run automatically by VNgen draw functions, it is almost never necessary to run `sys_deform_perform` manually.

See [Animations](#) for a list of included deformation scripts and how to make your own.

## 6.6.16. The "sys\_deform\_draw" Function

### Syntax:

```
sys_deform_draw(entity, index, tex, width, height, x, y, xscale, yscale, rot, alpha)  
;
```

Argument	Type	Description
entity	integer	The data structure of the entity to end animation
index	integer	The index of the row containing the target entity
tex	texture	The texture to be drawn deformed

---

width	real	The original width of the source sprite or surface
height	real	The original height of the source sprite or surface
x	real	The horizontal position to draw the deformation
y	real	The vertical position to draw the deformation
xscale	real	The horizontal scale multiplier to apply to the deformation
yscale	real	The vertical scale multiplier to apply to the deformation
rot	real	The rotation to apply to the deformation, in degrees
alpha	real	The alpha transparency to apply to the deformation

**Description:**

Draws a deformation which has been previously processed with `sys_deform_perform`.

As this script is run automatically by any deformation scripts tailored to specific entity types, it is almost never necessary to run `sys_deform_draw` manually.

See [Animations](#) for a list of included deformation scripts and how to make your own.

### 6.6.17. The "sys\_deform\_term" Function

**Syntax:**

```
sys_deform_term(entity, index);
```

Argument	Type	Description
entity	integer	The data structure of the entity to end animation
index	integer	The index of the row containing the target entity

**Description:**

Terminates a scripted deformation which has been performed on the input entity with `sys_deform_perform`. By nature, only looped deformations need to be terminated.

As this script is run automatically by any deformation scripts tailored to specific entity types, it is almost never necessary to run `sys_deform_term` manually.

See [Animations](#) for a list of included deformation scripts and how to make your own.

### 6.6.18. The "sys\_effect\_init" Function

**Syntax:**

```
sys_effect_init(index, effect, duration, loop, reverse, ease);
```

Argument	Type	Description
index	integer	The index of the row containing the target entity
effect	script	The effect script to be performed
duration	real	Sets the duration of the entire effect, in seconds
loop	boolean	Enables or disables endlessly looping the effect
reverse	boolean	Enables or disables reversing the effect
ease	integer/macro	Sets the effect easing override

### Description:

Initializes a given scripted effect which can be performed with `sys_effect_perform`.

As this script is run automatically by `vnngen_effect_start`, it is never necessary to run `sys_effect_init` manually.

See [Effects](#) for a list of included effect scripts and how to make your own.

## 6.6.19. The "sys\_effect\_perform" Function

### Syntax:

```
sys_effect_perform(index, x, y);
```

Argument	Type	Description
index	integer	The index of the row containing the target entity
x	real	The global horizontal offset to display all effects
y	real	The global vertical offset to display all effects

### Description:

Performs effects which have been initialized with `vnngen_effect_start`. Also returns true or false depending on whether the effect is active.

As this script is run automatically by `vnngen_object_draw`, it is never necessary to run `sys_effect_perform` manually.

Note that the offsets supplied here only provide input values to effect scripts. Actually using these offsets is voluntary.

See [Effects](#) for a list of included animation scripts and how to make your own.

## 6.6.20. The "sys\_effect\_term" Function

**Syntax:**

```
sys_effect_term(index);
```

Argument	Type	Description
index	integer	The index of the row containing the target entity

**Description:**

Terminates an effect which has been performed with `sys_effect_perform`. By nature, only looped effects need to be terminated.

As this script is run automatically by `vnngen_effect_stop`, it is never necessary to run `sys_effect_term` manually.

See [Effects](#) for a list of included animation scripts and how to make your own.

## 6.6.21. The "sys\_event\_skip" Function

**Syntax:**

```
sys_event_skip([index]);
```

Argument	Type	Description
[index]	integer	<i>Optional:</i> Sets the numeric event ID to skip to

**Description:**

Checks whether event skipping is currently enabled and returns the result, or if a numeric value is supplied, enables or disables skipping to the target event.

As this script is run automatically by other engine functions, it is almost never necessary to run `sys_event_skip` manually.

## 6.6.22. The "sys\_read\_skip" Function

**Syntax:**

```
sys_read_skip([enable]);
```

Argument	Type	Description
[enable]	integer	<i>Optional:</i> Enables or disables read event skipping

**Description:**

Checks whether read skipping is currently enabled and returns the result, or if a boolean value is supplied, enables or disables skipping to the nearest option block or unread event.



As this script is run automatically by other engine functions, it is almost never necessary to run `sys_read_skip` manually.

### 6.6.23. The "sys\_grid\_delete" Function

#### Syntax:

```
sys_grid_delete(ds_grid, row);
```

Argument	Type	Description
ds_grid	data structure	The data structure to remove row from
row	integer	The index of the row to remove

#### Description:

Removes a row from the specified `ds_grid` while preserving other data and returns the final grid.

Note that the final grid **must** be assigned to an external variable to prevent memory leaks! It is recommended to use the same variable as is assigned to the original input grid.

As this script is run automatically by other engine functions, it is almost never necessary to run `sys_grid_delete` manually.

### 6.6.24. The "sys\_grid\_last" Function

#### Syntax:

```
sys_grid_last(ds_grid);
```

Argument	Type	Description
ds_grid	data structure	The data structure to check

#### Description:

Returns the last available index in the input `ds_grid`, or 'undefined' if the grid does not exist or is empty.

As this script is run automatically by other engine functions, it is almost never necessary to run `sys_grid_last` manually.

### 6.6.25. The "sys\_layer\_set\_target" Function

#### Syntax:

```
sys_layer_set_target(x, y);
```

Argument	Type	Description
x	real	horizontal offset for all elements
y	real	vertical offset for all elements

#### Description:

Initializes perspective and other functions necessary for drawing VNgen layers

### 6.6.26. The "sys\_layer\_draw\_scene" Function

#### Syntax:

```
sys_layer_draw_scene( foreground );
```

Argument	Type	Description
foreground	boolean	Enables or disables drawing foregrounds (as opposed to backgrounds)

#### Description:

Draws scenes as either backgrounds or foregrounds.

Also has a legacy counterpart, `sys_layer_draw_scene_legacy`, which renders scenes without certain advanced features.

### 6.6.27. The "sys\_layer\_draw\_char" Function

#### Syntax:

```
sys_layer_draw_char( highlight );
```

Argument	Type	Description
highlight	boolean	Enables or disables highlighting the current speaking character(s)

#### Description:

Draws characters and character attachments with an optional highlighting effect to indicate the current speaker, if any.

Also has a legacy counterpart, `sys_layer_draw_char_legacy`, which renders characters without certain advanced features.

### 6.6.28. The "sys\_layer\_draw\_emote" Function

#### Syntax:

```
sys_layer_draw_emote( );
```

Argument	Type	Description
N/A	N/A	No arguments

#### Description:

Draws emotes.

Also has a legacy counterpart, `sys_layer_draw_emote_legacy`, which renders emotes without certain advanced features.

### 6.6.29. The "sys\_layer\_draw\_perspective" Function

#### Syntax:

```
sys_layer_draw_perspective();
```

Argument	Type	Description
N/A	N/A	No arguments

#### Description:

Draws perspective replace transitions.

Note that this script does NOT calculate perspective values themselves, as these are handled by `sys_layer_draw_init`.

### 6.6.30. The "sys\_layer\_draw\_effect" Function

#### Syntax:

```
sys_layer_draw_effect();
```

Argument	Type	Description
N/A	N/A	No arguments

#### Description:

Draws and/or performs effects.

### 6.6.31. The "sys\_layer\_draw\_textbox" Function

#### Syntax:

```
sys_layer_draw_textbox();
```

Argument	Type	Description
N/A	N/A	No arguments

#### Description:

Draws textboxes.

Also has a legacy counterpart, `sys_layer_draw_textbox_legacy`, which renders scenes without certain advanced features.

### 6.6.32. The "sys\_layer\_draw\_text" Function

**Syntax:**

```
sys_layer_draw_text();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Draws text.

Also has a legacy counterpart, `sys_layer_draw_text_legacy`, which renders text without certain advanced features.

### 6.6.33. The "sys\_layer\_draw\_label" Function

**Syntax:**

```
sys_layer_draw_label();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Draws labels.

Also has a legacy counterpart, `sys_layer_draw_label_legacy`, which renders labels without certain advanced features.

### 6.6.34. The "sys\_layer\_draw\_prompt" Function

**Syntax:**

```
sys_layer_draw_prompt();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Draws text prompts.

Also has a legacy counterpart, `sys_layer_draw_prompt_legacy`, which renders prompts without certain advanced features.

### 6.6.35. The "sys\_layer\_draw\_button" Function

**Syntax:**

```
sys_layer_draw_button();
```

---

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Draws buttons.

Also has a legacy counterpart, `sys_layer_draw_button_legacy`, which renders buttons without certain advanced features.

### 6.6.36. The "sys\_layer\_draw\_option" Function

**Syntax:**

```
sys_layer_draw_option();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Draws options.

Also has a legacy counterpart, `sys_layer_draw_option_legacy`, which renders options without certain advanced features.

### 6.6.37. The "sys\_layer\_reset\_target" Function

**Syntax:**

```
sys_layer_reset_target();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Finalizes drawing VNgen layers and draws debug information, if enabled

### 6.6.38. The "sys\_layer\_log\_set\_target" Function

**Syntax:**

```
sys_layer_log_set_target();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Initializes visibility, audio, and other functions necessary for drawing the VNgen backlog

## 6.6.39. The "sys\_layer\_draw\_log" Function

### Syntax:

```
sys_layer_draw_log(spr_background, spr_divider, spr_paused, spr_playing, sep, linebreak, font, color);
```

Argument	Type	Description
spr_background	sprite	Sprite to draw as a fullscreen background while the log is open (or keyword 'none' for none)
spr_divider	sprite	Sprite to draw as a divider between logged entries (or keyword 'none' for none)
spr_paused	sprite	Sprite to draw as paused audio icon (or keyword 'none' for none)
spr_playing	sprite	Sprite to draw as playing audio icon (or keyword 'none' for none)
sep	real	Distance in pixels between log entries and dividers (if any)
linebreak	real	Width in pixels before text is wrapped into a new line
font	font	Override font to draw logged text in (or keyword 'inherit' for logged style)
color	color	Override color to draw logged text in (or keyword 'inherit' for logged style)

### Description:

Draws the backlog, including text and stylization.

Also has a legacy counterpart, `sys_layer_draw_log_legacy`, which renders the backlog without certain advanced features.

## 6.6.40. The "sys\_layer\_draw\_log\_button" Function

### Syntax:

```
sys_layer_draw_log_button();
```

Argument	Type	Description
N/A	N/A	No arguments

### Description:

Draws log buttons.

Also has a legacy counterpart, `sys_layer_draw_log_button_legacy`, which renders buttons without certain advanced features.

## 6.6.41. The "sys\_layer\_log\_reset\_target" Function

### Syntax:

```
sys_layer_log_reset_target();
```

Argument	Type	Description
N/A	N/A	No arguments

### Description:

Finalizes drawing the backlog and handles scrolling and mouse state management

## 6.6.42. The "sys\_log\_init" Function

### Syntax:

```
sys_log_init(entity, index, linebreak, font);
```

Argument	Type	Description
entity	integer	The data structure of the text entity to process
index	integer	The index of the row containing the target entity
linebreak	real	The width in pixels before text is wrapped into a new line
font	font	The font to process text in, where fnt_default is default

### Description:

Processes a string of text for the backlog and calculates dimensions on initialization. Also returns the final surface created for further processing.

As this script is run automatically by the backlog, it is almost never necessary to run `sys_log_init` manually.

## 6.6.43. The "sys\_log\_perform" Function

### Syntax:

```
sys_log_perform(entity, index, font, color);
```

Argument	Type	Description
entity	integer	The data structure of the entity for which to perform text
index	integer	The index of the row containing the target entity ID
font	font	Override font to draw backlog text in (optional, use keyword 'inherit' to disable)
color	color	Override color to draw backlog text in

(optional, use keyword 'inherit' to disable)

**Description:**

Renders text which has been pre-processed with `sys_log_init` to a surface which can then be drawn to the backlog.

**6.6.44. The "sys\_log\_get\_style" Function****Syntax:**

```
sys_log_get_style(name);
```

Argument	Type	Description
name	string	The speaker name to check for associated style data

**Description:**

Checks the input speaker name for associated label style data and returns the name string enclosed in color markup for style data, if found.

**6.6.45. The "sys\_log\_get\_xoffset" Function****Syntax:**

```
sys_log_get_xoffset(entity, index, char_index);
```

Argument	Type	Description
entity	integer	The data structure of the entity for which to perform text
index	integer	The index of the row containing the target entity ID
char_index	integer	The index of the character in the string before which to check for linebreaks

**Description:**

Returns the horizontal offset of the current line of text based on the current paragraph alignment set by `vnngen_set_halign` at the time when logged text was recorded.

**6.6.46. The "sys\_mouse\_hover" Function****Syntax:**

```
sys_mouse_hover(state);
```

Argument	Type	Description
state	boolean	Sets the mouse cursor state to hover



(true) or default (false)

**Description:**

Enables or disables setting mouse cursors to the hover state. System cursors will be used by default unless sprite-based cursors have been set with `vnngen_set_cursor`.

As this script is run automatically by other engine functions, it is almost never necessary to run `sys_mouse_hover` manually.

## 6.6.47. The "sys\_option\_init" Function

**Syntax:**

```
sys_option_init();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Automatically run by option scripts. Assigns an ID to the running option and tests whether the option should be performed, returning true or false based on the result.

As this script is run automatically by option scripts, it is never necessary to run `sys_option_init` manually.

## 6.6.48. The "sys\_orig\_init" Function

**Syntax:**

```
sys_orig_init(entity, index, xorig, yorig);
```

Argument	Type	Description
entity	integer	The data structure of the entity to set origin of
index	integer	The index of the row containing the target entity
xorig	real/macro	The horizontal texture offset, or origin point, relative to the top-left corner
yorig	real/macro	The horizontal texture offset, or origin point, relative to the top-left corner

**Description:**

Applies the given origin points to the input entity on initialization. Origin can be set as either a literal value in pixels, or calculated automatically to set origins to left, top, center, right, or bottom.

As this script is run automatically by any actions that allow setting origins, it is almost never necessary to run `sys_orig_init` manually.

See [Macros & Keywords](#) for a list of available automatic origin modes.

## 6.6.49. The "sys\_queue\_enqueue" Function

### Syntax:

```
sys_queue_enqueue(id, [value0], [value1] ...);
```

Argument	Type	Description
id	string	The ID of the queue to add data to
value0 ... value14	real/string	<i>Optional:</i> Up to 15 values which can be enqueued as a single set of data

### Description:

VNgen uses a custom queue system for certain data to afford greater control over when and how it is applied. This is **not** the same as GameMaker Studio's built-in queue data structures and should not be used as a replacement for them.

Data enqueued with this script will be stored as a group occupying one slot in the queue, rather than a typical queue which only accepts one value in a slot.

If the target queue does not exist, it will be created. Note that queues created this way can never be *entirely* removed from memory, therefore VNgen queues should never be used for regular data.

As this script is run automatically by any actions that require delayed data execution, it is almost never necessary to run `sys_queue_enqueue` manually.

## 6.6.50. The "sys\_queue\_submit" Function

### Syntax:

```
sys_queue_submit(id);
```

Argument	Type	Description
id	string	The ID of the queue to read data from

### Description:

VNgen uses a custom queue system for certain data to afford greater control over when and how it is applied. This is **not** the same as GameMaker Studio's built-in queue data structures and should not be used as a replacement for them.

This script is specially designed to read queued data into the global backlog. Once data has been read, it will be removed from the queue. This script expects data to already be formatted as backlog data, and improperly formatted data will halt the game with an error.

As this script is run automatically by any actions that require delayed data execution, it is almost never necessary to run `sys_queue_submit` manually.

## 6.6.51. The "sys\_queue\_destroy" Function

**Syntax:**

```
sys_queue_destroy(id);
```

Argument	Type	Description
id	string	The ID of the queue to destroy

**Description:**

VNgen uses a custom queue system for certain data to afford greater control over when and how it is applied. This is **not** the same as GameMaker Studio's built-in queue data structures and should not be used as a replacement for them.

This script will clear the target queue of all data, if any exists. Note that queues destroyed this way can never be *entirely* removed from memory, therefore VNgen queues should never be used for regular data.

As this script is run automatically by any actions that require delayed data execution, it is almost never necessary to run `sys_queue_destroy` manually.

## 6.6.52. The "sys\_queue\_empty" Function

**Syntax:**

```
sys_queue_empty(id);
```

Argument	Type	Description
id	string	The ID of the queue to check

**Description:**

VNgen uses a custom queue system for certain data to afford greater control over when and how it is applied. This is **not** the same as GameMaker Studio's built-in queue data structures and should not be used as a replacement for them.

This script will check whether the target queue has data and return true if empty or false if not empty.

## 6.6.53. The "sys\_scale\_init" Function

**Syntax:**

```
sys_scale_init(entity, index, width, height, width_init, height_init, scaling);
```

Argument	Type	Description
entity	integer	The data structure of the entity to be scaled
index	integer	The index of the row containing the target entity
width	real	The current width in pixels of the reference area being scaled against
height	real	The current height in pixels of the reference area being scaled against
width_init	real	The initial width in pixels of the

height_init	real	reference area being scaled against The initial height in pixels of the reference area being scaled against
scaling	integer/macro	Sets the scaling mode to perform on the target entity

### Description:

Applies a given scaling technique to the input entity on initialization. Scaling *modes* are unique from scale *modifications* and will persist through regular modifications.

As this script is run automatically by any actions that require scaling, it is almost never necessary to run `sys_scale_init` manually.

See [Macros & Keywords](#) for a list of available scaling modes.

**Note:** This script should not be confused with `vnngen_set_scale`, which is completely unrelated.

## 6.6.54. The "sys\_shader\_init" Function

### Syntax:

```
sys_shader_init(entity, index, shader);
```

Argument	Type	Description
entity	integer	The data structure of the entity to be animated
index	integer	The index of the row containing the target entity
shader	shader	The shader to be performed

### Description:

Initializes a given shader for the input entity which can be performed with `sys_shader_perform`.

As this script is run automatically by any shader scripts tailored to specific entity types, it is almost never necessary to run `sys_shader_init` manually.

See [Shaders](#) for a list of included shaders.

## 6.6.55. The "sys\_shader\_perform" Function

### Syntax:

```
sys_shader_perform(entity, index);
```

Argument	Type	Description
entity	integer	The data structure of the entity to be animated
index	integer	The index of the row containing the

target entity

**Description:**

Performs a shader on the input entity which has been initialized with `sys_shader_init`.

As this script is run automatically by VNgen draw functions, it is almost never necessary to run `sys_shader_perform` manually.

See [Shaders](#) for a list of included shaders.

## 6.6.56. The "sys\_shader\_exists" Function

**Syntax:**

```
sys_shader_exists(shader);
```

Argument	Type	Description
shader	shader	The shader to check

**Description:**

Checks to see whether the given shader *likely* exists and returns true or false. If the shader *asset* exists, but was not successfully compiled due to errors or incompatibility with the current platform, false will also be returned.

Unfortunately, there is no universal way to determine whether a shader exists in GameMaker, so this script is tailored to run checks consequential to VNgen only.

See [Shaders](#) for a list of included shaders.

## 6.6.57. The "sys\_shader\_set\_sampler" Function

**Syntax:**

```
sys_shader_set_sampler(sampler, source, [index]);
```

Argument	Type	Description
sampler	integer	The index of the shader sampler to assign source to
source	sprite/string	The source sprite or surface to assign to sampler (string required for surface)
[index]	real	The source image index, if source is a sprite

**Description:**

Checks whether the source sprite or surface exists, and if so, assigns it as a sampler to the current shader

As this script is run automatically by VNgen draw functions, it is almost never necessary to run `sys_anim_term` manually.



---

See [Shaders](#) for a list of included shaders.

## 6.6.58. The "sys\_text\_init" Function

### Syntax:

```
sys_text_init(entity, index, text, name, linebreak, lineheight, font, col1, col2, col3, col4, shadow, outline, speed);
```

Argument	Type	Description
entity	integer	The data structure of the text entity to process
index	integer	The index of the row containing the target entity
text	string	The text string to process
name	string	The speaker name associated with the text, for style inheritance
linebreak	real	The width in pixels before text is wrapped into a new line
lineheight	real	The distance between lines of text, as a multiplier of the height of one line
font	font	The font to process text in, where fnt_default is default
col1	color	The text top-left gradient color, where c_white is default
col2	color	The text top-right gradient color, where c_white is default
col3	color	The text bottom-right gradient color, where c_white is default
col4	color	The text bottom-left gradient color, where c_white is default
shadow	color/macro	The text shadow color, where 'none' is default
outline	color/macro	The text outline color, where 'none' is default
speed/event	real	The rate at which text is printed onto the screen, as a value of characters per-second

### Description:

Processes a string of text for markup and calculates dimensions on initialization. Also returns the final surface created for further processing. This script does not handle style inheritance, which must be processed externally beforehand and the final results passed into this script.

As this script is run automatically by any actions that require processing text, it is almost never necessary to run `sys_text_init` manually.

## 6.6.59. The "sys\_text\_perform" Function

### Syntax:

```
sys_text_perform(entity, index);
```

Argument	Type	Description
----------	------	-------------

---

entity	integer	The data structure of the entity for which to perform text
index	integer	The index of the row containing the target entity ID

**Description:**

Renders text which has been pre-processed with `sys_text_init`, including performing most special markup functions.

## 6.6.60. The "sys\_text\_get\_label" Function

**Syntax:**

```
sys_text_get_label();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Scans text data for character names and stores the results in memory for use with auto labels. If multiple simultaneous speakers exist, names will be concatenated and separated with a slash.

As this script is run automatically by action scripts, it is almost never necessary to run `sys_text_get_label` manually.

## 6.6.61. The "sys\_text\_get\_xoffset" Function

**Syntax:**

```
sys_text_get_xoffset(entity, index);
```

Argument	Type	Description
entity	integer	The data structure of the entity for which to perform text
index	integer	The index of the row containing the target entity ID

**Description:**

Returns the horizontal offset of the current line of text based on the current paragraph alignment set by `vnngen_set_halign`.

## 6.6.62. The "sys\_text\_style\_init" Function

**Syntax:**

```
sys_text_style_init(entity, index, name, font, col1, col2, col3, col4, shadow, outli
```



```
ne);
```

Argument	Type	Description
entity	integer	The data structure of the entity to be styled
index	integer	The index of the row containing the target entity
name	string	The character name under which inheritance data is/will be stored
font	font/macro	The font to draw text in, where <code>fnt_default</code> is default (or keyword <i>'inherit'</i> for saved style)
col1	color/macro	The text top-left gradient color, where <code>c_white</code> is default (or keyword <i>'previous'</i> for no change, or keyword <i>'inherit'</i> for saved style)
col2	color/macro	The text top-right gradient color, where <code>c_white</code> is default (or keyword <i>'previous'</i> for no change, or keyword <i>'inherit'</i> for saved style)
col3	color/macro	The text bottom-right gradient color, where <code>c_white</code> is default (or keyword <i>'previous'</i> for no change, or keyword <i>'inherit'</i> for saved style)
col4	color/macro	The text bottom-left gradient color, where <code>c_white</code> is default (or keyword <i>'previous'</i> for no change, or keyword <i>'inherit'</i> for saved style)
shadow	color/macro	The text shadow color, where <i>'none'</i> is default (or keyword <i>'previous'</i> for no change, keyword <i>'inherit'</i> for saved style, or <i>'none'</i> for none)
outline	color/macro	The text outline color, where <i>'none'</i> is default (or keyword <i>'previous'</i> for no change, keyword <i>'inherit'</i> for saved style, or <i>'none'</i> for none)

### Description:

Processes text stylization, including support for style inheritance, and returns the results as a `ds_map`.

**Important note:** The resulting `ds_map` is temporary and must be destroyed outside this script when finished to prevent memory leaks!

As this script is run automatically by any text scripts supporting style inheritance, it is never necessary to run `sys_text_style_init` manually.

## 6.6.63. The "sys\_trans\_init" Function

### Syntax:

```
sys_trans_init(entity, index, trans, duration, reverse);
```



Argument	Type	Description
entity	integer	The data structure of the entity to transition
index	integer	The index of the row containing the target entity
trans	integer/macro	Sets the transition to perform on the target entity
duration	real	Sets the duration of the entire transition, in seconds
reverse	boolean	Enables or disables reversing the transition

### Description:

When a new entity is created, before a transition can be applied, it must be initialized so that the entity begins with its transitioned properties rather than the properties to which it will be transitioned.

As this script is run automatically by any actions that require forward transitions, it is almost never necessary to run `sys_trans_init` manually.

See [Macros & Keywords](#) for a list of available transitions.

## 6.6.64. The "sys\_trans\_perform" Function

### Syntax:

```
sys_trans_perform(entity, index, ease);
```

Argument	Type	Description
entity	integer	The data structure of the entity to transition
index	integer	The index of the row containing the target entity
ease	integer /macro	Sets the transition easing override

### Description:

Performs a transition on the target entity when created.

As this script is run automatically by any actions that require transitions, it is almost never necessary to run `sys_trans_perform` manually.

See [Macros & Keywords](#) for a list of available transitions.

## 6.6.65. The "sys\_vox\_add" Function

### Syntax:

```
sys_vox_add(entity, index, source);
```

Argument	Type	Description
entity	integer	The data structure of the entity to add vox to
index	integer	The index of the row containing the target entity
source	sound/array	The source sound or array of sounds to add

## Description:

Adds a new sound or array of sounds to the specified vox data structure.

As this script is run automatically by vox actions, it is almost never necessary to run `sys_vox_add` manually.

## 6.7. Global Functions

In addition to events and actions, VNgen relies on a number of global functions which exist outside the scope of the Quantum framework. These functions handle input, common logic processes, saving and loading, display settings, and more.

In this section we'll examine all available global functions and their use-cases both in VNgen and otherwise. While many of these functions are operated automatically, others can be used to customize the user experience, and a few are absolute essentials for any VNgen developer. By familiarizing yourself with VNgen's global functions you'll gain a better understanding of how the engine works and how to harness its power for your own creations.

### 6.7.1. The "vngen\_do\_auto" Function

#### Syntax:

```
vngen_do_auto(toggle, [delay], [sound]);
```

Argument	Type	Description
toggle	boolean/macro	Enables, disables, or toggles automatic progression
[delay]	real	<i>Optional:</i> Sets the delay before auto-continue is triggered, in seconds
[sound]	sound	<i>Optional:</i> A sound to be played when the script is run

#### Description:

By default, text actions will pause further event progression until `vngen_do_continue` is run, but if auto mode is active, the `vngen_do_continue` script will be automatically run after any delay has expired. This script can be used to toggle or explicitly enable/disable auto mode, as well as optionally set the delay duration and play a sound on input.

If only two arguments are supplied, the second argument will be interpreted as a delay duration. To specify a sound with no delay, the delay argument must be set to 0.

**!Important note:** If auto mode is enabled, negative pause values (e.g. `[pause=-1]`) will be interpreted literally, not indefinitely. This allows crafting a fully automated experience that retains temporary pauses where appropriate. To change this behavior, see `vngen_set_auto_type`.

Recommended for use in mouse, keyboard, or gamepad input events.

#### Example:

```
vngen_do_auto(toggle);  
vngen_do_auto(true, 3, snd_input);
```

## 6.7.2. The "vngen\_is\_auto" Function

### Syntax:

```
vngen_is_auto();
```

Argument	Type	Description
N/A	N/A	No arguments

### Description:

Checks whether auto mode is enabled and returns true or false.

### Example:

```
if (vngen_is_auto()) {  
    draw_text(32, 24, "AUTO");  
}
```

## 6.7.3. The "vngen\_do\_continue" Function

### Syntax:

```
vngen_do_continue([sound]);
```

Argument	Type	Description
[sound]	sound	<i>Optional:</i> A sound to be played when the script is run

### Description:

Skips any ongoing actions in the current event or continues any actions which pause event progression to wait for user input. This script is highly contextual and will both take action or refuse to take action as is appropriate for many different situations built in to VNgen.

Recommended for use in mouse, keyboard, or gamepad input events.

### Example:

```
vngen_do_continue(snd_continue);
```

## 6.7.4. The "vngen\_do\_pause" Function

### Syntax:

```
vngen_do_pause(toggle, [sound]);
```

Argument	Type	Description
toggle	boolean/macro	Enables, disables, or toggles pausing VNgen
[sound]	sound	<i>Optional:</i> A sound to be played when



the script is run

## Description:

Toggles or explicitly enables/disables pausing VNgen. While paused, all ongoing actions are suspended and functions such as `vnngen_do_continue` are ignored. (Buttons are an exception to this rule, so they can be used to create pause menus.)

Recommended for use in mouse, keyboard, or gamepad input events.

### Example:

```
vnngen_do_pause(toggle, snd_input);  
vnngen_do_pause(false);
```

## 6.7.5. The "vnngen\_is\_paused" Function

### Syntax:

```
vnngen_is_paused();
```

Argument	Type	Description
N/A	N/A	No arguments

## Description:

Checks whether VNgen is paused and returns true or false.

### Example:

```
if (vnngen_is_paused()) {  
    draw_text(960, 540, "PAUSED");  
}
```

## 6.7.6. The "vnngen\_do\_ui\_display" Function

### Syntax:

```
vnngen_do_ui_display(toggle, [stop], [sound]);
```

Argument	Type	Description
toggle	boolean/macro	Enables, disables, or toggles visibility of UI elements
[stop]	boolean	<i>Optional:</i> Enables or disables progression while UI is hidden
[sound]	sound	<i>Optional:</i> A sound to be played when the script is run

## Description:

Toggles or explicitly enables/disables visibility of VNgen UI elements. This includes textboxes, text, labels, prompts, options, and buttons, allowing the user to appreciate scene and character art free of obstruction.



Typically, if the UI is hidden when `vnngen_do_continue` is run, it will be unhidden automatically. This behavior can be disabled by setting the 'stop' argument to false, allowing progression without any UI.

If only two arguments are provided, the second argument will be interpreted as 'stop' if true or false, and as 'sound' if a sound asset.

Recommended for use in mouse, keyboard, or gamepad input events.

**Example:**

```
vnngen_do_ui_display(toggle, snd_input);  
vnngen_do_ui_display(false, false, snd_input);
```

### 6.7.7. The "vnngen\_is\_ui\_displayed" Function

**Syntax:**

```
vnngen_is_ui_displayed();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Checks whether UI elements are visible and returns true or false.

**Example:**

```
if (vnngen_is_ui_displayed()) {  
    virtual_key_show(my_key);  
} else {  
    virtual_key_hide(my_key);  
}
```

### 6.7.8. The "vnngen\_count" Function

**Syntax:**

```
vnngen_count(type, [name]);
```

Argument	Type	Description
type	integer/macro	Sets which type of entity to check
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment

**Description:**

Returns the number of entities of the given type that currently exist.

Note that in the case of buttons, results will differ depending on whether the log is currently open or not. If so, only log buttons will be counted, while if not, only non-log buttons will be counted instead.

See [Macros & Keywords](#) for a full list of available entity types. In addition to standard types, this script also supports checking the current speaking character(s) which can be used, for example, to trigger different UI styles depending on whose lines are in focus.

**Example:**

```
draw_text(x, y, "There are currently " + string(vngen_count(vngen_type_char)) + " characters.");
```

## 6.7.9. The "vngen\_exists" Function

**Syntax:**

```
vngen_exists(id, type, [name]);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to check (or keyword 'any' for any)
type	integer/macro	Sets which type of entity to check
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment

**Description:**

Returns true if the input entity exists, or false if it does not.

Note that in the case of buttons, results will differ depending on whether the log is currently open or not. If so, only log buttons will be counted, while if not, only non-log buttons will be counted instead.

See [Macros & Keywords](#) for a full list of available entity types. In addition to standard types, this script also supports checking the current speaking character(s) which can be used, for example, to trigger different UI styles depending on whose lines are in focus.

**Example:**

```
if (vngen_exists("John Doe", vngen_type_speaker)) {  
    vngen_textbox_modify_style("textbox", c_red, c_red, c_yellow, c_yellow, 1, 0.5);  
}
```

## 6.7.10. The "vngen\_goto" Function

**Syntax:**

```
vngen_goto(event, [object], [perform]);
```

Argument	Type	Description
event	integer/string	The Quantum event to go to (numeric ID or label)
[object]	object	<i>Optional:</i> The object containing the target event (default: self)
[perform]	boolean	<i>Optional:</i> Enables or disables performing skipped events (default:

---

enabled)

## Description:

Jumps from the current Quantum event to a different one, creating the containing object and destroying the current object, if necessary. This is a very important script, as its most common uses include restoring saved locations and navigating to different story branches from dialog options.

By default, running this script will perform skipped events in the background before arriving at the target event. This is necessary so that the target event will appear as if the user had viewed each event normally. To disable this behavior and perform **only** the target event, the optional 'perform' argument may be set to 'false'. Note that an object **must** be supplied in order to disable performing skipped events.

**Tip:** The keyword 'self' can be supplied in place of an object to target an event in the current object.

Note that **`vnngen_goto` is not an action**, and therefore requires being wrapped in `vnngen_script_execute` to behave as one. The exception to this is when used in conjunction with `vnngen_get_option` in a dialog option segment: as `vnngen_get_option` is itself not an action either (but behaves like one), in this case `vnngen_goto` should be used normally.

### Example:

```
vnngen_goto(5, obj_script);
vnngen_goto("my_event");
vnngen_goto(10, self, false);
```

## 6.7.11. The "vnngen\_goto\_unread" Function

### Syntax:

```
vnngen_goto_unread([perform]);
```

Argument	Type	Description
[perform]	boolean	<i>Optional:</i> Enables or disables performing skipped events (default: enabled)

## Description:

Jumps from the current Quantum event to the first *unread* event, stopping at any option blocks which are encountered along the way. This is a very important script, as its most common uses include quickly navigating to different story branches from dialog options.

By default, running this script will perform skipped events in the background before arriving at the target event. This is necessary so that the target event will appear as if the user had viewed each event normally. To disable this behavior and perform **only** the target event, the optional 'perform' argument may be set to 'false'.

Note that **`vnngen_goto_unread` is not an action**, and therefore requires being wrapped in `vnngen_script_execute` to behave as one. The exception to this is when used in conjunction with `vnngen_get_option` in a dialog option segment: as `vnngen_get_option` is itself not an action either (but behaves like one), in this case `vnngen_goto_unread` should be used normally.

### Example:



```
vnngen_goto_unread();  
vnngen_goto_unread(false);
```

## 6.7.12. The "vnngen\_instance\_change" Function

### Syntax:

```
vnngen_instance_change(object, [perform]);
```

Argument	Type	Description
object	object	The object to create a new instance of
[perform]	boolean	<i>Optional:</i> Enables or disables performing Destroy and Create Events on replace

### Description:

Safely clears the current VNgen object from memory while creating a new object in its place. While it is not required for the new object to contain VNgen script, it is required to run this script to change an object that does.

By default, this script will perform the Create and Destroy events of the new object. To disable this behavior, set the 'perform' argument to false.

To end a VNgen object without replacing it with a new one, see `vnngen_object_clear`.

### Example:

```
vnngen_instance_change(obj_new);  
vnngen_instance_change(obj_new, false);
```

## 6.7.13. The "vnngen\_room\_goto" Function

### Syntax:

```
vnngen_room_goto(room, [event, object], [perform]);
```

Argument	Type	Description
room	room	The room index to go to
event	integer/string	<i>Optional:</i> The VNgen event ID or label to jump to in the new room
object	object	<i>Optional:</i> The VNgen object to jump to in the new room
perform	boolean	<i>Optional:</i> Enables or disables performing skipped events

### Description:

Ends the current room and activates the specified room index in its place, safely clearing VNgen data from memory in the process to prevent errors and memory leaks. Optionally can also jump to a specific VNgen object and event in the new room.

This function should **always** be used in place of the standard `room_goto` function when one or more VNgen objects exist in the current room.

Note that **`vnngen_room_goto` is not an action**, and therefore requires being wrapped in `vnngen_script_execute` to behave as one. The exception to this is when used in conjunction with `vnngen_get_option` in a dialog option segment: as `vnngen_get_option` is itself not an action either (but behaves like one), in this case `vnngen_room_goto` should be used normally.

**Example:**

```
vnngen_room_goto(my_room);  
vnngen_room_goto(my_room, "choice_good", obj_vnngen);  
vnngen_room_goto(my_room, "choice_bad", obj_vnngen, false);
```

## 6.7.14. The "vnngen\_set\_auto\_type" Function

**Syntax:**

```
vnngen_set_auto_type(type);
```

Argument	Type	Description
type	integer	Sets the behavior for infinite pause values when auto mode is enabled

**Description:**

As of version 1.0.8, VNgen treats indefinite pause values (e.g. `[pause=-1]`) as literal durations when auto mode is enabled. This behavior can be reverted to match 1.0.7 and older versions, so that indefinite pauses are always respected, and will require running `vnngen_do_continue` even in auto mode.

To use the new behavior, set the auto type to 0 (enabled by default).

To use the old behavior, set the auto type to 1.

**Example:**

```
vnngen_set_auto_type(1);
```

## 6.7.15. The "vnngen\_set\_cursor" Function

**Syntax:**

```
vnngen_set_cursor(default, pointer);
```

Argument	Type	Description
default	sprite/constant	Sprite or cursor state to assign as default cursor
pointer	sprite/constant	Sprite or cursor state to assign as pointer cursor

**Description:**

VNgen uses cursor states for visual feedback on desktop platforms when hovering clickable entities. By default, the system cursors are used, but some users may wish to replace system cursors with custom images of their own. This script assigns a pair of sprites to be displayed in place of system cursors for both default and hovering states. It is also possible to assign built-in cursor states as the default and pointer cursor, or hide cursors entirely with the constant `cr_none`.

**Example:**

```
vnngen_set_cursor(spr_cur_default, spr_cur_pointer);
```

## 6.7.16. The "vnngen\_set\_renderlevel" Function

**Syntax:**

```
vnngen_set_renderlevel(level);
```

Argument	Type	Description
level	integer	sets the global renderlevel ( <i>default 0</i> )

**Description:**

VNgen uses multiple levels of rendering to accommodate the capabilities of different platforms. While VNgen will automatically apply the best renderlevel for most devices, in some cases you may wish to set renderlevel manually for better performance or compatibility.

VNgen currently supports three different renderlevels to choose from:

- 0, or full rendering (default)
- 1, or reduced rendering
- 2, or legacy rendering

Reduced rendering will prioritize compatibility while maintaining mostly the same features as full rendering, while legacy rendering is aimed at low-end devices and HTML5 which do not properly support some advanced effects or require additional performance. Note, however, that legacy rendering is not faster in all cases.

**Example:**

```
vnngen_set_renderlevel(2);
```

## 6.7.17. The "vnngen\_set\_scale" Function

**Syntax:**

```
vnngen_set_scale(view);
```

Argument	Type	Description
view	integer (0-7)	The viewport to scale ( <i>optional, use keyword 'none' to disable scaling</i> )

**Description:**

Enables automatically adjusting the input viewport to fill the screen at any resolution and aspect ratio, maintaining

width and adding height as necessary to eliminate black bars. The input viewport must be configured in the GameMaker Studio room editor and enabled as visible before it can be used with this script.

**Important note:** Setting a new viewport to scale reinitializes scaling to match the input viewport. As such, `vnngen_set_scale` should only be run in events where it is executed once, such as Create—NOT Step or Draw!

How each on-screen element responds to changes in display scale depends on the scaling modes and coordinates set in `vnngen_*_create` actions. With proper configuration, it is possible to create visual novels that fit appropriately on any size or shape of screen.

Note that this script provides only basic scaling support which may not be sufficient for projects which mix visual novels with other types of gameplay. Advanced users are recommended to replace this script with scaling functions tailored to their specific projects.

**Example:**

```
vnngen_set_scale(0);
```

## 6.7.18. The "vnngen\_set\_shader\_float" Function

**Syntax:**

```
vnngen_set_shader_float(id, type, [name], uniform, value);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to modify
type	integer/macro	Sets which type of entity to modify
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment
uniform	string	Sets the shader uniform to modify, as a string
value	real	Sets the value to assign to the shader uniform

**Description:**

When writing a shader, it can be useful to pass information from GML to the shader. Normally, shader variables are completely separate from GML and can't be modified externally, however it is possible to pass in variables called **uniforms** which are then referenced in shaders, allowing GML and shaders to communicate.

VNgen uses its own implementation of shaders on a per-entity basis. With this script, it is possible to assign a floating-point (real) value to a shader uniform defined within the shader itself.

Note that VNgen automatically passes in 5 default input values for handling fade transitions, global time, mouse and view coordinates, and the parent entity dimensions. These values are read-only and cannot be modified, but are very useful for designing shaders themselves.

They are:

```
uniform float in_Amount; //Transition percentage (0-1)
uniform float in_Time; //Seconds active
uniform vec2 in_Mouse; //X/Y
uniform vec2 in_Offset; //View X/Y
```

```
uniform vec2 in_Resolution; //Width/Height
```

Note the names and functions of these uniform values when supplying uniform names to be modified.

**Example:**

```
vngen_set_shader_float("bg", vngen_type_scene, "my_uniform", 0.0);
```

## 6.7.19. The "vngen\_set\_shader\_matrix" Function

### Syntax:

```
vngen_set_shader_matrix(id, type, [name], uniform);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to modify
type	integer/macro	Sets which type of entity to modify
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment
uniform	string	Sets the shader uniform to modify, as a string

### Description:

When writing a shader, it can be useful to pass information from GML to the shader. Normally, shader variables are completely separate from GML and can't be modified externally, however it is possible to pass in variables called **uniforms** which are then referenced in shaders, allowing GML and shaders to communicate.

VNgen uses its own implementation of shaders on a per-entity basis. With this script, it is possible to assign the current transform matrix to a shader uniform defined within the shader itself.

Note that VNgen automatically passes in 5 default input values for handling fade transitions, global time, mouse and view coordinates, and the parent entity dimensions. These values are read-only and cannot be modified, but are very useful for designing shaders themselves.

They are:

```
uniform float in_Amount; //Transition percentage (0-1)
uniform float in_Time; //Seconds active
uniform vec2 in_Mouse; //X/Y
uniform vec2 in_Offset; //View X/Y
uniform vec2 in_Resolution; //Width/Height
```

Note the names and functions of these uniform values when supplying uniform names to be modified.

**Example:**

```
vngen_set_shader_matrix("bg", vngen_type_scene, "my_matrix");
```

## 6.7.20. The "vngen\_set\_shader\_sampler" Function

### Syntax:

```
vngen_set_shader_sampler(id, type, [name], uniform, source);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to modify
type	integer/macro	Sets which type of entity to modify
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment
uniform	string	Sets the shader uniform to modify, as a string
source	sprite/string	Sets the sprite or surface to assign to the shader uniform (string required for surface)

### Description:

When writing a shader, it can be useful to pass information from GML to the shader. Normally, shader variables are completely separate from GML and can't be modified externally, however it is possible to pass in variables called **uniforms** which are then referenced in shaders, allowing GML and shaders to communicate.

VNgen uses its own implementation of shaders on a per-entity basis. With this script, it is possible to assign a sprite or surface to a shader uniform defined within the shader itself. While it might seem strange, surfaces must be input as a string, not the associated variable itself, as this is the only way to differentiate between whether the intended source is a sprite or not.

**Important note:** Surfaces passed into shaders as samplers **must** be created and drawn to *before* `vngen_object_draw` is run.

Note that VNgen automatically passes in 5 default input values for handling fade transitions, global time, mouse and view coordinates, and the parent entity dimensions. These values are read-only and cannot be modified, but are very useful for designing shaders themselves.

They are:

```
uniform float in_Amount; //Transition percentage (0-1)
uniform float in_Time; //Seconds active
uniform vec2 in_Mouse; //X/Y
uniform vec2 in_Offset; //View X/Y
uniform vec2 in_Resolution; //Width/Height
```

Note the names and functions of these uniform values when supplying uniform names to be modified.

### Example:

```
vngen_set_shader_sampler("bg", vngen_type_scene, "my_sampler", my_sprite);
vngen_set_shader_sampler("bg", vngen_type_scene, "my_sampler", "my_surf");
```

## 6.7.21. The "vngen\_set\_halign" Function

## Syntax:

```
vnngen_set_halign(id, type, halign);
```

Argument	Type	Description
id	real/string	The ID of the entity to set alignment (or keyword 'all' for all entities)
type	integer/macro	Sets whether to set alignment for text or labels
halign	constant	Sets font alignment to either <code>fa_left</code> , <code>fa_center</code> , or <code>fa_right</code> (default <code>fa_left</code> )

## Description:

Sets the horizontal alignment of the specified text or label entity. It is also possible to permanently set alignment of all entities of either type by supplying the 'all' keyword in place of an ID. In this case, alignment will also apply to new entities created in the future.

See [Macros & Keywords](#) for a full list of available entity types. This script supports only the entity types `vnngen_type_text` and `vnngen_type_label`.

### Example:

```
vnngen_set_halign("text", vnngen_type_text, fa_center);  
vnngen_set_halign(all, vnngen_type_label, fa_left);
```

**Note:** VNgen alignment properties exist independently of GameMaker's `draw_set_halign` property. Changing VNgen alignment will not affect GameMaker's alignment and vice versa.

## 6.7.22. The "vnngen\_get\_halign" Function

### Syntax:

```
vnngen_get_halign(id, type);
```

Argument	Type	Description
id	real/string	The ID of the entity to check alignment (use keyword 'all' for all entities)
type	integer/macro	Sets whether to check alignment for text or labels

### Description:

Returns the horizontal alignment of the specified text or label entity.

### Example:

```
if (vnngen_get_halign("text", vnngen_type_text) != vnngen_get_halign(all, vnngen_type_text)) {
```

```
vngen_set_halign("text", vngen_type_text, vngen_get_halign(all, vngen_type_text)
);
}
```

### 6.7.23. The "vngen\_set\_lineheight" Function

#### Syntax:

```
vngen_set_lineheight(id, type, multiplier);
```

Argument	Type	Description
id	real/string	The ID of the entity to set lineheight (or keyword 'all' for all entities)
type	integer/macro	Sets whether to set lineheight for text or labels
multiplier	real	Sets the global lineheight as a multiplier of font size

#### Description:

VNgen uses a global lineheight multiplier to calculate the vertical space between lines of text (relative to the height of the current font, in pixels). This script changes the lineheight multiplier for all forms of text drawn by VNgen, or overrides it for a specific element, where a value of 1 equals no extra separation between lines.

Lineheight is set to 1.5 by default.

**Tip:** If changes to lineheight aren't persisting, try running this script with `vngen_script_execute_ext!`

#### Example:

```
vngen_set_lineheight(all, vngen_type_text, 2);
vngen_set_lineheight("label_names", vngen_type_label, 1);
```

### 6.7.24. The "vngen\_get\_lineheight" Function

#### Syntax:

```
vngen_get_lineheight(id, type);
```

Argument	Type	Description
id	real/string	The ID of the entity to check lineheight (use keyword 'all' for all entities)
type	integer/macro	Sets whether to check lineheight for text or labels

#### Description:

Returns the vertical line separation multiplier of the specified text or label entity.

#### Example:



```
if (vngen_get_lineheight("text", vngen_type_text) != vngen_get_lineheight(all, vngen_type_text)) {  
    vngen_set_lineheight("text", vngen_type_text, vngen_get_lineheight(all, vngen_type_text));  
}
```

## 6.7.25. The "vngen\_set\_speed" Function

### Syntax:

```
vngen_set_speed([id], speed);
```

Argument	Type	Description
[id]	real/string	<i>Optional:</i> ID of the text entity to apply speed to ( <i>use keyword 'all' for all text entities</i> )
speed	real	Sets the speed at which VNgen text increments, in characters per-second

### Description:

Sets the rate at which text increments when the typewriter effect is enabled. This value is in characters per-second (CPS), and will automatically be adapted to framerate and delta time. Speeds of 0 or less will disable the typewriter effect entirely.

Speed is set to 25 by default.

Note that while this value sets the initial speed, speed can also be adjusted inline with [ speed ] markup. Markup values act as a multiplier of the initial speed rather than an explicit CPS value.

**Tip:** If changes to speed aren't persisting, try running this script with `vngen_script_execute_ext!`

### Example:

```
vngen_set_speed(50);  
vngen_set_speed("text", 50);
```

## 6.7.26. The "vngen\_get\_speed" Function

### Syntax:

```
vngen_get_speed(id);
```

Argument	Type	Description
id	real/string	The ID of the text entity to check speed ( <i>use keyword 'all' for all text entities</i> )

### Description:

Returns the typewriter effect speed of the specified text entity, as a value of characters per-second (CPS).

### Example:

```
if (vngen_get_speed("text") != vngen_get_speed(all)) {  
    vngen_set_speed("text", vngen_get_speed(all));  
}
```

### 6.7.27. The "vngen\_set\_vol" Function

#### Syntax:

```
vngen_set_vol(type, vol, [fade]);
```

Argument	Type	Description
type	integer/macro/keyword	Sets the type of sound to modify ( <i>use keyword 'all' for all types</i> )
volume	real (0-1)	Sets the global sound volume, where 1 is default and 0 is silent
[fade]	real	<i>Optional:</i> Sets the length of time to fade between volumes, in seconds

#### Description:

Sets the global volume multiplier for VNgen sound, music, voice, and other audio. Note that this volume is an *offset*, not an *override*, and is relative to any volume set in audio actions themselves. Also note that not all sounds support fading between volume levels.

See [Macros & Keywords](#) for a full list of available audio types.

#### Example:

```
vngen_set_vol(all, 0.5);  
vngen_set_vol(audio_type_music, 0.8, 3);
```

### 6.7.28. The "vngen\_get\_vol" Function

#### Syntax:

```
vngen_get_vol(type);
```

Argument	Type	Description
type	integer/macro	Sets the type of sound to get volume for

#### Description:

Returns the global volume multiplier for all VNgen sound, music, voice, and other audio. Note that this volume is an *offset*, not an *override*, and is relative to any volume set in audio actions themselves.

See [Macros & Keywords](#) for a full list of available audio types.

#### Example:

```
var vol_sfx = vngen_get_vol(audio_type_sound);
```

## 6.8. File Functions

For very long projects, it's imperative to offer users the ability to save their progress to return to later. As such, VNgen includes basic file functions for saving, loading, and deleting stored data. These functions are not meant as a replacement for custom save functions, but rather to serve as templates for advanced users while providing a serviceable solution for users who only need to save and load VNgen data without any additional game data of their own.

As of version 1.0.8, it is also possible to save and load data from a `ds_map`, allowing users to read and write VNgen save data in their own files, combined with any custom data and save format their unique projects demand.

In this section we'll examine available file functions which will allow you to save, load, and delete your progress through VNgen projects.

### 6.8.1. The "vngen\_file\_save" Function

#### Syntax:

```
vngen_file_save(filename, [encrypt]);
```

Argument	Type	Description
filename	string	The complete filename of the target save file, including path and extension
[encrypt]	boolean	<i>Optional:</i> Enables or disables encrypting save data to prevent external modification (Default: enabled)

#### Description:

Saves the current room, VNgen object, room coordinates, Quantum event, option choices, style inheritance, paragraph alignment, and language data to a file with optional encryption (enabled by default). The resulting file will be located in the current working directory (on Windows, `%localappdata%` by default). This script will also return 'true' or 'false' depending on whether the save operation was successful.

Note that due to GameMaker Studio being sandboxed, save files cannot be written to an external location on the hard drive by default. In GameMaker Studio 2, this behavior can be disabled in Game Settings, allowing files to be written anywhere (applies to desktop platforms only).

To save data without automatically writing it to a file, see the `vngen_file_save_map` function.

#### Example:

```
vngen_file_save(working_directory + "save.dat");  
vngen_file_save(working_directory + "save.dat", false);
```

### 6.8.2. The "vngen\_file\_save\_map" Function

#### Syntax:

```
vngen_file_save_map();
```

Argument	Type	Description
N/A	N/A	No arguments

### Description:

Returns a `ds_map` containing basic VNgen progress and engine data, which can then be written to save files along with other custom game data.

If no VNgen data could be found in the current room, `undefined` will be returned instead.

Also note that **data structures are volatile** and may become inaccessible if the variables referencing them are redefined. As such, it is highly recommended to always destroy the data structure when it no longer needed.

See the `vngen_file_load_map` function for a list of values included in the generated `ds_map`.

### Example:

```
var save_data = vngen_file_save_map();
var file = file_text_open_write(working_directory + "save.dat");
file_text_write_string(file, ds_map_write(save_data));
file_text_close(file);

ds_map_destroy(save_data);
```

## 6.8.3. The "vngen\_file\_load" Function

### Syntax:

```
vngen_file_load(filename);
```

Argument	Type	Description
filename	string	The complete filename of the target save file, including path and extension

### Description:

Reads basic VNgen data from the target file (if it exists), then recreates the stored object with all saved properties. Encryption is automatically detected and decrypted if found.

To load data without automatically applying it, see the `vngen_file_load_map` function.

### Example:

```
vngen_file_load(working_directory + "save.dat");
```

## 6.8.4. The "vngen\_file\_load\_map" Function

### Syntax:



```
vnngen_file_load_map(filename/ds_map);
```

Argument	Type	Description
filename/ds_map	string/ds_map	The complete filename of the target save file, including path and extension, OR a ds_map generated by vnngen_file_save_map

### Description:

Reads basic VNgen data from the target file or ds\_map (if it exists), then returns a ds\_map populated with all saved properties, or undefined if saved properties do not exist. Encryption is automatically detected and decrypted if found.

The returned map will contain the following key/value pairs which can be accessed using either the built-in ds\_map\_find\_value function or the ds\_map[? "key"] accessor:

Key	Value Description
"room"	The active room index (not name) when the save file was created
"object"	The object index (not name) of the saved VNgen object
"x"	The saved object horizontal room position
"y"	The saved object vertical room position
"event"	The saved VNgen event index (not label)
"halign_text"	The saved <b>global</b> VNgen text alignment setting
"halign_label"	The saved <b>global</b> VNgen label alignment setting
"lineheight_text"	The saved <b>global</b> VNgen text lineheight setting
"lineheight_label"	The saved <b>global</b> VNgen label lineheight setting
"	"
"auto_type"	The saved auto mode behavior setting
"lang_text"	The saved VNgen text language setting
"lang_audio"	The saved VNgen audio language setting

Note that not all saved data is user-accessible and will be restored immediately upon loading, such as global option choices, text and label style data, and event read logs. **Any existing data of this kind will be overwritten.** Other data (listed above) must be applied manually to take effect.

Also note that **data structures are volatile** and may become inaccessible if the variables referencing them are redefined. As such, it is highly recommended to always destroy the data structure when it no longer needed.

To automatically apply all loaded data, see the vnngen\_file\_load function.

### Example:

```
var map_load = vnngen_file_load_map(working_directory + "save.dat");
if (!is_undefined(map_load)) {
    vnngen_room_goto(map_load[? "room"], map_load[? "event"], map_load[? "object"]);

    ds_map_destroy(map_load);
}
```

## 6.8.5. The "vnngen\_file\_delete" Function

### Syntax:

```
vnngen_file_delete(filename);
```

Argument	Type	Description
filename	string	The complete filename of the target save file, including path and extension

### Description:

Physically removes the target file from the hard drive (if it exists) and returns true or false depending on if the operation was successful.

Note that this function will only delete files previously created by GameMaker Studio.

### Example:

```
if (vnngen_file_delete(working_directory + "save.dat") == false) {  
    show_message("Delete failed!");  
} else {  
    show_message("File deleted.");  
}
```

## 6.9. Language Functions

VNgen includes built-in support for creating content in multiple languages. This is achieved by setting **language flags** for text and audio which are then reflected in individual text and audio actions, assigning them to a particular language. In this way, it is possible to have separate active languages for text and audio.

Using language flags is entirely optional, and it's even possible to implement multi-language support of your own instead. For example, you may wish to store text and audio in data structures for each language which are simply swapped in to memory as needed. For more basic purposes, however, VNgen's built-in solution is an easy and effective way to reach whole new audiences with your visual novels.

In this section we'll examine available language functions which can be used to read and write your own language flags.

**Tip:** Blank text when drawing non-English characters? Check out the [Additional Language Setup](#) guide!

### 6.9.1. The "vnngen\_set\_lang" Function

#### Syntax:

```
vnngen_set_lang(lang, [type]);
```

Argument	Type	Description
lang	real/string	The language value to set
[type]	integer/macro	Sets whether to apply language value to text or audio ( <i>Optional, use no argument for both</i> )

## Description:

Sets the active language for language-sensitive actions such as text, labels, options, and voice. If a language-sensitive action specifies a certain language which does not match the language flag set by this script, it will not be performed.

Language flags are arbitrary and can be stored as reals or strings so long as they match the language set in language-sensitive actions. If actions do not specify a language, the active language will be ignored.

Text and audio language are set separately as `vnngen_type_text` and `vnngen_type_audio`, but can be assigned simultaneously by omitting the 'type' argument instead.

See [Macros & Keywords](#) for a full list of available entity types.

### Example:

```
vnngen_set_lang("en-US", vnngen_type_text);
```

## 6.9.2. The "vnngen\_get\_lang" Function

### Syntax:

```
vnngen_get_lang(type);
```

Argument	Type	Description
type	integer/macro	Sets whether to return text or audio language value

### Description:

Returns the active language for the input language type. If no language has been set, -1 will be returned instead.

Note that this script is not required to perform built-in text, label, option, or voice actions in different languages, as each provides optional language restrictions already.

See [Macros & Keywords](#) for a full list of available entity types.

### Example:

```
if (vnngen_get_lang(vnngen_type_text) == "en-US") {  
    show_message("Current text language is American English");  
}
```

## 6.10. Property Functions

Oftentimes, when you create a VNgen entity at a certain position, or modify it with a certain rotation and scale, the final values shown on-screen don't match the values specifically set in Q-script. The perspective, global offset, animations, and other factors all contribute to the final values applied to a given entity before being drawn. This entire process happens transparently, and in some cases you may not even be aware VNgen is making adjustments to ensure everything behaves as you expect! However, this becomes a challenge when integrating your own code and assets with VNgen scenes, as VNgen elements may not appear with the exact coordinates, dimensions, and rotation they were created at.

To solve this problem, each frame, VNgen records the final properties for major entities where they can be retrieved using `vngen_get_*` scripts. While seemingly insignificant, using just this simple data it's possible to integrate your own visual elements or even create entirely new forms of interactivity for your projects. For example, try combining these functions with `mouse_check_region` to detect when the user has clicked on an object in the scene!

### 6.10.1. The "vngen\_set\_prop" Function

#### Syntax:

```
vngen_set_prop(id, type, [name], prop, val);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to check
type	integer/macro	Sets which type of entity to check
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment
prop	enum	The property to check, as an enumerator
val	any	The value to assign to the property

#### Description:

Modifies the value of the specified property of the given entity, if it exists.

Properties are written with the syntax "prop.\_property". A full list of valid properties can be found by editing `sys_vngen_config`. However, note that not all entities possess all available properties.

See [Macros & Keywords](#) for a full list of available entity types.

**Warning:** This script modifies internal VNgen behavior and is intended for advanced users only. Improper usage can cause errors or crashes. Use at your own risk!

#### Example:

```
var john_x = vngen_get_prop("John Doe", vngen_type_char, prop._x);  
vngen_set_prop("John Doe", vngen_type_char, prop._x, john_x + 5);
```

### 6.10.2. The "vngen\_get\_prop" Function

#### Syntax:

```
vngen_get_prop(id, type, [name], prop);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to check
type	integer/macro	Sets which type of entity to check
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment



---

prop	enum	The property to check, as an enumerator
------	------	---

**Description:**

Returns the value of the specified property of the given entity, or 'undefined' if the entity or property does not exist.

Properties are written with the syntax "prop.\_property". A full list of valid properties can be found by editing `sys_vngen_config`. However, note that not all entities possess all available properties.

See [Macros & Keywords](#) for a full list of available entity types.

**Example:**

```
var john_x = vngen_get_prop("John Doe", vngen_type_char, prop._x);
```

### 6.10.3. The "vngen\_get\_index" Function

**Syntax:**

```
vngen_get_index(id, type, [name]);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to check
type	integer/macro	Sets which type of entity to check
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment

**Description:**

Returns the numerical index of the specified entity of the given ID and type, or 'undefined' if the entity or type does not exist.

See [Macros & Keywords](#) for a full list of available entity types. In addition to standard types, this script also supports checking the current speaking character(s) which will return the index of the *text* element containing the speaking character's dialog.

**Example:**

```
var index = vngen_get_index("John Doe", vngen_type_char);
```

### 6.10.4. The "vngen\_get\_struct" Function

**Syntax:**

```
vngen_get_struct(id, type, [name]);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to check
type	integer/macro	Sets which type of entity to check



---

[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment
--------	--------	---

### Description:

Returns the data structure index of the specified entity of the given ID and type, or 'undefined' if the entity or type does not exist.

See [Macros & Keywords](#) for a full list of available entity types.

#### Example:

```
var ds_struct = vngen_get_struct("John Doe", vngen_type_char);
```

## 6.10.5. The "vngen\_get\_width" Function

### Syntax:

```
vngen_get_width(id, type, [name]);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to check
type	integer/macro	Sets which type of entity to check
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment

### Description:

Returns the absolute width of the specified entity, taking into account any active modifications and animations.

See [Macros & Keywords](#) for a full list of available entity types.

#### Example:

```
if mouse_region(vngen_get_x("John", 2), vngen_get_y("John", 2), vngen_get_width("John", 2), vngen_get_height("John", 2)) {  
    if (mouse_check_button_pressed(mb_left)) {  
        show_message: "John: Hey! You clicked me!";  
    }  
}
```

## 6.10.6. The "vngen\_get\_height" Function

### Syntax:

```
vngen_get_height(id, type, [name]);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to check
type	integer/macro	Sets which type of entity to check
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment

## Description:

Returns the absolute height of the specified entity, taking into account any active modifications and animations.

See [Macros & Keywords](#) for a full list of available entity types.

### Example:

```
if mouse_region(vngen_get_x("John", 2), vngen_get_y("John", 2), vngen_get_width("John", 2), vngen_get_height("John", 2)) {  
    if (mouse_check_button_pressed(mb_left)) {  
        show_message: "John: Hey! You clicked me!";  
    }  
}
```

## 6.10.7. The "vngen\_get\_x" Function

### Syntax:

```
vngen_get_x(id, type, [name]);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to check
type	integer/macro	Sets which type of entity to check
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment

## Description:

Returns the absolute horizontal position of the specified entity, taking into account any active modifications and animations.

See [Macros & Keywords](#) for a full list of available entity types.

### Example:

```
if mouse_region(vngen_get_x("John", 2), vngen_get_y("John", 2), vngen_get_width("John", 2), vngen_get_height("John", 2)) {  
    if (mouse_check_button_pressed(mb_left)) {  
        show_message: "John: Hey! You clicked me!";  
    }  
}
```

## 6.10.8. The "vngen\_get\_y" Function

### Syntax:

```
vngen_get_y(id, type, [name]);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to check



---

type [name]	integer/macro string	Sets which type of entity to check <i>Optional:</i> Sets the character name to check, if entity is an attachment
----------------	-------------------------	---

### Description:

Returns the absolute vertical position of the specified entity, taking into account any active modifications and animations.

See [Macros & Keywords](#) for a full list of available entity types.

#### Example:

```
if mouse_region(vngen_get_x("John", 2), vngen_get_y("John", 2), vngen_get_width("John", 2), vngen_get_height("John", 2)) {  
    if (mouse_check_button_pressed(mb_left)) {  
        show_message: "John: Hey! You clicked me!";  
    }  
}
```

## 6.10.9. The "vngen\_get\_xscale" Function

### Syntax:

```
vngen_get_xscale(id, type, [name]);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to check
type	integer/macro	Sets which type of entity to check
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment

### Description:

Returns the absolute horizontal scale multiplier of the specified entity, taking into account any active modifications and animations.

See [Macros & Keywords](#) for a full list of available entity types.

#### Example:

```
var xscale = vngen_get_xscale("bg", 1);  
var yscale = vngen_get_yscale("bg", 1);  
var rot = vngen_get_rot("bg", 1);  
draw_sprite_ext(my_sprite, -1, x, y, xscale, yscale, rot, c_white, 1);
```

## 6.10.10. The "vngen\_get\_yscale" Function

### Syntax:

```
vngen_get_yscale(id, type, [name]);
```



Argument	Type	Description
id	real/string	The ID of the specific entity to check
type	integer/macro	Sets which type of entity to check
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment

### Description:

Returns the absolute vertical scale multiplier of the specified entity, taking into account any active modifications and animations.

See [Macros & Keywords](#) for a full list of available entity types.

#### Example:

```
var xscale = vngen_get_xscale("bg", 1);  
var yscale = vngen_get_yscale("bg", 1);  
var rot = vngen_get_rot("bg", 1);  
draw_sprite_ext(my_sprite, -1, x, y, xscale, yscale, rot, c_white, 1);
```

## 6.10.11. The "vngen\_get\_rot" Function

### Syntax:

```
vngen_get_rot(id, type, [name]);
```

Argument	Type	Description
id	real/string	The ID of the specific entity to check
type	integer/macro	Sets which type of entity to check
[name]	string	<i>Optional:</i> Sets the character name to check, if entity is an attachment

### Description:

Returns the absolute rotation of the specified entity in degrees, taking into account any active modifications and animations.

See [Macros & Keywords](#) for a full list of available entity types.

#### Example:

```
var xscale = vngen_get_xscale("bg", 1);  
var yscale = vngen_get_yscale("bg", 1);  
var rot = vngen_get_rot("bg", 1);  
draw_sprite_ext(my_sprite, -1, x, y, xscale, yscale, rot, c_white, 1);
```

## 6.11. The Backlog

In addition to Quantum-based functions, VNgen includes a second category of functions intended to be used in a separate object from the main script, which then records text and voice actions in a list for reviewing later. This visual history is called the **backlog**.

Using a backlog object is optional, and adding data to it is automatic. Nevertheless, while little developer action is



*required*, VNgen provides as many options as possible for creating and customizing the backlog for those who want it.

In this section we'll examine available backlog functions as well as a variety of navigation methods designed to suit any platform.

### 6.11.1. Backlog Buttons

In addition to standard on-screen buttons, VNgen provides a separate set of button functions specifically for the backlog. Both button types are functionally identical and even share a single set of IDs, but with one key difference: log buttons are only enabled while the backlog is open, and other buttons are only enabled while the backlog is closed.

In this section we'll examine available backlog button functions, which can be used to scroll the log, close it, or even act as a full pause menu.

#### 6.11.1.1. The "vnngen\_log\_button\_create" Function

##### Syntax:

```
vnngen_log_button_create(id, text, sprite, spr_hover, spr_select, x, y, z, font, color, [ease]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new button
text	string	The button text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a button background
spr_hover	sprite/macro	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_active	sprite/macro	The sprite to display as a background when selected (or keyword 'none' for default)
x	real	The horizontal position to display the button, relative to the GUI layer
y	real	The vertical position to display the button, relative to the GUI layer
z	real	The drawing depth and list order of the button, relative to other log buttons only
font	font	The font to draw text in, where fnt_default is default
color	color	The color to draw text in, where c_white is default
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the button animations

##### Description:

Creates a new button which will be displayed while the backlog is open. The chosen button ID will be recorded as the selected choice until cleared with `vngen_get_log_button` or `vngen_log_button_clear`. Buttons can be selected by mouse/touch, or by other input devices with `vngen_log_button_nav` and `vngen_log_button_select`.

Unlike other entities, button z-index determines not just drawing depth, but also navigation order. Buttons with lower z-index will be considered lower in the list of buttons. If z-index is equal between two or more buttons, navigation order will automatically be set by *reverse* creation order, but this may or may not be visually correct. Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

Note that although this script resembles an action, it is not and should not be executed within VNgen events. Instead, log buttons should ideally be created in the Create Event of a dedicated backlog object.

See [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vngen_log_button_create("up", "", spr_up, spr_uhover, spr_uactive, display_get_gui_w  
idth(), display_get_gui_height()*0.5, -1, fnt_default, c_white);  
vngen_log_button_create("dn", "", spr_dn, spr_dhover, spr_dactive, display_get_gui_w  
idth(), display_get_gui_height()*0.5, -2, fnt_default, c_white);
```

### 6.11.1.2. The "vngen\_log\_button\_create\_ext" Function

**Syntax:**

```
vngen_log_button_create_ext(id, text, sprite, spr_hover, spr_select, xorig, yorig, x  
, y, z, tx, ty, scaling, font, color, snd_hover, snd_select, ease);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new button
text	string	The button text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a button background
spr_hover	sprite/macro	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_active	sprite/macro	The sprite to display as a background when selected (or keyword 'none' for default)
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the button, relative to the GUI layer
y	real	The vertical position to display the button, relative to the GUI layer
z	real	The drawing depth and list order of the button, relative to other log

tx	real/macro	buttons only The horizontal position to display text, relative to the button sprite top-left corner ( <i>or keyword 'auto' for center</i> )
ty	real/macro	The vertical position to display text, relative to the button sprite top-left corner ( <i>or keyword 'auto' for center</i> )
scaling	integer/macro	Sets the automatic scaling mode for the button
font	font	The font to draw text in, where <code>fnt_default</code> is default
color	color	The color to draw text in, where <code>c_white</code> is default
snd_hover	sound	The sound to play when the button is hovered ( <i>Optional, use keyword 'none' for none</i> )
snd_select	sound	The sound to play when the button is selected ( <i>Optional, use keyword 'none' for none</i> )
ease	integer/macro	Sets the ease override for the button animations

## Description:

Creates a new button with extra options which will be displayed while the backlog is open. The chosen button ID will be recorded as the selected choice until cleared with `vngen_get_log_button` or `vngen_log_button_clear`. Buttons can be selected by mouse/touch, or by other input devices with `vngen_log_button_nav` and `vngen_log_button_select`.

Unlike other entities, button z-index determines not just drawing depth, but also navigation order. Buttons with lower z-index will be considered lower in the list of buttons. If z-index is equal between two or more buttons, navigation order will automatically be set by *reverse* creation order, but this may or may not be visually correct. Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

Note that although this script resembles an action, it is not and should not be executed within VNgen events. Instead, log buttons should ideally be created in the Create Event of a dedicated backlog object.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_log_button_create_ext("up", "", spr_up, spr_uhover, spr_uactive, orig_right, orig_bottom, display_get_gui_width(), display_get_gui_height()*0.5, -1, 0, 0, scale_none, fnt_default, c_white, snd_hover, snd_select, ease_sin_in_out);
vngen_log_button_create_ext("dn", "", spr_dn, spr_dhover, spr_dactive, orig_right, orig_top, display_get_gui_width(), display_get_gui_height()*0.5, -2, 0, 0, scale_none, fnt_default, c_white, snd_hover, snd_select, ease_sin_in_out);
```

## 6.11.1.3. The "vngen\_log\_button\_create\_transformed" Function



## Syntax:

```
vngen_log_button_create_transformed(id, text, sprite, spr_hover, spr_select, x, y, z, font, color, hov_x, hov_y, hov_scale, hov_color, sel_x, sel_y, sel_scale, sel_color, sel_duration, [ease]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new button
text	string	The button text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a button background
spr_hover	sprite/macro	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_active	sprite/macro	The sprite to display as a background when selected (or keyword 'none' for default)
x	real	The horizontal position to display the button, relative to the GUI layer
y	real	The vertical position to display the button, relative to the GUI layer
z	real	The drawing depth and list order of the button, relative to other log buttons only
font	font	The font to draw text in, where fnt_default is default
color	color	The color to draw text in, where c_white is default
hov_x	real	The horizontal offset to shift the button when hovered
hov_y	real	The vertical offset to shift the button when hovered
hov_scale	real	The scale multiplier to shift the button when hovered
hov_color	color	The color to draw text in when hovered
sel_x	real	The horizontal offset to shift the button when selected
sel_y	real	The vertical offset to shift the button when selected
sel_scale	real	The scale multiplier to shift the button when selected
sel_color	color	The color to draw text in when selected
sel_duration	real	Sets the duration of hover/select animations, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the button animations

## Description:

Creates a new button with hover and select transformations which will be displayed while the backlog is open. The chosen button ID will be recorded as the selected choice until cleared with `vngen_get_log_button` or

`vnngen_log_button_clear`. Buttons can be selected by mouse/touch, or by other input devices with `vnngen_log_button_nav` and `vnngen_log_button_select`.

Unlike other entities, button z-index determines not just drawing depth, but also navigation order. Buttons with lower z-index will be considered lower in the list of buttons. If z-index is equal between two or more buttons, navigation order will automatically be set by *reverse* creation order, but this may or may not be visually correct. Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

Note that although this script resembles an action, it is not and should not be executed within VNgen events. Instead, log buttons should ideally be created in the Create Event of a dedicated backlog object.

See [Macros & Keywords](#) for a list of available ease modes.

#### Example:

```
vnngen_log_button_create_transformed("up", "", spr_up, spr_uhover, spr_uactive, display_get_gui_width(), display_get_gui_height()*0.5, -1, fnt_default, c_white, 32, 0, 1.25, c_black, 0, 0, 1, c_yellow, 0.1);
vnngen_log_button_create_transformed("dn", "", spr_dn, spr_dhover, spr_dactive, display_get_gui_width(), display_get_gui_height()*0.5, -2, fnt_default, c_white, 32, 0, 1.25, c_black, 0, 0, 1, c_yellow, 0.1);
```

### 6.11.1.4. The "vnngen\_log\_button\_create\_ext\_transformed" Function

#### Syntax:

```
vnngen_log_button_create_ext_transformed(id, text, sprite, spr_hover, spr_select, xorig, yorig, x, y, z, tx, ty, scaling, font, color, hov_x, hov_y, hov_xscale, hov_yscale, hov_color, sel_x, sel_y, sel_xscale, sel_yscale, sel_color, sel_duration, snd_hover, snd_select, ease);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new button
text	string	The button text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a button background
spr_hover	sprite/macro	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_active	sprite/macro	The sprite to display as a background when selected (or keyword 'none' for default)
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the button, relative to the GUI layer
y	real	The vertical position to display the button, relative to the GUI layer
z	real	The drawing depth and list order of

		the button, relative to other log buttons only
tx	real/macro	The horizontal position to display text, relative to the button sprite top-left corner ( <i>or keyword 'auto' for center</i> )
ty	real/macro	The vertical position to display text, relative to the button sprite top-left corner ( <i>or keyword 'auto' for center</i> )
scaling	integer/macro	Sets the automatic scaling mode for the button
font	font	The font to draw text in, where <code>fnt_default</code> is default
color	color	The color to draw text in, where <code>c_white</code> is default
hov_x	real	The horizontal offset to shift the button when hovered
hov_y	real	The vertical offset to shift the button when hovered
hov_xscale	real	The horizontal scale multiplier to shift the button when hovered
hov_yscale	real	The vertical scale multiplier to shift the button when hovered
hov_color	color	The color to draw text in when hovered
sel_x	real	The horizontal offset to shift the button when selected
sel_y	real	The vertical offset to shift the button when selected
sel_xscale	real	The horizontal scale multiplier to shift the button when selected
sel_yscale	real	The vertical scale multiplier to shift the button when selected
sel_color	color	The color to draw text in when selected
sel_duration	real	Sets the duration of hover/select animations, in seconds
snd_hover	sound	The sound to play when the button is hovered ( <i>Optional, use keyword 'none' for none</i> )
snd_select	sound	The sound to play when the button is selected ( <i>Optional, use keyword 'none' for none</i> )
ease	integer/macro	Sets the ease override for the button animations

## Description:

Creates a new button with extra options and hover and select transformations which will be displayed while the backlog is open. The chosen button ID will be recorded as the selected choice until cleared with `vnngen_get_log_button` or `vnngen_log_button_clear`. Buttons can be selected by mouse/touch, or by other input devices with `vnngen_log_button_nav` and `vnngen_log_button_select`.

Unlike other entities, button z-index determines not just drawing depth, but also navigation order. Buttons with lower z-index will be considered lower in the list of buttons. If z-index is equal between two or more buttons, navigation order will automatically be set by *reverse* creation order, but this may or may not be visually correct.

Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

Note that although this script resembles an action, it is not and should not be executed within VNgen events. Instead, log buttons should ideally be created in the Create Event of a dedicated backlog object.

See [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vnngen_log_button_create_transformed_ext("up", "", spr_up, spr_uhover, spr_uactive, orig_right, orig_bottom, display_get_gui_width(), display_get_gui_height()*0.5, -1, 0, 0, fnt_default, c_white, 32, 0, 1.25, 1.25, c_black, 0, 0, 1, 1, c_yellow, 0.1, snd_hover, snd_select, ease_sin_in_out);
vnngen_log_button_create_transformed_ext("dn", "", spr_dn, spr_dhover, spr_dactive, orig_right, orig_top, display_get_gui_width(), display_get_gui_height()*0.5, -2, 0, 0, fnt_default, c_white, 32, 0, 1.25, 1.25, c_black, 0, 0, 1, 1, c_yellow, 0.1, snd_hover, snd_select, ease_sin_in_out);
```

### 6.11.1.5. The "vnngen\_log\_button\_destroy" Function

**Syntax:**

```
vnngen_log_button_destroy(id);
```

Argument	Type	Description
id	real/string	The unique ID of the button to destroy (or keyword 'all' for all buttons)

**Description:**

Removes a previously-created on-screen button from the backlog.

**Example:**

```
vnngen_log_button_destroy("up");
vnngen_log_button_destroy("dn");
```

### 6.11.1.6. The "vnngen\_log\_button\_clear" Function

**Syntax:**

```
vnngen_log_button_clear();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Manually clears log button **results** (not data) to prevent repeat execution of button-based triggers. As this functionality is also handled by `vnngen_get_log_button`, this script should not typically need to be run except in special cases.

This script is not an action.

**Example:**

```
if (vnngen_get_log_button(false) == "up") {  
    vnngen_log_nav(-1);  
    vnngen_log_button_clear();  
}
```

### 6.11.1.7. The "vnngen\_get\_log\_button" Function

**Syntax:**

```
vnngen_get_log_button([clear]);
```

Argument	Type	Description
[clear]	boolean	<i>Optional:</i> Enables or disables clearing the result from memory after returning it (enabled by default)

**Description:**

When run, this script will return the result of the most recently-selected log button, which is stored as the ID of the selected button. If buttons exist but none has been selected, -1 will be returned instead.

This script is intended to be used in conditional statements such as 'if' and 'switch' to take action based on the selected button. In order to prevent this conditional statement from repeatedly executing code, button results will be cleared from memory after being returned. To disable this behavior, it is possible to set the 'clear' argument to 'false', in which case button results can be cleared later with `vnngen_log_button_clear`.

**Example:**

```
if (vnngen_get_log_button(false) == "up") {  
    //Action  
}  
switch (vnngen_get_log_button()) {  
    case "up": //Action; break;  
    case "dn": //Action; break;  
}
```

### 6.11.1.8. The "vnngen\_do\_log\_button\_nav" Function

**Syntax:**

```
vnngen_do_log_button_nav(amount);
```

Argument	Type	Description
amount	integer	The number of buttons to scroll

**Description:**

Navigates through a linear list of log buttons, if any exist. The amount of buttons scrolled is arbitrary, where

negative values scroll up and positive values scroll down. If a hover sound has been assigned in `vngen_log_button_create_ext`, it will be played upon running this script. If the input amount exceeds the size of the buttons menu, the selection will loop back around to the other side.

Note that navigation order is determined by the button z-index, and therefore improper configuration of options can result in unexpected navigation with this script.

This script is not an action, and is intended to be run in keyboard and gamepad input events. Mouse and touch support are built-in and do not require `vngen_do_log_button_nav` for navigation.

**Note:** This script is not to be confused with `vngen_do_log_nav`, which navigates the backlog itself. This script navigates backlog buttons as a menu instead.

**Example:**

```
//Scroll up
vngen_do_log_button_nav(-1);
//Scroll down
vngen_do_log_button_nav(1);
```

### 6.11.1.9. The "vngen\_is\_log\_button\_hovered" Function

**Syntax:**

```
vngen_is_log_button_hovered(id);
```

Argument	Type	Description
id	real/string	The log button ID to check

**Description:**

Checks whether or not the specified log button is hovered (either by mouse/touch or `vngen_do_log_button_nav`) and returns 'true' or 'false'.

**Example:**

```
if (vngen_is_log_button_hovered("up")) {
    effect_create_above(ef_ellipse, mouse_x, mouse_y, 2, c_white);
}
```

### 6.11.1.10. The "vngen\_do\_log\_button\_select" Function

**Syntax:**

```
vngen_do_log_button_select([id]);
```

Argument	Type	Description
id	real/string	<i>Optional:</i> The button ID to select (default: current highlighted button)

**Description:**

Selects the currently-highlighted log button, if any, and records its ID. If a selection sound has been assigned in `vngen_log_button_create_ext`, it will be played upon running this script.

If a button ID is supplied, it will be selected directly, ignoring navigation.

This script is not an action, and is intended to be run in keyboard and gamepad input events. Mouse and touch support are built-in and do not require `vngen_do_log_button_select` to make selections.

**Example:**

```
if (keyboard_check_released(vk_enter)) {
    vngen_do_log_button_select();
}

if (keyboard_check_released(vk_up)) {
    vngen_do_log_button_select("up");
}
```

### 6.11.1.11. The "vngen\_is\_log\_button\_selected" Function

**Syntax:**

```
vngen_is_log_button_selected(id);
```

Argument	Type	Description
<code>id</code>	real/string	The log button ID to check

**Description:**

Checks whether or not the specified log button is selected (either by mouse/touch or `vngen_do_log_button_select`) and returns 'true' or 'false'.

Note that this script is different from `vngen_get_log_button` in that it will return 'true' while the button is *held down*, not just when it is released and activated.

**Example:**

```
if (vngen_is_log_button_selected("up")) {
    effect_create_above(ef_ellipse, mouse_x, mouse_y, 2, c_white);
}
```

## 6.11.2. Backlog Input

VNgen's backlog supports a variety of input methods designed to suit any platform, including keyboard, mouse, touch, and on-screen button support. While setting up each of these control methods is up to you, VNgen provides a set of standardized functions they can all use to navigate the backlog.

In this section we'll examine available input functions and how to use them in your own backlog objects.

### 6.11.2.1. The "vngen\_do\_log\_display" Function



## Syntax:

```
vnngen_do_log_display(toggle, [sound]);
```

Argument	Type	Description
toggle	boolean/macro	Enables, disables, or toggles visibility of the backlog
[sound]	sound	<i>Optional:</i> A sound to be played when the script is run

## Description:

Toggles or explicitly enables/disables visibility of the VNgen backlog. While the backlog is visible, VNgen itself will pause and ignore most background input.

Recommended for use in mouse, keyboard, or gamepad input events.

### Example:

```
vnngen_do_log_display(toggle);  
vnngen_do_log_display(true, snd_input);
```

## 6.11.2.2. The "vnngen\_is\_log\_displayed" Function

### Syntax:

```
vnngen_is_log_displayed();
```

Argument	Type	Description
N/A	N/A	No arguments

### Description:

Checks whether or not the backlog is visible and returns 'true' or 'false'.

### Example:

```
if (vnngen_is_log_displayed()) {  
    vnngen_set_vol(audio_type_music, 0.5, 1);  
}
```

## 6.11.2.3. The "vnngen\_do\_log\_nav" Function

### Syntax:

```
vnngen_do_log_nav(amount);
```

Argument	Type	Description
amount	integer	The number of backlog entries to scroll

### Description:



Scrolls the backlog by the specified amount, if visible.

The amount of log entries scrolled is arbitrary, where negative values scroll up and positive values scroll down. With this script it is possible to, for example, use the arrow keys to scroll one entry at a time and the Home/End keys to jump directly to the start or end of the log.

Intended to be used in keyboard, mouse, or gamepad input events.

**Example:**

```
vngen_do_log_nav(-1); //Up  
vngen_do_log_nav(1); //Down
```

### 6.11.2.4. The "vngen\_do\_log\_nav\_touch" Function

**Syntax:**

```
vngen_do_log_nav_touch();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Listens for touch input while the log is visible and scrolls when swipe gestures are detected.

Intended to be run in the Step Event.

Note that while this script is intended for touch input, it is also possible to click and drag the mouse to scroll the backlog while this script is active.

**Example:**

```
vngen_do_log_nav_touch();
```

### 6.11.2.5. The "vngen\_do\_log\_play" Function

**Syntax:**

```
vngen_do_log_play();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Plays all audio associated with the highlighted log entry, if any. By default, only audio played with `vngen_play_voice` will be included in the backlog. If multiple sounds are recorded for the highlighted event, they will be played back in sequence.

Intended for use in keyboard, mouse, or gamepad input events.

Note that if audio icons are supplied in `vnngen_log_draw`, it is also possible to click/tap the audio icon for playback.

**Example:**

```
vnngen_do_log_play();
```

### 6.11.2.6. The "vnngen\_is\_log\_playing" Function

**Syntax:**

```
vnngen_is_log_playing();
```

Argument	Type	Description
N/A	N/A	No arguments

**Description:**

Checks whether or not backlog voice audio is playing and returns 'true' or 'false'.

**Example:**

```
if (!vnngen_is_log_playing()) {  
    vnngen_log_nav(-1);  
}
```

### 6.11.3. The "vnngen\_log\_init" Function

**Syntax:**

```
vnngen_log_init(size);
```

Argument	Type	Description
size	integer	Sets the maximum number of logged events to store in memory

**Description:**

Initializes all necessary functions for running the VNgen backlog in the current object. Must be run in the Create Event.

To contain memory usage, the backlog is limited to a certain quantity of events. Once the maximum number of logged entries has been reached, the oldest will be deleted. This also prevents lists from growing so large that they become cumbersome to navigate. As such, recommended values would be either 25 or 50 even though most devices can handle much more.

**Example:**

```
vngen_log_init(25);
```

## 6.11.4. The "vngen\_log\_add" Function

### Syntax:

```
vngen_log_add(index, data, [name], [font, col1, col2, col3, col4, shadow, outline, h  
align, lineheight]);
```

Argument	Type	Description
index	integer/macro	The numeric ID of the event being logged (labels not accepted) ( <i>or use 'auto' to add to system queue</i> )
data	string/sound	The text or audio data to be added to the backlog
[name]	string	<i>Optional:</i> The speaking character name associated with the text
[font]	font	<i>Optional:</i> The original font of the logged text
[col1]	color	<i>Optional:</i> The original top-left gradient color of the text
[col2]	color	<i>Optional:</i> The original top-right gradient color of the text
[col3]	color	<i>Optional:</i> The original bottom-right gradient color of the text
[col4]	color	<i>Optional:</i> The original bottom-left gradient color of the text
[shadow]	color/macro	<i>Optional:</i> The original shadow color of the text ( <i>or keyword 'none' for none</i> )
[outline]	color/macro	<i>Optional:</i> The original outline color of the text ( <i>or keyword 'none' for none</i> )
[halign]	keyword	<i>Optional:</i> The paragraph alignment of the text (fa_left, fa_center, fa_right)
[lineheight]	real	<i>Optional:</i> The vertical separation between lines of text, as a multiplier

**Note:** Text style properties are treated as a group. While including them is optional, they must either be included together or not at all.

### Description:

Adds a new entry to the backlog at the specified event index. Input data can be either text or audio. If existing backlog data is found for the target event, the new data will be merged in as a single entry.

Data can also be added to the system queue for backlogged events by supplying the 'auto' keyword instead of a numerical event index. In this case, data will not be applied to the backlog immediately, but will automatically apply to the latest entry into the backlog when the current event is completed.

This script is triggered automatically by text, option, and voice functions and typically does not need to be run manually.

### Example:

```
vngen_log_add(5, "John Doe: Hello, world!");  
vngen_log_add(5, snd_hello);  
vngen_log_add(auto, "Hello, world!", "John Doe", fnt_Arial, c_white, c_white, c_gray  
, c_gray, c_black, c_black, fa_left, 1.5);
```

### 6.11.5. The "vngen\_log\_draw" Function

#### Syntax:

```
vngen_log_draw(spr_background, spr_divider, spr_paused, spr_playing, sep, linebreak,  
font, color);
```

Argument	Type	Description
spr_background	sprite/macro	Sprite to draw as a fullscreen background while the log is open (or keyword 'none' for none)
spr_divider	sprite/macro	Sprite to draw as a divider between logged entries (or keyword 'none' for none)
spr_paused	sprite/macro	Sprite to draw as paused audio icon (or keyword 'none' for none)
spr_playing	sprite/macro	Sprite to draw as playing audio icon (or keyword 'none' for none)
sep	real	Distance in pixels between log entries and dividers (if any)
linebreak	real	Width in pixels before text is wrapped into a new line
font	font/macro	Override font to draw logged text in (or keyword 'inherit' for logged style)
color	color/macro	Override color to draw logged text in (or keyword 'inherit' for logged style)

#### Description:

Draws the backlog with the given stylization, when visible. By default, the log is not visible and must be toggled with `vngen_toggle_log` to be displayed even while this script is running. This script is intended to be run in the Draw GUI Event.

Backlog elements are designed to be responsive and can adapt to real-time changes in resolution and aspect ratio, provided the linebreak setting allows it. For this reason, it is highly recommended to use a linebreak setting relative to the dimensions of the GUI layer itself. The GUI layer can then be scaled to match the display resolution with the GameMaker Studio function `display_set_gui_maximise`.

**Note:** As `vngen_set_scale` already scales the GUI along with other elements, if this script is active it is not necessary to run `display_set_gui_maximise` manually.

#### Example:

```
vngen_log_draw(spr_backlog, spr_divider, spr_audio_off, spr_audio_on, 64, 1200, fnt_Arial, c_white);
```

```
vngen_log_draw(spr_backlog, spr_divider, spr_audio_off, spr_audio_on, 64, display_get_gui_width()*0.75, inherit, inherit);
```

### 6.11.6. The "vngen\_log\_clear" Function

#### Syntax:

```
vngen_log_clear([destroy]);
```

Argument	Type	Description
[destroy]	boolean	<i>Optional:</i> Enables or disables destroying the running object when complete (Default: disabled)

#### Description:

The VNgen backlog stores a complex system of data in memory which must be removed when the object is no longer needed to prevent memory leaks. This script frees occupied memory so that the running object can be safely destroyed.

Although it is recommended *not* to destroy the backlog, it is possible to continue using other VNgen functions after the backlog has been purged from memory. In this case, to resume using the backlog, `vngen_log_init` must be run again.

Note that this script may be overridden by any VNgen function which attempts to add new data to the backlog, in which case it will be recreated automatically.

#### Example:

```
vngen_log_clear(false);
```

### 6.11.7. The "vngen\_log\_count" Function

#### Syntax:

```
vngen_log_count();
```

Argument	Type	Description
N/A	N/A	No arguments

#### Description:

Returns the current number of events stored in the backlog, or 0 if it does not exist.

#### Example:

```
show_message("The backlog currently contains " + string(vngen_log_count()) + " events.");
```

### 6.11.8. The "vngen\_log\_get\_index" Function



### Syntax:

```
vngen_log_get_index(entry);
```

Argument	Type	Description
entry	integer/macro	The on-screen entry to get historical index from, starting from 0 (or keyword 'previous' for most recent)

### Description:

This script might seem confusing at first, but its function is actually quite simple. VNgen essentially stores backlog entries with two indices: 1) the order in which entries have appeared *for the entire session*, and 2) the order in which entries appear on the screen. Typically, the log will be limited to just 25 or 50 entries before purging old data, at which point these two indices will become desynchronized.

While you shouldn't often need it, at times it can be helpful to know the historical index of an on-screen backlog entry. These entries are ordered from zero to max size, from top to bottom. By inputting the on-screen index of a backlog entry in `vngen_log_get_index`, the historical index will be returned and can be used, for example, to add new log data at a certain position even without knowing exactly how many entries the user has viewed in total throughout their entire session.

### Example:

```
var index_latest = vngen_log_get_index(previous);
```

## 6.12. Object Functions

VNgen operates in a hierarchy that goes something like this:

Global functions > local object functions > Quantum events > Quantum actions

While most of your time with VNgen will be spent in the last two categories, operating events and actions is only possible with the proper initial setup.

In this section we'll examine primary local object functions used for initializing and drawing VNgen content, as well as safely clearing VNgen from memory when the running object needs to be destroyed. All three of these processes are highly automated, but the first two must be executed in every VNgen object manually before it can be used. With this basic object setup completed, you'll be off to creating great content in no time.

### 6.12.1. The "vngen\_object\_init" Function

#### Syntax:

```
vngen_object_init();
```

Argument	Type	Description
N/A	N/A	No arguments

#### Description:

Initializes all the necessary functions for running VNgen in the current object.

Must be run in the Create Event of each object containing VNgen code.

**Example:**

```
vngen_object_init();
```

## 6.12.2. The "vngen\_object\_draw" Function

**Syntax:**

```
vngen_object_draw(x, y, [highlight]);
```

Argument	Type	Description
x	real	The global horizontal offset for displaying all VNgen entities
y	real	The global vertical offset for displaying all VNgen entities
[highlight]	boolean	<i>Optional:</i> Enables or disables highlighting effects on speaking characters and text links

**Description:**

Draws all VNgen entities at the specified location. Must be run in the Draw Event of each object containing VNgen code.

**Note:** It is also possible to draw VNgen in the Draw GUI Event, however this is not recommended.

While each VNgen entity possesses its own unique coordinates, this script can be used to also set global coordinates which determine where in the room VNgen appears as a whole. This is extremely useful when combining visual novels with other genres, as you may wish for your VNgen elements to appear where other game elements are physically located in the room. Another option is to draw VNgen in the GUI Event as opposed to the normal Draw Event, in which case the global offset can be set to 0, 0?no individual action coordinates must be altered to switch between drawing processes.

Additionally, VNgen characters are drawn with built-in fade/highlight support to emphasize who is currently speaking. If the name of a speaker set in `vngen_text_*` functions matches that of an active character, that specific character will be highlighted while all others are dimmed. This behavior is optional, however, and must be enabled or disabled in this script to function.

**Example:**

```
vngen_object_draw(camera_get_view_x(view_camera[0]), camera_get_view_y(view_camera[0]), true);
```

## 6.12.3. The "vngen\_object\_clear" Function

**Syntax:**

```
vngen_object_clear([destroy]);
```

Argument	Type	Description
----------	------	-------------



---

[destroy]	boolean	<i>Optional:</i> Enables or disables destroying the running object when complete (Default: disabled)
-----------	---------	--

### Description:

Each VNgen object stores a complex system of data in memory which must be removed when the object is no longer needed to prevent memory leaks. This script frees occupied memory so that the running object can be safely destroyed.

Although it is recommended to always destroy a VNgen object after clearing it, it is possible to continue using the object for other purposes after VNgen has been purged from memory. In this case, to resume using VNgen functions, `vnngen_object_init` must be run again.

To destroy a VNgen object while creating a new one in its place, see `vnngen_instance_change`.

This script is run automatically by any VNgen function which requires destroying the current object (e.g. `vnngen_goto` when jumping to a different object) and therefore does not typically need to be executed manually.

### Example:

```
vnngen_object_clear(true);
```

## 6.13. Events

Events are the fundamental building blocks of projects made in VNgen. Not to be confused with the events used by GameMaker Studio, VNgen uses an event/action system built on the [Quantum framework](#) by XGASOFT to form a timeline of operations comprising your visual novel. In this system, one event remains active until all of its actions are complete, at which point the current event will deactivate and the next event will be activated in its place. This process repeats itself until all events have been executed or the Q-script is terminated prematurely by the user.

All Q-script events are intended to be written in GameMaker Studio's Step Event. In this section, we'll examine available event functions and their proper execution order in Q-script.

### 6.13.1. The "vnngen\_event\_set\_target" Function

#### Syntax:

```
vnngen_event_set_target();
```

Argument	Type	Description
N/A	N/A	No arguments

#### Description:

Initializes a segment of code containing Quantum events and actions. Once this script has been run, as many events as desired can be executed after it. It is also required to run the `vnngen_event_reset_target` script after the last event in the sequence.

#### Example:



```
vngen_event_set_target();  
  
if vngen_event() {  
    //Actions  
}  
  
vngen_event_reset_target();
```

## 6.13.2. The "vngen\_event" Function

### Syntax:

```
vngen_event([pause], [noskip], [label]);
```

Argument	Type	Description
[pause]	real	<i>Optional:</i> Sets a delay in seconds before the event is executed
[noskip]	boolean	<i>Optional:</i> Enables or disables responding to user input for the duration of the event
[label]	string	<i>Optional:</i> Sets a string label for the event which can be used in place of a numeric ID

### Description:

Initializes a segment of code containing Quantum actions. Once this script has been run, as many actions as desired can be executed inside it, provided no two actions of the same type are applied to the same entity. By default, all actions contained in the event will be executed simultaneously. Once all actions are complete, the event will deactivate and the following event (if any) will be activated in its place.

For the sake of performance, `vngen_event` is intended to be used within an 'if' statement. This guarantees that actions within an event will only be executed while the event is active.

While no arguments are required to create an event, there are three optional parameters which can be used to customize the event's behavior. Unlike other functions, these parameters can be input in any order, however be aware that numeric values will always be interpreted as 'pause' first, then 'noskip'.

**Tip:** Although 'noskip' may be a boolean value, because 'true' and 'false' can be interpreted as 1 and 0, respectively, there is no other way to differentiate between 'pause' and 'noskip' than their input order.

If no pause value is supplied, the event will begin execution immediately upon activation. Otherwise, execution will be delayed by the given number of seconds unless skipped by running `vngen_continue`. It is also possible to set the event pause to -1, delaying the event indefinitely until `vngen_continue` is run. In either case, only the event pause will be skipped and not the event itself.

**Important note:** If auto mode is enabled, negative pause values will be interpreted literally, not indefinitely. This allows crafting a fully automated experience that retains temporary pauses where appropriate. To change this behavior, see `vngen_set_auto_type`.

Both event pauses and event actions can be prevented from being skipped by setting the 'noskip' value to 'true'. If 'noskip' mode is enabled, `vngen_continue` will be ignored for the duration of the current event.

Naturally, this poses a problem for features like indefinite pauses and text actions which require user input to continue, so in these instances the event will automatically progress on its own.

The third and final optional argument is the event label. Every event is automatically assigned a numeric ID which is used to determine execution order and to manage unique behaviors. However, determining a specific event's ID can be cumbersome. Typically this isn't a problem, as VNgen handles event IDs for you, but in some cases you may wish to have a more memorable way to identify particular events than by their numeric IDs. Labels act as string identifiers to complement internal numeric IDs, so that when you refer to an event by its label, the corresponding numeric ID is chosen instead. Most events will not require labels at all, but for those that do, labels are a very powerful feature.

Note that it is required to run the `vnngen_event_set_target` script before the first event and `vnngen_event_reset_target` script after the last event in the sequence.

**Example:**

```
vnngen_event_set_target();

if vnngen_event() {
    //Actions
}

if vnngen_event(1, true, "my_event") {
    //Actions
}

vnngen_event_reset_target();
```

### 6.13.3. The "vnngen\_event\_pause" Function

**Syntax:**

```
vnngen_event_pause(pause);
```

Argument	Type	Description
pause	real	Sets a delay in seconds between actions within an event

**Description:**

Despite its name, `vnngen_event_pause` is an action. By default, all actions within an event are executed simultaneously, but there may be times when it is best for actions to perform asynchronously instead. This script inserts a delay between actions, so that all actions following `vnngen_event_pause` will not be executed until after the delay is complete.

Note that this script is intended for creating *asynchronous* actions and not *consecutive* ones. In general, *consecutive* actions should be separated into multiple events, as they may not behave as expected if they are simply delayed until after previous actions in the same event are complete.

**Example:**

```
if vnngen_event() {
    //Action
    vnngen_event_pause(1);
}
```

```
//Delayed action  
}
```

#### 6.13.4. The "vngen\_event\_reset\_target" Function

##### Syntax:

```
vngen_event_reset_target();
```

Argument	Type	Description
N/A	N/A	No arguments

##### Description:

Finalizes a segment of code containing Quantum events and actions. Once this script has been run, no more events can be executed after it for the current frame. It is also required to run the `vngen_event_set_target` script prior to the first event in the sequence.

##### Example:

```
vngen_event_set_target();  
  
if vngen_event() {  
    //Actions  
}  
  
vngen_event_reset_target();
```

#### 6.13.5. The "vngen\_event\_count" Function

##### Syntax:

```
vngen_event_count();
```

Argument	Type	Description
N/A	N/A	No arguments

##### Description:

Returns the total number of events in the active object, or 0 if VNgen is not initialized in the active object.

##### Example:

```
show_message("The active object currently contains " + string(vngen_event_count()) +  
    " events.");
```

#### 6.13.6. The "vngen\_event\_get\_index" Function

##### Syntax:

```
vngen_event_get_index([label]);
```

---

Argument	Type	Description
[label]	string	<i>Optional:</i> The label of the event to get index of

**Description:**

Returns the numeric ID of the current active event, or -1 if VNgen is not initialized in the running object.

Can also be used to return the numeric index of an event by label, if any is supplied.

**Example:**

```
if (vngen_event_get_index() != -1) {  
    draw_text(x, y, string(vngen_event_get_index()));  
}
```

### 6.13.7. The "vngen\_event\_get\_label" Function

**Syntax:**

```
vngen_event_get_label([index]);
```

Argument	Type	Description
[index]	integer	<i>Optional:</i> The numeric index of the event to get label of

**Description:**

Returns the string label of the current active event, or "" if VNgen is not initialized in the running object.

Can also be used to return the label of an event by numeric index, if any is supplied.

**Example:**

```
if (vngen_event_get_label() != "") {  
    draw_text(x, y, vngen_event_get_label());  
}
```

### 6.13.8. The "vngen\_event\_get\_read" Function

**Syntax:**

```
vngen_event_get_read([index]);
```

Argument	Type	Description
[index]	integer	<i>Optional:</i> The numeric index of the event to get label of

**Description:**

Checks whether the specified event has been previously viewed or not and returns 'true' or 'false'. If no event is specified, the current event will be checked instead. If the event cannot be found, 'false' will be returned.

This can be used, for example, to create a “skip read” button for visual novels where users are expected to reread common sections while navigating through branching story paths.

**Example:**

```
if (vnngen_event_get_read(vnngen_event_get_index() + 1)) {  
    vnngen_do_continue();  
}
```

## 6.14. Actions

By now you should have a working understanding of the fundamentals of VNgen. You’ve got the toolbelt?now it’s time to start putting tools in it!

In the sections to follow we’ll examine the various actions available for use in Quantum events. It might seem like a lot at first glance, but as already mentioned, VNgen is highly standardized and many actions share behaviors across the board, making them quick and easy to learn.

## 6.15. Perspective Actions

VNgen uses a built-in ‘camera’ to display scenes with simulated depth and perspective. This global perspective is always enabled, but is set to neutral values by default so the scene appears flat. Like a real camera, VNgen perspective can be adjusted for position, offset (or angle), zoom, rotation, and parallax strength (or FOV). Changes to perspective affect all non-UI entities, such as scenes, characters, and emotes. UI elements such as textboxes, text, and options are not affected by perspective.

Note that individual entities’ properties can affect perspective as well. Specifically, z-index determines whether an entity appears nearer or farther from the camera compared to other entities of the same type. These properties operate at a ratio of 1:100, where a 100-point change in z-index equals a 1x change in perspective ‘distance’. This can be used to either exaggerate or counteract the appearance of distance as desired for entities using these modifications.

In this section we’ll examine available perspective modification and animation functions. It may seem complex at first, and manipulating perspective is entirely optional, but learning to use it effectively is recommended as it will enable the creation of highly dynamic and engaging scenes.

**Note:** VNgen’s perspective exists independently of GameMaker’s built-in viewports and cameras, however it can be linked to them. See GameMaker Studio documentation for available viewport and camera properties.

### 6.15.1. The "vnngen\_perspective\_modify\_pos" Function

**Syntax:**

```
vnngen_perspective_modify_pos(x, y, xoffset, yoffset, zoom, rot, strength, duration,  
[ease]);
```

Argument	Type	Description
x	real	The horizontal perspective position, where 0 is center
y	real	The vertical perspective position,



xoffset	real	where 0 is center The horizontal perspective offset, or 'angle', where 0 is center
yoffset	real	The vertical perspective offset, or 'angle', where 0 is center
zoom	real	The perspective zoom multiplier, where 1 is neutral
rot	real	The perspective rotation, in degrees
strength	real	The parallax strength multiplier, or 'FOV', where 1 is default and 0 is no parallax
duration	real	Sets the length of the perspective transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode for the transition animation

**Note:** Zoom is capped at a minimum of 0.005. Lower values will be clamped to this value.

## Description:

Applies a new position, orientation, and focal point to the global perspective while transitioning from previous properties over the input duration.

All modifications made with this script are permanent and will persist until another modification is performed.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vnngen_event() {  
    vnngen_perspective_modify_pos(0, 0, -50, 0, 1, 0, 1, 0.5, ease_quad_in_out);  
}
```

This will center the perspective camera while offsetting the 'angle' 50 pixels to the left, effectively shifting scene elements to the right using parallax.

## 6.15.2. The "vnngen\_perspective\_modify\_direct" Function

### Syntax:

```
vnngen_perspective_modify_direct(x, y, xoffset, yoffset, zoom, rot, strength);
```

Argument	Type	Description
x	real	The horizontal perspective position, where 0 is center
y	real	The vertical perspective position, where 0 is center
xoffset	real	The horizontal perspective offset, or 'angle', where 0 is center
yoffset	real	The vertical perspective offset, or

zoom	real	'angle', where 0 is center The perspective zoom multiplier, where 1 is neutral
rot	real	The perspective rotation, in degrees
strength	real	The parallax strength multiplier, or 'FOV', where 1 is default and 0 is no parallax

### Description:

Applies a new position, orientation, and focal point to the global perspective directly, outside the Q-script loop.

All modifications made with this script are permanent and will persist until another modification is performed.

As **this script is not an action**, care should be taken in deciding when and where to execute it. If both `*_modify` and `*_modify_direct` scripts are executed together, whichever is run last will override the other.

### Example:

```
var mx = window_mouse_get_x() - (window_get_width()*0.5);
var my = window_mouse_get_y() - (window_get_height()*0.5);
vngen_perspective_modify_direct(0, 0, mx, my, 1, 0, 1);
```

This will map the perspective angle to the mouse, allowing the user to look around the scene.

## 6.15.3. The "vngen\_perspective\_replace" Function

### Syntax:

```
vngen_perspective_replace(x, y, xoffset, yoffset, zoom, rot, strength, duration, [ease]);
```

Argument	Type	Description
x	real	The horizontal perspective position, where 0 is center
y	real	The vertical perspective position, where 0 is center
xoffset	real	The horizontal perspective offset, or 'angle', where 0 is center
yoffset	real	The vertical perspective offset, or 'angle', where 0 is center
zoom	real	The perspective zoom multiplier, where 1 is neutral
rot	real	The perspective rotation, in degrees
strength	real	The parallax strength multiplier, or 'FOV', where 1 is default and 0 is no parallax
duration	real	Sets the length of the fade transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode for the transition animation

## Description:

Unlike other \*\_replace functions, `vngen_perspective_replace` behaves quite similarly to `vngen_perspective_modify`, but rather than transition the global perspective properties directly, this script *fades* from the previous perspective to a new one.

As with other types of modifications, perspective replacements are permanent and will persist until another replacement is performed.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {  
    vngen_perspective_replace(0, 0, -50, 0, 1, 0, 1, 0.5, true);  
}
```

This will center the perspective camera while fading the 'angle' 50 pixels to the left, effectively shifting scene elements to the right using parallax.

## 6.15.4. The "vngen\_perspective\_anim\_start" Function

### Syntax:

```
vngen_perspective_anim_start(anim, duration, loop, reverse, [ease]);
```

Argument	Type	Description
anim	script	The animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

### Description:

Performs a keyframe animation script on the global perspective with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Animations](#) for included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {  
    vngen_perspective_anim_start(anim_wiggle, 0.5, false, false);  
}
```



### 6.15.5. The "vngen\_perspective\_anim\_stop" Function

#### Syntax:

```
vngen_perspective_anim_stop();
```

Argument	Type	Description
N/A	N/A	No arguments

#### Description:

Stops the active animation being performed on the global perspective, if any. Note that this script only needs to be run to stop *looped* animations.

#### Example:

```
vngen_event() {  
    vngen_perspective_anim_stop();  
}
```

### 6.15.6. The "vngen\_perspective\_shader\_start" Function

#### Syntax:

```
vngen_perspective_shader_start(shader, fade, [ease]);
```

Argument	Type	Description
shader	shader	The shader to perform
fade	real	Sets the length of time to fade shader in, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

#### Description:

Applies a shader with optional fade in transition. Shaders are written externally and can be used to modify the color and position of vertices and/or pixels.

**!Important note:** Some shaders require extra input values to function properly. These values must be set externally with `vngen_set_shader_*` scripts.

See [Shaders](#) for a list of included shaders.

#### Example:

```
vngen_event() {  
    vngen_perspective_shader_start(shade_sepia, 0.5);  
}
```

### 6.15.7. The "vngen\_perspective\_shader\_stop" Function

### Syntax:

```
vngen_perspective_shader_stop(fade, [ease]);
```

Argument	Type	Description
fade	real	Sets the length of time to fade shader out, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

### Description:

Stops the active shader being performed on the global perspective, if any.

#### Example:

```
vngen_event() {  
    vngen_perspective_shader_stop(0.5);  
}
```

## 6.16. Scene Actions

VNgen scenes come in two flavors: **backgrounds** and **foregrounds**. Both exist as sprite resources and appear beneath UI elements such as textboxes and text, but as their names imply, backgrounds appear behind characters while foregrounds appear in front of them.

While a single background is all that's necessary to set the stage for an entire scene, it is recommended to use multiple layers of both types for maximum effect. This becomes especially powerful when combined with perspective functions. In addition, foregrounds are quite useful for scene effects such as fades to black.

In this section we'll examine available scene actions covering five basic operations: create, modify, replace, destroy, and animate.

### 6.16.1. The "vngen\_scene\_create" Function

#### Syntax:

```
vngen_scene_create(id, sprite, x, y, z, repeat, foreground, transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new scene
sprite	sprite	The sprite to draw as a scene
x	real	The horizontal position to display the scene, relative to the global offset
y	real	The vertical position to display the scene, relative to the global offset
z	real	The drawing depth of the scene, relative to other scenes only
repeat	boolean	Enables or disables endlessly tiling the scene



foreground	boolean	Enables or disables drawing the scene as a foreground
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

### Description:

Creates a new scene as either a background (if 'foreground' is 'false') or foreground (if 'foreground' is 'true') which will be displayed until `vnngen_scene_destroy` is run. Multiple scenes can exist simultaneously, however no two scenes may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

VNgen uses texture extrapolation to draw tiled scenes by default. This allows tiled scenes to behave exactly like non-tiled ones, with support for rotation, gradient color blending, and deformations. However, not all platforms correctly support drawing scenes this way. For these platforms, see [vnngen\\_set\\_legacy](#) to enable drawing tiled scenes using the old method. Legacy tiled scenes will still appear tiled, but will lack the features described above.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

### Example:

```
vnngen_event() {  
    vnngen_scene_create("bg", spr_bg, 960, 540, 0, false, false, trans_fade, 2);  
}
```

## 6.16.2. The "vnngen\_scene\_create\_ext" Function

### Syntax:

```
vnngen_scene_create_ext(id, sprite, xorig, yorig, x, y, z, scaling, repeat, foreground,  
d, transition, duration, ease);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new scene
sprite	sprite	The sprite to draw as a scene
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the scene, relative to the global offset
y	real	The vertical position to display the scene, relative to the global offset
z	real	The drawing depth of the scene, relative to other scenes only
scaling	integer/macro	Sets the automatic scaling mode for the scene
repeat	boolean	Enables or disables endlessly tiling the scene

foreground	boolean	Enables or disables drawing the scene as a foreground
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
ease	integer/macro	Sets the ease override for the transition script

### Description:

Creates a new scene with extra options as either a background (if 'foreground' is 'false') or foreground (if 'foreground' is 'true') which will be displayed until `vngen_scene_destroy` is run. Multiple scenes can exist simultaneously, however no two scenes may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

VNgen uses texture extrapolation to draw tiled scenes by default. This allows tiled scenes to behave exactly like non-tiled ones, with support for rotation, gradient color blending, and deformations. However, not all platforms correctly support drawing scenes this way. For these platforms, see [vngen\\_set\\_legacy](#) to enable drawing tiled scenes using the old method. Legacy tiled scenes will still appear tiled, but will lack the features described above.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available origin points, scaling modes, transition animations, and ease modes.

### Example:

```
vngen_event() {
    vngen_scene_create_ext("bg", spr_bg, orig_center, orig_center, 960, 540, 0, scale_none, false, false, trans_fade, 2, ease_sin_in_out);
}
```

## 6.16.3. The "vngen\_scene\_modify\_style" Function

### Syntax:

```
vngen_scene_modify_style(id, col1, col2, col3, col4, alpha, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the scene to modify (or keyword 'all' for all scenes)
col1	color	The top-left gradient color, where <code>c_white</code> is default
col2	color	The top-right gradient color, where <code>c_white</code> is default
col3	color	The bottom-right gradient color, where <code>c_white</code> is default
col4	color	The bottom-left gradient color, where <code>c_white</code> is default
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to

---

perform the transition in

### Description:

Applies a four-color gradient and transparency modifications to the input entity ID. Color values can be input using GameMaker Studio's built-in color constants or functions like VNgen's `make_color_hex_to_rgb`.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vnngen_event() {  
    vnngen_scene_modify_style("bg", c_yellow, c_yellow, c_orange, c_orange, 1, 2);  
}
```

This will apply a warm set of colors to the scene "bg", simulating a sunset.

## 6.16.4. The "vnngen\_scene\_modify\_pos" Function

### Syntax:

```
vnngen_scene_modify_pos(id, x, y, z, scale, rot, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the scene to modify ( <i>or keyword 'all' for all scenes</i> )
x	real	The horizontal position to display the scene, relative to the global offset
y	real	The vertical position to display the scene, relative to the global offset
z	real	The drawing depth of the scene, relative to other scenes only
scale	real	The scene scale multiplier, where 1 is default
rot	real	The scene rotation, in degrees
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

**Note:** The scale *multiplier* here should not be confused with the scaling *mode* available in `*_create_ext` functions. The scale *mode* sets the entity's default scale, while the scale *multiplier* modifies the default scale.

### Description:

Applies a new position, rotation, and scale to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This

script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vngen_event() {  
    vngen_scene_modify_pos("bg", 150, 150, -1, 1.5, 45, 2);  
}
```

## 6.16.5. The "vngen\_scene\_modify\_ext" Function

**Syntax:**

```
vngen_scene_modify_ext(id, x, y, z, xscale, yscale, rot, col1, col2, col3, col4, alpha, duration, ease);
```

Argument	Type	Description
id	real/string	The ID of the scene to modify ( <i>or keyword 'all' for all scenes</i> )
x	real	The horizontal position to display the scene, relative to the global offset
y	real	The vertical position to display the scene, relative to the global offset
z	real	The drawing depth of the scene, relative to other scenes only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The scene rotation, in degrees
col1	color	The top-left gradient color, where c_white is default
col2	color	The top-right gradient color, where c_white is default
col3	color	The bottom-right gradient color, where c_white is default
col4	color	The bottom-left gradient color, where c_white is default
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification transition, in seconds
ease	integer/macro	Sets the ease mode for the modification transition

**Description:**

Combines `vngen_scene_modify_style` and `vngen_scene_modify_pos` with a few extra options, applying a new position, scale, rotation, four-color gradient, and transparency modifications to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vngen_event() {  
    vngen_scene_modify_ext("bg", 0, 0, -1, 1.5, 1.5, 0, c_yellow, c_yellow, c_orange,  
        c_orange, 1, 2, ease_sin_out);  
}
```

This will increase the scene's scale, simulating a zoom while fading to a warm gradient simulating a sunset.

## 6.16.6. The "vngen\_scene\_modify\_direct" Function

**Syntax:**

```
vngen_scene_modify_direct(id, x, y, z, xscale, yscale, rot);
```

Argument	Type	Description
id	real/string	The ID of the scene to modify ( <i>or keyword 'all' for all scenes</i> )
x	real	The horizontal position to display the scene, relative to the global offset
y	real	The vertical position to display the scene, relative to the global offset
z	real	The drawing depth of the scene, relative to other scenes only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The scene rotation, in degrees

**Description:**

Applies a new position, orientation, and scale to the input entity ID directly, outside the Q-script loop.

All modifications made with this script are permanent and will persist until another modification is performed.

As **this script is not an action**, care should be taken in deciding when and where to execute it. If both `*_modify` and `*_modify_direct` scripts are executed together, whichever is run last will override the other.

**Example:**

```
var mx = window_mouse_get_x() - (window_get_width()*0.5);  
var my = window_mouse_get_y() - (window_get_height()*0.5);  
  
if (mouse_check_button(mb_left)) {  
    vngen_scene_modify_direct("bg", mx, my, 0, 1, 1, 0);  
}
```

This will map the scene to the mouse, allowing the user to click and drag the scene to a new position on the screen.

### 6.16.7. The "vngen\_scene\_replace" Function

#### Syntax:

```
vngen_scene_replace(id, sprite, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the scene to replace
sprite	sprite	The new sprite to draw as a scene
duration	real	Sets the duration of the fade transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

#### Description:

Replaces the input entity ID with a new sprite and fades the old sprite to the new one over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

#### Example:

```
if (vngen_event()) {  
    vngen_scene_replace("bg", spr_new, 1);  
}
```

### 6.16.8. The "vngen\_scene\_replace\_ext" Function

#### Syntax:

```
vngen_scene_replace_ext(id, sprite, xorig, yorig, scaling, duration, ease);
```

Argument	Type	Description
id	real/string	The ID of the scene to replace
sprite	sprite	The new sprite to draw as a scene
xorig	real/macro	The new horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The new vertical sprite offset, or origin point, relative to the top-left corner
scaling	integer/macro	Sets the new automatic scaling mode for the scene
duration	real	Sets the duration of the fade transition, in seconds



ease

integer/macro

Sets the ease mode to perform the  
fade transition in**Description:**

Replaces the input entity ID with a new sprite and extra sprite properties and fades the old sprite to the new one over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available origin points, scaling modes, and ease modes.

**Example:**

```
vnngen_event() {  
    vnngen_scene_replace_ext("bg", spr_new, orig_center, orig_center, scale_none, 1, ease_sin_in_out);  
}
```

### 6.16.9. The "vnngen\_scene\_destroy" Function

**Syntax:**

```
vnngen_scene_destroy(id, transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the scene to destroy ( <i>or keyword 'all' for all scenes</i> )
transition	script	Sets the transition animation to perform
duration	real	Sets the length of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

**Description:**

Destroys the input entity ID, freeing its resources from memory. Can also perform an exit transition animation before destroying. Once a given ID has been destroyed, it can be created again with the same ID.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

**Example:**

```
vnngen_event() {  
    vnngen_scene_destroy("bg", trans_fade, 2, ease_sin_in);  
}
```

### 6.16.10. The "vnngen\_scene\_anim\_start" Function

## Syntax:

```
vnngen_scene_anim_start(id, anim, duration, loop, reverse, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the scene to animate (or keyword 'all' for all scenes)
anim	script	The animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

## Description:

Performs a keyframe animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vnngen_event() {  
    vnngen_scene_anim_start("bg", anim_impact, 0.5, false, false);  
}
```

## 6.16.11. The "vnngen\_scene\_anim\_stop" Function

## Syntax:

```
vnngen_scene_anim_stop(id);
```

Argument	Type	Description
id	real/string	The ID of the scene to stop animating (or keyword 'all' for all scenes)

## Description:

Stops the active keyframe animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

### Example:

```
vnngen_event() {  
    vnngen_scene_anim_stop("bg");  
}
```

## 6.16.12. The "vngen\_scene\_deform\_start" Function

### Syntax:

```
vngen_scene_deform_start(id, def, duration, loop, reverse, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the scene to animate (or keyword 'all' for all scenes)
def	script	The deformation animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

### Description:

Performs a keyframe deformation animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {  
    vngen_scene_deform_start("bg", def_wave, 0.5, false, false);  
}
```

## 6.16.13. The "vngen\_scene\_deform\_stop" Function

### Syntax:

```
vngen_scene_deform_stop(id);
```

Argument	Type	Description
id	real/string	The ID of the scene to stop animating (or keyword 'all' for all scenes)

### Description:

Stops the active keyframe deformation animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

### Example:

```
vngen_event() {  
    vngen_scene_deform_stop("bg");  
}
```

```
}
```

### 6.16.14. The "vngen\_scene\_shader\_start" Function

#### Syntax:

```
vngen_scene_shader_start(id, shader, fade, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the scene to apply shader to (or keyword 'all' for all scenes)
shader	shader	The shader to perform
fade	real	Sets the length of time to fade shader in, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

#### Description:

Applies a shader with optional fade in transition. Shaders are written externally and can be used to modify the color and position of vertices and/or pixels.

**Important note:** Some shaders require extra input values to function properly. These values must be set externally with `vngen_set_shader_*` scripts.

See [Shaders](#) for a list of included shaders.

#### Example:

```
vngen_event() {  
    vngen_scene_shader_start("bg", shade_sepia, 0.5);  
}
```

### 6.16.15. The "vngen\_scene\_shader\_stop" Function

#### Syntax:

```
vngen_scene_shader_stop(id, fade, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the scene to stop shader on (or keyword 'all' for all scenes)
fade	real	Sets the length of time to fade shader out, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

#### Description:

Stops the active shader being performed on the specified entity, if any.

#### Example:

```
vngen_event() {  
    vngen_scene_shader_stop("bg", 0.5);  
}
```

## 6.17. Character Actions

Characters are one of the most unique and dynamic components of VNgen. While they share a familiar set of actions with other entity types, unlike other entities, Characters are comprised of many sprites composited together to create a single animated persona. These sprites themselves, however, are divided into three separate categories: the **body**, the **face**, and **attachments**.

Although only a body is required to create a new character, it is recommended to take advantage of all three for the most compelling results. Using separate sprites for faces allows characters to automatically animate in sync with text or audio speech, and attachments provide an easy way to create clothing and body elements which can be animated and manipulated individually in real-time!

In this section we'll examine character actions covering five basic operations: create, modify, replace, destroy, and animate, with attachments to follow in a section of their own.

### 6.17.1. The "vngen\_char\_create" Function

#### Syntax:

```
vngen_char_create(name, spr_body, spr_face_idle, spr_face_talk, x, y, z, face_x, face_y, flip, transition, duration, [ease]);
```

Argument	Type	Description
name	string	The unique identifier to use for the new character
spr_body	sprite	The sprite to draw as a character body
spr_face_idle	sprite	The sprite to draw as an idle face (or keyword 'none' for none)
spr_face_talk	sprite	The sprite to draw as a talking face (or keyword 'none' for none)
x	real	The horizontal position to display the character, relative to the global offset
y	real	The vertical position to display the character, relative to the global offset
z	real	The drawing depth of the character, relative to other characters only
face_x	real	The horizontal position to display the face, relative to the body sprite top-left corner
face_y	real	The vertical position to display the face, relative to the body sprite top-left corner
flip	boolean	Enables or disables horizontally flipping the character
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition

[ ease ]

integer/macro

animation, in seconds

*Optional:* Sets the ease override for the transition script

## Description:

Creates a new character which will be displayed until `vngen_char_destroy` is run. Multiple characters can exist simultaneously, however no two characters may share the same ID. Also note that unlike other entities, character IDs must be strings, not real numbers.

Note that using separate face sprites is optional. If only a body sprite is desired, 'spr\_face\_idle' and 'spr\_face\_talk' can be set to -1 or the keyword 'none' to disable them.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

### Example:

```
vngen_event() {
    vngen_char_create("John Doe", spr_body, spr_face, spr_talking, 0, view_hview[0] -
        900, -1, 384, 128, false, trans_slide_right, 2);
}
```

## 6.17.2. The "vngen\_char\_create\_ext" Function

### Syntax:

```
vngen_char_create_ext(name, spr_body, spr_face_idle, spr_face_talk, xorig, yorig, x,
    y, z, face_x, face_y, scaling, flip, idle, transition, duration, ease);
```

Argument	Type	Description
name	string	The unique identifier to use for the new character
spr_body	sprite	The sprite to draw as a character body
spr_face_idle	sprite	The sprite to draw as an idle face ( <i>or keyword 'none' for none</i> )
spr_face_talk	sprite	The sprite to draw as a talking face ( <i>or keyword 'none' for none</i> )
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the character, relative to the global offset
y	real	The vertical position to display the character, relative to the global offset
z	real	The drawing depth of the character, relative to other characters only
face_x	real	The horizontal position to display the face, relative to the body sprite top-left corner
face_y	real	The vertical position to display the face, relative to the body sprite top-left corner



scaling	integer/macro	Sets the automatic scaling mode for the character
flip	boolean	Enables or disables horizontally flipping the character
idle	real	Sets the delay in seconds between loops of the idle face sprite animation, where 3 is default
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
ease	integer/macro	Sets the ease override for the transition script

### Description:

Creates a new character with extra options which will be displayed until `vngen_char_destroy` is run. Multiple characters can exist simultaneously, however no two characters may share the same ID. Also note that unlike other entities, character IDs must be strings, not real numbers.

Note that using separate face sprites is optional. If only a body sprite is desired, 'spr\_face\_idle' and 'spr\_face\_talk' can be set to -1 or the keyword 'none' to disable them.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available keywords, origin points, scaling modes, transition animations, and ease modes.

### Example:

```
vngen_event() {  
    vngen_char_create_ext("John Doe", spr_body, spr_face, spr_talking, orig_left, orig_bottom, 0, view_hview[0], -1, 384, 128, scale_none, false, trans_slide_right, 2, ease_quad_out);  
}
```

## 6.17.3. The "vngen\_char\_modify\_style" Function

### Syntax:

```
vngen_char_modify_style(name, col1, col2, col3, col4, alpha, duration, [ease]);
```

Argument	Type	Description
name	string	The ID of the character to modify (or keyword 'all' for all characters)
col1	color	The top-left gradient color, where <code>c_white</code> is default
col2	color	The top-right gradient color, where <code>c_white</code> is default
col3	color	The bottom-right gradient color, where <code>c_white</code> is default
col4	color	The bottom-left gradient color, where <code>c_white</code> is default
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification

[ease]

integer/macro

transition, in seconds

*Optional:* Sets the ease mode to perform the transition in

## Description:

Applies a four-color gradient and transparency modifications to the input entity ID. Color values can be input using GameMaker Studio's built-in color constants or functions like VNgen's `make_color_hex_to_rgb`.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vnngen_event() {  
    vnngen_char_modify_style("John Doe", c_yellow, c_yellow, c_orange, c_orange, 1, 2)  
;  
}
```

This will apply a warm set of colors to the character "John Doe", suggesting an evening environment.

## 6.17.4. The "vnngen\_char\_modify\_pos" Function

### Syntax:

```
vnngen_char_modify_pos(name, x, y, z, scale, rot, duration, [ease]);
```

Argument	Type	Description
name	string	The ID of the character to modify ( <i>or keyword 'all' for all characters</i> )
x	real	The horizontal position to display the character, relative to the global offset
y	real	The vertical position to display the character, relative to the global offset
z	real	The drawing depth of the character, relative to other characters only
scale	real	The character scale multiplier, where 1 is default
rot	real	The character rotation, in degrees
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

**Note:** The scale *multiplier* here should not be confused with the scaling *mode* available in `*_create_ext` functions. The scale *mode* sets the entity's default scale, while the scale *multiplier* modifies the default scale.

## Description:



Applies a new position, rotation, and scale to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vngen_event() {  
    vngen_char_modify_pos("John Doe", 150, 150, -1, 1.5, 45, 2);  
}
```

## 6.17.5. The "vngen\_char\_modify\_ext" Function

**Syntax:**

```
vngen_char_modify_ext(name, x, y, z, xscale, yscale, rot, col1, col2, col3, col4, alpha, duration, ease);
```

Argument	Type	Description
name	string	The ID of the character to modify ( <i>or keyword 'all' for all characters</i> )
x	real	The horizontal position to display the character, relative to the global offset
y	real	The vertical position to display the character, relative to the global offset
z	real	The drawing depth of the character, relative to other characters only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The character rotation, in degrees
col1	color	The top-left gradient color, where c_white is default
col2	color	The top-right gradient color, where c_white is default
col3	color	The bottom-right gradient color, where c_white is default
col4	color	The bottom-left gradient color, where c_white is default
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification transition, in seconds
ease	integer/macro	Sets the ease mode for the modification transition

**Description:**

Combines `vngen_char_modify_style` and `vngen_char_modify_pos` with a few extra options, applying a new position, scale, rotation, four-color gradient, and transparency modifications to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vngen_event() {  
    vngen_char_modify_ext("John Doe", 0, 0, -1, 1.5, 1.5, 0, c_yellow, c_yellow, c_or  
ange, c_orange, 1, 2, ease_sin_out);  
}
```

This will increase the characters's scale, simulating a zoom while fading to a warm gradient suggesting an evening environment.

## 6.17.6. The "vngen\_char\_modify\_direct" Function

**Syntax:**

```
vngen_char_modify_direct(name, x, y, z, xscale, yscale, rot);
```

Argument	Type	Description
name	string	The ID of the character to modify ( <i>or keyword 'all' for all characters</i> )
x	real	The horizontal position to display the character, relative to the global offset
y	real	The vertical position to display the character, relative to the global offset
z	real	The drawing depth of the character, relative to other characters only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The character rotation, in degrees

**Description:**

Applies a new position, orientation, and scale to the input entity ID directly, outside the Q-script loop.

All modifications made with this script are permanent and will persist until another modification is performed.

As **this script is not an action**, care should be taken in deciding when and where to execute it. If both \*\_modify and \*\_modify\_direct scripts are executed together, whichever is run last will override the other.

**Example:**

```
var mx = window_mouse_get_x() - (window_get_width()*0.5);  
var my = window_mouse_get_y() - (window_get_height()*0.5);  
  
if (mouse_check_button(mb_left)) {  
    vngen_character_modify_direct("John Doe", mx, my, 0, 1, 1, 0);  
}
```

```
}
```

This will map the character to the mouse, allowing the user to click and drag them to a new position on the screen.

### 6.17.7. The "vnngen\_char\_replace" Function

#### Syntax:

```
vnngen_char_replace(name, spr_body, spr_face_idle, spr_face_talk, face_x, face_y, flip, duration, [ease]);
```

Argument	Type	Description
name	string	The ID of the character to replace
spr_body	sprite	The new sprite to draw as a character body
spr_face_idle	sprite	The new sprite to draw as an idle face (or keyword 'none' for none)
spr_face_talk	sprite	The new sprite to draw as a talking face (or keyword 'none' for none)
face_x	real	The horizontal position to display the face, relative to the body sprite top-left corner (or keyword 'previous' for no change)
face_y	real	The vertical position to display the face, relative to the body sprite top-left corner (or keyword 'previous' for no change)
flip	boolean	Enables or disables horizontally flipping the character
duration	real	Sets the duration of the fade transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

#### Description:

Replaces the input entity ID with a new set of sprites and fades the old character to the new one over the input duration.

Note that using separate face sprites is optional. If only a body sprite is desired, 'spr\_face\_idle' and 'spr\_face\_talk' can be set to -1 or the keyword 'none' to disable them.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

#### Example:

```
if (vnngen_event()) {
```

```
vngen_char_replace("John Doe", spr_new_body, spr_new_idle, spr_new_talk, 256, 384, true, 1);  
}
```

### 6.17.8. The "vngen\_char\_replace\_ext" Function

#### Syntax:

```
vngen_char_replace_ext(name, newname, spr_body, spr_face_idle, spr_face_talk, xorig, yorig, face_x, face_y, scaling, flip, idle_pause, duration, ease);
```

Argument	Type	Description
name	string	The ID of the character to replace
newname	string/macro	The new ID for the character being replaced (or keyword 'previous' for no change)
spr_body	sprite	The new sprite to display as a character body
spr_face_idle	sprite	The new sprite to display as an idle face (or keyword 'none' for none)
spr_face_talk	sprite	The new sprite to display as a talking face (or keyword 'none' for none)
xorig	real/macro	The new horizontal body sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The new vertical body sprite offset, or origin point, relative to the top-left corner
face_x	real	The horizontal position to display the face, relative to the body sprite top-left corner (or keyword 'previous' for no change)
face_y	real	The vertical position to display the face, relative to the body sprite top-left corner (or keyword 'previous' for no change)
scaling	integer/macro	Sets the new automatic scaling mode for the character
flip	boolean	Enables or disables horizontally flipping the character
idle	real	Sets the delay in seconds between loops of the idle face sprite animation, where 3 is default
duration	real	Sets the duration of the fade transition, in seconds
ease	integer/macro	Sets the ease mode to perform the fade transition in

#### Description:

Replaces the input entity ID with a new set of sprites and extra sprite properties and fades the old character to the new one over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until

another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

Unlike other modifications, this script allows changing the character's name, which doubles as its entity ID. As such, be aware that changing a character's ID will require further modification actions to use the new ID instead of the old one in the future.

See [Macros & Keywords](#) for a list of available origin points, scaling modes, and ease modes.

**Example:**

```
vngen_event() {  
    vngen_char_replace_ext("John Doe", previous, spr_new_body, spr_new_idle, spr_new_  
talk, orig_left, orig_bottom, 256, 384, scale_none, true, 3, 1, ease_sin_in_out);  
}
```

## 6.17.9. The "vngen\_char\_destroy" Function

**Syntax:**

```
vngen_char_destroy(name, transition, duration, [ease]);
```

Argument	Type	Description
name	string	The ID of the character to destroy (or keyword 'all' for all characters)
transition	script	Sets the transition animation to perform
duration	real	Sets the length of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

**Description:**

Destroys the input entity ID, freeing its resources from memory. Can also perform an exit transition animation before destroying. Once a given ID has been destroyed, it can be created again with the same ID.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

**Example:**

```
vngen_event() {  
    vngen_char_destroy("John Doe", trans_slide_left, 2, ease_sin_in);  
}
```

## 6.17.10. The "vngen\_char\_anim\_start" Function

**Syntax:**

```
vngen_char_anim_start(name, anim, duration, loop, reverse, [ease]);
```

Argument	Type	Description
----------	------	-------------



name	string	The ID of the character to animate ( <i>or keyword 'all' for all characters</i> )
anim	script	The animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

### Description:

Performs a keyframe animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

#### Example:

```
vngen_event() {  
    vngen_char_anim_start("John Doe", anim_wobble, 0.5, false, false);  
}
```

## 6.17.11. The "vngen\_char\_anim\_stop" Function

### Syntax:

```
vngen_char_anim_stop(name);
```

Argument	Type	Description
name	string	The ID of the character to stop animating ( <i>or keyword 'all' for all characters</i> )

### Description:

Stops the active keyframe animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

#### Example:

```
vngen_event() {  
    vngen_char_anim_stop("John Doe");  
}
```

## 6.17.12. The "vngen\_char\_deform\_start" Function

### Syntax:

```
vngen_char_deform_start(name, def, duration, loop, reverse, [ease]);
```

Argument	Type	Description
name	string	The ID of the character to animate ( <i>or keyword 'all' for all characters</i> )
def	script	The deformation animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

### Description:

Performs a keyframe deformation animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

#### Example:

```
vngen_event() {  
    vngen_char_deform_start("John Doe", def_breathe, 2.5, true, false);  
}
```

## 6.17.13. The "vngen\_char\_deform\_stop" Function

### Syntax:

```
vngen_char_deform_stop(name);
```

Argument	Type	Description
name	string	The ID of the character to stop animating ( <i>or keyword 'all' for all characters</i> )

### Description:

Stops the active keyframe deformation animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

#### Example:

```
vngen_event() {  
    vngen_scene_deform_stop("John Doe");  
}
```

## 6.17.14. The "vngen\_char\_shader\_start" Function

## Syntax:

```
vnngen_char_shader_start(name, shader, fade, [ease]);
```

Argument	Type	Description
name	string	The ID of the character to apply shader to ( <i>or keyword 'all' for all characters</i> )
shader	shader	The shader to perform
fade	real	Sets the length of time to fade shader in, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

## Description:

Applies a shader with optional fade in transition. Shaders are written externally and can be used to modify the color and position of vertices and/or pixels.

**Important note:** Some shaders require extra input values to function properly. These values must be set externally with `vnngen_set_shader_*` scripts.

See [Shaders](#) for a list of included shaders.

### Example:

```
vnngen_event() {  
    vnngen_char_shader_start("John Doe", shade_sepia, 0.5);  
}
```

## 6.17.15. The "vnngen\_char\_shader\_stop" Function

### Syntax:

```
vnngen_char_shader_stop(name, fade, [ease]);
```

Argument	Type	Description
name	string	The ID of the character to stop shader on ( <i>or keyword 'all' for all characters</i> )
fade	real	Sets the length of time to fade shader out, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

### Description:

Stops the active shader being performed on the specified entity, if any.

### Example:

```
vnngen_event() {  
    vnngen_char_shader_stop("John Doe", 0.5);  
}
```



```
}
```

## 6.18. Character Attachment Actions

Unlike other entities, character attachments do not exist independently, but rather are composited directly onto characters themselves. Attachments can be anything?from hair and body parts to clothing and accessories to weapons and equipment and beyond?and each attachment can be independently modified and animated while also inheriting the modifications and animations of the parent character. This makes attachments a very powerful feature of VNgen.

In this section we'll examine available attachment actions and how to create, modify, replace, destroy, and animate attachments on your characters.

### 6.18.1. The "vnngen\_attach\_create" Function

#### Syntax:

```
vnngen_attach_create(name, id, sprite, x, y, z, transition, duration, [ease]);
```

Argument	Type	Description
name	string	The ID of the character to apply the attachment to
id	real/string	The unique identifier to use for the new attachment
sprite	sprite	The sprite to draw as an attachment
x	real	The horizontal position to display the attachment, relative to the character top-left corner
y	real	The vertical position to display the attachment, relative to the character top-left corner
z	real	The drawing depth of the attachment, relative to other attachments only
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

#### Description:

Creates a new attachment for the input character ID which will be displayed until `vnngen_attach_destroy` is run. Multiple attachments can exist simultaneously, however no two attachments may share the same ID on the same character. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID. Unlike other entities, because attachments are specific to each character, attachments on other characters can use the same ID without conflict.

Note that all coordinates are relative to the parent character, including the z-index. For attachments, negative z-index will display the attachment in front of the parent character, while positive z-index will display the attachment behind them.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

**Example:**

```
vngen_event() {  
    vngen_attach_create("John Doe", "attach", spr_attach, 50, 150, -1, trans_wipe_right, 0.5);  
}
```

## 6.18.2. The "vngen\_attach\_create\_ext" Function

**Syntax:**

```
vngen_attach_create_ext(name, id, sprite, xorig, yorig, x, y, z, scaling, transition, duration, ease);
```

Argument	Type	Description
name	string	The ID of the character to apply attachment to
id	real/string	The unique identifier to use for the new attachment
sprite	sprite	The sprite to draw as an attachment
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the attachment, relative to the character top-left corner
y	real	The vertical position to display the attachment, relative to the character top-left corner
z	real	The drawing depth of the attachment, relative to other attachments only
scaling	integer/macro	Sets the automatic scaling mode for the attachment, relative to the parent character
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
ease	integer/macro	Sets the ease override for the transition script

**Description:**

Creates a new attachment with extra options for the input character ID which will be displayed until `vngen_attach_destroy` is run. Multiple attachments can exist simultaneously, however no two attachments may share the same ID on the same character. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID. Unlike other entities, because attachments are specific to each character, attachments on other characters can use the same ID without conflict.

Note that all coordinates are relative to the parent character, including the z-index. For attachments, negative z-index will display the attachment in front of the parent character, while positive z-index will display the attachment behind them. The same applies to the scaling mode, which is relative to the parent character rather than to the entire display.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available origin points, scaling modes, transition animations, and ease modes.

**Example:**

```
vnngen_event() {  
    vnngen_attach_create_ext("John Doe", "attach", spr_attach, orig_center, orig_center, 50, 150, -1, scale_none, trans_wipe_right, 0.5, ease_sin_out);  
}
```

### 6.18.3. The "vnngen\_attach\_modify\_style" Function

**Syntax:**

```
vnngen_attach_modify_style(name, id, col1, col2, col3, col4, alpha, duration, [ease])  
;
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to modify (or keyword 'all' for all attachments)
col1	color	The top-left gradient color, where c_white is default
col2	color	The top-right gradient color, where c_white is default
col3	color	The bottom-right gradient color, where c_white is default
col4	color	The bottom-left gradient color, where c_white is default
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

**Description:**

Applies a four-color gradient and transparency modifications to the input entity ID. Color values can be input using GameMaker Studio's built-in color constants or functions like VNgen's `make_color_hex_to_rgb`.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vnngen_event() {  
    vnngen_attach_modify_style("John Doe", "attach", c_yellow, c_yellow, c_orange, c_orange, 1, 2);  
}
```

This will give the attachment a yellow/orange gradient from top to bottom. Use attachments as clothing to apply custom colors in real-time!

## 6.18.4. The "vngen\_attach\_modify\_pos" Function

### Syntax:

```
vngen_attach_modify_pos(name, id, x, y, z, scale, rot, duration, [ease]);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to modify (or keyword 'all' for all attachments)
x	real	The horizontal position to display the attachment, relative to the parent character
y	real	The vertical position to display the attachment, relative to the parent character
z	real	The drawing depth of the attachment, relative to other attachments only
scale	real	The attachment scale multiplier, where 1 is default
rot	real	The attachment rotation, in degrees
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

**Note:** The scale *multiplier* here should not be confused with the scaling *mode* available in \*\_create\_ext functions. The scale *mode* sets the entity's default scale, while the scale *multiplier* modifies the default scale.

### Description:

Applies a new position, rotation, and scale to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {  
    vngen_attach_modify_pos("John Doe", "attach", 150, 150, -1, 1.5, 45, 2);  
}
```

## 6.18.5. The "vngen\_attach\_modify\_ext" Function

## Syntax:

```
vngen_attach_modify_ext(name, id, x, y, z, xscale, yscale, rot, col1, col2, col3, col4, alpha, duration, ease);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to modify (or keyword 'all' for all attachments)
x	real	The horizontal position to display the attachment, relative to the parent character
y	real	The vertical position to display the attachment, relative to the parent character
z	real	The drawing depth of the attachment, relative to other attachments only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The attachment rotation, in degrees
col1	color	The top-left gradient color, where c_white is default
col2	color	The top-right gradient color, where c_white is default
col3	color	The bottom-right gradient color, where c_white is default
col4	color	The bottom-left gradient color, where c_white is default
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification transition, in seconds
ease	integer/macro	Sets the ease mode for the modification transition

## Description:

Combines `vngen_attach_modify_style` and `vngen_attach_modify_pos` with a few extra options, applying a new position, scale, rotation, four-color gradient, and transparency modifications to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

## Example:

```
vngen_event() {  
    vngen_attach_modify_ext("John Doe", "attach", 0, 0, -1, 1.5, 1.5, 0, c_yellow, c_  
yellow, c_orange, c_orange, 1, 2, ease_sin_out);  
}
```

## 6.18.6. The "vngen\_attach\_modify\_direct" Function

### Syntax:

```
vngen_attach_modify_direct(name, id, x, y, z, xscale, yscale, rot);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to modify (or keyword 'all' for all attachments)
x	real	The horizontal position to display the attachment, relative to the parent character
y	real	The vertical position to display the attachment, relative to the parent character
z	real	The drawing depth of the attachment, relative to other attachments only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The attachment rotation, in degrees

### Description:

Applies a new position, orientation, and scale to the input entity ID directly, outside the Q-script loop.

All modifications made with this script are permanent and will persist until another modification is performed.

As **this script is not an action**, care should be taken in deciding when and where to execute it. If both \*\_modify and \*\_modify\_direct scripts are executed together, whichever is run last will override the other.

### Example:

```
var mx = window_mouse_get_x() - (window_get_width()*0.5);
var my = window_mouse_get_y() - (window_get_height()*0.5);

if (mouse_check_button(mb_left)) {
    vngen_attach_modify_direct("John Doe", "attach", mx, my, 0, 1, 1, 0);
}
```

This will map the attachment to the mouse, allowing the user to click and drag the attachment to a new position on the parent character.

## 6.18.7. The "vngen\_attach\_replace" Function

### Syntax:

```
vnngen_attach_replace(name, id, sprite, duration, [ease]);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to replace
sprite	sprite	The new sprite to display as an attachment
duration	real	Sets the duration of the fade transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

### Description:

Replaces the input entity ID with a new sprite and fades the old sprite to the new one over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
if (vnngen_event()) {  
    vnngen_attach_replace("John Doe", "attach", spr_new, 1);  
}
```

## 6.18.8. The "vnngen\_attach\_replace\_ext" Function

### Syntax:

```
vnngen_attach_replace_ext(name, id, sprite, xorig, yorig, scaling, duration, ease);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to replace
sprite	sprite	The new sprite to display as a attachment
xorig	real/macro	The new horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The new vertical sprite offset, or origin point, relative to the top-left corner
scaling	integer/macro	Sets the new automatic scaling mode for the attachment, relative to the parent character
duration	real	Sets the duration of the fade transition, in seconds
ease	integer/macro	Sets the ease mode to perform the

---

fade transition in

### Description:

Replaces the input entity ID with a new sprite and extra sprite properties and fades the old sprite to the new one over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available origin points, scaling modes, and ease modes.

### Example:

```
vnngen_event() {  
    vnngen_attach_replace_ext("John Doe", "attach", spr_new, orig_center, orig_center,  
    scale_none, 1, ease_sin_in_out);  
}
```

## 6.18.9. The "vnngen\_attach\_destroy" Function

### Syntax:

```
vnngen_attach_destroy(name, id, transition, duration, [ease]);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to destroy (or keyword 'all' for all attachments)
transition	script	Sets the transition animation to perform
duration	real	Sets the length of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

### Description:

Destroys the input entity ID, freeing its resources from memory. Can also perform an exit transition animation before destroying. Once a given ID has been destroyed, it can be created again with the same ID.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

### Example:

```
vnngen_event() {  
    vnngen_attach_destroy("John Doe", "attach", trans_fade, 2, ease_sin_in);  
}
```

## 6.18.10. The "vnngen\_attach\_anim\_start" Function



## Syntax:

```
vnngen_attach_anim_start(name, id, anim, duration, loop, reverse, [ease]);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to animate (or keyword 'all' for all attachments)
anim	script	The animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

## Description:

Performs a keyframe animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vnngen_event() {  
    vnngen_attach_anim_start("John Doe", "attach", anim_wobble, 0.5, false, false);  
}
```

## 6.18.11. The "vnngen\_attach\_anim\_stop" Function

### Syntax:

```
vnngen_attach_anim_stop(name, id);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to stop animating (or keyword 'all' for all attachments)

## Description:

Stops the active keyframe animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

### Example:

```
vnngen_event() {
```

```
vngen_attach_anim_stop("John Doe", "attach");  
}
```

## 6.18.12. The "vngen\_attach\_deform\_start" Function

### Syntax:

```
vngen_attach_deform_start(name, id, def, duration, loop, reverse, [ease]);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to animate (or keyword 'all' for all attachments)
def	script	The deformation animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

### Description:

Performs a keyframe deformation animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {  
    vngen_attach_deform_start("John Doe", "attach", def_wave, 0.5, false, false);  
}
```

## 6.18.13. The "vngen\_attach\_deform\_stop" Function

### Syntax:

```
vngen_attach_deform_stop(name, id);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to stop animating (or keyword 'all' for all attachments)

## Description:

Stops the active keyframe deformation animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

### Example:

```
vngen_event() {  
    vngen_attach_deform_stop("John Doe", "attach");  
}
```

## 6.18.14. The "vngen\_attach\_shader\_start" Function

### Syntax:

```
vngen_attach_shader_start(name, id, shader, fade, [ease]);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to apply shader to ( <i>or keyword 'all' for all attachments</i> )
shader	shader	The shader to perform
fade	real	Sets the length of time to fade shader in, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

### Description:

Applies a shader with optional fade in transition. Shaders are written externally and can be used to modify the color and position of vertices and/or pixels.

**!Important note:** Some shaders require extra input values to function properly. These values must be set externally with `vngen_set_shader_*` scripts.

See [Shaders](#) for a list of included shaders.

### Example:

```
vngen_event() {  
    vngen_attach_shader_start("John Doe", "attach", shade_sepia, 0.5);  
}
```

## 6.18.15. The "vngen\_attach\_shader\_stop" Function

### Syntax:

```
vngen_attach_shader_stop(id, name, fade, [ease]);
```

Argument	Type	Description
name	string	The ID of the attachment parent character
id	real/string	The ID of the attachment to stop shader on ( <i>or keyword 'all' for all attachments</i> )
fade	real	Sets the length of time to fade shader out, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

### Description:

Stops the active shader being performed on the specified entity, if any.

### Example:

```
vngen_event() {  
    vngen_attach_shader_stop("John Doe", "attach", 0.5);  
}
```

## 6.19. Emote Actions

Once created, most VNgen entities persist until manually destroyed by the user. However, sometimes you may wish to perform one-time animations as well. In VNgen, this type of temporary entity is called an **emote**. Most commonly, emotes in visual novels are used to augment expressions of character emotion in the form of animated icons which appear around their heads. In situations where the exact emotion is difficult to convey with simple art assets (or if the situation is somewhat comedic) emotes are a powerful communication tool.

But don't let the term limit your idea of what emotes can do: performing single sprite animations opens the door for all kinds of special effects. In this section, we'll examine how to create emotes for whatever sprite animations you can imagine.

### 6.19.1. The "vngen\_emote\_create" Function

#### Syntax:

```
vngen_emote_create(id, sprite, x, y, z, [loop]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new emote
sprite	sprite	The sprite to display as an emote
x	real	The horizontal position to display the emote, relative to the global offset
y	real	The vertical position to display the emote, relative to the global offset
z	real	The drawing depth of the emote, relative to other emotes only
[loop]	boolean	<i>Optional:</i> Enables or disables endlessly looping the emote

### Description:

Creates a new emote which will be displayed until the sprite animation is complete, at which point it will self-destruct and be removed from memory. Alternatively, the emote can be set to loop endlessly, in which case it will continue to exist until destroyed manually.

Multiple emotes can exist simultaneously, however no two emotes may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

**Example:**

```
vngen_event() {  
    vngen_emote_create("emote", spr_emote, 50, 50, -1, true);  
}
```

## 6.19.2. The "vngen\_emote\_create\_ext" Function

**Syntax:**

```
vngen_emote_create_ext(id, sprite, xorig, yorig, x, y, z, xscale, yscale, rot, col1,  
    col2, col3, col4, alpha, [loop]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new emote
sprite	sprite	The sprite to display as an emote
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the emote, relative to the global offset
y	real	The vertical position to display the emote, relative to the global offset
z	real	The drawing depth of the emote, relative to other emotes only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The emote rotation, in degrees
col1	color	The top-left gradient color, where c_white is default
col2	color	The top-right gradient color, where c_white is default
col3	color	The bottom-right gradient color, where c_white is default
col4	color	The bottom-left gradient color, where c_white is default
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
[loop]	boolean	<i>Optional:</i> Enables or disables endlessly looping the emote

## Description:

Creates a new emote with extra options which will be displayed until the sprite animation is complete, at which point it will self-destruct and be removed from memory. Alternatively, the emote can be set to loop endlessly, in which case it will continue to exist until destroyed manually.

Multiple emotes can exist simultaneously, however no two emotes may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

See [Macros & Keywords](#) for a list of available origin points.

### Example:

```
vnngen_event() {  
    vnngen_emote_create_ext("emote", spr_emote, orig_center, orig_center, 50, 50, -1,  
1, 1, 0, c_yellow, c_yellow, c_orange, c_orange, 1);  
}
```

## 6.19.3. The "vnngen\_emote\_destroy" Function

### Syntax:

```
vnngen_emote_destroy(id, wait);
```

Argument	Type	Description
id	real/string	The ID of the emote to destroy ( <i>or keyword 'all' for all emotes</i> )
wait	boolean	Enables or disables waiting for the sprite animation to complete before destroying

### Description:

Destroys the input entity ID, freeing its resources from memory. Can also wait until the current sprite animation is complete before destroying. Once a given ID has been destroyed, it can be created again with the same ID.

Note that this script is only required to destroy *looped* emotes, as non-looped emotes self-destruct automatically.

### Example:

```
vnngen_event() {  
    vnngen_emote_destroy("emote_tears", true);  
}
```

## 6.20. Effect Actions

VNgen effects are a very special type of entity. They are functionally very similar to animations performed on other entities, only effects are completely independent and execute arbitrary code instead. This code is executed in the Draw Event, making effects suitable for a variety of visual and experiential tasks (rather than general programming tasks).

Effects are displayed beneath textboxes and above all other entities, and are not affected by the global perspective.



See [Effects](#) for a list of included effect scripts and how to make your own. In this section, we'll examine how to perform and terminate effects.

### 6.20.1. The "vngen\_effect\_start" Function

#### Syntax:

```
vngen_effect_start(id, effect, duration, loop, reverse, [ease]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new effect
effect	script	The effect script to perform
duration	real	Sets the duration of the entire effect
loop	boolean	Enables or disables looping the effect
reverse	boolean	Enables or disables performing the effect in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the effect script

#### Description:

Performs a keyframe effect script with the input duration. As effects are independent, unlike animations and modifications, multiple effects can be performed simultaneously.

See [Included Effects](#) for a list of included effect scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

#### Example:

```
vngen_event() {  
    vngen_effect_start("ef", ef_scrn_flash, 0.5, false, false);  
}
```

### 6.20.2. The "vngen\_effect\_stop" Function

#### Syntax:

```
vngen_effect_stop(id);
```

Argument	Type	Description
id	real/string	The ID of the effect to stop (or keyword 'all' for all effects)

#### Description:

Stops the specified keyframe effect being performed, if any. Note that this script only needs to be run to stop *looped* effects.

#### Example:

```
vngen_event() {
```

```
vngen_effect_stop("ef");  
}
```

## 6.21. Textbox Actions

One of the most basic entities in VNgen, textboxes are purely decorative and do not serve any mechanical purpose. However, a good textbox is an important part of any visual novel's UI design, and more importantly, ensures readability across a variety of backgrounds and scenes. And basic or not, textboxes still support the full range of available actions, including color blending and scripted animation support! This means textboxes are easily customizable for each character on-screen, taking them far above and beyond being just a colored box.

In this section we'll examine available textbox actions covering five basic operations: create, modify, replace, destroy, and animate.

### 6.21.1. The "vngen\_textbox\_create" Function

#### Syntax:

```
vngen_textbox_create(id, sprite, x, y, z, transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new textbox
sprite	sprite	The sprite to draw as a textbox
x	real	The horizontal position to display the textbox, relative to the global offset
y	real	The vertical position to display the textbox, relative to the global offset
z	real	The drawing depth of the textbox, relative to other textboxes only
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

#### Description:

Creates a new textbox which will be displayed until `vngen_textbox_destroy` is run. Multiple textboxes can exist simultaneously, however no two textboxes may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

#### Example:

```
vngen_event() {  
    vngen_textbox_create("tb", spr_textbox, 0, 1080, 0, trans_wipe_right, 2);  
}
```

### 6.21.2. The "vngen\_textbox\_create\_ext" Function





## Syntax:

```
vngen_textbox_create_ext(id, sprite, xorig, yorig, x, y, z, scaling, transition, duration, ease);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new textbox
sprite	sprite	The sprite to draw as a textbox
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the textbox, relative to the global offset
y	real	The vertical position to display the textbox, relative to the global offset
z	real	The drawing depth of the textbox, relative to other textboxes only
scaling	integer/macro	Sets the automatic scaling mode for the textbox
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
ease	integer/macro	Sets the ease override for the transition script

## Description:

Creates a new textbox with extra options which will be displayed until `vngen_textbox_destroy` is run. Multiple textboxes can exist simultaneously, however no two textboxes may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available origin points, scaling modes, transition animations, and ease modes.

### Example:

```
vngen_event() {  
    vngen_textbox_create_ext("tb", spr_textbox, orig_left, orig_bottom, 0, 1080, 0, scale_stretch_x, trans_wipe_right, 2, ease_sin_in_out);  
}
```

## 6.21.3. The "vngen\_textbox\_modify\_style" Function

### Syntax:

```
vngen_textbox_modify_style(id, col1, col2, col3, col4, alpha, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the textbox to modify ( <i>or keyword 'all' for all textboxes</i> )

col1	color	The top-left gradient color, where c_white is default
col2	color	The top-right gradient color, where c_white is default
col3	color	The bottom-right gradient color, where c_white is default
col4	color	The bottom-left gradient color, where c_white is default
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

### Description:

Applies a four-color gradient and transparency modifications to the input entity ID. Color values can be input using GameMaker Studio's built-in color constants or functions like VNgen's `make_color_hex_to_rgb`.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

#### Example:

```
vngen_event() {
    vngen_textbox_modify_style("tb", c_red, c_red, c_white, c_white, 1, 2);
}
```

This will apply a red tint along the top of the textbox, perhaps to represent a character whose theme color is red.

## 6.21.4. The "vngen\_textbox\_modify\_pos" Function

### Syntax:

```
vngen_textbox_modify_pos(id, x, y, z, scale, rot, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the textbox to modify ( <i>or keyword 'all' for all textboxes</i> )
x	real	The horizontal position to display the textbox, relative to the global offset
y	real	The vertical position to display the textbox, relative to the global offset
z	real	The drawing depth of the textbox , relative to other textboxes only
scale	real	The textbox scale multiplier, where 1 is default
rot	real	The textbox rotation, in degrees

duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

**Note:** The scale *multiplier* here should not be confused with the scaling *mode* available in \*\_create\_ext functions. The scale *mode* sets the entity's default scale, while the scale *multiplier* modifies the default scale.

## Description:

Applies a new position, rotation, and scale to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {
    vngen_textbox_modify_pos("tb", 150, 150, -1, 1.5, 45, 2);
}
```

## 6.21.5. The "vngen\_textbox\_modify\_ext" Function

### Syntax:

```
vngen_textbox_modify_ext(id, x, y, z, xscale, yscale, rot, col1, col2, col3, col4, alpha, duration, ease);
```

Argument	Type	Description
id	real/string	The ID of the textbox to modify ( <i>or keyword 'all' for all textboxes</i> )
x	real	The horizontal position to display the textbox, relative to the global offset
y	real	The vertical position to display the textbox, relative to the global offset
z	real	The drawing depth of the textbox, relative to other textboxes only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The textbox rotation, in degrees
col1	color	The top-left gradient color, where c_white is default
col2	color	The top-right gradient color, where c_white is default
col3	color	The bottom-right gradient color, where c_white is default
col4	color	The bottom-left gradient color, where c_white is default



alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification transition, in seconds
ease	integer/macro	Sets the ease mode for the modification transition

### Description:

Combines `vnngen_textbox_modify_style` and `vnngen_textbox_modify_pos` with a few extra options, applying a new position, scale, rotation, four-color gradient, and transparency modifications to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vnngen_event() {  
    vnngen_textbox_modify_ext("tb", 0, 0, -1, 1.5, 1.5, 0, c_yellow, c_yellow, c_orange,  
    c_orange, 1, 2, ease_sin_out);  
}
```

## 6.21.6. The "vnngen\_textbox\_modify\_direct" Function

### Syntax:

```
vnngen_textbox_modify_direct(id, x, y, z, xscale, yscale, rot);
```

Argument	Type	Description
id	real/string	The ID of the textbox to modify ( <i>or keyword 'all' for all textboxes</i> )
x	real	The horizontal position to display the textbox, relative to the global offset
y	real	The vertical position to display the textbox, relative to the global offset
z	real	The drawing depth of the textbox, relative to other textboxes only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The textbox rotation, in degrees

### Description:

Applies a new position, orientation, and scale to the input entity ID directly, outside the Q-script loop.

All modifications made with this script are permanent and will persist until another modification is performed.

As **this script is not an action**, care should be taken in deciding when and where to execute it. If both `*_modify` and `*_modify_direct` scripts are executed together, whichever is run last will override the other.

**Example:**

```
var mx = window_mouse_get_x() - (window_get_width()*0.5);
var my = window_mouse_get_y() - (window_get_height()*0.5);

if (mouse_check_button(mb_left)) {
    vngen_textbox_modify_direct("tb", mx, my, 0, 1, 1, 0);
}
```

This will map the textbox to the mouse, allowing the user to click and drag the textbox to a new position on the screen.

## 6.21.7. The "vngen\_textbox\_replace" Function

**Syntax:**

```
vngen_textbox_replace(id, sprite, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the textbox to replace
sprite	sprite	The new sprite to draw as a textbox
duration	real	Sets the duration of the fade transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

**Description:**

Replaces the input entity ID with a new sprite and fades the old sprite to the new one over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
if (vngen_event()) {
    vngen_textbox_replace("tb", spr_new, 1);
}
```

## 6.21.8. The "vngen\_textbox\_replace\_ext" Function

**Syntax:**

```
vngen_textbox_replace_ext(id, sprite, xorig, yorig, scaling, duration, ease);
```

Argument	Type	Description
id	real/string	The ID of the textbox to replace

sprite	sprite	The new sprite to draw as a textbox
xorig	real/macro	The new horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The new vertical sprite offset, or origin point, relative to the top-left corner
scaling	integer/macro	Sets the new automatic scaling mode for the textbox
duration	real	Sets the duration of the fade transition, in seconds
ease	integer/macro	Sets the ease mode to perform the fade transition in

### Description:

Replaces the input entity ID with a new sprite and extra sprite properties and fades the old sprite to the new one over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available origin points, scaling modes, and ease modes.

#### Example:

```
vngen_event() {
    vngen_textbox_replace_ext("tb", spr_new, orig_center, orig_center, scale_none, 1,
    ease_sin_in_out);
}
```

## 6.21.9. The "vngen\_textbox\_destroy" Function

### Syntax:

```
vngen_textbox_destroy(id, transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the textbox to destroy ( <i>or keyword 'all' for all textboxes</i> )
transition	script	Sets the transition animation to perform
duration	real	Sets the length of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

### Description:

Destroys the input entity ID, freeing its resources from memory. Can also perform an exit transition animation before destroying. Once a given ID has been destroyed, it can be created again with the same ID.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

**Example:**

```
vngen_event() {  
    vngen_textbox_destroy("tb", trans_fade, 2, ease_sin_in);  
}
```

## 6.21.10. The "vngen\_textbox\_anim\_start" Function

**Syntax:**

```
vngen_textbox_anim_start(id, anim, duration, loop, reverse, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the textbox to animate ( <i>or keyword 'all' for all textboxes</i> )
anim	script	The animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

**Description:**

Performs a keyframe animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vngen_event() {  
    vngen_textbox_anim_start("tb", anim_quake, 1, false, false);  
}
```

## 6.21.11. The "vngen\_textbox\_anim\_stop" Function

**Syntax:**

```
vngen_textbox_anim_stop(id);
```

Argument	Type	Description
id	real/string	The ID of the textbox to stop animating ( <i>or keyword 'all' for all textboxes</i> )

**Description:**

Stops the active keyframe animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

**Example:**

```
vngen_event() {  
    vngen_textbox_anim_stop("tb");  
}
```

## 6.21.12. The "vngen\_textbox\_deform\_start" Function

**Syntax:**

```
vngen_textbox_deform_start(id, def, duration, loop, reverse, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the textbox to animate ( <i>or keyword 'all' for all textboxes</i> )
def	script	The deformation animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

**Description:**

Performs a keyframe deformation animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vngen_event() {  
    vngen_textbox_deform_start("tb", def_wave, 0.5, false, false);  
}
```

## 6.21.13. The "vngen\_textbox\_deform\_stop" Function

**Syntax:**

```
vngen_textbox_deform_stop(id);
```

Argument	Type	Description
id	real/string	The ID of the textbox to stop animating ( <i>or keyword 'all' for all</i>



---

*textboxes)***Description:**

Stops the active keyframe deformation animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

**Example:**

```
vngen_event() {  
    vngen_textbox_deform_stop("tb");  
}
```

## 6.21.14. The "vngen\_textbox\_shader\_start" Function

**Syntax:**

```
vngen_textbox_shader_start(id, shader, fade, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the textbox to apply shader to (or keyword 'all' for all textboxes)
shader	shader	The shader to perform
fade	real	Sets the length of time to fade shader in, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

**Description:**

Applies a shader with optional fade in transition. Shaders are written externally and can be used to modify the color and position of vertices and/or pixels.

**Important note:** Some shaders require extra input values to function properly. These values must be set externally with `vngen_set_shader_*` scripts.

See [Shaders](#) for a list of included shaders.

**Example:**

```
vngen_event() {  
    vngen_textbox_shader_start("tb", shade_sepia, 0.5);  
}
```

## 6.21.15. The "vngen\_textbox\_shader\_stop" Function

**Syntax:**

```
vngen_textbox_shader_stop(id, fade, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the textbox to stop shader

---

fade	real	on (or keyword 'all' for all textboxes) Sets the length of time to fade shader out, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

### Description:

Stops the active shader being performed on the specified entity, if any.

### Example:

```
vngen_event() {  
    vngen_textbox_shader_stop("tb", 0.5);  
}
```

## 6.22. Text Actions

Text is one of the most important aspects of any visual novel. While writing great content is up to you, VNgen provides many options for displaying text in the most engaging ways possible. As one of the most specialized of all entities, text both shares effects and animations with other entities and possesses properties all its own. VNgen even features inline markup exclusive to text actions which adds support for variable colors, fonts, typewriter speeds, and more!

At the same time, text is also quick and easy to use. Advanced stylization is stored in memory and can be reapplied with a single keyword, saving time rewriting the same styles over and over. After all, your focus should be on telling stories?not code! Any time a font, gradient color, shadow color, or outline color is declared, it is automatically stored in memory and assigned to the speaking character. If ever the same speaking character is used again, fonts and colors using the 'inherit' keyword will restore this saved data. In this way, each character can easily have their own fonts and colors without the need for performing stylization over and over again!

In this section we'll examine available text actions, as well as special features such as inline markup, style inheritance, and more.

### 6.22.1. Inline Text Markup

In addition to standard stylization features, VNgen text actions support inline markup for modifying style and behavior in ways not possible externally. This markup is written directly in text strings and is interpreted by the engine in real-time, meaning you can even use VNgen markup in external word processing software before copying in your script.

With some exceptions, markup is typically written as tags in brackets with opening and closing pairs. However, using closing tags is optional, and omitting them will simply apply markup to the rest of the entire string.

In this section we will examine available markup tags and their individual behaviors in detail.

## Speaker Names

### Syntax:

```
"Name | |Text "
```

## Description:

Though not always necessary, the name of the current speaking character associated with the text can be declared with inline markup. This is done by writing the name of the character followed immediately by “||”, after which text can be written as normal. Any text preceding “||” will be assigned as the speaking character and ignored when drawing the rest of the string onto the screen.

If the speaking character is declared both inline and as an argument, the inline declaration will take precedence.

Note that in order for speaker names to affect other engine features such as labels and character highlighting, the name declared here must match the name set in `vngen_char_create` **exactly**.

## Newline

### Syntax:

```
\n
```

## Description:

While VNgen word-wraps text automatically, there may be times when you wish to manually trigger a line break prematurely. This is achieved using a combination of the primary escape character, “\”, and “n” for “newline”.

Note that VNgen uses “\n” for manual line breaks in both GameMaker Studio 1 and 2, even though only GameMaker Studio 2 supports “\n” markup by default.

## Escape

### Syntax:

```
^[
```

## Description:

By default, VNgen interprets all brackets as markup. However, there may be times when you wish to draw brackets literally. This is achieved using a combination of the secondary escape character, “^”, and “[”.

Note that only opening brackets need to be escaped.

## Font

### Syntax:

```
[font=my_font]custom text[/font]
```

## Description:

Draws the enclosed text with a custom font. This font must be previously created as a font resource in GameMaker Studio, and can include bold or italic variants of the default font, or different fonts entirely. That being said, it is recommended to use fonts of similar size when possible. It is also possible to supply a variable in place of an actual font resource.

## Color

### Syntax:

```
[color=#FFFFFF]colored text[/color]  
[color=#FFF, #000]2-color gradient text[/color]  
[color=#FFF, #FFF, #000]3-color gradient text[/color]  
[color=#FFF, #FFF, #000, #000]4-color gradient text[/color]
```

### Description:

Draws the enclosed text with up to four colors specified in hex color notation and separated by commas. Hex color can be written in either three or six characters representing the red, green, and blue channels, respectively. Where multiple colors are specified, text will be drawn with a gradient: two colors providing a horizontal gradient, three colors providing a triangular gradient, and four colors providing a square gradient.

## Shadow

### Syntax:

```
[shadow=#000]shaded text[/shadow]
```

### Description:

Draws the enclosed text with a shadow of the color specified in hex color notation. Hex color can be written in either three or six characters representing the red, green, and blue channels, respectively. Unlike regular text color, shadow color modifications do not support comma-separated gradients.

## Outline

### Syntax:

```
[outline=#FFF]outlined text[/outline]
```

### Description:

Draws the enclosed text with an outline of the color specified in hex color notation. Hex color can be written in either three or six characters representing the red, green, and blue channels, respectively. Unlike regular text color, outline color modifications do not support comma-separated gradients.

## Event

---

### Syntax:

```
[event=ev_other, ev_user0]
```

### Description:

Executes the specified GameMaker Studio Event when reached by the typewriter effect. Two values must always be supplied here, separated by a comma, with the second value being 0 when no other value applies. Supports Create, Destroy, Step, Alarm, Draw, and Other Events (see documentation on GameMaker's `event_perform` function for a list of available events in these categories). Other > User Events are recommended.

The desired Event must be set up in the running object in order for markup to take effect. In this way, event markup can be used to execute any code possible, but bear in mind that it will only be executed **once** when the text is drawn.

## Link

### Syntax:

```
[link=ev_other, ev_user0]linked text[/link]
```

### Description:

Turns the enclosed text into a clickable hotspot which will execute the specified GameMaker Studio Event when selected. Two values must always be supplied here, separated by a comma, with the second value being 0 when no other value applies. Supports Create, Destroy, Step, Alarm, Draw, and Other Events (see documentation on GameMaker's `event_perform` function for a list of available events in these categories). Other > User Events are recommended.

The desired Event must be set up in the running object in order for links to take effect. In this way, VNgen links can be used to execute any code possible, including acting as links to webpages with the `url_open` command built-in to GameMaker Studio.

It is recommended to combine link tags with other style tags to draw attention to them as clickable. Links will automatically highlight when hovered, however highlighting will not be visible on white text.

## Speed

### Syntax:

```
[speed=0.5]slow text[/speed]  
[speed=2]fast text[/speed]
```

### Description:

Multiplies the speed at which enclosed text is printed onto the screen. Unlike the default speed, which is written in characters per-second, speed tags use a multiplier to achieve the same effect at all text speeds.



## Pause

### Syntax:

```
[pause=5]temporarily delayed text  
[pause=-1]indefinitely delayed text
```

### Description:

Suspends the typewriter effect for the specified duration, in seconds. If -1 is supplied, the effect will be paused indefinitely until `vnngen_continue` is run. Unlike other tags, pause markup has no corresponding closing tag.

**Important note:** If auto mode is enabled, negative pause values will be interpreted literally, not indefinitely. This allows crafting a fully automated experience that retains temporary pauses where appropriate. To change this behavior, see `vnngen_set_auto_type`.

## 6.22.2. The "vnngen\_text\_create" Function

### Syntax:

```
vnngen_text_create(id, name, text, x, y, z, linebreak, font, color, transition, duration, [ease], [lang]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new text
name	string	The speaker name to associate with the text
text	string	The text string to display
x	real/macro	The horizontal position to display the text, relative to the global offset (or keyword 'auto' to place after previous text)
y	real/macro	The vertical position to display the text, relative to the global offset (or keyword 'auto' to place after previous text)
z	real	The drawing depth of the text, relative to other text only
linebreak	real	The width in pixels before text is wrapped into a new line
font	font/macro	The font to draw text in, where <code>fnt_default</code> is default (or keyword 'inherit' for saved style)
color	color/macro	The color to draw text in, where <code>c_white</code> is default (or keyword 'inherit' for saved style)
transition	script	Sets the transition animation to perform



duration	real	Sets the duration of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script
[lang]	real/string	<i>Optional:</i> Sets the language flag for the text

## Description:

Creates a new string of text which will be displayed until `vngen_text_destroy` is run. Multiple text elements can exist simultaneously, however no two may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

By default, text will be printed onto the screen with a typewriter effect at the rate set by `vngen_set_speed`, progressing one character at a time until the entire string is displayed. Once the string is complete, the engine will wait for user input before continuing to the next event. Running `vngen_continue` before the effect is complete will skip it.

Newly-created text will be automatically added to the backlog.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available style modes, transition animations, and ease modes. See [Language Functions](#) for details on multi-language support. See [Inline Markup](#) for available markup tags.

### Example:

```
vngen_event() {  
    vngen_text_create("text", "John Doe", "Hello, world!", 200, 900, -1, 1500, inherit, inherit, trans_fade, 2);  
    vngen_text_create("text", "John Doe", "Hello, world!", 200, 900, -1, 1500, fnt_Arial, c_white, trans_fade, 2, "en-US");  
}
```

## 6.22.3. The "vngen\_text\_create\_ext" Function

### Syntax:

```
vngen_text_create_ext(id, name, text, xorig, yorig, x, y, z, scaling, linebreak, font, color, transition, duration, ease, [lang]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new text
name	string	The character name to associate with the text
text	string	The text string to display
xorig	real/macro	The horizontal texture offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical texture offset, or origin point, relative to the top-left corner
x	real/macro	The horizontal position to display the text, relative to the global offset ( <i>or keyword 'auto' to place after previous text</i> )

y	real/macro	The vertical position to display the text, relative to the global offset (or keyword 'auto' to place after previous text)
z	real	The drawing depth of the text, relative to other text only
scaling	integer/macro	Sets the automatic scaling mode for the text
linebreak	real	The width in pixels before text is wrapped into a new line
font	font/macro	The font to draw text in, where fnt_default is default (or keyword 'inherit' for saved style)
color	color/macro	The color to draw text in, where c_white is default (or keyword 'inherit' for saved style)
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
ease	integer/macro	Sets the ease override for the transition script
[lang]	real/string	Optional: Sets the language flag for the text

## Description:

Creates a new string of text with extra options which will be displayed until `vnngen_text_destroy` is run. Multiple text elements can exist simultaneously, however no two may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

By default, text will be printed onto the screen with a typewriter effect at the rate set by `vnngen_set_speed`, progressing one character at a time until the entire string is displayed. Once the string is complete, the engine will wait for user input before continuing to the next event. Running `vnngen_continue` before the effect is complete will skip it.

Newly-created text will be automatically added to the backlog.

Note that text elements are displayed as a surface, or texture, and the exact dimensions of this texture may not be known. As a result, functions like 'xorig', 'yorig', and 'scaling' may not behave as expected. It is important to test your text actions and design your fonts and linebreaks to a standard which results in predictable texture dimensions.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available style modes, origin points, scaling modes, transition animations, and ease modes. See [Language Functions](#) for details on multi-language support. See [Inline Markup](#) for available markup tags.

## Example:

```
vnngen_event() {  
    vnngen_text_create("text", "John Doe", "Hello, world!", orig_left, orig_top, 200,  
900, -1, scale_none, 1500, inherit, inherit, trans_fade, 2, ease_sin_out);  
    vnngen_text_create("text", "", "John Doe|Hello, world!", orig_left, orig_top, 200  
, 900, -1, scale_none, 1500, fnt_Arial, c_white, trans_fade, 2, ease_sin_out, "en-
```



```
US" );  
}
```

## 6.22.4. The "vngen\_text\_modify\_style" Function

### Syntax:

```
vngen_text_modify_style(id, col1, col2, col3, col4, shadow, outline, alpha, duration  
, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the text to modify (or keyword 'all' for all text)
col1	color/macro	The text top-left gradient color, where c_white is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
col2	color/macro	The text top-right gradient color, where c_white is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
col3	color/macro	The text bottom-right gradient color, where c_white is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
col4	color/macro	The text bottom-left gradient color, where c_white is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
shadow	color/macro	The text shadow color, where 'none' is default (or keyword 'previous' for no change, keyword 'inherit' for saved style, or 'none' for none)
outline	color/macro	The text outline color, where 'none' is default (or keyword 'previous' for no change, keyword 'inherit' for saved style, or 'none' for none)
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

### Description:

Applies a four-color gradient and transparency modifications to the input entity ID. Color values can be input using GameMaker Studio's built-in color constants or functions like VNgen's `make_color_hex_to_rgb`.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

Unlike other entities, color modifications applied to text are applied to each individual character in the string rather than to the string as a whole. The exception to this is animations, which are applied as normal.

See [Macros & Keywords](#) for a list of available style modes and ease modes.

#### Example:

```
vngen_event() {
    vngen_text_modify_style("text", c_red, c_blue, c_green, c_yellow, c_black, c_white, 1, 2);
}
```

### 6.22.5. The "vngen\_text\_modify\_pos" Function

#### Syntax:

```
vngen_text_modify_pos(id, x, y, z, scale, rot, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the text to modify ( <i>or keyword 'all' for all text</i> )
x	real	The horizontal position to display the text, relative to the global offset
y	real	The vertical position to display the text, relative to the global offset
z	real	The drawing depth of the text, relative to other text only
scale	real	The text scale multiplier, where 1 is default
rot	real	The text rotation, in degrees
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

**Note:** The scale *multiplier* here should not be confused with the scaling *mode* available in \*\_create\_ext functions. The scale *mode* sets the entity's default scale, while the scale *multiplier* modifies the default scale.

#### Description:

Applies a new position, rotation, and scale to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

#### Example:

```
vngen_event() {
    vngen_text_modify_pos("text", 150, 150, -1, 1.5, 45, 2);
}
```

### 6.22.6. The "vngen\_text\_modify\_ext" Function



## Syntax:

```
vnngen_text_modify_ext(id, x, y, z, xscale, yscale, rot, col1, col2, col3, col4, shadow, outline, alpha, duration, ease);
```

Argument	Type	Description
id	real/string	The ID of the text to modify (or keyword 'all' for all text)
x	real	The horizontal position to display the text, relative to the global offset
y	real	The vertical position to display the text, relative to the global offset
z	real	The drawing depth of the text, relative to other text only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The text rotation, in degrees
col1	color/macro	The text top-left gradient color, where c_white is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
col2	color/macro	The text top-right gradient color, where c_white is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
col3	color/macro	The text bottom-right gradient color, where c_white is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
col4	color/macro	The text bottom-left gradient color, where c_white is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
shadow	color/macro	The text shadow color, where 'none' is default (or keyword 'previous' for no change, keyword 'inherit' for saved style, or 'none' for none)
outline	color/macro	The text outline color, where 'none' is default (or keyword 'previous' for no change, keyword 'inherit' for saved style, or 'none' for none)
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification transition, in seconds
ease	integer/macro	Sets the ease mode for the modification transition

## Description:

Combines `vnngen_text_modify_style` and `vnngen_text_modify_pos` with a few extra options, applying a new position, scale, rotation, four-color gradient, and transparency modifications to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

Unlike other entities, color modifications applied to text elements are applied to each individual character in the string rather than to the string as a whole. The exception to this is animations, which are applied as normal.

See [Macros & Keywords](#) for a list of available style modes and ease modes.

**Example:**

```
vngen_event() {  
    vngen_text_modify_ext("text", 0, 0, -1, 1.5, 1.5, 0, c_yellow, c_yellow, c_orange  
    , c_orange, 1, 2, ease_sin_out);  
}
```

## 6.22.7. The "vngen\_text\_modify\_direct" Function

**Syntax:**

```
vngen_text_modify_direct(id, x, y, z, xscale, yscale, rot);
```

Argument	Type	Description
id	real/string	The ID of the text to modify ( <i>or keyword 'all' for all text</i> )
x	real	The horizontal position to display the text, relative to the global offset
y	real	The vertical position to display the text, relative to the global offset
z	real	The drawing depth of the text, relative to other text only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The text rotation, in degrees

**Description:**

Applies a new position, orientation, and scale to the input entity ID directly, outside the Q-script loop.

All modifications made with this script are permanent and will persist until another modification is performed.

As **this script is not an action**, care should be taken in deciding when and where to execute it. If both `*_modify` and `*_modify_direct` scripts are executed together, whichever is run last will override the other.

**Example:**

```
var mx = window_mouse_get_x() - (window_get_width()*0.5);  
var my = window_mouse_get_y() - (window_get_height()*0.5);  
  
if (mouse_check_button(mb_left)) {  
    vngen_text_modify_direct("text", mx, my, 0, 1, 1, 0);  
}
```

This will map the text element to the mouse, allowing the user to click and drag the text to a new position on the screen.

## 6.22.8. The "vngen\_text\_replace" Function

### Syntax:

```
vngen_text_replace(id, name, text, font, color, duration, [ease], [lang]);
```

Argument	Type	Description
id	real/string	The ID of the text to replace
name	string/macro	The new speaker name to associate with the text ( <i>or keyword 'previous' for no change</i> )
text	string	The new text string to display
font	font/macro	The font to draw text in, where <code>fnt_default</code> is default ( <i>or keyword 'previous' for no change, or keyword 'inherit' for saved style</i> )
color	color/macro	The color to draw text in, where <code>c_white</code> is default ( <i>or keyword 'previous' for no change, or keyword 'inherit' for saved style</i> )
duration	real	Sets the duration of the fade transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in
[lang]	real/string	<i>Optional:</i> Sets the language flag for the text

### Description:

Replaces the input entity ID with a new string of text and fades the old text to the new one over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

Replaced text will be automatically added to the backlog.

See [Macros & Keywords](#) for a list of available style modes and ease modes. See [Language Functions](#) for details on multi-language support. See [Inline Markup](#) for available markup tags.

### Example:

```
vngen_event() {  
    vngen_text_replace("text", previous, "Hello, world!", previous, inherit, 3);  
    vngen_text_replace("text", "Jane Doe", "Hello, world!", fnt_Arial, c_white, 3, "en-US");  
}
```

## 6.22.9. The "vngen\_text\_replace\_ext" Function

## Syntax:

```
vnngen_text_replace_ext(id, name, text, xorig, yorig, scaling, linebreak, font, color  
, duration, ease, [lang]);
```

Argument	Type	Description
id	real/string	The ID of the text to replace
name	string	The character name to associate with the text
text	string	The new text string to display
xorig	real/macro	The horizontal texture offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical texture offset, or origin point, relative to the top-left corner
scaling	integer/macro	Sets the automatic scaling mode for the text
linebreak	real	The width in pixels before text is wrapped into a new line
font	font/macro	The font to draw text in, where <i>fnt_default</i> is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
color	color/macro	The color to draw text in, where <i>c_white</i> is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
duration	real	Sets the duration of the fade transition, in seconds
ease	integer/macro	Sets the ease mode to perform the transition in
[lang]	real/string	<i>Optional:</i> Sets the language flag for the text

## Description:

Replaces the input entity ID with a new string of text and fades the old text to the new one over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

Note that text elements are displayed as a surface, or texture, and the exact dimensions of this texture may not be known. As a result, functions like 'xorig', 'yorig', and 'scaling' may not behave as expected. It is important to test your text actions and design your fonts and linebreaks to a standard which results in predictable texture dimensions.

Replaced text will be automatically added to the backlog.

See [Macros & Keywords](#) for a list of available style modes. See [Language Functions](#) for details on multi-language support. See [Inline Markup](#) for available markup tags.

## Example:

```
vnngen_event() {  
    vnngen_text_replace("text", "Jane Doe", "Hello, world!", orig_left, orig_top, scal
```

```
e_none, 1500, previous, inherit, 3, ease_sin_out);  
    vngen_text_replace("text", "", "Jane Doe||Hello, world!", orig_left, orig_top, scale_none, 1500, fnt_Arial, c_white, 3, ease_sin_out, "en-US");  
}
```

## 6.22.10. The "vngen\_text\_destroy" Function

### Syntax:

```
vngen_text_destroy(id, transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the text to destroy (or keyword 'all' for all text)
transition	script	Sets the transition animation to perform
duration	real	Sets the length of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

### Description:

Destroys the input entity ID, freeing its resources from memory. Can also perform an exit transition animation before destroying. Once a given ID has been destroyed, it can be created again with the same ID.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

### Example:

```
vngen_event() {  
    vngen_text_destroy("text", trans_fade, 2, ease_sin_in);  
}
```

## 6.22.11. The "vngen\_text\_anim\_start" Function

### Syntax:

```
vngen_text_anim_start(id, anim, duration, loop, reverse, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the text to animate (or keyword 'all' for all text)
anim	script	The animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

## Description:

Performs a keyframe animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {  
    vngen_text_anim_start("text", anim_impact, 0.5, false, false);  
}
```

## 6.22.12. The "vngen\_text\_anim\_stop" Function

### Syntax:

```
vngen_text_anim_stop(id);
```

Argument	Type	Description
id	real/string	The ID of the text to stop animating (or keyword 'all' for all text)

### Description:

Stops the active keyframe animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

### Example:

```
vngen_event() {  
    vngen_text_anim_stop("text");  
}
```

## 6.22.13. The "vngen\_text\_deform\_start" Function

### Syntax:

```
vngen_text_deform_start(id, def, duration, loop, reverse, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the text to animate (or keyword 'all' for all text)
def	script	The deformation animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the



[ease]

integer/macro

animation in reverse keyframe order  
*Optional:* Sets the ease override for the animation script**Description:**

Performs a keyframe deformation animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vngen_event() {  
    vngen_text_deform_start("text", def_wave, 0.5, false, false);  
}
```

## 6.22.14. The "vngen\_text\_deform\_stop" Function

**Syntax:**

```
vngen_text_deform_stop(id);
```

**Argument**

id

**Type**

real/string

**Description**

The ID of the text to stop animating  
(or keyword 'all' for all text)

**Description:**

Stops the active keyframe deformation animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

**Example:**

```
vngen_event() {  
    vngen_text_deform_stop("text");  
}
```

## 6.22.15. The "vngen\_text\_shader\_start" Function

**Syntax:**

```
vngen_text_shader_start(id, shader, fade, [ease]);
```

**Argument**

id

**Type**

real/string

**Description**

The ID of the text to apply shader to  
(or keyword 'all' for all text)

shader

shader

The shader to perform

fade

real

Sets the length of time to fade shader in, in seconds

ease

integer/macro

*Optional:* Sets the ease mode for

fade transition

## Description:

Applies a shader with optional fade in transition. Shaders are written externally and can be used to modify the color and position of vertices and/or pixels.

**Important note:** Some shaders require extra input values to function properly. These values must be set externally with `vngen_set_shader_*` scripts.

See [Shaders](#) for a list of included shaders.

### Example:

```
vngen_event() {  
    vngen_text_shader_start("text", shade_sepia, 0.5);  
}
```

## 6.22.16. The "vngen\_text\_shader\_stop" Function

### Syntax:

```
vngen_text_shader_stop(id, fade, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the text to stop shader on (or keyword 'all' for all text)
fade	real	Sets the length of time to fade shader out, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

## Description:

Stops the active shader being performed on the specified entity, if any.

### Example:

```
vngen_event() {  
    vngen_text_shader_stop("text", 0.5);  
}
```

## 6.23. Label Actions

In addition to its robust text system, VNgen also includes a second type of text entity called **labels**. Although similar in many ways, labels lack the advanced effects and functionality of normal text, making them ideal for use in UI elements. Most importantly, this includes the speaker name typically listed above or below text itself.

Unlike regular text elements, labels can automatically populate with the current speaker(s), even if multiple characters are speaking at once! Like text, however, labels also store advanced stylization in memory, making it quick and easy to create unique labels for every character in your visual novels. Any time a font, gradient color, shadow color, or outline color is declared, it is automatically stored in memory and assigned to the label string. If

ever the same label string is used again, fonts and colors using the 'inherit' keyword will restore this saved data so that stylization does not have to be performed again. This is especially powerful when using labels as indicators of the current speaking character(s).

In this section we'll examine available label actions as well as special features such as style inheritance and more.

### 6.23.1. The "vnngen\_label\_create" Function

#### Syntax:

```
vnngen_label_create(id, label, x, y, z, linebreak, font, color, transition, duration,  
[ease], [lang]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new label
label	string	The text string to display as a label (or keyword 'auto' for speaker names)
x	real	The horizontal position to display the label, relative to the global offset
y	real	The vertical position to display the label, relative to the global offset
z	real	The drawing depth of the label, relative to other labels only
linebreak	real	The width in pixels before text is wrapped into a new line
font	font/macro	The font to draw text in, where <code>fnt_default</code> is default (or keyword 'inherit' for saved style)
color	color/macro	The color to draw text in, where <code>c_white</code> is default (or keyword 'inherit' for saved style)
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script
[lang]	real/string	<i>Optional:</i> Sets the language flag for the label

#### Description:

Creates a new static string of text which will be displayed until `vnngen_label_destroy` is run. Multiple labels can exist simultaneously, however no two labels may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

While labels share many properties with standard text elements, labels are static and do not affect progression, character speaking animations, or the backlog, and do not support advanced features such as a typewriter effect or markup.

While the actual label text can be anything, it is also possible to set labels to automatically display the name of the

current speaker(s) defined by other text elements. This value is updated automatically at the start of every event, meaning the label does not require replacing to reflect each new speaker.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available style modes, transition animations, and ease modes. See [Language Functions](#) for details on multi-language support.

**Example:**

```
vnngen_event() {  
    vnngen_label_create("label", auto, 100, 750, -1, 384, fnt_Arial, c_white, trans_wipe_right, 2);  
    vnngen_label_create("label", "John Doe", 100, 750, -1, 384, fnt_Arial, c_white, trans_wipe_right, 2, "en-US");  
}
```

## 6.23.2. The "vnngen\_label\_create\_ext" Function

**Syntax:**

```
vnngen_label_create_ext(id, label, xorig, yorig, x, y, z, scaling, linebreak, font, color, transition, duration, ease, [lang]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new label
label	string/macro	The text string to display as a label (or keyword 'auto' for speaker names)
xorig	real/macro	The horizontal texture offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical texture offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the label, relative to the global offset
y	real	The vertical position to display the label, relative to the global offset
z	real	The drawing depth of the label, relative to other labels only
scaling	integer/macro	Sets the automatic scaling mode for the label
linebreak	real	The width in pixels before text is wrapped into a new line
font	font/macro	The font to draw text in, where fnt_default is default (or keyword 'inherit' for saved style)
color	color/macro	The color to draw text in, where c_white is default (or keyword 'inherit' for saved style)
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
ease	integer/macro	Sets the ease override for the transition script



[lang]	real/string	<i>Optional:</i> Sets the language flag for the label
--------	-------------	---

## Description:

Creates a new static string of text with extra options which will be displayed until `vngen_label_destroy` is run. Multiple labels can exist simultaneously, however no two labels may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

While labels share many properties with standard text elements, labels are static and do not affect progression, character speaking animations, or the backlog, and do not support advanced features such as a typewriter effect or markup.

While the actual label text can be anything, it is also possible to set labels to automatically display the name of the current speaker(s) defined by other text elements. This value is updated automatically at the start of every event, meaning the label does not require replacing to reflect each new speaker.

Note that labels are displayed as a surface, or texture, and the exact dimensions of this texture may not be known. As a result, functions like 'xorig', 'yorig', and 'scaling' may not behave as expected. It is important to test your label actions and design your fonts and linebreaks to a standard which results in predictable texture dimensions.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available origin points, scaling modes, transition animations, and ease modes. See [Language Functions](#) for details on multi-language support.

### Example:

```
vngen_event() {  
    vngen_label_create_ext("label", auto, orig_left, orig_top, 100, 750, -1, scale_none, 384, fnt_Arial, c_white, trans_wipe_right, 2, ease_sin_out);  
    vngen_label_create_ext("label", "John Doe", orig_left, orig_top, 100, 750, -1, scale_none, 384, fnt_Arial, c_white, trans_wipe_right, 2, ease_sin_out, "en-US");  
}
```

## 6.23.3. The "vngen\_label\_modify\_style" Function

### Syntax:

```
vngen_label_modify_style(id, col1, col2, col3, col4, shadow, outline, alpha, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the label to modify (or keyword 'all' for all labels)
col1	color/macro	The text top-left gradient color, where <code>c_white</code> is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
col2	color/macro	The text top-right gradient color, where <code>c_white</code> is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
col3	color/macro	The text bottom-right gradient color,

col4	color/macro	where c_white is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style) The text bottom-left gradient color, where c_white is default (or keyword 'previous' for no change, or keyword 'inherit' for saved style)
shadow	color/macro	The text shadow color, where 'none' is default (or keyword 'previous' for no change, keyword 'inherit' for saved style, or 'none' for none)
outline	color/macro	The text outline color, where 'none' is default (or keyword 'previous' for no change, keyword 'inherit' for saved style, or 'none' for none)
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

## Description:

Applies a four-color gradient and transparency modifications to the input entity ID. Color values can be input using GameMaker Studio's built-in color constants or functions like VNgen's `make_color_hex_to_rgb`.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

Unlike other entities, color modifications applied to labels are applied to each individual character in the string rather than to the string as a whole. The exception to this is animations, which are applied as normal.

See [Macros & Keywords](#) for a list of available style modes and ease modes.

### Example:

```
vnngen_event() {
    vnngen_label_modify_style("label", c_red, c_blue, c_green, c_yellow, c_black, c_white, 1, 2);
}
```

## 6.23.4. The "vnngen\_label\_modify\_pos" Function

### Syntax:

```
vnngen_label_modify_pos(id, x, y, z, scale, rot, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the label to modify (or keyword 'all' for all labels)
x	real	The horizontal position to display the label, relative to the global offset
y	real	The vertical position to display the

z	real	label, relative to the global offset The drawing depth of the label, relative to other labels only
scale	real	The label scale multiplier, where 1 is default
rot	real	The label rotation, in degrees
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

**Note:** The scale *multiplier* here should not be confused with the scaling *mode* available in \*\_create\_ext functions. The scale *mode* sets the entity's default scale, while the scale *multiplier* modifies the default scale.

## Description:

Applies a new position, rotation, and scale to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {
    vngen_label_modify_pos("label", 150, 150, -1, 1.5, 45, 2);
}
```

## 6.23.5. The "vngen\_label\_modify\_ext" Function

### Syntax:

```
vngen_label_modify_ext(id, x, y, z, xscale, yscale, rot, col1, col2, col3, col4, shadow, outline, alpha, duration, ease);
```

Argument	Type	Description
id	real/string	The ID of the label to modify ( <i>or keyword 'all' for all labels</i> )
x	real	The horizontal position to display the label, relative to the global offset
y	real	The vertical position to display the label, relative to the global offset
z	real	The drawing depth of the label, relative to other labels only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The label rotation, in degrees
col1	color/macro	The text top-left gradient color, where c_white is default ( <i>or keyword 'previous' for no change, or keyword</i>

col2	color/macro	<i>'inherit'</i> for saved style) The text top-right gradient color, where <code>c_white</code> is default (or keyword <i>'previous'</i> for no change, or keyword <i>'inherit'</i> for saved style)
col3	color/macro	The text bottom-right gradient color, where <code>c_white</code> is default (or keyword <i>'previous'</i> for no change, or keyword <i>'inherit'</i> for saved style)
col4	color/macro	The text bottom-left gradient color, where <code>c_white</code> is default (or keyword <i>'previous'</i> for no change, or keyword <i>'inherit'</i> for saved style)
shadow	color/macro	The text shadow color, where <i>'none'</i> is default (or keyword <i>'previous'</i> for no change, keyword <i>'inherit'</i> for saved style, or <i>'none'</i> for none)
outline	color/macro	The text outline color, where <i>'none'</i> is default (or keyword <i>'previous'</i> for no change, keyword <i>'inherit'</i> for saved style, or <i>'none'</i> for none)
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent
duration	real	Sets the length of the modification transition, in seconds
ease	integer/macro	Sets the ease mode for the modification transition

## Description:

Combines `vnngen_label_modify_style` and `vnngen_label_modify_pos` with a few extra options, applying a new position, scale, rotation, four-color gradient, and transparency modifications to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

Unlike other entities, color modifications applied to labels are applied to each individual character in the string rather than to the string as a whole. The exception to this is animations, which are applied as normal.

See [Macros & Keywords](#) for a list of available style modes and ease modes.

### Example:

```
vnngen_event() {
    vnngen_label_modify_ext("label", 0, 0, -1, 1.5, 1.5, 0, c_yellow, c_yellow, c_oran
ge, c_orange, 1, 2, ease_sin_out);
}
```

## 6.23.6. The "vnngen\_label\_modify\_direct" Function

### Syntax:

```
vnngen_label_modify_direct(id, x, y, z, xscale, yscale, rot);
```



Argument	Type	Description
id	real/string	The ID of the label to modify ( <i>or keyword 'all' for all labels</i> )
x	real	The horizontal position to display the label, relative to the global offset
y	real	The vertical position to display the label, relative to the global offset
z	real	The drawing depth of the label, relative to other labels only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The label rotation, in degrees

### Description:

Applies a new position, orientation, and scale to the input entity ID directly, outside the Q-script loop.

All modifications made with this script are permanent and will persist until another modification is performed.

As **this script is not an action**, care should be taken in deciding when and where to execute it. If both `*_modify` and `*_modify_direct` scripts are executed together, whichever is run last will override the other.

#### Example:

```
var mx = window_mouse_get_x() - (window_get_width()*0.5);
var my = window_mouse_get_y() - (window_get_height()*0.5);

if (mouse_check_button(mb_left)) {
    vngen_label_modify_direct("label", mx, my, 0, 1, 1, 0);
}
```

This will map the label to the mouse, allowing the user to click and drag the label to a new position on the screen.

## 6.23.7. The "vngen\_label\_replace" Function

### Syntax:

```
vngen_label_replace(id, label, font, color, duration, [ease], [lang]);
```

Argument	Type	Description
id	real/string	The ID of the label to replace
label	string/macro	The text string to display as a label ( <i>or keyword 'auto' for speaker names</i> )
font	font/macro	The font to draw text in, where <code>fnt_default</code> is default ( <i>or keyword 'previous' for no change, or keyword 'inherit' for saved style</i> )
color	color/macro	The color to draw text in, where <code>c_white</code> is default ( <i>or keyword</i>

duration	real	<i>'previous' for no change, or keyword 'inherit' for saved style)</i> Sets the duration of the fade transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in
[lang]	real/string	<i>Optional:</i> Sets the language flag for the label

## Description:

Replaces the input entity ID with a new string of text and fades the old label to the new one over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available style modes and ease modes.

### Example:

```
vngen_event() {
    vngen_label_replace("label", auto, previous, inherit, 3);
    vngen_label_replace("label", "Jane Doe", fnt_Arial, c_white, 3, "en-US");
}
```

## 6.23.8. The "vngen\_label\_replace\_ext" Function

### Syntax:

```
vngen_label_replace_ext(id, label, xorig, yorig, scaling, linebreak, font, color, duration, ease, [lang]);
```

Argument	Type	Description
id	real/string	The ID of the label to replace
label	string/macro	The text string to display as a label (or keyword <i>'auto'</i> for speaker names)
xorig	real/macro	The horizontal texture offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical texture offset, or origin point, relative to the top-left corner
scaling	integer/macro	Sets the automatic scaling mode for the label
linebreak	real	The width in pixels before text is wrapped into a new line
font	font/macro	The font to draw text in, where <code>fnt_default</code> is default (or keyword <i>'previous'</i> for no change, or keyword <i>'inherit'</i> for saved style)
color	color/macro	The color to draw text in, where <code>c_white</code> is default (or keyword <i>'previous'</i> for no change, or keyword

---

duration	real	<i>'inherit' for saved style)</i> Sets the duration of the fade transition, in seconds
ease	integer/macro	Sets the ease mode to perform the transition in
[lang]	real/string	<i>Optional:</i> Sets the language flag for the label

### Description:

Replaces the input entity ID with extra options and fades the old label to the new one over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

Note that labels are displayed as a surface, or texture, and the exact dimensions of this texture may not be known. As a result, functions like 'xorig', 'yorig', and 'scaling' may not behave as expected. It is important to test your label actions and design your fonts and linebreaks to a standard which results in predictable texture dimensions.

See [Macros & Keywords](#) for a list of available style modes and transition animations. See [Language Functions](#) for details on multi-language support.

### Example:

```
vngen_event() {  
    vngen_label_replace("label", auto, orig_left, orig_top, scale_none, 384, previous  
    , inherit, 3, ease_sin_out);  
    vngen_label_replace("label", "John Doe", orig_left, orig_top, scale_none, 384, fn  
t_Arial, c_white, 3, ease_sin_out, "en-US");  
}
```

## 6.23.9. The "vngen\_label\_destroy" Function

### Syntax:

```
vngen_label_destroy(id, transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the label to destroy ( <i>or keyword 'all' for all labels</i> )
transition	script	Sets the transition animation to perform
duration	real	Sets the length of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

### Description:

Destroys the input entity ID, freeing its resources from memory. Can also perform an exit transition animation before destroying. Once a given ID has been destroyed, it can be created again with the same ID.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

**Example:**

```
vngen_event() {  
    vngen_label_destroy("label", trans_fade, 2, ease_sin_in);  
}
```

## 6.23.10. The "vngen\_label\_anim\_start" Function

**Syntax:**

```
vngen_label_anim_start(id, anim, duration, loop, reverse, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the label to animate ( <i>or keyword 'all' for all labels</i> )
anim	script	The animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

**Description:**

Performs a keyframe animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vngen_event() {  
    vngen_label_anim_start("label", anim_shake, 1, false, false);  
}
```

## 6.23.11. The "vngen\_label\_anim\_stop" Function

**Syntax:**

```
vngen_label_anim_stop(id);
```

Argument	Type	Description
id	real/string	The ID of the label to stop animating ( <i>or keyword 'all' for all labels</i> )

**Description:**

Stops the active keyframe animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

**Example:**

```
vngen_event() {  
    vngen_label_anim_stop("label");  
}
```

## 6.23.12. The "vngen\_label\_deform\_start" Function

**Syntax:**

```
vngen_label_deform_start(id, def, duration, loop, reverse, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the label to animate (or keyword 'all' for all labels)
def	script	The deformation animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

**Description:**

Performs a keyframe deformation animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vngen_event() {  
    vngen_label_deform_start("label", def_wave, 0.5, false, false);  
}
```

## 6.23.13. The "vngen\_label\_deform\_stop" Function

**Syntax:**

```
vngen_label_deform_stop(id);
```

Argument	Type	Description
id	real/string	The ID of the label to stop animating (or keyword 'all' for all labels)

## Description:

Stops the active keyframe deformation animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

### Example:

```
vngen_event() {  
    vngen_label_deform_stop("label");  
}
```

## 6.23.14. The "vngen\_label\_shader\_start" Function

### Syntax:

```
vngen_label_shader_start(id, shader, fade, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the label to apply shader to (or keyword 'all' for all labels)
shader	shader	The shader to perform
fade	real	Sets the length of time to fade shader in, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

### Description:

Applies a shader with optional fade in transition. Shaders are written externally and can be used to modify the color and position of vertices and/or pixels.

**Important note:** Some shaders require extra input values to function properly. These values must be set externally with `vngen_set_shader_*` scripts.

See [Shaders](#) for a list of included shaders.

### Example:

```
vngen_event() {  
    vngen_label_shader_start("label", shade_sepia, 0.5);  
}
```

## 6.23.15. The "vngen\_label\_shader\_stop" Function

### Syntax:

```
vngen_label_shader_stop(id, fade, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the label to stop shader on (or keyword 'all' for all labels)

fade	real	Sets the length of time to fade shader out, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

### Description:

Stops the active shader being performed on the specified entity, if any.

#### Example:

```
vngen_event() {
    vngen_label_shader_stop("label", 0.5);
}
```

## 6.24. Prompt Actions

In VNgen terminology, prompts are animated icons which sit on top of the textbox and display different sprites to indicate to the user whether the text typewriter effect is in-progress, waiting for user input, or in auto mode. Although using prompts is entirely optional, taking advantage of them is a simple and effective way to add visual flair and, more importantly, visual feedback to your UI.

Prompts support the full range of actions available to other entities, and can even be customized for each character on-screen, taking them far above and beyond being just a simple decoration.

In this section we'll examine available prompt actions covering five basic operations: create, modify, replace, destroy, and animate.

### 6.24.1. The "vngen\_prompt\_create" Function

#### Syntax:

```
vngen_prompt_create(id, sprite_idle, sprite_active, sprite_auto, x, y, z, transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new prompt
sprite_idle	sprite	The sprite to draw as a prompt when waiting for user input
sprite_active	sprite	The sprite to draw as a prompt when the typewriter effect is active ( <i>or keyword 'none' for none</i> )
sprite_auto	sprite	The sprite to draw when auto mode is enabled ( <i>or keyword 'none' for none</i> )
x	real/macro	The horizontal position to display the prompt, relative to the global offset ( <i>or keyword 'auto' to position by text</i> )
y	real/macro	The vertical position to display the prompt, relative to the global offset ( <i>or keyword 'auto' to position by text</i> )

z	real	<i>by text)</i> The drawing depth of the prompt, relative to other prompts only
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

## Description:

Creates a new prompt icon which will be displayed until `vngen_prompt_destroy` is run. Multiple prompts can exist simultaneously, however no two prompts may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

Unlike other entities, prompts feature an automatic position mode which will display prompts next to text?even while the typewriter effect is active.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

### Example:

```
vngen_event() {
    vngen_prompt_create("prompt", spr_prompt_idle, spr_prompt_active, spr_prompt_auto
, auto, auto, 0, trans_fade, 1);
}
```

## 6.24.2. The "vngen\_prompt\_create\_ext" Function

### Syntax:

```
vngen_prompt_create_ext(id, trigger, sprite_idle, sprite_active, sprite_auto, xorig,
    yorig, x, y, z, scaling, transition, duration, ease);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new prompt
trigger	name	The character name to use as a trigger for displaying the prompt ( <i>or keyword 'any' for all characters</i> )
sprite_idle	sprite	The sprite to draw as a prompt when waiting for user input
sprite_active	sprite	The sprite to draw as a prompt when the typewriter effect is active ( <i>or keyword 'none' for none</i> )
sprite_auto	sprite	The sprite to draw when auto mode is enabled ( <i>or keyword 'none' for none</i> )
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the



y	real	prompt, relative to the global offset The vertical position to display the prompt, relative to the global offset
z	real	The drawing depth of the prompt, relative to other prompts only
scaling	integer/macro	Sets the automatic scaling mode for the prompt
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animation, in seconds
ease	integer/macro	Sets the ease override for the transition script

## Description:

Creates a new prompt icon with extra options which will be displayed until `vngen_prompt_destroy` is run. Multiple prompts can exist simultaneously, however no two prompts may share the same ID. VNgen entity IDs are arbitrary and most can be either numbers or strings, but bear in mind that -1 is reserved as 'null' and cannot be used as an ID.

Unlike other entities, prompts feature an automatic position mode which will display prompts next to text?even while the typewriter effect is active.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available origin points, scaling modes, transition animations, and ease modes.

### Example:

```
vngen_event() {
    vngen_prompt_create_ext("prompt", "John Doe", spr_prompt_idle, spr_prompt_active,
        spr_prompt_auto, orig_left, orig_center, auto, auto, 0, scale_none, trans_fade, 1,
        ease_sin_out);
}
```

## 6.24.3. The "vngen\_prompt\_modify\_style" Function

### Syntax:

```
vngen_prompt_modify_style(id, col1, col2, col3, col4, alpha, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the prompt to modify ( <i>or keyword 'all' for all prompts</i> )
col1	color	The top-left gradient color, where <code>c_white</code> is default
col2	color	The top-right gradient color, where <code>c_white</code> is default
col3	color	The bottom-right gradient color, where <code>c_white</code> is default
col4	color	The bottom-left gradient color, where <code>c_white</code> is default
alpha	real (0-1)	The alpha transparency value, where 1 is default and 0 is fully transparent



duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

### Description:

Applies a four-color gradient and transparency modifications to the input entity ID. Color values can be input using GameMaker Studio's built-in color constants or functions like VNgen's `make_color_hex_to_rgb`.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {  
    vngen_prompt_modify_style("prompt", c_yellow, c_yellow, c_orange, c_orange, 1, 2)  
;  
}
```

This will apply a yellow-orange gradient to the prompt, perhaps to represent a character whose theme color is orange.

## 6.24.4. The "vngen\_prompt\_modify\_pos" Function

### Syntax:

```
vngen_prompt_modify_pos(id, x, y, z, scale, rot, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the prompt to modify ( <i>or keyword 'all' for all prompts</i> )
x	real	The horizontal position to display the prompt, relative to the global offset ( <i>or keyword 'auto' to position by text</i> )
y	real	The vertical position to display the prompt, relative to the global offset ( <i>or keyword 'auto' to position by text</i> )
z	real	The drawing depth of the prompt, relative to other prompts only
scale	real	The prompt scale multiplier, where 1 is default
rot	real	The prompt rotation, in degrees
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

**Note:** The scale *multiplier* here should not be confused with the scaling *mode* available in \*\_create\_ext functions. The scale *mode* sets the entity's default scale, while the scale *multiplier* modifies the default scale.

## Description:

Applies a new position, rotation, and scale to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {
    vngen_prompt_modify_pos("prompt", 1600, 900, -1, 1.5, 45, 2);
}
```

## 6.24.5. The "vngen\_prompt\_modify\_ext" Function

### Syntax:

```
vngen_prompt_modify_ext(id, x, y, z, xscale, yscale, rot, col1, col2, col3, col4, alpha, duration, ease);
```

Argument	Type	Description
id	real/string	The ID of the prompt to modify (or keyword 'all' for all prompts)
x	real	The horizontal position to display the prompt, relative to the global offset (or keyword 'auto' to position by text)
y	real	The vertical position to display the prompt, relative to the global offset (or keyword 'auto' to position by text)
z	real	The drawing depth of the prompt, relative to other prompts only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The prompt rotation, in degrees
col1	color	The top-left gradient color, where c_white is default
col2	color	The top-right gradient color, where c_white is default
col3	color	The bottom-right gradient color, where c_white is default
col4	color	The bottom-left gradient color, where c_white is default
alpha	real (0-1)	The alpha transparency value, where

duration	real	1 is default and 0 is fully transparent Sets the length of the modification transition, in seconds
ease	integer/macro	Sets the ease mode for the modification transition

### Description:

Combines `vnngen_prompt_modify_style` and `vnngen_prompt_modify_pos` with a few extra options, applying a new position, scale, rotation, four-color gradient, and transparency modifications to the input entity ID.

All modifications made with this script are permanent and will persist until another modification is performed. This script cannot be performed simultaneously with other modification actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

#### Example:

```
vnngen_event() {
    vnngen_prompt_modify_ext("prompt", 1600, 900, -1, 1.5, 1.5, 0, c_yellow, c_yellow,
    c_orange, c_orange, 1, 2, ease_sin_out);
}
```

## 6.24.6. The "vnngen\_prompt\_modify\_direct" Function

### Syntax:

```
vnngen_prompt_modify_direct(id, x, y, z, xscale, yscale, rot);
```

Argument	Type	Description
id	real/string	The ID of the prompt to modify ( <i>or keyword 'all' for all prompts</i> )
x	real	The horizontal position to display the prompt, relative to the global offset
y	real	The vertical position to display the prompt, relative to the global offset
z	real	The drawing depth of the prompt, relative to other prompts only
xscale	real	The horizontal scale multiplier, where 1 is default
yscale	real	The vertical scale multiplier, where 1 is default
rot	real	The prompt rotation, in degrees

### Description:

Applies a new position, orientation, and scale to the input entity ID directly, outside the Q-script loop.

All modifications made with this script are permanent and will persist until another modification is performed.

As **this script is not an action**, care should be taken in deciding when and where to execute it. If both `*_modify` and `*_modify_direct` scripts are executed together, whichever is run last will override the other.

#### Example:

```
var mx = window_mouse_get_x() - (window_get_width()*0.5);
var my = window_mouse_get_y() - (window_get_height()*0.5);

if (mouse_check_button(mb_left)) {
    vngen_prompt_modify_direct("prompt", mx, my, 0, 1, 1, 0);
}
```

This will map the prompt to the mouse, allowing the user to click and drag the prompt to a new position on the screen.

## 6.24.7. The "vngen\_prompt\_replace" Function

### Syntax:

```
vngen_prompt_replace(id, sprite_idle, sprite_active, sprite_auto, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the prompt to replace
sprite_idle	sprite	The new sprite to draw as a prompt when waiting for user input
sprite_active	sprite	The new sprite to draw as a prompt when the typewriter effect is active (or keyword 'none' for none)
sprite_auto	sprite	The new sprite to draw when auto mode is enabled (or keyword 'none' for none)
duration	real	Sets the duration of the fade transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

### Description:

Replaces the input entity ID with new sprites and fades the old sprites to the new ones over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
if (vngen_event()) {
    vngen_prompt_replace("prompt", spr_new, 1);
}
```

## 6.24.8. The "vngen\_prompt\_replace\_ext" Function

## Syntax:

```
vngen_prompt_replace_ext(id, trigger, sprite_idle, sprite_active, sprite_auto xorig,  
    yorig, scaling, duration, ease);
```

Argument	Type	Description
id	real/string	The ID of the prompt to replace
trigger	string	The character name to use as a new trigger for displaying the prompt ( <i>or keyword 'any' for all characters</i> )
sprite_idle	sprite	The new sprite to draw as a prompt when waiting for user input
sprite_active	sprite	The new sprite to draw as a prompt when the typewriter effect is active ( <i>or keyword 'none' for none</i> )
sprite_auto	sprite	The new sprite to draw when auto mode is enabled ( <i>or keyword 'none' for none</i> )
xorig	real/macro	The new horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The new vertical sprite offset, or origin point, relative to the top-left corner
scaling	integer/macro	Sets the new automatic scaling mode for the prompt
duration	real	Sets the duration of the fade transition, in seconds
ease	integer/macro	Sets the ease mode to perform the fade transition in

## Description:

Replaces the input entity ID with new sprites and extra sprite properties and fades the old sprites to the new ones over the input duration.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available origin points, scaling modes, and ease modes.

### Example:

```
vngen_event() {  
    vngen_prompt_replace_ext("prompt", "John Doe", spr_prompt_idle, spr_prompt_active  
    , spr_prompt_auto, orig_left, orig_center, scale_none, 1, ease_sin_in_out);  
}
```

## 6.24.9. The "vngen\_prompt\_destroy" Function

### Syntax:

```
vngen_prompt_destroy(id, transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the prompt to destroy (or keyword 'all' for all prompts)
transition	script	Sets the transition animation to perform
duration	real	Sets the length of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

### Description:

Destroys the input entity ID, freeing its resources from memory. Can also perform an exit transition animation before destroying. Once a given ID has been destroyed, it can be created again with the same ID.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

#### Example:

```
vnngen_event() {
    vnngen_prompt_destroy("prompt", trans_fade, 2, ease_sin_in);
}
```

## 6.24.10. The "vnngen\_prompt\_anim\_start" Function

### Syntax:

```
vnngen_prompt_anim_start(id, anim, duration, loop, reverse, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the prompt to animate (or keyword 'all' for all prompts)
anim	script	The animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

### Description:

Performs a keyframe animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.

#### Example:

```
vnngen_event() {
    vnngen_prompt_anim_start("prompt", anim_wobble, 1, false, false);
}
```

```
}
```

### 6.24.11. The "vngen\_prompt\_anim\_stop" Function

#### Syntax:

```
vngen_prompt_anim_stop(id);
```

Argument	Type	Description
id	real/string	The ID of the prompt to stop animating ( <i>or keyword 'all' for all prompts</i> )

#### Description:

Stops the active keyframe animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

#### Example:

```
vngen_event() {  
    vngen_prompt_anim_stop("prompt");  
}
```

### 6.24.12. The "vngen\_prompt\_deform\_start" Function

#### Syntax:

```
vngen_prompt_deform_start(id, def, duration, loop, reverse, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the prompt to animate ( <i>or keyword 'all' for all prompts</i> )
def	script	The deformation animation script to perform
duration	real	Sets the duration of the entire animation
loop	boolean	Enables or disables looping the animation
reverse	boolean	Enables or disables performing the animation in reverse keyframe order
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the animation script

#### Description:

Performs a keyframe deformation animation script on the input entity ID with the input duration. As animations are temporary and relative, animations and modifications can be performed simultaneously.

See [Included Animations](#) for a list of included animation scripts and how to create your own, and [Macros & Keywords](#) for a list of available ease modes.





**Example:**

```
vngen_event() {  
    vngen_prompt_deform_start("prompt", def_wave, 0.5, false, false);  
}
```

## 6.24.13. The "vngen\_prompt\_deform\_stop" Function

**Syntax:**

```
vngen_prompt_deform_stop(id);
```

Argument	Type	Description
id	real/string	The ID of the prompt to stop animating (or keyword 'all' for all prompts)

**Description:**

Stops the active keyframe deformation animation being performed on the input entity ID, if any. Note that this script only needs to be run to stop *looped* animations.

**Example:**

```
vngen_event() {  
    vngen_prompt_deform_stop("prompt");  
}
```

## 6.24.14. The "vngen\_prompt\_shader\_start" Function

**Syntax:**

```
vngen_prompt_shader_start(id, shader, fade, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the prompt to apply shader to (or keyword 'all' for all prompts)
shader	shader	The shader to perform
fade	real	Sets the length of time to fade shader in, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

**Description:**

Applies a shader with optional fade in transition. Shaders are written externally and can be used to modify the color and position of vertices and/or pixels.

**Important note:** Some shaders require extra input values to function properly. These values must be set externally with `vngen_set_shader_*` scripts.

See [Shaders](#) for a list of included shaders.

**Example:**

```
vngen_event() {  
    vngen_prompt_shader_start("prompt", shade_sepia, 0.5);  
}
```

## 6.24.15. The "vngen\_prompt\_shader\_stop" Function

**Syntax:**

```
vngen_prompt_shader_stop(id, fade, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the prompt to stop shader on (or keyword 'all' for all prompts)
fade	real	Sets the length of time to fade shader out, in seconds
ease	integer/macro	<i>Optional:</i> Sets the ease mode for fade transition

**Description:**

Stops the active shader being performed on the specified entity, if any.

**Example:**

```
vngen_event() {  
    vngen_prompt_shader_stop("prompt", 0.5);  
}
```

## 6.25. Button Actions

In addition to specialized forms of navigation and UI, VNgen features a general purpose buttons system which can be used for both in-game functions and main menus themselves. Taking action based on which button is selected might seem complex at first, but the open-ended nature of VNgen buttons is precisely what makes them so powerful.

Simply put, there is no predetermined outcome when a user selects a given button. Rather, that button's ID is stored in memory where it can be read and used in conditional statements like 'if' and 'switch' to execute code of your own based on the result. Check out the [Getting Started](#) guide for a full breakdown of this process using options as an example.

In this section we'll examine available button functions so you'll be off to creating menus in no time!

**Tip:** Buttons ignore the engine pause state, meaning you can even create pause menus with them!

### 6.25.1. The "vngen\_button\_create" Function

## Syntax:

```
vngen_button_create(id, text, sprite, spr_hover, spr_select, x, y, z, font, color, transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new button
text	string	The button text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a button background
spr_hover	sprite/macro	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_active	sprite/macro	The sprite to display as a background when selected (or keyword 'none' for default)
x	real	The horizontal position to display the button, relative to the GUI layer
y	real	The vertical position to display the button, relative to the GUI layer
z	real	The drawing depth and list order of the button, relative to other log buttons only
font	font	The font to draw text in, where fnt_default is default
color	color	The color to draw text in, where c_white is default
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animations, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the button animations

## Description:

Creates a new button which will be displayed until `vngen_button_destroy` is run. The chosen button ID will be recorded as the selected choice until cleared with `vngen_get_button` or `vngen_button_clear`. Buttons can be selected by mouse/touch, or by other input devices with `vngen_button_nav` and `vngen_button_select`.

Unlike other entities, button z-index determines not just drawing depth, but also navigation order. Buttons with lower z-index will be considered lower in the list of buttons. If z-index is equal between two or more buttons, navigation order will automatically be set by reverse creation order, but this may or may not be visually correct. Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

## Example:

```
vngen_button_create("btn", "Click me!", spr_btn, spr_btn_hover, spr_btn_active, 256, 512, -1, fnt_Arial, c_white, trans_fade, 1);
```

## 6.25.2. The "vngen\_button\_create\_ext" Function

### Syntax:

```
vngen_button_create_ext(id, text, sprite, spr_hover, spr_select, xorig, yorig, x, y, z, tx, ty, scaling, font, color, snd_hover, snd_select, transition, duration, ease)
;
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new button
text	string	The button text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a button background
spr_hover	sprite/macro	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_active	sprite/macro	The sprite to display as a background when selected (or keyword 'none' for default)
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the button
y	real	The vertical position to display the button
z	real	The drawing depth and list order of the button, relative to other buttons only
tx	real/macro	The horizontal position to display text, relative to the button sprite top-left corner (or keyword 'auto' for center)
ty	real/macro	The vertical position to display text, relative to the button sprite top-left corner (or keyword 'auto' for center)
scaling	integer/macro	Sets the automatic scaling mode for the button
font	font	The font to draw text in, where fnt_default is default
color	color	The color to draw text in, where c_white is default
snd_hover	sound	The sound to play when the button is hovered (or keyword 'none' for none)
snd_select	sound	The sound to play when the button is selected (or keyword 'none' for none)
transition	script	Sets the transition animation to perform

duration	real	Sets the duration of the transition animation, in seconds
ease	integer/macro	Sets the ease override for all button animations

### Description:

Creates a new button with extra features which will be displayed until `vngen_button_destroy` is run. The chosen button ID will be recorded as the selected choice until cleared with `vngen_get_button` or `vngen_button_clear`. Buttons can be selected by mouse/touch, or by other input devices with `vngen_button_nav` and `vngen_button_select`.

Unlike other entities, button z-index determines not just drawing depth, but also navigation order. Buttons with lower z-index will be considered lower in the list of buttons. If z-index is equal between two or more buttons, navigation order will automatically be set by *reverse* creation order, but this may or may not be visually correct. Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

### Example:

```
vngen_button_create_ext("btn", "Click me!", spr_btn, spr_btn_hover, spr_btn_active,
    orig_center, orig_center, 256, 512, -1, 72, 16, scale_none, fnt_Arial, c_white, snd_hover,
    snd_select, trans_fade, 1, ease_sin_out);
```

## 6.25.3. The "vngen\_button\_create\_transformed" Function

### Syntax:

```
vngen_button_create_transformed(id, text, sprite, spr_hover, spr_select, x, y, z, font,
    color, hov_x, hov_y, hov_scale, hov_color, sel_x, sel_y, sel_scale, sel_color, sel_duration,
    transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new button
text	string	The button text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a button background
spr_hover	sprite/macro	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_active	sprite/macro	The sprite to display as a background when selected (or keyword 'none' for default)
x	real	The horizontal position to display the button
y	real	The vertical position to display the button
z	real	The drawing depth and list order of the button, relative to other buttons

font	font	only The font to draw text in, where <code>fnt_default</code> is default
color	color	The color to draw text in, where <code>c_white</code> is default
hov_x	real	The horizontal offset to shift the button when hovered
hov_y	real	The vertical offset to shift the button when hovered
hov_scale	real	The scale multiplier to shift the button when hovered
hov_color	color	The color to draw text in when hovered
sel_x	real	The horizontal offset to shift the button when selected
sel_y	real	The vertical offset to shift the button when selected
sel_scale	real	The scale multiplier to shift the button when selected
sel_color	color	The color to draw text in when selected
sel_duration	real	Sets the duration of hover/select animations, in seconds
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animations, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for all button animations

## Description:

Creates a new button with hover and select transformations which will be displayed until `vnngen_button_destroy` is run. The chosen button ID will be recorded as the selected choice until cleared with `vnngen_get_button` or `vnngen_button_clear`. Buttons can selected by mouse/touch, or by other input devices with `vnngen_button_nav` and `vnngen_button_select`.

Unlike other entities, button z-index determines not just drawing depth, but also navigation order. Buttons with lower z-index will be considered lower in the list of buttons. If z-index is equal between two or more buttons, navigation order will automatically be set by *reverse* creation order, but this may or may not be visually correct. Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

### Example:

```
vnngen_button_create_transformed("btn", "Click me!", spr_btn, spr_btn_hover, spr_btn_active, 256, 512, -1, fnt_Arial, c_white, 32, 0, 1.25, c_black, 0, 0, 1, c_yellow, 0.1, trans_fade, 1);
```

## 6.25.4. The "vnngen\_button\_create\_ext\_transformed" Function

## Syntax:

```
vnngen_button_create_ext_transformed(id, text, sprite, spr_hover, spr_select, xorig, yorig, x, y, z, tx, ty, scaling, font, color, hov_x, hov_y, hov_xscale, hov_yscale, hov_color, sel_x, sel_y, sel_xscale, sel_yscale, sel_color, sel_duration, snd_hover, snd_select, transition, duration, ease);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new button
text	string	The button text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a button background
spr_hover	sprite/macro	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_active	sprite/macro	The sprite to display as a background when selected (or keyword 'none' for default)
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the button
y	real	The vertical position to display the button
z	real	The drawing depth and list order of the button, relative to other buttons only
tx	real/macro	The horizontal position to display text, relative to the button sprite top-left corner (or keyword 'auto' for center)
ty	real/macro	The vertical position to display text, relative to the button sprite top-left corner (or keyword 'auto' for center)
scaling	integer/macro	Sets the automatic scaling mode for the button
font	font	The font to draw text in, where fnt_default is default
color	color	The color to draw text in, where c_white is default
hov_x	real	The horizontal offset to shift the button when hovered
hov_y	real	The vertical offset to shift the button when hovered
hov_xscale	real	The horizontal scale multiplier to shift the button when hovered
hov_yscale	real	The vertical scale multiplier to shift the button when hovered
hov_color	color	The color to draw text in when hovered
sel_x	real	The horizontal offset to shift the button when selected

sel_y	real	The vertical offset to shift the button when selected
sel_xscale	real	The horizontal scale multiplier to shift the button when selected
sel_yscale	real	The vertical scale multiplier to shift the button when selected
sel_color	color	The color to draw text in when selected
sel_duration	real	Sets the duration of hover/select animations, in seconds
snd_hover	sound	The sound to play when the button is hovered (or keyword 'none' for none)
snd_select	sound	The sound to play when the button is selected (or keyword 'none' for none)
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animations, in seconds
ease	integer/macro	Sets the ease override for all button animations

## Description:

Creates a new button with extra features and hover and select transformations which will be displayed until `vnngen_button_destroy` is run. The chosen button ID will be recorded as the selected choice until cleared with `vnngen_get_button` or `vnngen_button_clear`. Buttons can be selected by mouse/touch, or by other input devices with `vnngen_button_nav` and `vnngen_button_select`.

Unlike other entities, button z-index determines not just drawing depth, but also navigation order. Buttons with lower z-index will be considered lower in the list of buttons. If z-index is equal between two or more buttons, navigation order will automatically be set by reverse creation order, but this may or may not be visually correct. Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

## Example:

```
vnngen_button_create_ext_transformed("btn", "Click me!", spr_btn, spr_btn_hover, spr_btn_active, orig_center, orig_center, 256, 512, -1, 72, 16, fnt_Arial, c_white, 32, 0, 1.25, 1.25, c_black, 0, 0, 1, 1, c_yellow, 0.1, snd_hover, snd_select, trans_fade, 1, ease_sin_out);
```

## 6.25.5. The "vnngen\_button\_destroy" Function

### Syntax:

```
vnngen_button_destroy(id, transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The ID of the button to destroy (or



---

transition	script	<i>keyword 'all' for all buttons)</i> Sets the transition animation to perform
duration	real	Sets the length of the transition animation, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

### Description:

Destroys the input entity ID, freeing its resources from memory. Can also perform an exit transition animation before destroying. Once a given ID has been destroyed, it can be created again with the same ID.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

### Example:

```
vngen_event() {  
    vngen_button_destroy("btn", trans_fade, 1, ease_sin_out);  
}
```

## 6.25.6. The "vngen\_button\_clear" Function

### Syntax:

```
vngen_button_clear();
```

Argument	Type	Description
N/A	N/A	No arguments

### Description:

Manually clears button **results** (not data) to prevent repeat execution of button-based triggers. As this functionality is also handled by `vngen_get_button`, this script should not typically need to be run except in special cases.

This script is not an action.

### Example:

```
if (vngen_get_button(false) == "btn") {  
    myvar += 1;  
    vngen_button_clear();  
}
```

## 6.25.7. The "vngen\_get\_button" Function

### Syntax:

```
vngen_get_button([clear]);
```

Argument	Type	Description
----------	------	-------------

[clear]	boolean	<i>Optional:</i> Enables or disables clearing the result from memory after returning it (enabled by default)
---------	---------	--

### Description:

When run, this script will return the result of the most recently-selected button, which is stored as the ID of the selected button. If buttons exist but none has been selected, -1 will be returned instead.

This script is intended to be used in conditional statements such as 'if' and 'switch' to take action based on the selected button. In order to prevent this conditional statement from repeatedly executing code, button results will be cleared from memory after being returned. To disable this behavior, it is possible to set the 'clear' argument to 'false', in which case button results can be cleared later with `vnngen_button_clear`.

#### Example:

```
if (vnngen_get_button(false) == "btn") {  
    //Action  
}  
switch (vnngen_get_button()) {  
    case "btn1": //Action; break;  
    case "btn2": //Action; break;  
}
```

## 6.25.8. The "vnngen\_do\_button\_nav" Function

### Syntax:

```
vnngen_do_button_nav(amount);
```

Argument	Type	Description
amount	integer	The number of buttons to scroll

### Description:

Navigates through a linear list of buttons, if any exist. The amount of buttons scrolled is arbitrary, where negative values scroll up and positive values scroll down. If a hover sound has been assigned in `vnngen_button_create_ext`, it will be played upon running this script. If the input amount exceeds the size of the buttons menu, the selection will loop back around to the other side.

Note that navigation order is determined by the button z-index, and therefore improper configuration of options can result in unexpected navigation with this script.

This script is not an action, and is intended to be run in keyboard and gamepad input events. Mouse and touch support are built-in and do not require `vnngen_do_button_nav` for navigation.

#### Example:

```
//Scroll up  
vnngen_do_button_nav(-1);  
//Scroll down  
vnngen_do_button_nav(1);
```

## 6.25.9. The "vngen\_is\_button\_hovered" Function

### Syntax:

```
vngen_is_button_hovered(id);
```

Argument	Type	Description
id	real/string	The button ID to check

### Description:

Checks whether or not the specified button is hovered (either by mouse/touch or `vngen_do_button_nav`) and returns 'true' or 'false'.

### Example:

```
if (vngen_is_button_hovered("save")) {  
    effect_create_above(ef_ellipse, mouse_x, mouse_y, 2, c_white);  
}
```

## 6.25.10. The "vngen\_do\_button\_select" Function

### Syntax:

```
vngen_do_button_select([id]);
```

Argument	Type	Description
id	real/string	<i>Optional:</i> The button ID to select (default: current highlighted button)

### Description:

Selects the currently-highlighted button, if any, and records its ID. If a selection sound has been assigned in `vngen_button_create_ext`, it will be played upon running this script.

If a button ID is supplied, it will be selected directly, ignoring navigation.

This script is not an action, and is intended to be run in keyboard and gamepad input events. Mouse and touch support are built-in and do not require `vngen_do_button_select` to make selections.

### Example:

```
if (keyboard_check_released(vk_enter)) {  
    vngen_do_button_select();  
}  
  
if (keyboard_check_released(ord("2"))) {  
    vngen_do_button_select("btn2");  
}
```

## 6.25.11. The "vngen\_is\_button\_selected" Function

**Syntax:**

```
vnngen_is_button_selected(id);
```

Argument	Type	Description
id	real/string	The log button ID to check

**Description:**

Checks whether or not the specified button is selected (either by mouse/touch or `vnngen_do_button_select`) and returns 'true' or 'false'.

Note that this script is different from `vnngen_get_button` in that it will return 'true' while the button is *held down*, not just when it is released and activated.

**Example:**

```
if (vnngen_is_button_selected("save")) {  
    effect_create_above(ef_ellipse, mouse_x, mouse_y, 2, c_white);  
}
```

## 6.26. Option Actions

VNgen features a simple, yet robust options menu system which can be used for both in-game dialog choices and your main menus themselves. Taking action based on which choice is selected might seem complex at first, but the open-ended nature of VNgen options is precisely what makes them so powerful.

Simply put, there is no predetermined outcome when a user selects a given option. Rather, that option's ID is stored in memory where it can be read and used in conditional statements like 'if' and 'switch' to execute code of your own based on the result. Check out the [Getting Started](#) guide for a full breakdown of this process.

In this section we'll examine available options functions so you'll be off to creating interactive dialogs in no time!

### 6.26.1. The "vnngen\_option" Function

**Syntax:**

```
vnngen_option(id, x, y, delay_in, delay_out, snd_hover, snd_select, [lang]);
```

Argument	Type	Description
id	real/string	The identifier to use for the entire options block, as a group
x	real	The horizontal position to display the entire options block, relative to the global offset
y	real	The vertical position to display the entire options block, relative to the global offset
delay_in	real	The length of time in seconds to delay displaying options
delay_out	real	The length of time in seconds to delay destroying options



snd_hover	sound	The sound to play when an option is hovered (or keyword 'none' for none)
snd_select	sound	The sound to play when an option is selected (or keyword 'none' for none)
[lang]	real/string	Optional: Sets the language flag for the entire options block

### Description:

Initializes a segment of code containing VNgen options. Once this script has been run, as many options as desired can be executed inside it. While it may behave like an event, this script is itself an action and will pause all further actions in the current event until a selection has been made.

As of version 1.0.7, options have both a block (or group) ID and individual option IDs. The ID of previously-selected options can be recalled at any time by supplying the option block ID in the `vnngen_get_option` function.

For the sake of performance, `vnngen_option` is intended to be used within an 'if' statement. This guarantees that individual options within will only be performed while the option block is active.

See [Language Functions](#) for details on multi-language support.

#### Example:

```
if vnngen_event() {  
    if vnngen_option("op_choice", view_wview[0]*0.5, view_hview[0]*0.5, 0.5, 0.5, snd_  
hover, snd_select) {  
        //Options  
    }  
}
```

## 6.26.2. The "vnngen\_option\_create" Function

### Syntax:

```
vnngen_option_create(id, text, sprite, spr_hover, spr_select, x, y, z, font, color, t  
ransition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new option
text	string	The option text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a button background
spr_hover	sprite	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_select	sprite	The sprite to display as a background when selected (or keyword 'none' for default)
x	real	The horizontal position to display the

		option, relative to the option block offset
y	real	The vertical position to display the option, relative to the option block offset
z	real	The drawing depth and list order of the option, relative to other options only
font	font	The font to draw text in, where <code>fnt_default</code> is default
color	color	The color to draw text in, where <code>c_white</code> is default
transition	script	Sets the transition animation to perform when created and destroyed
duration	real	Sets the duration of the transition animations, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for the transition script

## Description:

Creates a new option which will be displayed until any option in the current option block has been selected. The chosen option ID will be recorded as the selected choice. Options can be selected by mouse/touch, or by other input devices with `vnngen_option_nav` and `vnngen_option_select`.

Unlike other entities, option z-index determines not just drawing depth, but also navigation order. Options with lower z-index will be considered lower in the list of options. If z-index is equal between two or more options, navigation order will automatically be set by *reverse* creation order, but this may or may not be visually correct. Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

Note that although this script is an action, it cannot be executed outside of a `vnngen_option` block. In this way, an entire option block can be thought of as a single action.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

### Example:

```
if vnngen_event() {
    if vnngen_option("op_choice", 0, 0, 0.5, 0.5) {
        vnngen_option_create("op1", "Option 1", spr_option, spr_option_hover, spr_option_select, 0, 0, -1, fnt_Arial, c_white, trans_slide_right, 1);
        vnngen_option_create("op2", "Option 2", spr_option, spr_option_hover, spr_option_select, 0, 50, -2, fnt_Arial, c_white, trans_slide_right, 1.5);
    }
}
```

## 6.26.3. The "vnngen\_option\_create\_ext" Function

### Syntax:

```
vnngen_option_create_ext(id, text, sprite, spr_hover, spr_select, xorig, yorig, x, y, z, tx, ty, scaling, font, color, trans_in, trans_out, duration, ease);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new option
text	string	The option text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a button background
spr_hover	sprite	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_select	sprite	The sprite to display as a background when selected (or keyword 'none' for default)
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the option, relative to the option block offset
y	real	The vertical position to display the option, relative to the option block offset
z	real	The drawing depth and list order of the option, relative to other options only
tx	real/macro	The horizontal position to display text, relative to the option sprite top-left corner (or keyword 'auto' for center)
ty	real/macro	The vertical position to display text, relative to the option sprite top-left corner (or keyword 'auto' for center)
scaling	integer/macro	Sets the automatic scaling mode for the option
font	font	The font to draw text in, where fnt_default is default
color	color	The default color to draw text in, where c_white is default
trans_in	script	Sets the transition animation to perform when created
trans_out	script	Sets the transition animation to perform when destroyed
duration	real	Sets the duration of the transition animations, in seconds
ease	integer/macro	Sets the ease override for the transition scripts

## Description:

Creates a new option with extra features which will be displayed until any option in the current option block has been selected. The chosen option ID will be recorded as the selected choice. Options can be selected by mouse/touch, or by other input devices with `vnngen_option_nav` and `vnngen_option_select`.

Unlike other entities, option z-index determines not just drawing depth, but also navigation order. Options with lower z-index will be considered lower in the list of options. If z-index is equal between two or more options,

navigation order will automatically be set by *reverse* creation order, but this may or may not be visually correct. Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

Note that although this script is an action, it cannot be executed outside of a `vnngen_option` block. In this way, an entire option block can be thought of as a single action.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

#### Example:

```
if vnngen_event() {
    if vnngen_option("op_choice", 0, 0, 0.5, 0.5) {
        vnngen_option_create_ext("op1", "Option 1", spr_option, spr_option_hover, spr_option_select, orig_center, orig_center, 0, 0, -1, 72, 12, scale_none, fnt_Arial, c_white, trans_slide_right, trans_slide_left, 1, ease_elastic_out);
        vnngen_option_create_ext("op2", "Option 2", spr_option, spr_option_hover, spr_option_select, orig_center, orig_center, 0, 50, -2, 72, 12, scale_none, fnt_Arial, c_white, trans_slide_right, trans_slide_left, 1.5, ease_elastic_out);
    }
}
```

## 6.26.4. The "vnngen\_option\_create\_transformed" Function

### Syntax:

```
vnngen_option_create_transformed(id, text, sprite, spr_hover, spr_select, x, y, z, font, color, hov_x, hov_y, hov_scale, hov_color, sel_x, sel_y, sel_scale, sel_color, sel_duration, transition, duration, [ease]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new option
text	string	The option text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a background
spr_hover	sprite/macro	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_active	sprite/macro	The sprite to display as a background when selected (or keyword 'none' for default)
x	real	The horizontal position to display the option
y	real	The vertical position to display the option
z	real	The drawing depth and list order of the option, relative to other options only
font	font	The font to draw text in, where <code>fnt_default</code> is default
color	color	The color to draw text in, where <code>c_white</code> is default
hov_x	real	The horizontal offset to shift the



		option when hovered
hov_y	real	The vertical offset to shift the option when hovered
hov_scale	real	The scale multiplier to shift the option when hovered
hov_color	color	The color to draw text in when hovered
sel_x	real	The horizontal offset to shift the option when selected
sel_y	real	The vertical offset to shift the option when selected
sel_scale	real	The scale multiplier to shift the option when selected
sel_color	color	The color to draw text in when selected
sel_duration	real	Sets the duration of hover/select animations, in seconds
transition	script	Sets the transition animation to perform
duration	real	Sets the duration of the transition animations, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease override for all option animations

## Description:

Creates a new option with hover and select transformations which will be displayed until any option in the current option block has been selected. The chosen option ID will be recorded as the selected choice. Options can be selected by mouse/touch, or by other input devices with `vnngen_option_nav` and `vnngen_option_select`.

Unlike other entities, option z-index determines not just drawing depth, but also navigation order. Options with lower z-index will be considered lower in the list of options. If z-index is equal between two or more options, navigation order will automatically be set by *reverse* creation order, but this may or may not be visually correct. Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

Note that although this script is an action, it cannot be executed outside of a `vnngen_option` block. In this way, an entire option block can be thought of as a single action.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

## Example:

```
if vnngen_event() {
    if vnngen_option("op_choice", 0, 0, 0.5, 0.5) {
        vnngen_option_create_transformed("op1", "Option 1", spr_option, spr_option_hover,
        spr_option_select, 0, 0, -1, fnt_Arial, c_white, 32, 0, 1.25, c_black, 0, 0, 1, c_
        _yellow, 0.1, trans_slide_right, 1);
        vnngen_option_create_transformed("op2", "Option 2", spr_option, spr_option_hover,
        spr_option_select, 0, 50, -2, fnt_Arial, c_white, 32, 0, 1.25, c_black, 0, 0, 1,
        c_yellow, 0.1, trans_slide_right, 1.5);
    }
}
```

## 6.26.5. The "vngen\_option\_create\_ext\_transformed" Function

### Syntax:

```
vngen_option_create_ext_transformed(id, text, sprite, spr_hover, spr_select, xorig,
yorig, x, y, z, tx, ty, scaling, font, color, hov_x, hov_y, hov_xscale, hov_yscale,
hov_color, sel_x, sel_y, sel_xscale, sel_yscale, sel_color, sel_duration, trans_in,
trans_out, duration, ease);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new option
text	string	The option text to display over the background (or "" for none)
sprite	sprite	The sprite to display as a background
spr_hover	sprite/macro	The sprite to display as a background when hovered (or keyword 'none' for default)
spr_active	sprite/macro	The sprite to display as a background when selected (or keyword 'none' for default)
xorig	real/macro	The horizontal sprite offset, or origin point, relative to the top-left corner
yorig	real/macro	The vertical sprite offset, or origin point, relative to the top-left corner
x	real	The horizontal position to display the option
y	real	The vertical position to display the option
z	real	The drawing depth and list order of the option, relative to other options only
tx	real/macro	The horizontal position to display text, relative to the option sprite top-left corner (or keyword 'auto' for center)
ty	real/macro	The vertical position to display text, relative to the option sprite top-left corner (or keyword 'auto' for center)
scaling	integer/macro	Sets the automatic scaling mode for the option
font	font	The font to draw text in, where fnt_default is default
color	color	The color to draw text in, where c_white is default
hov_x	real	The horizontal offset to shift the option when hovered
hov_y	real	The vertical offset to shift the option when hovered
hov_xscale	real	The horizontal scale multiplier to shift the option when hovered
hov_yscale	real	The vertical scale multiplier to shift the option when hovered
hov_color	color	The color to draw text in when hovered

---

sel_x	real	The horizontal offset to shift the option when selected
sel_y	real	The vertical offset to shift the option when selected
sel_xscale	real	The horizontal scale multiplier to shift the option when selected
sel_yscale	real	The vertical scale multiplier to shift the option when selected
sel_color	color	The color to draw text in when selected
sel_duration	real	Sets the duration of hover/select animations, in seconds
trans_in	script	Sets the transition in animation to perform
trans_out	script	Sets the transition out animation to perform
duration	real	Sets the duration of the transition animations, in seconds
ease	integer/macro	Sets the ease override for all button animations

## Description:

Creates a new option with extra features and hover and select transformations which will be displayed until any option in the current option block has been selected. The chosen option ID will be recorded as the selected choice. Options can be selected by mouse/touch, or by other input devices with `vnngen_option_nav` and `vnngen_option_select`.

Unlike other entities, option z-index determines not just drawing depth, but also navigation order. Options with lower z-index will be considered lower in the list of options. If z-index is equal between two or more options, navigation order will automatically be set by *reverse* creation order, but this may or may not be visually correct. Therefore, it is recommended to always set z-index explicitly. The actual values used are inconsequential so long as they are ordered from highest to lowest.

Note that although this script is an action, it cannot be executed outside of a `vnngen_option` block. In this way, an entire option block can be thought of as a single action.

See [Included Animations](#) and [Macros & Keywords](#) for a list of available transition animations and ease modes.

## Example:

```
if vnngen_event() {
    if vnngen_option("op_choice", 0, 0, 0.5, 0.5) {
        vnngen_option_create_ext_transformed("op1", "Option 1", spr_option, spr_option_
hover, spr_option_select, orig_center, orig_center, 0, 0, -1, scale_none, fnt_Arial,
c_white, 32, 0, 1.25, 1.25, c_black, 0, 0, 1, c_yellow, 0.1, trans_slide_right, tra
ns_slide_left, 1, ease_sin_out);
        vnngen_option_create_ext_transformed("op2", "Option 2", spr_option, spr_option_
hover, spr_option_select, orig_center, orig_center, 0, 50, -2, scale_none, fnt_Arial
, c_white, 32, 0, 1.25, 1.25, c_black, 0, 0, 1, c_yellow, 0.1, trans_slide_right, tr
ans_slide_left, 1.5, ease_sin_out);
    }
}
```

## 6.26.6. The "vngen\_option\_clear" Function

### Syntax:

```
vngen_option_clear([id]);
```

Argument	Type	Description
[id]	real/string	<i>Optional:</i> The ID of the option block to clear results from

### Description:

Manually clears option **results** (not data) to prevent repeat execution of option-based triggers. As this functionality is also handled by `vngen_get_option`, this script should not typically need to be run except in special cases.

If an ID is supplied, the option result will be cleared from the *history*, and it will be as if the choice was never made. However, the most recent option in *memory* will not be cleared. In this case, this script should be run twice to clear both, if needed.

This script is not an action.

### Example:

```
if (vngen_get_option() == "op1") {  
    my_var += 1;  
    vngen_option_clear();  
}  
  
var result = vngen_get_option("my_option");  
vngen_option_clear("my_option");
```

## 6.26.7. The "vngen\_get\_option" Function

### Syntax:

```
vngen_get_option([id], [clear]);
```

Argument	Type	Description
[id]	real/string	<i>Optional:</i> The ID of the option block to return result from
[clear]	boolean	<i>Optional:</i> Enables or disables clearing the result from memory after returning it (enabled by default)

### Description:

Despite its name, `vngen_get_option` is not an action. However, it behaves like one so that it can be used both inside and outside the context of Quantum events.

When run, this script will return the result of the specified option block, or the most recent option block if no group ID is supplied. Results are stored as the ID of the selected option. If options exist but none has been selected, -1 will be returned instead, or if no options exist and have been cleared from memory, -2.

If only one argument is supplied, booleans/reals will be interpreted as 'clear', and strings will be interpreted as an option block ID.

If no arguments are supplied, the most recent option block will be assumed, and clearing will be enabled by default.

This script is intended to be used in conditional statements such as 'if' and 'switch' to take action based on the selected option in a previous option block. In order to prevent this conditional statement from repeatedly executing code, option results will be cleared from memory after being returned. To disable this behavior, it is possible to set the 'clear' argument to 'false', in which case option results can be cleared later with `vnngen_option_clear`.

**Example:**

```
if (vnngen_get_option() == "op1") {  
    //Action  
}  
switch (vnngen_get_option("op_choice", false)) {  
    case "op2": //Action; break;  
    case "op3": //Action; break;  
}  
vnngen_option_clear();
```

## 6.26.8. The "vnngen\_do\_option\_nav" Function

**Syntax:**

```
vnngen_do_option_nav(amount);
```

Argument	Type	Description
amount	integer	The number of options to scroll

**Description:**

Navigates through a linear list of options, if any exist. The amount of options scrolled is arbitrary, where negative values scroll up and positive values scroll down. If a hover sound has been assigned in `vnngen_option`, it will be played upon running this script. If the input amount exceeds the size of the options menu, the selection will loop back around to the other side.

Note that navigation order is determined by the option z-index, and therefore improper configuration of options can result in unexpected navigation with this script.

This script is not an action, and is intended to be run in keyboard and gamepad input events. Mouse and touch support are built-in and do not require `vnngen_do_option_nav` for navigation.

**Example:**

```
//Scroll up  
vnngen_do_option_nav(-1);
```

```
//Scroll down  
vnngen_do_option_nav(1);
```

## 6.26.9. The "vngen\_is\_option\_hovered" Function

### Syntax:

```
vngen_is_option_hovered(id);
```

Argument	Type	Description
id	real/string	The option ID to check

### Description:

Checks whether or not the specified option is hovered (either by mouse/touch or `vngen_do_option_nav`) and returns 'true' or 'false'.

#### Example:

```
if (vngen_is_option_hovered("choice_yes")) {  
    effect_create_above(ef_ellipse, mouse_x, mouse_y, 2, c_white);  
}
```

## 6.26.10. The "vngen\_do\_option\_select" Function

### Syntax:

```
vngen_do_option_select([id]);
```

Argument	Type	Description
id	real/string	<i>Optional:</i> The option ID to select (default: current highlighted option)

### Description:

Selects the currently-highlighted option, if any, and records its ID. If a selection sound has been assigned in `vngen_option`, it will be played upon running this script.

If an option ID is supplied, it will be selected directly, ignoring navigation.

This script is not an action, and is intended to be run in keyboard and gamepad input events. Mouse and touch support are built-in and do not require `vngen_do_option_select` to make selections.

#### Example:

```
if (keyboard_check_released(vk_enter)) {  
    vngen_do_option_select();  
}  
  
if (keyboard_check_released(ord("Y"))) {  
    vngen_do_option_select("yes");  
}
```

## 6.26.11. The "vngen\_is\_option\_selected" Function

## Syntax:

```
vnngen_is_option_selected(id);
```

Argument	Type	Description
id	real/string	The option ID to check

## Description:

Checks whether or not the specified option is selected (either by mouse/touch or `vnngen_do_option_select`) and returns 'true' or 'false'.

Note that this script is different from `vnngen_get_option` in that it will return 'true' while the option is *held down*, not just when it is released and activated.

### Example:

```
if (vnngen_is_option_selected("choice_no")) {  
    effect_create_above(ef_ellipse, mouse_x, mouse_y, 2, c_white);  
}
```

## 6.27. Audio Actions

While GameMaker Studio possesses its own audio functions which can be used in conjunction with VNgen, there are many times when you'll want to perform audio as actions in Quantum events. VNgen provides four categories of audio actions, all of which are specialized to behave in certain ways useful for creating engaging visual novels. These categories include **sound effects**, **music**, **voice**, and **vox**.

In this section we'll examine each, as well as each category's unique properties and how to modify sounds after they've started playing.

### 6.27.1. The "vnngen\_audio\_play\_sound" Function

## Syntax:

```
vnngen_audio_play_sound(id, sound, volume, fade, [ease], [loop]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new sound
sound	sound	The sound resource to play
volume	real (0-1)	The sound volume, where 1 is default and 0 is silent
fade	real	The length of time to fade sound in, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the fade in
[loop]	boolean	<i>Optional:</i> Enables or disables endlessly looping the sound (disabled by default)

## Description:

Plays a new sound effect with an optional fade in and volume control. Sound effects are the most generic type of sound in VNgen and can be looped or played once and will be paused if `vnngen_toggle_pause` is run. Looped audio actions will be marked complete immediately and will not delay the next event until any fade transition is complete.

See [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vnngen_event() {  
    vnngen_audio_play_sound("explode", snd_explosion, 1, 0, ease_none, false);  
}
```

## 6.27.2. The "vnngen\_audio\_play\_music" Function

**Syntax:**

```
vnngen_audio_play_music(id, sound, volume, fade, [ease], [start, end, full]);
```

Argument	Type	Description
id	real/string	The unique identifier to use for the new sound
sound	sound	The sound resource to play
volume	real (0-1)	The sound volume, where 1 is default and 0 is silent
fade	real	The length of time to fade sound in, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the fade in
[start]	real	<i>Optional:</i> Sets the starting point to clip the sound, in seconds
[end]	real	<i>Optional:</i> Sets the ending point to clip the sound, in seconds
[full]	boolean	<i>Optional:</i> Enables or disables playing from beginning before clipping

**Note:** Clip arguments are a group. All three must be input together, or not at all.

**Description:**

Plays a new music track with an optional fade in, volume control, and real-time loop clipping. Music is always looped and will not be paused if `vnngen_toggle_pause` is run. Looped audio actions will be marked complete immediately and will not delay the next event until any fade transition is complete.

See [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vnngen_event() {  
    vnngen_audio_play_music("music", mus_battle, 1, 0.5, ease_sin_out);  
    vnngen_audio_play_music("music", mus_battle, 1, 0.5, 5, 20, true);  
}
```



### 6.27.3. The "vngen\_audio\_play\_voice" Function

#### Syntax:

```
vngen_audio_play_voice(name, sound, volume, fade, [ease], [lang]);
```

Argument	Type	Description
name	string	The ID of the speaking character to match this sound to
sound	sound	The sound resource to play as voiceover
volume	real (0-1)	The sound volume, where 1 is default and 0 is silent
fade	real	The length of time to fade sound in, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the fade in
[lang]	real/string	<i>Optional:</i> Sets the language flag for this action

#### Description:

Plays a new sound as voiceover with an optional fade in and volume control. Voiceover is never looped and will be paused if `vngen_toggle_pause` is run. Voice audio will be added to the backlog for the current event and will trigger facial animations for the associated character, if any.

See [Macros & Keywords](#) for a list of available ease modes. See [Language Functions](#) for details on multi-language support.

#### Example:

```
vngen_event() {  
    vngen_audio_play_voice("John Doe", snd_voice, 1, 0, ease_none);  
    vngen_audio_play_voice("John Doe", snd_voice, 1, 0, "en-US");  
}
```

### 6.27.4. The "vngen\_audio\_modify" Function

#### Syntax:

```
vngen_audio_modify(id, pitch, volume, duration, [ease], [start, end, full]);
```

Argument	Type	Description
id	real/string	The ID of the sound to modify ( <i>or keyword 'all' for all sounds</i> )
pitch	real	The pitch multiplier, where 1 is default
volume	real (0-1)	The sound volume, where 1 is default and 0 is silent
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode for the modification transition
[start]	real	<i>Optional:</i> Sets the starting point to

[end]	real	clip the sound, in seconds <i>Optional:</i> Sets the ending point to clip the sound, in seconds
[full]	boolean	<i>Optional:</i> Enables or disables playing from the current position before clipping

**Note:** Clip arguments are a group. All three must be input together, or not at all.

## Description:

Modifies a sound previously played with VNgen audio actions (except vox audio, which uses a separate function for modifications) with pitch, volume, and optional real-time clipping adjustments. As this script is itself an action, generally it is only practical to modify looped sound effects and music.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {
    vngen_audio_modify("explode", 0.8, 1, 0.5, ease_sin_out);
    vngen_audio_modify("music", 1, 0.75, 1, ease_sin_out, 10, 30, false);
}
```

## 6.27.5. The "vngen\_audio\_replace" Function

### Syntax:

```
vngen_audio_replace(id, sound, [fade], [ease]);
```

Argument	Type	Description
id	real/string	The ID of the audio entity to replace
sound	sound	The new sound to play
[fade]	real	<i>Optional:</i> Sets the duration of the fade transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

### Description:

Replaces the input entity ID with a new sound and fades the old sound to the new one over the input duration. Playback position and other properties will be synchronized between sounds, making this function ideal for transitioning between music tracks.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
if (vngen_event()) {
```

```
vngen_audio_replace("music", music_intense);  
}
```

## 6.27.6. The "vngen\_audio\_pause" Function

### Syntax:

```
vngen_audio_pause(id);
```

Argument	Type	Description
id	real/string	The ID of the sound to pause (or keyword 'all' for all sounds)

### Description:

Pauses a sound previously played with VNgen audio actions (except vox audio, which has a separate pause function). To unpause the sound later, see `vngen_audio_resume`. All audio except music will be automatically paused and unpaused if `vngen_toggle_pause` is run.

### Example:

```
vngen_event() {  
    vngen_audio_pause("explode");  
    vngen_audio_pause("John Doe");  
}
```

This will pause one sound effect and one voiceover audio line. Remember, voice audio uses the name of the associated character as an ID.

## 6.27.7. The "vngen\_audio\_resume" Function

### Syntax:

```
vngen_audio_resume(id);
```

Argument	Type	Description
id	real/string	The unique identifier of the sound to unpause (or keyword 'all' for all sounds)

### Description:

Unpauses a sound previously played with VNgen audio actions (except vox audio, which uses a separate pause function). All audio except music will be automatically paused and unpaused if `vngen_toggle_pause` is run.

### Example:

```
vngen_event() {  
    vngen_audio_resume("explode");  
    vngen_audio_resume("John Doe");  
}
```



```
}
```

This will unpause one sound effect and one voiceover audio line. Remember, voice audio uses the name of the associated character as an ID.

### 6.27.8. The "vngen\_audio\_stop" Function

#### Syntax:

```
vngen_audio_stop(id, [fade], [ease]);
```

Argument	Type	Description
id	real/string	The unique identifier of the sound to stop ( <i>or keyword 'all' for all sounds</i> )
[fade]	real	<i>Optional:</i> The length of time to fade sound out, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the fade in

#### Description:

Stops the input sound with an optional fade out and removes the audio from memory.

See [Macros & Keywords](#) for a list of available ease modes.

#### Example:

```
vngen_event() {  
    vngen_audio_stop("explode");  
    vngen_audio_stop("explode", 2, ease_sin_in);  
}
```

### 6.27.9. The "vngen\_vox\_play" Function

#### Syntax:

```
vngen_vox_play(name, sound, pitch_min, pitch_max, volume);
```

Argument	Type	Description
name	string	The ID of the speaking character to associate with this sound
sound	sound/array	The sound resource to play
pitch_min	real (0-255)	The minimum pitch multiplier, where 1 is default
pitch_max	real (0-255)	The maximum pitch multiplier, where 1 is default
volume	real (0-1)	The sound volume, where 1 is default and 0 is silent

#### Description:

Unlike other sound actions, this script does not play a sound right away, but rather assigns a sound or array of sounds to be played whenever the specified character is speaking. This sound will be looped for the duration of the text typewriter effect to fill in for the absence of real voice audio. As such, it is recommended to use a brief 'blip' sound effect with this script. To further simulate a natural voice, pitch will be randomized between the min and max values each time text increments on the screen.

If multiple sounds are supplied in an array, a sound will be chosen at random each time text increments on the screen. Additional sounds can also be added or removed later with the `vngen_vox_add` and `vngen_vox_remove` functions.

Vox audio actions will be marked complete immediately and will not delay the next event until any fade transition is complete. Vox will be paused if `vngen_toggle_pause` is run.

**Example:**

```
vngen_event() {  
    vngen_vox_play("John Doe", snd_blip, 0.75, 1.25, 1);  
}
```

## 6.27.10. The "vngen\_vox\_modify" Function

**Syntax:**

```
vngen_vox_modify(name, pitch_min, pitch_max, volume, duration, [ease]);
```

Argument	Type	Description
name	string	The ID of the vox sound to modify (or keyword 'all' for all vox)
pitch_min	real (0-255)	The minimum pitch multiplier, where 1 is default
pitch_max	real (0-255)	The maximum pitch multiplier, where 1 is default
volume	real (0-1)	The sound volume, where 1 is default and 0 is silent
duration	real	Sets the length of the modification transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode for the modification transition

**Description:**

Modifies a vox sound previously played with `vngen_vox_play` with pitch and volume adjustments. As vox is only played during text actions, it is possible modifications will not be audible during transition, however a transition will still be applied.

See [Macros & Keywords](#) for a list of available ease modes.

**Example:**

```
vngen_event() {  
    vngen_vox_modify("John Doe", 1, 0.75, 1, ease_sin_in_out);  
}
```

## 6.27.11. The "vngen\_vox\_replace" Function

### Syntax:

```
vngen_vox_replace(name, sound, [fade], [ease]);
```

Argument	Type	Description
name	string	The ID of the vox entity to replace
sound	sound/array	The new sound(s) to play
[fade]	real	<i>Optional:</i> Sets the duration of the fade transition, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the transition in

### Description:

Replaces the input entity ID with a new sound or array of sounds and fades the old sound(s) to the new one(s) over the input duration. Playback position and other properties will be synchronized between sounds.

As with other types of modifications, replacements made with this script are permanent and will persist until another replacement is performed. This script cannot be performed simultaneously with other replacement actions operating on the same entity ID.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
if (vngen_event()) {  
    vngen_vox_replace("John Doe", vox_happy);  
}
```

## 6.27.12. The "vngen\_vox\_add" Function

### Syntax:

```
vngen_vox_add(name, sound);
```

Argument	Type	Description
name	string	The ID of the vox sound to modify ( <i>or keyword 'all' for all vox</i> )
sound	sound/array	The sound resource to add

### Description:

Adds a new sound or array of sounds to a vox entity previously created with `vngen_vox_play`. Unlike `vngen_vox_replace`, adding sounds with this script will not clear existing resources from the entity, but assign multiple sounds to the same entity which will then be chosen at random each time a new text element increments on-screen.

### Example:

```
vngen_event() {  
    vngen_vox_add("John Doe", snd_vox_happy);  
}
```

```
}
```

### 6.27.13. The "vngen\_vox\_remove" Function

#### Syntax:

```
vngen_vox_remove(name, sound);
```

Argument	Type	Description
name	string	The ID of the vox sound to modify ( <i>or keyword 'all' for all vox</i> )
sound	sound/array	The sound resource to remove

#### Description:

Removes a sound or array of sounds from a vox entity previously created with `vngen_vox_play`. Unlike `vngen_vox_stop`, removing sounds with this script will not clear the entity from memory, but only remove sounds from a selection to be chosen at random each time a new text element increments on-screen.

#### Example:

```
vngen_event() {  
    vngen_vox_remove("John Doe", snd_vox_sad);  
}
```

### 6.27.14. The "vngen\_vox\_pause" Function

#### Syntax:

```
vngen_vox_pause(name);
```

Argument	Type	Description
name	string	The ID of the vox sound to pause ( <i>or keyword 'all' for all vox</i> )

#### Description:

Pauses a vox sound previously played with `vngen_vox_play`. To unpause the sound later, see `vngen_vox_resume`.

#### Example:

```
vngen_event() {  
    vngen_vox_pause("John Doe");  
}
```

This will pause vox from being performed during text actions. Remember, vox audio uses the name of the associated character as an ID.

### 6.27.15. The "vngen\_vox\_resume" Function

## Syntax:

```
vngen_vox_resume(name);
```

Argument	Type	Description
name	string	The unique identifier of the vox sound to unpause (or keyword 'all' for all vox)

## Description:

Unpauses a vox sound previously paused with `vngen_vox_pause`.

### Example:

```
vngen_event() {  
    vngen_vox_resume("John Doe");  
}
```

This will unpause vox to be played during text actions. Remember, vox audio uses the name of the associated character as an ID.

## 6.27.16. The "vngen\_vox\_stop" Function

### Syntax:

```
vngen_vox_stop(name, [fade], [ease]);
```

Argument	Type	Description
name	string	The unique identifier of the vox sound to stop (or keyword 'all' for all vox)
[fade]	real	<i>Optional:</i> The length of time to fade sound out, in seconds
[ease]	integer/macro	<i>Optional:</i> Sets the ease mode to perform the fade in

## Description:

Stops the input vox sound with an optional fade out and removes the audio from memory.

See [Macros & Keywords](#) for a list of available ease modes.

### Example:

```
vngen_event() {  
    vngen_vox_stop("John Doe");  
    vngen_vox_stop("John Doe", 2, ease_sin_in);  
}
```

## 6.28. Code Actions



As explained in the [Getting Started](#) guide, it is generally not advisable to execute plain code in Quantum events, as this code will not be treated as an action and therefore will not inherit the benefits of Q-script.

Instead, VNgen offers 'wrappers' which allow executing your own scripts and code without modification as if they were actions. Although it's impossible to fully adapt arbitrary code automatically, using a wrapper is a powerful way to implement custom code without learning the inner workings of Q-script.

In this section we'll examine how to execute your own scripts and plain code as actions in Quantum events.

### 6.28.1. The "vngen\_code\_execute" Function

#### Syntax:

```
vngen_code_execute()
```

Argument	Type	Description
N/A	N/A	No arguments

#### Description:

Executes arbitrary code as an action in the context of Quantum events. This can be anything, but be aware that code executed with this function will be performed only once when the containing event first becomes active.

This script should always be run as an 'if' statement with the code to execute inside.

While this function may appear identical to `vngen_code_execute_ext`, it comes with one critical difference: code is prevented from being re-executed during `vngen_goto` event skip operations.

#### Example:

```
vngen_event() {  
    if (vngen_code_execute()) {  
        my_var += 1;  
    }  
}
```

### 6.28.2. The "vngen\_code\_execute\_ext" Function

#### Syntax:

```
vngen_code_execute_ext()
```

Argument	Type	Description
N/A	N/A	No arguments

#### Description:

Executes arbitrary code as an action in the context of Quantum events. This can be anything, but be aware that code executed with this function will be performed only once when the containing event first becomes active.

This script should always be run as an 'if' statement with the code to execute inside.

While this function may appear identical to `vngen_code_execute`, it comes with one critical difference: code will always be re-executed during `vngen_goto` event skip operations.

**Example:**

```
vngen_event() {  
    if (vngen_code_execute_ext()) {  
        my_var += 1;  
    }  
}
```

### 6.28.3. The "vngen\_script\_execute" Function

**Syntax:**

```
vngen_script_execute(script, argument0, argument1, argument2...);
```

Argument	Type	Description
script	script	The script to execute
argument0 ... argument31	any	Up to 32 values to pass into the script as arguments

**Description:**

Executes the input script as an action in the context of Quantum events. The specified script can be anything, but be aware that scripts executed with this function will be performed only once when the containing event first becomes active.

While this function may appear identical to `vngen_script_execute_ext`, it comes with one critical difference: scripts are prevented from being re-executed during `vngen_goto` event skip operations.

**Example:**

```
vngen_event() {  
    vngen_script_execute(vngen_goto, "my_event");  
}
```

This will cause the engine to jump to the event "my\_event" as soon as the example event is reached, but not if the event is skipped with `vngen_goto`. This use-case is exceptionally useful when managing branching paths in a single object.

### 6.28.4. The "vngen\_script\_execute\_ext" Function

**Syntax:**

```
vngen_script_execute_ext(script, argument0, argument1, argument2...);
```

Argument	Type	Description
script	script	The script to execute



argument0 ... argument31

any

Up to 32 values to pass into the script  
as arguments

## Description:

Executes the input script as an action in the context of Quantum events. The specified script can be anything, but be aware that scripts executed with this function will be performed only once when the containing event first becomes active.

While this function may appear identical to the standard `vngen_script_execute`, it comes with one critical difference: scripts will always be re-executed during `vngen_goto` event skip operations.

### Example:

```
vngen_event() {  
    vngen_script_execute_ext(vngen_set_speed, 50);  
}
```

This will set the global text typewriter effect speed, both when the example event is reached and when skipped with `vngen_goto`. This use-case is exceptionally useful when managing branching paths in a single object.

## 7. Special Thanks

### Patreon Credits

This product is made possible by the generous support of XGASOFT patrons on [Patreon](#). Every contribution counts, no matter how big or small. To all fans and patrons around the globe, thanks for being a part of XGASOFT's story!

Very special thanks goes out to:

### Patreon “Enthusiasts”

Marvin Mrzyglod

### Patreon “Developers”

AshleeVocals

AutumnInAprilArt

Daniel Sato

Darktoz

Dirty Sock Games

Josef Scott

Meyaoi Games

**Patreon “Gamers”**

Kampmichi (Forgers of Novelty)

**Patreon “Supporters”**

Adam Miller (Actawesome)

Cosmopath

D Luecke

Alex Lepinay

Tarquinn J Goodwin

## Creative Credits

XGASOFT is also privileged to work with other creators from around the world, in some cases on the very developer tools used to make XGASOFT products possible.

Special credit goes out to the following talents for their contributions:

### **VNgen Demo Voiceover**

Kanen (*as Miki and Mei*)

## 8. End-User License Agreement ("EULA")

### **LAST UPDATED: 12/16/2019**

We know that reading EULAs isn't very exciting, but this is important. Please take your time to review and ensure you understand the terms of this document before proceeding to use XGASOFT products in your own work.

If you have any questions or concerns about the terms outlined in this document, please feel free to contact us at [contact@xgasoft.com](mailto:contact@xgasoft.com) or by visiting our [Contact & Support](#) page.

## **LICENSE AGREEMENT**



This License Agreement (the “Agreement”) is entered into by and between XGASOFT (the “Licensor”), and you (the “Licensee”). This agreement is legally binding, and becomes effective when you purchase and/or download a free product from XGASOFT or authorized third-party distributors. If you do not agree to the terms of this Agreement, do not purchase, download, or otherwise use XGASOFT products.

In order to accept this Agreement, you must be at least eighteen (18) years of age or whatever age is of legal majority in your country. Otherwise, you must obtain your parent’s or legal guardian’s approval and acceptance of this Agreement in your stead. XGASOFT accepts no liability for your failure to meet this requirement.

XGASOFT delivers content through authorized third-party distributors, each of which may require its own separate End-User License Agreement (“EULA”). XGASOFT accepts no liability for the terms of any third-party agreements, nor for your failure to meet them.

## STANDARD LIFETIME LICENSE

This is a license, not a sale. XGASOFT retains ownership of all content (including but not limited to any copyright, trademarks, brand names, logos, software, images, animations, graphics, video, audio, music, text, and tutorials) comprising digital products and services offered by XGASOFT (the “Property”). All rights not expressly granted are reserved by XGASOFT.

Subject to your acceptance of the terms of this Agreement, XGASOFT grants you a worldwide, revocable, non-exclusive, non-transferable, and **perpetual** license to download, embed, and modify for your own purposes XGASOFT Property solely for incorporation with electronic applications and other interactive media, including both commercial and non-commercial works, wherever substantial value has been added by you.

Any source code included as part of XGASOFT Property must be compiled prior to redistribution as an incorporated work, whether for commercial or non-commercial purposes.

## PATREON LIMITED LICENSE

When you register as a recurring financial supporter of XGASOFT through Patreon (Patreon, Inc.), XGASOFT may provide free access to XGASOFT Property as a reward, subject to the terms of each contribution tier. This is a privilege, not a right.

XGASOFT retains ownership of all content (including but not limited to any copyright, trademarks, brand names, logos, software, images, animations, graphics, video, audio, music, text, and tutorials) comprising digital products and services offered by XGASOFT (the “Property”). All rights not expressly granted are reserved by XGASOFT.

Subject to your acceptance of the terms of this Agreement, XGASOFT grants you a worldwide, revocable, non-exclusive, non-transferable, and **temporary** license to download, embed, and modify for your own purposes XGASOFT Property solely for incorporation with electronic applications and other interactive media, including both commercial and non-commercial works, wherever substantial value has been added by you.

Any source code included as part of XGASOFT Property must be compiled prior to redistribution as an incorporated work, whether for commercial or non-commercial purposes.

This license shall remain effective for the duration of your subscription to XGASOFT through Patreon. In the event that you cancel or reduce your contribution to a lower tier not qualifying for free access to XGASOFT Property, this license will be considered revoked and void for any and all public commercial and non-commercial activities. In order to continue using XGASOFT Property publicly, you must purchase a standard lifetime license.



This limitation shall not be applied retroactively, so that any existing, complete, and publicly available commercial and non-commercial properties using XGASOFT Property will not be considered in violation of this agreement. Furthermore, this limitation shall not apply in the event that XGASOFT suspends, revokes, or disables the contribution of financial support to XGASOFT through Patreon. In such case as contributions are limited or prohibited by XGASOFT (and not the Licensee), the terms of the Standard Lifetime License shall apply to any and all XGASOFT Property granted as rewards for recurring financial support prior to the date of suspension.

## **SINGLE-USER**

This Agreement grants one (1) user an applicable license to use XGASOFT Property on unlimited devices. This license may not be transferred, shared with, or sold to other users.

However, you, the Licensee, may use XGASOFT Property along with a team or company of collaborators wherever substantial value has been added by you.

This limitation does not extend a license to other users. For any works unrelated to you, collaborators must purchase separate licenses.

## **MODIFICATIONS**

In accordance with the terms of this Agreement, you may freely modify, or alter the functionality of XGASOFT Property exclusively for your own use.

Modifying the Property will not terminate your license, however XGASOFT cannot guarantee the quality and functionality of modified versions of the Property, nor its compatibility with other products.

XGASOFT accepts no liability for any loss or damage incurred by the modified Property, and reserves the right to refuse technical support for the modified Property.

Modifications made to XGASOFT Property in no way represent a change of ownership of the Property.

You may not reverse-engineer XGASOFT Property for the purpose of commercial exploitation which may be in competition with XGASOFT.

## **MUTABILITY**

License fees are determined for each product and service on a case-by-case basis, and XGASOFT reserves the right to change fees on the Property with or without prior notice.

XGASOFT reserves the right to modify, suspend, or terminate this Agreement, the Property, or any service to which it connects with or without prior notice and without liability to you, the Licensee.

## **LIABILITY**

By using XGASOFT Property, you agree to indemnify and hold harmless XGASOFT, its employees, and agents



from and against any and all claims (including third party claims), demands, actions, lawsuits, expenses (including attorney's fees) and damages (including indirect or consequential loss) resulting in any way from your use or reliance on XGASOFT Property, any breach of terms of this Agreement, or any other act of your own.

This limitation will survive and apply even in the event of termination of this Agreement.

## **GOVERNING LAW AND JURISDICTION**

This Agreement shall be governed by and interpreted according to the laws of the United States of America and the State of Kansas.

If any provision of this Agreement is held to be unenforceable or invalid, such provision will be changed and interpreted to accomplish the objectives of such provision to the greatest extent possible under applicable law, and the remaining provisions will continue in full force and effect.

## **CONCLUSION**

This document contains the whole agreement between XGASOFT and you, the Licensee, relating to the Property and licenses thereof and supersedes all prior Agreements, arrangements and understandings between both parties regarding XGASOFT Property and licenses.