docs.xgasoft.com

Table of contents:

Welcome to Xzip - Robust Archive Format for Games

Xzip is a custom file archive utility for GameMaker Studio 2. Don't store your external files where anyone can see and edit them, use Xzip!

Features

Robust

Xzip is built using native GML functions for the widest platform compatibility. Unlike `zip_unzip`, which offers no flexibility, Xzip can create, read, and extract archives with individual file access. You can even access and modify files inside the archive--no extraction required!

Flexible

Xzip supports adding and manipulating files individually, as folders, or as arrays. All relative paths are preserved in the archive--but you don't need to remember that. Automatic folder management means a file name and extension are all you need to access data anywhere in the archive.

Can't remember that much? Numerical indices are supported too!

Optimal

Using a combination of custom metadata and compressed binary data, Xzip balances speed and security to serve the broadest possible uses for your external file needs. You can also read and write regular game data to the same archives as your files!

Secure

While anyone can access zip files included with your game, data stored in Xzip cannot be read with conventional archive software. To further protect against unwanted tampering, extracted files can be verified against their original archived copies and overwritten if any changes are detected.

For maximum security, combine with GameMaker's built-in MD5 encoding to determine if the archive itself has been tampered with, and your files are bullet-proof.

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

- Xzip archive basics, such as creating and extracting archives
- Advanced Xzip functions, such as direct archive access and modification
- Protecting archive integrity with data verification
- Individual script arguments, and what they mean

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

Version History

1.0.2

- Updated to GameMaker Studio 2.3.1 standards
- Updated GML+ dependencies to latest version

1.0.0

- Initial release

Xzip Reference Guide

GameMaker Studio supports the inclusion of external files with game projects... with caveats. By default, any included files will export completely unprotected, while your primary assets are tucked safely away in an archive that's difficult for the average user to access or modify. **Why shouldn't external files get the same treatment?**

Although GameMaker has built-in support for .zip archives with the `zip_unzip` command, this command offers no granular access to archive contents and will pause the game until the entire archive is extracted. It also lacks support for any kind of security features. What's more, there's no corresponding `zip_zip` command to create archives in the first place!

Xzip is an archive format reimagined just for GameMaker Studio 2 to manage external files discretely and securely. Although it utilizes compression, Xzip's focus is *not* on making big files smaller. Instead, Xzip aims to make using external archives **convenient**, **fast**, and **powerful** using only native GML and smart, simple commands.

In this reference guide, we'll examine each function in detail.

The "xzip_create" Function

Syntax

```
xzip_create(arch, file1, [file2], ...);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to create
`file1`	string/array	The full path and filename of a file to add to the archive, or array of file paths
`[file2]`	string	*Optional:* Additional files to add to the archive (arrays not accepted)

Description

Creates a new Xzip archive on the disk with the specified filename and adds one or more files to it. Also returns the archive path as a string to be stored in a variable for future reference.

Input files should be written as a string containing the full path of the file to add, including drive letter. If a path points to a folder, the contents of the folder will be added, preserving the relative path of files and subfolders inside.

WARNING

At present, Xzip requires all files to have an extension. Also, although folders are supported, no two files can have the exact same file name even if they are separated by different folders!

IMPORTANT

Due to GameMaker's string handling, slashes in paths should be escaped (i.e. `\\`, not `\`). **Do not** add a final slash to directories!

Instead of listing individual files in the `xzip_create` command, an array of file paths can be passed into the `file1` argument instead. Only the first file argument will accept an array as input, and once detected, no further file arguments will be processed. You cannot combine array and string inputs in a single command.

Files added to the archive will be compressed using zlib. However, Xzip uses additional metadata which may result in compressed files not being much smaller than their uncompressed originals. This is normal.

The resulting archive can be added to the game's **Included Files** area as a means of storing external assets for future use. To access files in the created archive, use `xzip_extract` or `xzip_read`.

Be warned that creation takes time, and archiving many files at once can cause the game to temporarily appear frozen.

TIP

It is recommended to disable the filesystem sandbox for this script. If the sandbox is enabled, archives can only be created and extracted in `working_directory`.

Example

```
my_zip = xzip_create(
            //Archive
            "C:\\archive.xz",

            //Files
            "C:\\file1.txt",
            "C:\\file2.gif",
            "C:\\file3.png"
         );
```

The "xzip_add" Function

Syntax

```
xzip_add(arch, file1, [file2], ...);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to create
`file1`	string/array	The full path and filename of a file to add to the archive, or array of file paths
`[file2]`	string	*Optional:* Additional files to add to the archive (arrays not accepted)

Description

Adds one or more files to an archive previously created with `xzip_create`. Also returns `true` or `false` to indicate if the operation succeeded or failed. Note that this includes failure to overwrite an existing file flagged as read-only. (This can be determined with `xzip_get_readonly`.)

NOTE

If multiple files are input, even a single error will return `false` even though other files succeeded. In this scenario, use `xzip_report` to retrieve a list of failed files.

Input files should be written as a string containing the full path of the file to add, including drive letter. If a path points to a folder, the contents of the folder will be added, preserving the relative path of files and subfolders inside.

WARNING

At present, Xzip requires all files to have an extension. Also, although folders are supported, no two files can have the exact same file name even if they are separated by different folders!

IMPORTANT

Due to GameMaker's string handling, slashes in paths should be escaped (i.e. `\\`, not `\`). **Do not** add a final slash to directories!

Instead of listing individual files in the `xzip_add` command, an array of file paths can be passed into the `file1` argument instead. Only the first file argument will accept an array as input, and once detected, no further file arguments will be processed. You cannot combine array and string inputs in a single command.

Be warned that adding files takes time, and archiving many files at once can cause the game to temporarily appear frozen.

Example

```
xzip_add("C:\\archive.xz", "C:\\file4.jpg", "C:\\file5.pdf",
"C:\\my\\source\\folder");
```

The "xzip_delete" Function

Syntax

```
xzip_delete(arch, file1, [file2], ...);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to modify
`file1`	string/integer/array/keyword	The name or index of a file to delete, or array of files *(or keyword 'all' for all files)*
`[file2]`	string/integer	*Optional:* Additional files to delete from the archive (arrays and keywords not accepted)

Description

Checks if one or more files exist in the given archive and deletes them. If a folder is supplied, all contents of the folder will be deleted.

Also returns `true` or `false` to indicate if the operation succeeded or failed. Note that this includes failure to delete a file flagged as read-only. (This can be determined with `xzip_get_readonly`.)

> **NOTE**
>
> If multiple files are input, even a single error will return `false` even though other files succeeded. In this scenario, use `xzip_report` to retrieve a list of failed files.

For files stored in the archive, a full path should **not** be used, just the file name with extension. Use `xzip_list` to see what file names are available in the archive. The numerical index from `xzip_list`, an array, or the `all` keyword can also be used.

Also note that **this script does not delete the archive itself**! For that, use the built-in `file_delete` function.

Be warned that deletion takes time, and deleting many files at once can cause the game to temporarily appear frozen.

Example

```
xzip_delete("C:\\archive.xz", "file1.txt");
xzip_delete(my_zip, "file2.gif", "my\\sub\\folder");
```

The "xzip_exists" Function

Syntax

```
xzip_exists(arch, file);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to check
`file`	string/integer	The name or index of a file to check

Description

Checks if a file or folder exists in the given archive and returns `true` or `false`.

Note that **this script does not check the archive itself**! For that, use the built-in `file_exists` function.

Example

```
if (!xzip_exists("C:\\archive.xz", "file4.jpg")) {
    xzip_add("C:\\archive.xz", "file4.jpg");
}
```

The "xzip_extract" Function

Syntax

```
xzip_extract(arch, dir, file1, [file2], ... );
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to create
`dir`	string	The full path of the destination folder to extract into, **excluding** final slash
`file1`	string/integer/array/keyword	The name or index of a file to extract, or array of files *(or keyword 'all' for all files)*
`[file2]`	string	*Optional:* Additional files to extract from the archive (arrays and keywords not accepted)

Description

Extracts one or more files from an archive created with `xzip_create` to the destination folder. Also returns `true` or `false` to indicate if the operation succeeded.

NOTE

If multiple files are input, even a single error will return `false` even though other files succeeded. In this scenario, use `xzip_report` to retrieve a list of failed files.

For files stored in the archive, a full path should **not** be used, just the file name with extension. Use `xzip_list` to see what file names are available in the archive. The numerical index from `xzip_list`, an array, or the `all` keyword can also be used. If a folder is input, all files from the folder will be extracted, preserving the relative path.

Be warned that extraction takes time, and extracting many files at once can cause the game to temporarily appear frozen. It is recommended to extract large archives over a series of Steps and display a loading screen. *(See example usage.)*

TIP

It is recommended to disable the filesystem sandbox for this script. If the sandbox is enabled, archives can only be created and extracted in `working_directory`.

Example

```
//CREATE
file_archive    = "C:\\archive.xz";
file_dest       = "C:\\my\\destination\\folder";
file_count      = xzip_count(file_archive);
file_current    = 0;
file_array      = xzip_list(file_archive);

//STEP
if (file_current < file_count) {
    xzip_extract(file_archive, file_dest, file_array[file_current]);

    file_current++;
}

//DRAW
var file_prog = file_current/file_count;

draw_text(25, 25, "Extraction: " + string(file_prog) + "% complete");
```

The "xzip_recurse" Variable

Description

`xzip_recurse` is not a function, but rather a macro for a built-in variable that enables or disables recursive folder operations in any function that accepts folders as inputs.

By default, `xzip_recurse` is set to `false`, meaning only files in the exact directory specified will be considered in folder operations. To include subfolders as well, set `xzip_recurse` to `true`. This setting is global and will apply to **all** Xzip functions.

This is especially useful when (but not limited to) using the `xzip_list_dir` and `xzip_count_dir` functions.

Example

```
xzip_recurse = true;

my_zip = "C:\\archive.xz";
my_folder = "my\\sub\\folder";
my_list = xzip_list_dir(my_zip, my_folder, true);
my_count = xzip_count_dir(my_zip, my_folder);

draw_text(25, 25, "Found " + string(my_count) + " items:");
for (var i = 0; i < array_length_1d(my_list); i++) {
    draw_text(25, 25 + (25*i), my_list[i]);
}
```

This will display a count of all items found in the given folder and any subfolders below, as well as list out the file names including relative paths.

The "xzip_report" Data Structure

Description

`xzip_report` is not a function, but rather a macro for a built-in `ds_list` that tracks errors in Xzip functions which work with multiple files simultaneously. This includes:

Monitored Functions

- `xzip_add`
- `xzip_delete`
- `xzip_extract`
- `xzip_move`
- `xzip_set_readonly`

These functions will return `true` or `false` to indicate whether all file operations completed successfully. In the event of errors, any problematic file names will be recorded in the data structure returned by `xzip_report`, as it is otherwise difficult to tell which files caused the errors to occur.

Any time a monitored function returns `false`, it is a good idea to parse `xzip_report` to take appropriate action in response. This is helpful not just in debugging, but also for handling errors gracefully in compiled applications.

NOTE

For simplicity, only file *names* are recorded, **not** file *paths*. Paths for each file are already known by virtue of the fact they were input as arguments to a script; using file *names only* makes `xzip_report` easier to parse.

Keep in mind that `xzip_report` is refreshed each time a new monitored function is run. To store the contents of a report for future usage, it should be copied to a secondary `ds_list` with `ds_list_copy`. (Assigning `xzip_report` to another variable with `=` will **not** work, as this will reassign the `ds_list` *index* only, not the *contents* of the `ds_list` itself.)

Example

```
if (xzip_delete("C:\\archive.xz", "file1.txt", "file2.jpg") == false) {
    if (ds_list_find_value(xzip_report(), 0) == "archive.xz") {
        //Archive not found
    }
}
```

This will check if an error occurs during a file deletion process, and if so, the first entry in `xzip_report` will be queried to determine if the archive itself is the problem.

The "xzip_move" Function

Syntax

```
xzip_move(arch, dir, file1, [file2], ...);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to create
`dir`	string	The internal path of the destination folder to move into
`file1`	string/integer/array/keyword	The name or index of a file to move, or array of files *(or keyword 'all' for all files)*
`[file2]`	string	*Optional:* Additional files to move (arrays and keywords not accepted)

Description

Moves one or more files to a subfolder within an archive previously created with `xzip_create`. Upon extraction, files will appear in the new subfolder. If any subfolder in the destination path does not exist, it will be created.

If a folder is input, all files inside will be moved to the new location--the relative path will **not** be preserved.

Also returns `true` or `false` to indicate if the operation succeeded or failed. Note that this includes failure to move a file flagged as read-only. (This can be determined with `xzip_get_readonly`.)

NOTE

If multiple files are input, even a single error will return `false` even though other files succeeded. In this scenario, use `xzip_report` to retrieve a list of failed files.

As archives have no mount point, you should **not** supply a drive letter or leading slash when specifying a destination folder. Use `""` to move files to the root directory of the archive.

IMPORTANT

Due to GameMaker's string handling, slashes in paths should be escaped (i.e. `\\`, not `\`). **Do not** add a final slash to directories!

Note that **this script does not move the archive itself**! For that, use the built-in `file_copy` and `file_delete` functions.

Example

```
xzip_move("C:\\archive.xz", "my\\destination\\folder", "file1.txt",
"file5.pdf");
```

The "xzip_rename" Function

Syntax

```
xzip_rename(arch, file, name);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to create
`file`	string/integer	The name or index of a file to rename
`name`	string	The new file name to apply

Description

Renames a file or folder inside an archive created with `xzip_create`. Also returns `true` or `false` to indicate if the operation succeeded. Note that this includes failure to rename a file flagged as read-only. (This can be determined with `xzip_get_readonly`.)

If a folder is input, the `file` argument must include the folder's relative path with no beginning or ending slash. However, no relative path should be supplied for regular files or for `name`.

Note that **this script does not rename the archive itself**! For that, use the built-in `file_rename` function.

Example

```
xzip_rename("C:\\archive.xz", "file3.png", "image.png");
```

The "xzip_verify" Function

Syntax

```
xzip_verify(arch, dir, file);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to check
`dir`	string	The full path of the destination folder to check, **excluding** final slash
`file`	string/integer	The name or index of a file to check

Description

Verifies an extracted file against its original archive and returns `true` or `false` if the files' data matches. This is useful not only to test for file corruption or incomplete extraction, but also to protect file integrity. If a file has been illegitimately modified, this script can be used to detect it and trigger re-extraction of the unmodified file from the archive.

TIP

To take security one step further, use GameMaker Studio's built-in MD5 functions to store a hash of the archive itself. Then your code can detect if illegitimate modifications have been made to the archive source file in an attempt to thwart `xzip_verify`.

For files stored in the archive, a full path should **not** be used, just the file name with extension. Use `xzip_list` to see what file names are available in the archive. The numerical index from `xzip_list` can also be used.

This function does not support folders as file inputs. Any relative paths inside the archive will be automatically applied to file names. Likewise, the archive file name will be automatically applied to the target directory to check. (Using a directory instead of explicit file names to compare enables simpler batch verifications.)

Be warned that verification takes time, and verifying many files at once can cause the game to temporarily appear frozen. It is recommended to verify large archives over a series of Steps and display a loading screen. *(See example usage.)*

Example

```
//CREATE
file_archive        = "C:\\archive.xz";
file_dest           = "C:\\my\\destination\\folder";
file_count          = xzip_count(file_archive);
file_current        = 0;
file_list           = xzip_list(file_archive);
file_fail_list      = 0;
file_fail_current   = 0;

//STEP
if (file_current < file_count) {
    //Verify files
    if (!xzip_verify(file_archive, file_dest, file_list[file_current])) {
        //Create an array of failed files
        file_fail_list[file_fail_current] = file_list[file_current];
        file_fail_current++;
    }

    file_current++;
}

//DRAW
var file_prog = file_current/file_count;

draw_text(25, 25, "Verification: " + string(file_prog) + "% complete");
draw_text(25, 50, "Failed: " + string(array_height(file_fail_list)) + "
files");
```

The "xzip_list" Function

Syntax

```
xzip_list(arch, [relative]);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to check
`[relative]`	boolean	*Optional:* Enables or disables including relative paths in the results (disabled by default)

Description

Returns an array of strings containing the file names and extensions (and optionally, relative paths) contained within the given archive. The array index for each file also corresponds to the archive index and can be used when extracting files in place of a file name itself.

Many functions support inputting an array of files, in which case an array returned by this function can be passed in directly.

Example

```
file_array = xzip_list("C:\\archive.xz");

xzip_extract("C:\\archive.xz", "C:\\extracted", file_array[0]);
```

This will retrieve a list of files contained in an archive and extract the first file in the list. Remember that because the array index corresponds to the archive index, we could also use just `0` here instead of `file_array[0]`.

The "xzip_list_dir" Function

Syntax

```
xzip_list_dir(arch, dir, [relative]);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to check
`dir`	string	The *relative* directory within the archive to list
`[relative]`	boolean	*Optional:* Enables or disables including relative paths in the results (disabled by default)

Description

Returns an array of strings containing the file names and extensions (and optionally, relative paths) contained within an *individual directory* within the given archive. An empty directory will return `0`, or `-1` if the archive doesn't exist at all.

Unlike file names and other directories, `dir` here should be written as a relative path. (i.e. If the directory is a subfolder, all parent folders must be included in the argument.)

As archives have no mount point, you should **not** supply a drive letter or leading slash when specifying a directory to list. Use `""` to list files in the root directory of the archive.

IMPORTANT

Due to GameMaker's string handling, slashes in paths should be escaped (i.e. `\\`, not `\`). **Do not** add a final slash to directories!

Unlike `xzip_list`, the resulting array indices **do not** correspond to archive indices. However, many functions support inputting an array of files, in which case an array returned by this function can be passed in directly.

TIP

By default, only the contents of the exact folder specified will be listed. To include the contents of any subfolders as well, see `xzip_recurse`.

Example

```
file_dir_array = xzip_list_dir("C:\\archive.xz", "my\\sub\\folder");

xzip_extract("C:\\archive.xz", "C:\\extracted", file_dir_array[0]);
```

This will retrieve a list of files contained in a specific subfolder of an archive and extract the first file in the list.

The "xzip_count" Function

Syntax

```
xzip_count(arch);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to check

Description

Returns the number of files in the given archive created with `xzip_create`. An empty archive will return `0`, or `-1` if it doesn't exist at all.

Example

```
my_zip = xzip_create("C:\\archive.xz");

if (xzip_count(my_zip) == 0) {
    xzip_add(my_zip, "C:\\file1.txt", "C:\\file2.zip");
}
```

This will create a new archive and only add contents to it if it is empty. This technique can be used to prevent repeatedly adding archive contents in Step events, which are run every frame, for example.

The "xzip_count_dir" Function

Syntax

```
xzip_count_dir(arch, dir);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to check
`dir`	string	The *relative* directory within the archive to count

Description

Returns the number of files contained within an *individual directory* within the given archive created with `xzip_create`. An empty directory will return `0`, or `-1` if the archive doesn't exist at all.

Unlike file names and other directories, `dir` here should be written as a relative path. (i.e. If the directory is a subfolder, all parent folders must be included in the argument.)

As archives have no mount point, you should **not** supply a drive letter or leading slash when specifying a directory to list. Use `""` to list files in the root directory of the archive.

IMPORTANT

Due to GameMaker's string handling, slashes in paths should be escaped (i.e. `\\`, not `\`). **Do not** add a final slash to directories!

TIP

By default, only the contents of the exact folder specified will be listed. To include the contents of any subfolders as well, see `xzip_recurse`.

Example

```
my_zip = xzip_create("C:\\archive.xz", "C:\\file1.txt", "C:\\file2.pdf");

if (xzip_count_dir(my_zip, "my\\destination\\folder") == 0) {
    xzip_move(my_zip, "my\\destination\\folder", "file1.txt", "file2.pdf");
}
```

The "xzip_write" Function

Syntax

```
xzip_write(arch, buff, file);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to check
`buff`	buffer	A buffer containing data to be written to the archive
`file`	string	The file name to assign to the buffer in the archive (including relative path, if any)

Description

Writes data from a buffer directly into the archive (instead of from a file on the disk) and assigns it a standard file name. Buffers will be compressed with zlib before writing. Once written, buffers can be restored with `xzip_read` or extracted and read from disk like any other file.

Also returns `true` or `false` to indicate if the operation succeeded. Note that this includes failure to overwrite a file flagged as read-only. (This can be determined with `xzip_get_readonly`.)

In GameMaker Studio, **buffers** are simply containers for binary data, and can store anything from text, to audio, to surfaces, and beyond. However, it's important to keep in mind that GameMaker must have a function to interpret the data in order to use it.

Example

```
var buf_surf = buffer_create(32, buffer_grow, 1);
buffer_get_surface(buf_surf, application_surface, 0);

xzip_write("C:\\archive.xz", buf_surf, "file5.surf");
```

This will copy the application surface in its current state and write it to the archive, essentially capturing a screenshot which can be viewed later.

The "xzip_read" Function

Syntax

```
xzip_read(arch, file);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to read from
`file`	string/integer	The name or index of a file to read

Description

Reads a file from an archive directly into memory (instead of extracting to the disk) and returns the result as a buffer. This is especially useful for game data like audio and surfaces which have built-in buffer functions in GameMaker Studio.

If the input archive or file within do not exist, `-1` will be returned instead, so it's a good idea to run `buffer_exists` before handling data returned by this script.

Example

```
var buf_surf = xzip_read("C:\\archive.xz", "file5.surf");
var my_surf = surface_create(1280, 720);

if (buffer_exists(buf_surf)) {
    buffer_set_surface(buf_surf, my_surf, 0);
}
```

This will load a surface from an archive and copy it to a pre-existing surface for drawing.

The "xzip_set_readonly" Function

Syntax

```
xzip_set_readonly(arch, enable, file1, [file2], ...);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to modify
`enable`	boolean	Sets whether to enable or disable the read-only flag
`file1`	string/integer/array/keyword	The name or index of a file to modify, or array of files *(or keyword 'all' for all files)*
`[file2]`	string	*Optional:* Additional files to modify (arrays and keywords not accepted)

Description

Enables or disables flagging files as **read-only** within the given archive created with `xzip_create`. Files flagged as read-only cannot be overwritten, deleted, renamed, or moved without first disabling the read-only flag.

If a folder is input, all files inside will be modified.

Also returns `true` or `false` to indicate if the operation succeeded or failed.

> NOTE
>
> If multiple files are input, even a single error will return `false` even though other files succeeded. In this scenario, use `xzip_report` to retrieve a list of failed files.

Example

```
xzip_set_readonly("C:\\archive.xz", true, all);
xzip_set_readonly("C:\\archive.xz", false, "file1.txt", "file3.png");
```

This will make all files read-only *except* "file1.txt" and "file3.png".

The "xzip_get_readonly" Function

Syntax

```
xzip_get_readonly(arch, file);
```

Argument	Type	Description
`arch`	string	The full path and filename of the archive to check
`file`	string/integer	The name or index of a file to check

Description

Checks whether the read-only flag is enabled or disabled for the given file and returns `true` or `false`. If the input archive or file within do not exist, `false` will also be returned, as this indicates the archive/file can be written.

This function does not support folders as file inputs.

Example

```
if (!xzip_get_readonly("C:\\archive.xz", "file1.txt")) {
    xzip_rename("C:\\archive.xz", "file1.txt", "textfile.txt");
}
```

This will check if a file is read-only before attempting to rename it.

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)

All Other 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)*

End-User License Agreement ("EULA")

NOTE

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 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

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.