Edge VN Reference Guide

by XGASOFT

1. Welcome to Edge VN

Thank you for choosing the Edge Visual Novel Engine—**Edge VN**, for short. **Edge Engine** is a fully cross-platform, modular framework built to augment Game Maker Studio with pre-made code and assets that serve as the foundation for a wide variety of game genres. All Edge Engine modules feature creative, human-readable code with helpful notations throughout, making them both powerful and easy to use.

Edge Visual Novel Engine (or Edge VN) is a powerful visual novel engine that raises the bar for presentation in the genre. With specialized tools for drawing animated scenes, characters, text, and more, it's easy for beginners to get a visual novel up-and-running in Game Maker Studio, and flexible for developers to create professional and engaging storytelling sequences in any genre.

In this guide you will learn the different components available in Edge VN and how to use them in constructing your very own visual novels. Many functions are standardized, and so as you familiarize yourself with Edge VN conventions you will quickly be able to take advantage of more advanced features present in the engine for near limitless customization using only Edge VN scripts.

2. Buy Now(https://marketplace.yoyogames.com/assets/1591/visual-novel-edge-engine)

3. Download PDF(https://docs.xgasoft.com/wp-content/uploads/sites/2/edge-vn-reference-guide-10.pdf)

4. What's New

v1.9.1

• Improved `edgevn_goto` to avoid crashes and handle skipping over previously unskippable actions

v1.9.0

• Improved HTML5 support

• Improved text log performance, now uses sprite background

• Improved transitions between text blocks

• Updated `edgevn_create_next` to return created instance ID

• Minor additional improvements to code

v1.8.7

• Fixed a regression in previous update causing audio to fail to play on string 0

v1.8.6

• Fixed a bug in Audio functions causing audio to loop even with loop disabled

• Fixed a bug causing auto mode to fail in certain conditions when multiple scenes are used simultaneously

v1.8.4

• Added support for fractional typewriter effect speeds

• Added `edgevn_set_volume` script to control voiceover audio as set in `edgevn_create_text`

• Updated `edgevn_draw_char_emote` to match character scale and orientation

• To save on processing, RTL mode is now disabled by default. It can be re-enabled by uncommenting a line of code in `edgevn_create_text`

• `edgevn_auto` now waits for scene transitions to continue

• Minor improvements to code and performance

v1.8.2

• Updated documentation to new format

• Added `edgevn_char_emote` script for displaying small, expressive animations over characters to demonstrate emotion

• Improvements to code for backtracking and dialog options

• Fixed an issue where text would fail to display on screens that are lower resolution than the game itself

v1.8.0

• Updated `edgevn_create_text` to set the drawing font, enabling variable fonts per string. The text font parameter in `edgevn_draw_text` is now an override and can be disabled

• Added optional auto ID mode to `edgevn_create_text` and `edgevn_create_char`

• Added option to `edgevn_create_text` to enable or disable breaking between strings to facilitate 'NVL'-style visual novel presentation

• Added passive parallax mode to scenes, enabling scenes to offset other scenes for enhanced effect

• Rewritten textbox script brings code up to parity with recent updates—adds style modification features and effects to textboxes just like characters and scenes!

• Updated `edgevn_draw_char_ext` to syntactical parity with scene and textbox effects, including a new 'loop' option (also applies to `edgevn_char_style_ext`)

• Added option to audio functions to stop sound if target string is skipped before sound is finished playing

• Added basic backtracking support to `edgevn_goto`

• Improved touch screen support for reference links (see documentation on mobile device support)

• Renamed `edgevn_prepare_block` to `edgevn_init_block` for clarity and uniformity with other Edge Engine modules

• Added support for # line breaking

• General additional improvements

v1.7.0

• Migrated all drawing functions to Draw GUI **(Warning: this could affect projects made using previous versions of Edge VN!)**

• Rewritten scene engine—now cleaner, easier to use, and includes optional parallax effect with characters!

• Added `edgevn_create_next_block` script for easily stringing multiple text blocks together

• Added character scaling and transparency options to `edgevn_char_style`

• Added audio functions for playing sounds and music within the context of Edge VN events

• Added `edgevn_toggle_pause` script for pausing/unpausing most engine functions

• Auto mode now pauses when game loses focus

• Added basic support for RTL (right-to-left) languages (text blocks only; log not supported)

• Fixed a bug causing crashes when typewriter mode is not enabled

• Minor additional bug fixes

• Numerous additional minor improvements

• Code cleanup and enhancements for efficiency and readability

v1.5.4

• Added `edgevn_goto` script to allow jumping to specific text blocks/lines of text, effectively skipping entire scenes

• Dialog options are now added to the text log when selected

• Fixed a bug causing log audio functions to crash when log is empty

• Minor improvements and bug fixes

v.1.5.1

• Improvements to dialog options—added delay to selected items and support for multiple option segments in a single text block

• Minor bug fixes

v1.5.0

• Numerous improvements to usability and code, including bug fixes

• Improved performance in both text blocks and text log

• Replaced scene zoom effect with full scaling in `edgevn_scene_style`

• Added ability to set which view to draw in via `edgevn_prepare_block`

• Added `edgevn_block_complete` script for testing block state in custom code

• Added audio playback to text log

v1.4.0

• Added `edgevn_toggle_visibility` script for showing/hiding textbox and text

• Fixed an issue preventing bold-italic from displaying properly when skipped

• Code cleanup! Numerous minor fixes and improvements

v1.3.6

• Added support for a third sprite in `edgevn_draw_prompt` as an ‘auto’ mode indicator.

• ‘Auto’ mode now prevents manual skipping/continuing while enabled

• Fixed a bug where ‘auto’ mode and dialog options would conflict, preventing options from working

• Improved touch scrolling in `edgevn_log_scroll_swipe`

• Improved text engine with support for arbitrary framerates and successive character fading

• Fixed multiple bugs regarding italic text

v1.3.0

• Added markup support for bold and italic text in `edgevn_create_text`

• Cleaned-up and improved text engine; variable typewriter speeds now supported

• Improved `edgevn_char_style` and added `edgevn_char_style_ext` for dynamic animations

• Added `edgevn_log_scroll_swipe` script for touch screen log scrolling

v1.2.1

• Minor additions to `edgevn_draw_option`. See documentation for details.

v1.2.0

• Rewritten dynamic characters with extensible animations

• Added branching dialog options system

• `edgevn_continue` can now play a sound effect when incrementing text block.

v1.1.2

• Added foreground option to `edgevn_draw_scene`

• Added option to disable textbox animations (primarily for stringing text blocks together)

• Minor other improvements

v1.1.0

• Revamped character system with easy positioning and styling; ‘if’ statements no longer needed

• Added scene system for drawing visual novel backgrounds with extensible visual effects

• Added auto scripts for hands-free progression of text

• Merged `edgevn_draw_text` and `edgevn_draw_text_ext`

• Cosmetic changes—renamed scripts for clarity

• General improvements and significant rewrites

v1.0.0

• Initial release

5. Migrating to New Versions of Edge VN

As Edge VN matures and develops, old features sometimes become incompatible with the latest updates. This does not mean projects built on older versions of Edge VN cannot be upgraded, but it does mean the process is not automatic and will require some effort on the part of the developer.

Note that the latest version of Edge VN should already be installed in your project *before* following this guide. To install Edge VN on top of older versions, remove Edge VN from your project’s ‘Extensions’ folder and delete the entire ‘Edge VN’ folder from your ‘Scripts’ folder. Also delete any Edge VN demo objects, sprites, and backgrounds, leaving only your own custom assets behind. To update these custom project assets, refer to the migration guide below.

• All drawing functions have been migrated to the Draw GUI event.

Projects made with previous versions of Edge VN may require substantial modification to be brought up-to-date with this change. All Edge VN functions previously used in the Draw event must be moved to Draw GUI, and any coordinates relying on views for positioning may require adjusting to display properly. As GameMaker’s GUI layer is positioned independent of the game room, a coordinate of 0,0 is always the top-left corner of the screen and all other coordinates are relative to that point on the screen, not the room or view. **In many cases no changes to code may be necessary**, however be prepared to adjust custom values accordingly. You may wish to run the built-in `display_set_gui_size` function to minimize the number of adjustments that must be made for an identical experience to previous versions of Edge VN. **It is still possible to run Edge VN in the Draw event**, however in this case you will need to position all Edge VN elements relative to a view manually, as this is no longer part of the engine itself.

• `edgevn_prepare_block` has been renamed to `edgevn_init_block`.

This change was made for clarity and to unify nomenclature across Edge Engine modules. All instances of this script must be renamed to match the new name. See documentation for further information.

• The ‘view’ argument has been removed from `edgevn_init_block`.

Instances of this script must be updated to remove the view setting, as all drawing functions now use the GUI layer. See documentation for further details on the updated syntax.

• `edgevn_create_text` now has an additional argument to set the break mode between strings.

This allows for the creation of ‘NVL’-style visual novels where the text occupies the majority of the screen and pauses after each line rather than printing an entire paragraph at once. See documentation for further information.

• `edgevn_draw_textbox` now has an additional argument to set the textbox scaling mode.

All instances of this script must be updated to match the new syntax. See documentation for further information.

• Audio playback functions now have an additional argument to make audio skippable.

This additional true/false parameter must be added into existing instances of `edgevn_play_sound` and `edgevn_play_sound_ext`. See documentation for further information on the new syntax.

• All scene scripts have been rewritten and now use a new syntax.

While the new scene engine retains all of the same features as the last, it now draws scenes in an entirely different way and even includes some new features that were not possible before. All occurrences of Edge VN scene scripts must be updated to match these new syntaxes, which can be referred to in the documentation below. Backgrounds formerly drawn as foregrounds must now be drawn literally on top of other elements by placing the `edgevn_draw_scene` script *last* in the Draw GUI event.

• All scene effects have been rewritten to use a new, simpler effect syntax.

In the same vein as the last point, scene effects have been rewritten as well and now function almost identically to dynamic character animations. The new effect syntax is simple and makes it very easy to write additional effects, however as a consequence any pre-existing custom scene effects must be rewritten to match the new scene effect syntax. **All effects previously included with Edge VN are still present.**

• `edgevn_draw_char_ext` now has an additional argument to set the animation loop mode.

All instances of this script must be updated to match the new syntax. See documentation for further information.

• All dynamic character animations have been rewritten to use a new, simpler effect syntax.

The changes here are minimal, but contribute significantly to Edge VN by creating a uniform syntax across scene effects, textbox effects, and dynamic character animations. As with other effect syntaxes, animations now declare the number of keyframes at the beginning of the animation and handle looping with the parent `edgevn_draw_char_ext` script. The `cutin_frame` variable has also been renamed to `cutin_keyframe` for uniformity. See the included demo animations for further information.

• `edgevn_char_style` and `edgevn_char_style_ext` have been given two additional arguments for scaling characters, and `edgevn_char_style_ext` a third argument for setting the animation loop mode.

Any instances of these scripts must be updated to include values for these new arguments. See documentation for further details on the updated syntax.

• Some scripts which previously adjusted for sprite offset have been optimized to adjust for sprite dimensions only.

This means that some sprites which previously were forced to display a certain way are no longer restricted by sprite offset, and any sprites using an irregular sprite offset may need to be reset to an offset of 0, 0 (the top-left corner). As this is already the case for new sprites by default, most users will be unaffected by this change, and the simpler calculations will result in a modest improvement in performance.

6. Overview

Creating a visual novel can be a tedious process. Even when included as only one element among many in a game, telling a story line-by-line takes many hours of intricate work, especially when other visual elements like backgrounds and character cut-ins are brought into the mix. But you're here to tell a story—not get tangled in string after string of code. That's where Edge VN comes in. Edge VN handles all the fundamental elements of visual novels for you and boils them down into the easiest-to-use package possible so you can create great-looking stories in no time. While there is a bit of a learning curve to knowing how, refer to the guide below to see how it all happens and you'll be well on your way to creating visual novels of your own in a flash!

Before we get started, first take note of EdgeVN's organizational structure. Rather than follow a strictly alphabetical order, EdgeVN scripts are sorted primarily by category and the order in which you'll need them. There are folders for **functions**, the text **log**, **text blocks**, and inside the text blocks folder, **global** functions, **scenes** and scene **effects**, **characters** and character **animations**, **textbox** and textbox **effects**, dialog **branches**, and **audio** functions. This breakdown should establish a clear workflow and further eases the learning curve for new users. This is also the order this documentation will follow in detailing each function of Edge VN.

6.1. Using the GUI Layer

As of version 1.7.0, all Edge VN drawing functions use the Draw GUI event as opposed to the regular Draw event. In Game Maker Studio, drawing operations are performed on different surfaces, or 'layers', the lowest of which is the **application surface**, and the highest of which is the **GUI**. Anything drawn to the GUI will automatically appear on top of anything drawn to the application surface, and unlike the application surface, the GUI is always relative to the *screen*, not the game room. This means coordinates regarding the GUI are always absolute—a coordinate of `0,0` will always be the top-left corner of the screen no matter what area of the room is visible or what views are used.

This also means Edge VN comes with a large amount of **scaling** built-in, and it is designed to adapt to changes in screen resolution as effectively as possible. It’s not *all* automatic, however. By default, the GUI layer will resize itself to match the physical screen resolution of the window or display being used, which can in some cases result in undesirable effects as Edge VN elements become grouped too closely or spread too far apart. To design your visual novels most effectively, it is recommended to approach this problem in one of two ways: 1) using the built-in `display_set_gui_size` function to force the GUI to a certain screen resolution, or 2) always positioning your Edge VN elements using the relative dimensions of the GUI, which can be obtained with the built-in `display_get_gui_width()` and `display_get_gui_height()` functions. All of these functions can be read about in the GameMaker documentation by pressing F1 from within Game Maker Studio at any time.

For more robust scaling options, also be sure to check out Edge Display Scaler!

By familiarizing yourself with the GUI layer and how it works, you’ll be off to creating great-looking visual novel content in no time. However, Edge VN remains compatible with the application surface and the normal Draw event should your project need it. Just bear in mind that it will still be scaled to match the GUI, and coordinates will have to be positioned relative to the active view, where a coordinate of `view_xview[0]`, `view_yview[0]`, not `0,0` is the top-left corner of the screen instead.

6.2. Mobile Device Support

While Edge VN will generally work out-of-the-box on all platforms supported by GameMaker Studio, sometimes special attention needs to be given to mobile and touch-based devices to ensure the best user experience for that specific type of hardware. Here are a few pointers to keep in mind:

• Use the Global Mouse Left *Released* event instead of Global Mouse Left *Pressed* for `edgevn_continue`, `edgevn_option_nav_select`, and other mouse input functions.

This will allow users to drag a finger around the screen before triggering the next string. Highly useful for navigating dialog options and reference links on touch displays.

• Use larger fonts and smaller margins.

Smaller screens mean decreased readability. Compensating for screen size with proportionally large fonts and graphics will make your visual novel easier to experience, and smaller margins will allow optimal room for character and scene graphics.

6.3. Functions

On top of providing specialized, easy-to-use scripts designed for providing specific functionality, Edge VN also extends the basic functionality of Game Maker Studio with a few generic functions that fit in line with existing commands. Edge VN simply relies on these scripts for its core functionality and also leaves them available for advanced users to expand Game Maker Studio’s capabilities even further.

6.3.1. animate_sprites(speed)

```
animate_sprites(1);
```

While under normal conditions sprites in Game Maker Studio animate themselves automatically, when drawing multiple sprites within a single object these sprites will be drawn static unless animated manually. The `animate_sprites` script does exactly that, while also adjusting the sprite animation speed for variable framerates and even pausing the sprite animation if Edge VN is paused with `edgevn_toggle_pause`. Note that using this script will animate **all** sprites in the object running the script. It has only one argument, **speed**, where a value of `1` equals 100% speed.

Note that this script must be run in either the Step or Draw GUI event to function.

> Note: This script currently does not support HTML5. To animate sprites in HTML5, create a new variable of your own (e.g. `frame_index = 0;`) in an object's Create event, and then replace every usage of `image_index` in Edge VN scripts with your own variable. This can be easily achieved with Game Maker Studio's script search function, found from the program menu under Scripts > Search in Scripts…, or by pressing Shift + Ctrl + F on your keyboard and searching for 'image_index'. Double-clicking on search results will open the script at the location where 'image_index' occurs, and you can easily replace it with your own variable by using the search-and-replace function from the script editor, opened with Ctrl + F. This process is necessary due to a bug in the Game Maker Studio HTML5 export module that is beyond Edge Engine's control.

6.3.2. make_color_hex("#RRGGBB")

```
make_color_hex("#0066FF");
```

The `make_color_hex` script augments built-in Game Maker Studio color support with yet another color standard: hex, also known for its usage as HTML color code. As Edge Engine imitates HTML/CSS in multiple ways, `make_color_hex` is a fundamental piece of the package, bringing this familiar standard to Game Maker Studio for the first time.

But `make_color_hex` can be used to create a color for anything, not just Edge VN text. The color format, as shown in the example above, follows a syntax of *red, red, green, green, blue, blue*, with each value ranging from 0-F. That is, each value can be: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F—that's a range of 16 for each value, resulting in 32-bit color notation (as each color in the RGB scale is represented by two consecutive values). In standard practice, a # should be included before the hex notation, but the script is designed to function with or without it (therefore `make_color_hex("#FFFFFF")` and `make_color_hex("FFFFFF")` would both return the color white). Note that the input hex color code **must be a string.** If any sort of malformed input is supplied, the script will either return `c_white` or its best guess at a color based on the given information.

> Note: You can download `make_color_hex` independently of Edge VN for free! Download it here.

6.4. Creating Text

While EdgeVN is comprised of many different scripts, using them correctly is actually quite simple when you know how, and you'll only need two types of objects to do it: a single **text log** object, and as many **text block** objects as your story requires.

The **text log** is by far the easier of the two to wrap your head around, and it also happens to come first, before any text blocks are created. Only one persistent text log object is needed in each game, and once it is set up, it will happily sit in the background and do its job without any further effort on the part of the developer. And what is its job, exactly? As the name implies, to keep a running record of all the text that goes on in your text blocks. That way if a line of text is accidentally skipped or just needs a bit of context the player can easily look back through past dialogue and reorient themselves. The log is displayed as an overlay and can be summoned or closed at any time with the touch of a button. **While mandatory for Edge VN**, if for some reason a text log is not desired, it can be disabled by simply not running the `edgevn_toggle_log` script and setting the log size to 0 from the included log object's Create event.

Text blocks are how your story is actually told, and they offer about as much freedom as the blank pages you'll be filling with your script. Each text block consists of 1) a background scene, 2) one or more character cut-ins, 3) a textbox, 4) a 'continue' prompt, and last but far from least, 5) one or more lines of text. But as this is *your* story, each element is highly customizable, and in many cases, even entirely optional. There is technically no limit to how much dialog can be contained in a single block, but it is recommended to only use 200 or fewer lines of plain text, or 100 or fewer lines of markup text (more on that later) to avoid causing a moment or two of lag upon a block's creation on less powerful hardware. This being the case, you are likely to have many text block objects in your game. For this reason, it is strongly recommended you organize your scenarios into folders of their own and give your text block objects simple, incremental names, so that one can be created right after another with the `edgevn_create_next` script. Think of it as a flowchart with Game Maker objects!

These are the two pillars of Edge VN, but naturally you'll need more than just a basic concept of each to use them effectively. In summary, here's what each object will look like. To gain a detailed understanding of how they work, refer to the rest of this reference guide to follow.

Summary

Text Log:

Create event:

```
edgevn_create_log(25);
```

Step event:

```
edgevn_log_scroll_swipe(5);
```

Draw GUI event:

```
edgevn_draw_log(spr_log_bg, -1, c_white, 24, 1000, spr_audio_icon);
edgevn_draw_log_controls(spr_scroll_up, spr_scroll_down);
```

Press Up event:

```
edgevn_log_scroll_up();
```

Press Down event:

```
edgevn_log_scroll_down();
```

Press Space event:

```
edgevn_log_play_sound();
```

Press L event:

```
edgevn_toggle_log();
```

Text Block:

Create event:

```
edgevn_init_block(true);
edgevn_create_char(0, “John Doe”);
edgevn_create_text(0, snd_mysound, Arial, “John Doe”, “Hello, world!”);
```

Step event:

```
animate_sprites(1);
edgevn_auto(2);
```

Global Mouse Left Pressed event:

```
edgevn_continue(snd_continue);
edgevn_option_nav_select(snd_select);
```

Draw GUI event:

```
edgevn_draw_scene(0, bg_demo, 0, 1, 0, 0, 1.5, 1, 1, false, true);
edgevn_draw_char(0, 0, 1, spr_char_body, spr_face_listening, spr_face_talking, 0,
display_get_gui_height(), 300, 175, true, true, 2);
edgevn_draw_textbox(spr_textbox_bg, 0, display_get_gui_height(), 2, 2);
edgevn_draw_prompt(display_get_gui_width() – 120, display_get_gui_height() – 40,
spr_cont_incomplete, spr_cont_complete, spr_continue_auto);
edgevn_draw_text(150, display_get_gui_height() – 200, Arial, c_yellow, 150,
display_get_gui_height() – 150, Arial, c_white, true, 1, 32, 1200, true);
```

Press Enter event:

```
edgevn_option_nav_select(snd_select);
```

Press Space event:

```
edgevn_continue(snd_continue);
```

Press Up event:

```
edgevn_option_nav_up(snd_hover);
```

Press Down event:

```
edgevn_option_nav_down(snd_hover);
```

Press A event:

```
edgevn_toggle_auto();
```

Press P event:

```
edgevn_toggle_pause();
```

Press V event:

```
edgevn_toggle_visibility();
```

6.5. The Text Log

The **text log** (or **backlog**) is a background element of visual novels, usually hidden, that records each line of text as it is displayed on-screen for users to review later for context or if some lines were accidentally skipped. Edge VN includes a simple, yet robust text log that records text, dialog choices, and dialog audio for review in a smooth, customizable interface.

Important note: it is recommended to set your text log object to **persistent** and a **depth** of **-999999**. While Edge VN will attempt to set both automatically, this automatic setting can be overridden in a number of ways, such as when a room is restarted or a saved game is loaded. These types of actions will bypass the Create event of the log object and thereby skip over automatic depth and persistence settings, in which case both must be set manually as a fallback.

6.5.1. edgevn_create_log(size)

```
edgevn_create_log(25);
```

Creates the Edge VN text log with the specified maximum **size**. When the maximum size has been reached, the oldest entry will be deleted to make room for a new one. Recommended size is 25, as very large logs can be cumbersome to sift through, while very small logs do not offer enough context to be useful. Note that this script should not need to be run more than once per game, as the text log itself should be a persistent object. **This script must be run *before* any text blocks are created, or else the game will crash!**

6.5.2. edgevn_draw_log(textbox, font, color, lineheight, linebreak, icon)

```
edgevn_draw_log(spr_log_bg, -1, c_white, 24, 1000, spr_audio_icon);
```

Draws the text log with the given stylization. In this case **textbox** refers **not** to the textbox of the current text block, but the textbox of each individual logged line of text. The textbox sprite also serves to determine the area in which text is drawn, so even if a textbox background is not desired a transparent rectangle must be used in its place.

Be careful to plan your text stylization ahead of time when designing your textbox sprite. Typically, you will want to use a **font** smaller than your text blocks and with a shorter linebreak and lineheight as well, but this is entirely up to you. To forego a custom font entirely, use `-1` to default to 12pt Arial. **Lineheight** and **linebreak** are two parameters you'll become very familiar with in Edge VN. Lineheight refers to the *vertical* space between lines of text, and linebreak refers to the *horizontal* space before a string of text is wrapped into multiple lines. Both parameters are measured in pixels.

Note that simply running `edgevn_draw_log` in your log object's Draw GUI event will not actually display the log on-screen! It will also need to be toggled as visible with the `edgevn_toggle_log script`!

6.5.3. edgevn_draw_log_controls(up sprite, down sprite)

```
edgevn_draw_log_controls(spr_scroll_up, spr_scroll_down);
```

Draws on-screen controls for scrolling through the text log while on display, using the given sprites as buttons. These buttons are automatically aligned to the middle right of the view and pre-configured to function with mouse and touch input methods. Note that this script is optional on platforms where keyboards or gamepads are mandatory. For such hardware controls, use the `edgevn_log_scroll_up` and `edgevn_log_scroll_down` scripts. It is also possible to use the `edgevn_log_scroll_swipe` script as a replacement if no on-screen buttons are desired.

6.5.4. edgevn_toggle_log()

```
edgevn_toggle_log();
```

Opens or closes the text log. Designed for use in keyboard and gamepad input events, but can be triggered in any event as desired (for example, by an on-screen button). **Must** be run in the text log object itself, or applied to the log object by means of a ‘with’ statement.

6.5.5. edgevn_clear_log(destroy)

```
edgevn_clear_log(true);
```

Resets the text log to its original, blank state and removes it from memory, optionally also **destroying** the log object itself, if enabled (set to ‘true’). While this is not usually necessary, it can be helpful to optimize long sections of gameplay in which the log will not be used. Note that if this script is run, `edgevn_create_log` must be run again before the log will be usable again, and in the meantime no further text blocks can be drawn, as any attempts to add new entries to the log will fail. **Must** be run in the text log object itself, or applied to the log object by means of a ‘with’ statement.

6.5.6. edgevn_log_scroll_up()

```
edgevn_log_scroll_up();
```

Scrolls the log upward by one entry. Designed for use in keyboard and gamepad input events, but can be triggered in any event as desired (for example, by an on-screen button). **Must** be run in the text log object itself, or applied to the log object by means of a ‘with’ statement.

6.5.7. edgevn_log_scroll_down()

```
edgevn_log_scroll_down();
```

Scrolls the log downward by one entry. Designed for use in keyboard and gamepad input events, but can be triggered in any event as desired (for example, by an on-screen button). **Must** be run in the text log object itself, or applied to the log object by means of a ‘with’ statement.

6.5.8. edgevn_log_scroll_swipe(sensitivity)

```
edgevn_log_scroll_swipe(5);
```

Scrolls the log up or down when swiped on a touchscreen or click-and-dragged with a mouse. The **sensitivity** parameter sets the number of ‘steps’ the screen is divided into, therefore higher values equal higher sensitivity. The ideal value for most displays will be `5-8`. A value of `0` is not acceptable and will crash the game, however negative values are acceptable to reverse scrolling orientation. This script **must** be run in the text log object itself, or applied to the log object by means of a ‘with’ statement.

6.5.9. edgevn_log_play_sound()

```
edgevn_log_play_sound();
```

Plays the logged audio associated with the currently-highlighted log entry, if any. Designed for use in keyboard and gamepad input events, but can be triggered in any event as desired (for example, by an on-screen button).

Must be run in the text log object itself, or applied to the log object by means of a ‘with’ statement.

6.6. Text Blocks

Key Variables

In addition to its many useful scripts, Edge VN also features a number of even lower-level functions as built-in variables that can be accessed and/or modified to achieve results beyond what the included scripts offer. In particular, the built-in **`str_current`**, **`scene_current`**, **`text_complete`**, and **`text_auto`, `text_visible`,** and **`global.edgevn_pause`** variables are critical pieces of Edge VN that exist at the core of basic Edge VN functionality and can be very useful when employed in custom code.

As its name implies, `str_current` is a variable that indicates the current string ID being drawn. While there is absolutely no need to manually tamper with `str_current` in basic usage of Edge VN, for advanced users it becomes a key to creating many interesting results. By using `str_current` in simple ‘if’ statements, any code can be triggered when the visual novel progresses to a certain line of text.

In the same vein, `scene_current` indicates the current scene being drawn, which can be useful for adding custom code to control how scenes are displayed without having to repeat code for each scene ID.

For more basic usage, `text_complete` is a simpler and likewise less specific approach to basically the same set of tasks as `str_current`. This true/false variable indicates whether the text is still being drawn onto the screen with the typewriter effect or if the effect is complete and the string is fully drawn.

The final set of variables, `text_auto`, `text_visible`, and `global.edgevn_pause` can be used to both set and detect whether their respective properties of Edge VN are enabled. This can be very useful in programming visual elements in particular so that your own custom graphics behave in-line with Edge VN. Note that `text_auto` and `text_visible` are local variables to the active text block and must be employed in custom code either within the text block object directly or by means of a ‘with’ statement. They will also be destroyed along with the text block object, so use them carefully.

Learn to use these variables in your own custom code effectively and you’ll be well on your way to mastering visual novel presentations.

6.6.1. edgevn_init_block(first)

```
edgevn_init_block(true);
```

Initializes a new text block with all necessary variables, functions and text drawing surfaces. Should only be run once per text block at the very beginning of the object’s Create event, **even before `edgevn_create_text`!** As of version 1.5.0, `edgevn_init_block` also has a special parameter, **first**, for flagging the text block being initialized as the first in a string or not the first. ‘First’ text blocks will initialize global auto mode and pausing variables that are better off not being initialized again with each successive block object. By setting this value to ‘true’ for the first block and ‘false’ for all other blocks following, auto mode can be continuous across blocks.

Note that multiple blocks can safely have this value set to ‘true’, but if all blocks are set to ‘false’ the game will crash upon attempting auto or pause functions.

6.6.2. edgevn_create_text(id, sound, font, break, speaker, text)

```
edgevn_create_text(0, snd_voice, fnt_Arial, true, “John Doe”, “Hello, world!”);
```

Creates a single line of text to be drawn in Edge VN format. Should be run in the Create event of a text block. This script is where your visual novel will actually be written, and as such it is important to have a good grasp on the parameters required to operate it.

The **text ID** is an integer which simply marks the order of each line of text in a text block. This integer must start at `0` and increment by 1. While theoretically there is no limit to how many lines of text a single text block can contain, because each line must be parsed for markup, having more than a few hundred occurrences of `edgevn_create_text` can cause a noticeable delay upon the text block's creation depending on amount of markup and the speed of the CPU being used. If very little or no markup is used, most devices will handle hundreds of strings without stuttering. Therefore, in most cases the ID parameter will be a number with a range of 0-200 or thereabouts, each representing its own line of text in the text block.

As of version 1.8.0, `-1` can also be supplied as the text ID to calculate the order automatically with **auto ID mode**. This is extremely useful for editing your script later, removing a line or adding a new line in, as you no longer have to change the ID of the rest of your text to reflect the changes to your script. However, the downside of relying on auto ID is that you have no clear indicator of what each string's ID is without manually counting the instances of this script from 0. Many Edge VN scripts rely on knowing which text ID is numerically which, so don't assume that auto ID is always the best choice.

The **sound** parameter allows a certain sound file to be played when the given line of text is displayed, such as voiceover or sound effects. This sound does not loop and will be stopped as soon as the next line is displayed, so do not attempt to trigger background music with this script (use Edge VN Audio functions instead).

The **font** parameter determines the default font for text to be drawn in `edgevn_draw_text` for the current string, and also sets the font to be used for processing certain forms of markup. Note that this font can be overridden in `edgevn_draw_text`, however certain markup may not display correctly if the processing and drawing fonts are of dissimilar size. In `edgevn_draw_text` a font value of `-1` will use the font set in this script, and in this script, a value of `-1` will default to 12pt Arial.

Next, the **break** parameter is a true/false value specifying whether the line of text should be drawn independently or as a continuation of the previous line. This can be useful for dramatic effect by introducing a pause in the middle of a sentence or, more importantly, to create 'NVL'-style visual novels where the text occupies most of the screen and prints line by line rather than entire paragraphs at once. In most cases this value should be set to 'true', with 'false' only being for these special scenarios. Note that non-broken lines will always override the speaker name to match the previous string.

Lastly, the **speaker** and **text** parameters are a pair of strings that make up the real meat of any visual novel: the name of who's speaking, and what they're saying. Get creative and let your imagination be your guide! Note that the name of the speaker must match **exactly** with the name of a character set in `edgevn_create_char` in order for character animations to function.

As mentioned, EdgeVN also supports multiple types of markup tags for strings in the 'text' parameter: **color**, **reference**, **bold**, and **italic**. **Right-to-left** text can also be triggered in this manner.

Color tags are written in the format `[color=#FFFFFF]colored text[/color]`, with color notation being in hex format by way of the `make_color_hex` script. This allows for certain parts of the string to be drawn in separate colors from the rest, such as for indicating keywords like an item or location. Note that because this markup is already part of a string, contrary to regular `make_color_hex usage` the color notation should **not** be enclosed in quotes, and **preceding the notation with a # is mandatory**.

The second type of markup, called reference tags, or **ref tags** for short, functions similarly to a link in a web browser. Written in the format `[ref=ev_user0]reference text[/ref]`, ref tags allow you to trigger certain actions when the text between tags is clicked (or tapped on a touch screen). This is achieved by *referring* to one of Game Maker Studio's **User events**, found in Other > User Defined in the Game Maker Studio event selector. Whenever a ref tag is selected, its respective user event will be executed on the spot, allowing for some very dynamic interactivity. With the `url_open` function built-in to Game Maker Studio, you really can even use ref tags as links!

Note that text enclosed in ref tags cannot be line broken, or else only the first line will function. It is also recommended to use color tags in conjunction with ref tags so that players know there is something to click on (unless you *want* hidden easter eggs)! It is also important to note that while Game Maker Studio supports up to 15 user events, Edge VN currently only supports user events 0-9 for use in ref tags, or in GML, `ev_user0`, `ev_user1`, `ev_user2`, and so on.

As of version 1.3 Edge VN also supports **bold** and **italic** markup, simply written in the format `[b]bold text[/b]` and `[i]italic text[/i]`. As with all Edge VN markup, it is possible to mix these tags for bold-italic text as well. Note, however, that due to the nature of Game Maker Studio heavy use of these styles can slightly affect performance, so it is recommended to use them sparingly on low-powered hardware.

> Note: Italic text may not display correctly on mobile and HTML5 target platforms due to different export modules' handling of textured primitives. Mileage will vary; some fonts will look better than others.

Finally, as of version 1.7.0, Edge VN also supports drawing text in reverse for use with **right-to-left** (RTL) languages, such as Arabic or Hebrew. To trigger RTL drawing, it must first be enabled by editing the edgevn_create_text script and uncommenting (removing the '//' from) line 163, which reads:

```
//rtl_count[str_id] = string_count("[rtl]", argument5); //UNCOMMENT TO ENABLE THIS FEATURE
```

RTL processing is disabled by default so as to save performance for users who do not need it. With RTL processing enabled, simply enclose your text in the appropriate tag, written as `[rtl]right-to-left text[/rtl]`. This will trigger drawing both the entire line of text and the name of the speaker in reverse so that RTL languages will be displayed correctly. To learn how to import non-Latin fonts for use with RTL languages, see documentation on Using Additional Languages in Edge VN.

> Note: Please allow at least two characters between occurrences of markup tags of the same type, otherwise parsing errors may occur. The game will not crash, but overlapping can occur between tags resulting in drawing errors. Note that this does **not** apply to mixing different types of tags, which can be directly inside and beside each other without issue.

6.6.3. edgevn_draw_prompt(x, y, sprite1, sprite2, sprite3)

```
edgevn_draw_prompt(1500, 600, spr_prompt_waiting, spr_prompt_ready, spr_prompt_auto);
```

Draws a three-state prompt at the given location to indicate the status of the text being drawn. This prompt can take any form desired, be it a spinning circle, a triangle that bobs up and down, or virtually anything else. As you'd expect, the **x** and **y** parameters determine where the prompt is drawn on the screen. The three sprite parameters that follow, however, are more specific: the first **sprite** parameter determines the sprite to be shown while text is *incomplete*, the second **sprite** parameter determines the sprite to be shown when text is *complete*, and the third **sprite** parameter determines the sprite to be shown when Edge VN is running in *auto* mode. If in any case transparency is desired over an actual sprite, `-1` may be supplied instead.

> Note: Sprites not animating? Make sure you're running the `animate_sprites` script!

6.6.4. edgevn_draw_text(speaker_x, speaker_y, speaker_font, speaker_color, text_x, text_y, text_font, text_color, shadow, speed, lineheight, linebreak, clear)

```
edgevn_draw_text(150, display_get_gui_height()-200, fnt_Arial, c_yellow, 150,
display_get_gui_height()-150, -1, c_white, true, 3, 32, 1200, true);
```

Draws all text created with `edgevn_create_text` line by line, in order from beginning to end, with the given stylization and markup (if any). Contrary to `edgevn_create_text`, which must be run once for every line, only one occurrence of `edgevn_draw_text` is required to draw every line. While it may appear complex at the onset, using it is actually fairly simple.

The **speaker x** and **speaker y** parameters determine where to draw the speaker name of the current line of text, with the drawn text itself being aligned to the top left corner. As its name implies, the **speaker font** parameter specifies the font to draw the speaker name in. This can be the same or an entirely different font than the rest of the text, with a value of `-1` defaulting to 12pt Arial. Likewise, **speaker color** sets a separate color to draw the speaker name in.

The same goes for **text x** and **text y—**these values determine where the text will be drawn. Alignment here is a bit different, however, as text can and often will wrap into multiple lines. As such, in order to maintain consistency the entire string is aligned by the top left corner of the *first line*, possibly of many, in the string of text. This also means that contrary to the text log, text block text is not automatically centered, leaving it more open to a variety of uses and stylizations. To center text yourself, first decide on your linebreak and then set 'text x' to *half* the remaining horizontal space in the screen. For example, with a screen resolution of 1600 horizontal pixels and a linebreak of 1200, 'text x' should be set to 200, which is half of 400 (1600 – 1200). It may be a good idea to use a slightly lower value to allow for some overflow (e.g. 150 instead of 200).

The **text font** parameter differs from the speaker font parameter in that it is merely an override for the font set in `edgevn_create_text`. To use the set font for the current string in this script, set the text font parameter to `-1`. Otherwise, supplying a specific font here will override the set font in all strings. Note that this is also different from setting the font in `edgevn_create_text`, where a value of `-1` will default to 12pt Arial. **Text color** operates on similar principles. This parameter determines the **base color** to draw text in, however this value will be temporarily overridden if color markup is used.

Shadow is a boolean (true/false) option to enable or disable drawing a shadow beneath the text. Beyond being a stylistic option, disabling a text shadow can improve performance on lower-powered devices.

> Note that shadows may not display correctly on mobile devices and HTML5 platforms without support for WebGL.

The **speed** parameter controls Edge VN's typewriter effect—that is, the rate at which the characters in a string are printed onto the screen. This can be any positive real number, and essentially determines how many characters are drawn in each step. For example, a speed of 2 would draw 2 characters in one step, 2 more characters the next step, and so on until the string is drawn in its entirety. Use `0` or `-1` to disable the typewriter effect and draw entire strings instantaneously. As of version 1.8.4, fractional values (e.g. `0.25`) are also acceptable.

Next, **lineheight** and **linebreak** are two parameters you'll become very familiar with in Edge VN. Lineheight refers to the vertical space between lines of text, and linebreak refers to the horizontal space before a string of text is wrapped into multiple lines. Both parameters are measured in pixels.

Lastly, the **clear** parameter is a true/false value enabling or disabling automatic clearing and destroying of the text block object when all text has been drawn. This value should be set to 'false' if `edgevn_create_next` is being used to create a string of text block objects, and 'true' if no further text blocks in the string are to be made.

Note that simply drawing your text block with this script will not make it continue through each line on its own! For that you'll need to run the `edgevn_continue` script, or alternatively `edgevn_toggle_auto`.

> Note: Due to Edge VN's use of surfaces, it is recommended to run the `edgevn_draw_text` script last when possible, as any drawing done on top of the text surface will be dramatically slower than drawing done below.

6.6.5. edgevn_continue(sound)

```
edgevn_continue(snd_cont);
```

Skips the typewriter effect of the current string (if enabled and in-progress), or continues from one line to the next if the current string is already fully drawn. As of version 1.2, also optionally plays a **sound** effect on navigation, but this sound only plays when continuing from one string to another, not when skipping the current string. If no sound is desired, use `-1` for none.

Designed for use in keyboard and gamepad input events, but can be triggered in any event as desired (for example, by a timer). **Must** be run in the current text block object itself, or applied to the block object by means of a 'with' statement.

6.6.6. edgevn_clear_block(destroy)

```
edgevn_clear_block(destroy);
```

Clears data from the text block object and removes it from memory, preparing for new blocks to be created. **Must** be run in the current text block object itself, or applied to the block object by means of a 'with' statement. Note however that running this script manually is almost never necessary, as it is automatically triggered by `edgevn_draw_text` (optional), `edgevn_draw_option`, `edgevn_goto`, and `edgevn_create_next`. If any of these scripts is set to run `edgevn_clear_block`, **do not** manually run `edgevn_clear_block` a second time, as running the script multiple times may cause the game to crash! On the other hand, **do** make sure to always run this script when destroying a text block prematurely, as failing to do so could cause a memory leak (bloated, wasted RAM usage)!

> Note: if that sounds a bit scary, fear not: Edge VN is highly optimized for minimum CPU and RAM usage. Failure to properly clear a single text block will only result in approximately 10-15MB of wasted RAM. While not a crippling amount, if repeated enough that number can ascend to problematic heights, so don't push it.

6.6.7. edgevn_create_next(object, immediate)

```
edgevn_create_next(obj_block2, false);
```

Creates a new text block object when the current text block is complete, clearing the current block from memory. As its name implies, the **object** parameter specifies the text block to be created. By default, this script should be run in the Step or Draw GUI event and allowed to detect when the block should be created on its own, however if more manual control is desired, the **immediate** parameter can be set to 'true' to create the new text block without waiting for it to end. If this type of operation is not desired, this value should be set to ‘false’ (recommended). Additionally, this script will return the instance ID of the newly created object.

6.6.8. edgevn_goto(text block, string)

```
edgevn_goto(obj_block2, 5);
```

Jumps directly to a certain **text block** object and **string ID** number within that text block, effectively scene-skipping all content up to the desired line of text. As of version 1.9.1, backtracking is fully supported, as is skipping over all forms of actions. Note that this includes any dialog options that may be between the current string and the target string being skipped to.

6.6.9. edgevn_auto(delay)

```
edgevn_auto(3);
```

Automatically triggers `edgevn_continue` after the number of seconds specified in the **delay** parameter, after the typewriter effect (if any) is complete. This creates a hands-off and even-paced reading experience, which may be more desirable for some users than clicking through line after line of text manually.

This script must be run in the Step event of the current text block object, however **simply running the script will not enable auto mode**! For that, run the `edgevn_toggle_auto` script.

Note that if audio is supplied in the `edgevn_create_text` script, `edgevn_continue` will not be triggered until the audio is complete. As of version 1.8.4, this script will also wait for any scenes to complete their fade transition before continuing. Otherwise `edgevn_continue` will be triggered when the typewriter effect is complete instead.

6.6.10. edgevn_toggle_auto()

```
edgevn_toggle_auto();
```

Enables or disables the `edgevn_auto` script, toggling automatic progression through the current text block. Intended for use in keyboard or gamepad input events, but can be triggered by any other means as well, such as an on-screen button.

6.6.11. edgevn_toggle_visibility()

```
edgevn_toggle_visibility();
```

Shows or hides both the textbox and text surface, allowing players to view background and character art free of obstruction. Hidden text cannot be skipped or continued. Otherwise this is a purely cosmetic feature. Note that text will be forced visible where dialog options are used. Intended for use in keyboard or gamepad input events, but can be triggered by any other means as well, such as an on-screen button.

6.6.12. edgevn_toggle_pause()

```
edgevn_toggle_pause();
```

Pauses or unpauses Edge VN as a whole, applying to the text typewriter effect, character animations, voiceover audio, and more. The exception here is Audio functions, which will continue to run in the background even while paused. Intended for use in keyboard or gamepad input events, but can be triggered by any other means as well, such as an on-screen button.

6.6.13. edgevn_block_complete()

```
if edgevn_block_complete() == true { … }
```

Unlike most other Edge VN scripts, edgevn_block_complete doesn't do anything on its own, however it can be used in custom code to achieve a variety of useful effects. When this script is run, it will return either 'true' or 'false' depending on if the text block has finished drawing its last string and is therefore 'complete'. This can be useful for manually triggering a variety of effects and integrating Edge VN into other genres of games.

6.7. Scenes and Scene Effects

While Edge VN can be drawn on top of literally any other code or style of game, for pure visual novels it is important to engage the player with scenery and set the physical context for the story being told. For that, Edge VN includes a robust scene system complete with background transitions and effects. This background system exists independently of Game Maker Studio's *room* backgrounds, but uses standard background *assets* in ways not possible with built-in room settings.

6.7.1. edgevn_draw_scene(id, background, start, end, x, y, fade, scale, zoom, repeat, parallax)

```
edgevn_draw_scene(0, bg_scene, 0, 10, 0, 0, 3, 1, 1, false, false);
```

Draws a background asset with built-in fade, positioning, scaling, tiling, and parallax options. Like many other Edge VN assets, scenes are handled in a series delineated by an **id**, or an integer starting at `0` and increasing by 1 for each new scene in the text block. As of version 1.7.0, scenes are completely independent of built-in GameMaker background functions, so performance is the only limit for how many scenes can be drawn. However, `edgevn_draw_scene` still relies on *background* assets for drawing, not sprites, and therefore the **background** parameter specifies a background asset, not a sprite, to be assigned to the scene ID.

The **start** parameter indicates which text ID to begin displaying the scene on, and the **end** parameter the text ID to stop displaying the scene on. Note that unlike other Edge VN assets, the 'end' parameter is not clamped to the end of the block, meaning the same scene can be seamlessly drawn across multiple text blocks with no transition in-between.

Next, the **x** and **y** parameters specify where to display the background relative to the GUI.

The **fade** parameter is a real number value setting the length of time for the background to fade in and out, in seconds. If no fade is desired, it can be disabled with `0` or `-1`.

Contrary to its name, the **scale** parameter does **not** set a direct scale multiplier for the background. Instead it offers three different modes of scaling the background to the screen automatically: `0` for no scaling, `1` for full scaling (cropped to fill the screen completely), or `2` for proportion scaling (scaled to match changes in resolution, not necessarily to fill the screen). The **zoom** parameter, on the other hand, can be thought of as a real number scale multiplier where a value of `1` equals 100% size, and any greater or smaller values will zoom the scene in or out, respectively. Zoom also affects passive parallax scenes by essentially setting the distance, where more distant (smaller zoom) scenes have reduced parallax compared to nearer (higher zoom) scenes.

The **repeat** parameter is a true/false value which enables or disables drawing the background as an endless tile both vertically and horizontally. Note that tiled backgrounds cannot be rotated, and any scene effects that use rotation will not apply.

Finally, the **parallax** parameter is a true/false value that enables or disables offsetting the position and scale of character and other scene elements relative to that of the current scene. With this feature enabled, scenes will appear to have a 3D effect during motion, creating a very engaging and dramatic look to the visual novel. As of version 1.8.0, a value of `other` instead of `true` or `false` will apply another scene's parallax to the current scene, allowing for some very dynamic compositions with a sense of real depth.

> Note: Looking for the 'foreground' option from earlier versions of Edge VN? As of version 1.7.0 it is no longer necessary—simply run `edgevn_draw_scene` after other Edge VN elements and it will be drawn on top! This allows for much greater simplicity and flexibility in your code.

> Note 2: As of version 1.7.0 Edge VN no longer uses Game Maker Studio's built-in backgrounds. This means you can employ built-in backgrounds yourself however you like without regard for Edge VN! Both GameMaker and Edge VN scenes can exist together completely without conflict.

6.7.2. edgevn_scene_pos(id, start, x, y, zoom, speed)

```
edgevn_scene_pos(0, 5, 100, 200, 2, 5);
```

Re-positions and re-scales the specified scene **id** at the text ID indicated by the **start** parameter. The **x** and **y** values specify the new location for the scene, while the **zoom** value specifies the new zoom multiplier. Scenes will be transitioned smoothly from current to new values. The rate of this transition is set by the **speed** multiplier, where a value of `1` is considered normal speed.

6.7.3. edgevn_scene_style(id, start, rotation, color, alpha, speed)

```
edgevn_scene_style(0, 5, 45, c_aqua, 0.75, 3);
```

Applies a number of styling options not available in `edgevn_draw_scene` to the specified scene **id** at the text ID indicated by the **start** parameter. The **rotation** parameter is a value in degrees tilting the scene around its top-left corner—therefore you may wish to run `edgevn_scene_pos` in tandem to offset the scene's position appropriately to compensate for rotation. The **color** parameter specifies a certain color to blend the scene with, where a value of `c_white` is default and effectively no blending. Colors can be set using built-in Game Maker Studio color values and functions or with Edge VN's `make_color_hex`. The **alpha** parameter is a transparency value ranging from 0-1, where `0` is completely transparent and `1` is completely opaque. Scenes will be transitioned smoothly from current to new values. The rate of this transition is set by the **speed** multiplier, where a value of `1` is considered normal speed.

6.7.4. edgevn_scene_effect(id, start, stop, effect, speed, loop)

```
edgevn_scene_effect(0, 5, 10, ef_earthquake, 3, true);
```

As of version 1.1, Edge VN has an extensible scene effects system with a number of included effects and a wide range of possible additions to the included effects library. These effects are found in the 'Effects' folder within the 'Scenes' folder, and are designed to be run with the `edgevn_scene_effect` script. In this script, the **id** parameter specifies the scene to apply an effect to, and the **start** and **stop** parameters indicate the text IDs to begin and end the effect on, with the **effect** parameter being the desired effect script and the **speed** parameter being a multiplier to affect the speed of the given effect, where a value of `1` is standard. Effects can be played once or looped endlessly by setting the **loop** parameter to either false or true, respectively. If an effect is not looped, the ‘stop’ parameter should ideally be only 1 greater than the ‘start’ parameter, as the scene is not manipulable by other scripts until the ‘stop’ string is reached regardless of whether the effect is playing or not.

Edge VN includes five scene effects total out of the box: `ef_earthquake`, `ef_flash`, `ef_impact`, `ef_lightning`, and `ef_siren`. Additional effect scripts can be easily written by following the keyframe syntax template in each existing effect.

Be creative! Perhaps a single, slow flash could be used to indicate a great idea, while high-speed lightning could simulate paparazzi, or a very slow earthquake could be used to imitate a ship at sea. Don’t get hung up on the name of the effect being used—chances are each one can be much more when used imaginatively!

6.8. Characters & Animations

Characters play a major role in visual novels. That’s true not only in terms of the story being told, but in crafting the experience of the game and creating a real connection between the player and the story at hand. Characters add life to the scene and convey even deeper meaning through their actions and expressions. For this reason, Edge VN features a robust character system every bit as powerful as its text and scene systems.

6.8.1. edgevn_create_char(id, name)

```
edgevn_create_char(0, “John Doe”);
```

Like `edgevn_create_text`, this script initializes a new character graphic to be drawn with `edgevn_draw_char`. This script must be run in a text block object’s Create event, either before or after `edgevn_create_text`. Also like `edgevn_create_text`, the **character ID** is a running count beginning at `0` and incrementing by `1` for each new character, allowing multiple characters to coexist. While there’s technically no limit to how many characters can be in a scene, bear in mind that each one consumes a small amount of resources, and having many characters at once can bog down less powerful devices.

As of version 1.8.0, `-1` can also be supplied as the character ID to calculate the order automatically with **auto ID mode**. This is extremely useful for editing your script later, removing a character or adding a new one in, as you no longer have to change the ID of the rest of your characters to reflect the changes to your script. However, the downside of relying on auto ID is that you have no clear indicator of what each character’s ID is without manually counting the instances of this script from 0. Many Edge VN scripts rely on knowing which character ID is numerically which, so don’t assume that auto ID is always the best choice.

The character **name** parameter is, as it sounds, a string identifying the character by name. This is used to determine when the character is speaking, and when to animate and focus/unfocus the character (if the focus effect is enabled in `edgevn_draw_char`). As such, the name must match a speaker name used in `edgevn_create_text` *exactly*. Both name strings are case sensitive. In simple terms, you’re creating a character and writing what they are saying, and the character’s name is the link between these two elements.

Note that while your character and text speaker *names* should match, character and text *IDs* are completely separate from each other.

6.8.2. edgevn_draw_char(id, entry, exit, body, listening face, talking face,

body x, body y, face xoffset, face yoffset, focus, flip, transition)

```
edgevn_draw_char(0, 0, 50, spr_john_body, spr_john_listening, spr_john_talking, 0,
1080, 200, 150, true, true, 2);
```

Draws a character created with `edgevn_create_char`. Must be run in a text block object, ideally *before* `edgevn_draw_textbox` so that characters appear underneath text elements.

The **character ID** specifies which created character to draw, and the **entry** and **exit** parameters set which text ID the character should *enter* the scene on and which text ID the character should *exit* the scene on, respectively. Note that the exit value is clamped, so `9999` for example would always equal the end of the text block and no more.

The next few parameters deal with actually drawing the character itself. Edge VN characters are intended to be comprised of two separate elements: a body and a face. Thus, the **body sprite** parameter should refer to a character sprite with a blank face, and the **listening sprite** and **talking sprite** to two different sprites for two different states of faces: a non-talking face, and a talking face. This technique is optional, however—to forgo using separate face sprites, simply use `-1` in place of these two parameters. Also note that the offset, or alignment of all of these sprites is not forced and must be set manually. It is recommended to align character body sprites to the bottom left corner.

The next group of parameters specifies where to display the character elements on the screen. The **body x** and **body y** parameters set the exact coordinates to draw the body sprite, and like them the **face xoffset** and **face yoffset** set the position to draw the face sprite. Unlike the body coordinates, however, **face coordinates are relative to the *body sprite***, making a coordinate of `0,0` the top-left corner of the body sprite regardless of the body sprite's alignment. Only one set of coordinates is needed for both states of face.

The **focus** parameter is a true/false value to enable or disable dimming and highlighting characters based on the current speaker. The functionality of this effect is based on the character name matching up with the speaker name of a certain line or lines of text in the text block.

The **flip** parameter is a true/false value to enable or disable flipping the character depending on which side of the screen they are drawn on. When using this functionality, characters should be designed *facing to the right* by default. Note that flipping a character also effectively flips its horizontal alignment (e.g. if the body sprite is aligned to the bottom left on the left side of the screen, it will be aligned to the bottom right on the right side of the screen). This functionality is not permanent whether enabled or disabled and can be set again while re-positioning later with the `edgevn_char_pos` script.

Lastly, the **transition** parameter sets which of three types of animation to play when the character enters or exits the screen. A value of `0` disables transition animations entirely, while a value of `1` will fade and a value of `2` will slide.

> Note: Want to go for a more classic look? Instead of full graphics, make small character portraits and draw them on top of your text boxes instead of underneath! And remember, having separate face sprites is optional. Edge VN character scripts can even be used for non-characters, like cut-in windows showing comic book-style frames!
>
> Note 2: Want a character to perform different transitions on entry and exit? Create a variable of your own ranging 0-2 and change the value after the character has been created, then assign the 'transition' parameter to your custom variable!

6.8.3. edgevn_draw_char_ext(id, entry, exit, animation, speed, loop, body, listening face, talking face, x, y, focus, flip, transition)

```
edgevn_draw_char_ext(0, 0, 50, anim_breathing, 2, true, spr_john_body,
spr_john_listening, spr_john_talking, 0, 1080, true, true, 2);
```

Draws a character created with `edgevn_create_char` with a real-time ‘dynamic’ vertex-based animation. Must be run in a text block object, ideally *before* `edgevn_draw_textbox` so that characters appear underneath text elements.

The **character ID** specifies which created character to draw, and the **entry** and **exit** parameters set which text ID the character should *enter* the scene on and which text ID the character should *exit* the scene on, respectively. Note that the exit value is clamped, so `9999` for example, would always equal the end of the text block and no more.

The next few parameters deal with actually drawing the character itself. The **animation** parameter specifies a dynamic animation script to apply to the drawn character, and the **speed** parameter determines how fast the animation should play, with standard speed being a value of `1`, regardless of FPS. As of version 1.8.0 the **loop** parameter can be used to enable or disable endlessly repeating the dynamic animation with either a ‘true’ or a ‘false’. In most cases this value should be set to ‘true’.

Edge VN characters are intended to be comprised of two separate elements: a body and a face. Thus, the **body sprite** parameter should refer to a character sprite with a blank face, and the **listening face sprite** and **talking face sprite** to two different sprites for two different states of faces: a non-talking face, and a talking face. This technique is optional, however—to forgo using separate face sprites, simply use `-1` in place of these two parameters. Also note that the offset, or alignment of all of these sprites is not forced and must be set manually. It is recommended to align character body sprites to the bottom left corner.

The next pair of parameters specifies where to display the character elements on the screen. The **x** and **y** parameters set the exact coordinates to draw the character body. Unlike the standard `edgevn_draw_char` script, face sprite positions are set within dynamic animation scripts themselves, with values being relative to the top-left corner of the *body sprite* regardless of the body sprite’s alignment. Only one set of coordinates is needed for both states of face.

The **focus** parameter is a true/false value to enable or disable dimming and highlighting characters based on the current speaker. The functionality of this effect is based on the character name matching up with the speaker name of a certain line or lines in the text block.

The **flip** parameter is a true/false value to enable or disable flipping the character depending on which side of the screen they are drawn on. When using this functionality, characters should be designed *facing to the right* by default. Note that flipping a character also effectively flips its horizontal alignment (e.g. if the body sprite is aligned to the bottom left on the left side of the screen, it will be aligned to the bottom right on the right side of the screen). This functionality is not permanent whether enabled or disabled and can be set again while re-positioning later with the `edgevn_char_pos` script.

Lastly, the **transition** parameter sets which of three types of animation to play when the character enters or exits the screen. A value of `0` disables transition animations entirely, while a value of `1` will fade and a value of `2` will slide.

> Note: Dynamic characters are relatively demanding and may cause poor performance on low-powered devices. Advanced users should ideally include an option in their games for the user to enable or disable dynamic characters.

> Note 2: Dynamic characters are not supported by most HTML5 platforms.

6.8.4. edgevn_draw_char_emote(id, start, sprite, xoffset, yoffset, speed)

```
edgevn_draw_char_emote(0, 5, spr_emote, 400, -800, 1);
```

Draws a brief, animated sprite over a character to demonstrate emotion. The **id** parameter sets which character ID to apply the 'emote' to, and the **start** parameter sets which text ID to perform the emote on, while the **sprite** parameter is the emote itself. Technically there is no restriction on what this sprite can be, however small sprites are recommended, and animated sprites are required. These sprites are intended to be icons such as a rain drop to show sadness, or a question mark to show confusion. The **xoffset** and **yoffset** parameters set the coordinates to draw the emote *relative to the character body* similarly to face sprites, however this script should not be used for drawing faces—see `edgevn_draw_char` or `edgevn_draw_char_ext`. Lastly, the **speed** parameter is a multiplier which sets the rate at which the sprite animation performs, where a value of `1` essentially equals 30 FPS. Emotes animate independently of `animate_sprites`.

As of version 1.8.4, emotes automatically adjust position and scale to match the character they are assigned to. This takes automatic character flipping into account as well—only one xoffset is necessary for characters facing either direction.

6.8.5. edgevn_char_pos(id, start, x, y, flip)

```
edgevn_char_pos(0, 25, 200, 1080, false);
```

Re-positions a character drawn with `edgevn_draw_char`. The **id** parameter specifies which character ID to re-position. The **start** parameter specifies which *text* ID to re-position the character on, and the **x** and **y** parameters dictate where on the screen to re-position the character at. Lastly, the **flip** parameter is a true/false value which enables or disables flipping the character based on which side of the room it is on. Characters will smoothly transition to the new location. This script should be run in the current text block's Draw GUI event, preferably after `edgevn_draw_char`.

6.8.6. edgevn_char_style(id, start, body, listening face, talking face, xscale, yscale, alpha)

```
edgevn_char_style(0, 10, spr_john_waving, spr_john_happy, spr_john_chatting, 1.25,
1.25, 1);
```

Assigns a new set of sprites and display options to the character specified in the **id** parameter, beginning at the text ID specified in the **start** parameter. See `edgevn_draw_char` for information on the **body**, **listening face**, and **talking face** parameters. Additional features not available in `edgevn_draw_char` include the **xscale, yscale,** and **alpha** parameters, which set the size and transparency of the character, respectively: **xscale** refers to the horizontal scale of the character sprite and **yscale** the vertical scale, where a value of `1` equals 100% size. **Alpha**, on the other hand, is a real number ranging from 0-1 where `0` is completely transparent (invisible) and `1` is completely opaque (visible).

Note that this script can be applied to both static and dynamic characters, however applying this script to a dynamic character will not convert it to a static character.

6.8.7. edgevn_char_style_ext(id, start, body, listening face, talking face, animation, animation speed, loop, xscale, yscale, alpha)

```
edgevn_char_style_ext(0, 10, spr_john_waving, spr_john_happy, spr_john_chatting,
anim_wave, 3, false, 1.25, 1.25, 1);
```

Assigns a new set of sprites and animations to the dynamic character specified in the **id** parameter, beginning at the text ID specified in the **start** parameter. See `edgevn_draw_char_ext` for information on the **body**, **listening face**, **talking face**, **animation**, **animation speed**, and **loop** parameters. Additional features not available in `edgevn_draw_char_ext` include the **xscale, yscale,** and **alpha** parameters, which set the size and transparency of the character, respectively: **xscale** refers to the horizontal scale of the character sprite and **yscale** the vertical scale, where a value of `1` equals 100% size. **Alpha**, on the other hand, is a real number ranging from 0-1 where `0` is completely transparent (invisible) and `1` is completely opaque (visible).

Note that unlike `edgevn_char_style`, this script can **only** be applied to dynamic characters. Applying this script to a static character will not convert it to a dynamic character, and will attempt to set values not present in standard `edgevn_draw_char`.

6.9. Textbox and Textbox Effects

The textbox is a vastly underappreciated element of any visual novel. It's almost always there, and it might not seem to do much, but in reality it isn't just a container for your story; your textbox communicates something all by itself. It's the focal point of your entire user interface, and with a bit of stylization and animation, it can also be a very visually engaging piece to enhance players' reading experience. Thus, as of version 1.8.0 Edge VN features a textbox system based on the same code as scenes, giving this subtle element the robust implementation it deserves.

6.9.1. edgevn_draw_textbox(sprite, x, y, scale, transition)

```
edgevn_draw_textbox(spr_textbox, 0, 1080, 2, 1);
```

Draws a textbox background with the given sprite, screen position, scaling, and one of three in/out animations. Note that the textbox bears no actual relation to the text drawing process, therefore it is up to you to design a background large or small enough to adequately contain your lines of text—or even to decide whether to use one at all.

First of all, the **sprite** parameter specifies the sprite asset to use as a textbox, and just like when drawing a regular sprite, the **x** and **y** parameters set where on the screen to draw the textbox. Unlike regular sprites, however, this script also features three different **scale** modes, where a value of `0` will not apply scaling at all, `1` will scale the textbox on both axes to fill the display horizontally, and `2` will scale on the x-axis only, essentially stretching the textbox rather than purely scaling it.

Finally, the **transition** parameter gives a few options for elegantly entering and exiting the textbox from the screen, with `0` being no transition, `1` being a simple fade, `2` being a vertical slide, and `3` being a horizontal scale effect. For new text blocks being strung together, it's a good idea to set this value to `0` so the transition between text blocks is not noticeable.

6.9.2. edgevn_textbox_pos(trigger, x, y)

```
edgevn_textbox_pos(5, 200, 800);
```

```
edgevn_textbox_pos(“John Doe”, obj_john.x, obj_john.y);
```

Smoothly re-positions the textbox on the given **trigger**, which can be either a text ID (an integer) or a character name (a string). Text ID triggers are permanent modifications that are applied when the specified string has been reached, whereas character name triggers apply the new scene position only when that character is speaking, and then resets when they are not. And of course, the **x** and **y** parameters specify the coordinates on the screen for the textbox to re-position itself to.

6.9.3. edgevn_textbox_style(trigger, sprite, color)

```
edgevn_textbox_style(5, spr_textbox2, c_yellow);

edgevn_textbox_style(“John Doe”, spr_textbox, c_blue);
```

Applies a new textbox sprite and color blending value on the given **trigger**, which can be either a text ID (an integer) or a character name (a string). Text ID triggers are permanent modifications that are applied when the specified string has been reached, whereas character name triggers apply the new scene styling only when that character is speaking, and then resets when they are not. The **sprite** parameter specifies a sprite asset to replace the textbox with, while the **color** parameter specifies a blending value, where `c_white` equals no blending.

6.9.4. edgevn_textbox_effect(start, stop, effect, speed, loop)

```
edgevn_textbox_effect(10, 11, ef_attention, 3, false);
```

As of version 1.8.0, Edge VN has an extensible textbox effects system with a number of included effects and a wide range of possible additions to the included effects library. These effects are found in the ‘Effects’ folder within the ‘Textbox’ folder, and are designed to be run with the `edgevn_textbox_effect` script. In this script, the **start** and **stop** parameters indicate the text IDs to begin and end the effect on, with the **effect** parameter being the desired effect script and the **speed** parameter being a multiplier to affect the speed of the given effect, where a value of `1` is standard. Effects can be played once or looped endlessly by setting the **loop** parameter to either false or true, respectively. If an effect is not looped, the ‘stop’ parameter should ideally be only 1 greater than the ‘start’ parameter, as the scene is not manipulable by other scripts until the ‘stop’ string is reached regardless of whether the effect is playing or not.

Edge VN includes four textbox effects total out of the box: `ef_attention`, `ef_exclamation`, `ef_nod`, and `ef_shake`. Additional effect scripts can be easily written by following the keyframe syntax template in each existing effect.

6.10. Branching Text Options

As of version 1.2.0, Edge VN has built-in support for branching story paths using on-screen text options and different text blocks to take your stories in as many different directions as you can imagine. Think of text blocks as nodes on a flowchart, where dialog options are at every split and intersection—in fact, it is recommended to create such a flowchart in planning your story ahead of time.

6.10.1. edgevn_draw_option(id, entry, text, text block, block string, x, y, font, color, main background, hover background, select background, sound, transition)

```
edgevn_draw_option(0, 5, “Choose me!”, obj_block2, 0, 500, 200, fnt_Arial, c_white,
spr_option, spr_option_hover, spr_option_select, snd_option_hover, 2);
```

The primary script for handling story branches, `edgevn_draw_option` creates a dialog option, displays it, and handles mouse/touch input all at once (aside from `edgevn_option_nav_select`).

Like other elements in Edge Engine, the text **option id** refers to an array forming a list—literally, in this case. Edge VN options are intended to be drawn vertically, with the topmost option having an ID of `0`, the next option down having an ID of `1`, and so on. And as always, it is critical to begin with `0`, else the game will crash.

The **entry** parameter specifies which text ID of the current text block object to display options on (see `edgevn_create_text`). The **text** parameter, as the name implies, is a string label for the dialog option itself (e.g. “This is your first option”, in quotes). The **text block** parameter specifies the object to be created if the text option is selected. By design this should be a text block containing the related dialog to the option itself, but technically any object at all can be supplied here to achieve a variety of effects. If the current text block is input, it will continue as normal or jump to the specified block string. The **block string** parameter can target a specific string in the text block, created with `edgevn_create_text`, so that when the given text option is selected the following text block will jump directly to the given block string. A value of `0` indicates the first string in the new text block.

The **x** and **y** parameters specify where to display the text option. Again, option items are intended to be drawn as a vertical list, however each option can be positioned freely. Be aware, however, that unusual positioning may result in awkward navigation with `edgevn_option_nav_up` and `edgevn_option_nav_down`.

The rest of the parameters in this script refer to stylization: the **font** and **color** parameters set the text font and color, respectively, and the **main background**, **hover background,** and **select background** set a trio of sprites to display when the option is non-highlighted, highlighted, and selected, respectively. The **sound** parameter sets a sound to be played when the option is hovered by mouse or touch, however this is optional and can be disabled by supplying `-1` instead of the name of an audio asset. Finally, the **transition** parameter sets the option to appear and disappear with no animation (`0`), a fade animation (`1`), or a slide animation (`2`).

As the name of the script implies, `edgevn_draw_option` must be run in the Draw GUI event of a text block object, ideally before `edgevn_draw_text` and after `edgevn_draw_char` for best performance and composition.

6.10.2. edgevn_option_nav_up(sound)

```
edgevn_option_nav_up(snd_nav);
```

Navigates upward in a list of dialog options, looping around to the bottom if necessary. Also optionally plays a sound effect on navigation via the **sound** parameter. If no sound is desired, use `-1` for none. This script is designed for use in keyboard or gamepad pressed input events.

6.10.3. edgevn_option_nav_down(sound)

```
edgevn_option_nav_down(snd_nav);
```

Navigates downward in a list of dialog options, looping around to the top if necessary. Also optionally plays a sound effect on navigation via the **sound** parameter. If no sound is desired, use `-1` for none. This script is designed for use in keyboard or gamepad pressed input events.

6.10.4. edgevn_option_nav_select(sound)

```
edgevn_option_nav_select(snd_select);
```

Selects the currently highlighted dialog option, setting the text block object and string ID specified in `edgevn_draw_option`. Also optionally plays a sound effect on navigation via the **sound** parameter. If no sound is desired, use `-1` for none. This script is designed for use in keyboard or gamepad pressed input events, or global mouse pressed input events.

6.11. Audio Functions

As of version 1.7.0, Edge VN also features a range of audio functions for playing sounds and music within the context of Edge VN events. These new functions extend the existing voiceover function present in `edgevn_create_text` and do not replace it. Instead, these functions are intended for music and sound effects rather than voices.

6.11.1. edgevn_play_sound(sound, start, volume, fade, loop, skippable)

```
edgevn_play_sound(my_sound, 15, 1, 500, false, true);
```

Plays the specified **sound** asset when the text ID set in the **start** parameter has been reached. The **volume** parameter is a real number ranging from 0-1 where `0` is silent and `1` is max volume. The **fade** parameter sets the length of time to fade the sound in, in milliseconds (a value of `1000` equals 1 second). The **loop** parameter is a true/false value enabling or disabling looping the sound being played. Looped sounds will need to be stopped with `edgevn_stop_sound` or by setting the **skippable** parameter to 'true'. A skippable sound will immediately stop if the 'start' string is exceeded while the sound is still playing, otherwise if set to 'false' the sound will continue playing until finished or stopped with `edgevn_stop_sound`.

Note that unlike other Edge VN assets, sounds played with this script, once played are not tied to the text block in any way and can continue playing into other text blocks. This script must be run in the Step event.

6.11.2. edgevn_play_sound_ext(sound, start, volume, fade, loop, skippable, begin, end, full)

```
edgevn_play_sound_ext(my_sound, 15, 1, 500, false, true, 0.5, 5, false);
```

Plays the specified **sound** asset when the text ID set in the **start** parameter has been reached, with several custom timing functions not present in `edgevn_play_sound`. The **volume** parameter is a real number ranging from 0-1 where `0` is silent and `1` is max volume. The **fade** parameter sets the length of time to fade the sound in, in milliseconds (a value of `1000` equals 1 second). The **loop** parameter is a true/false value enabling or disabling looping the sound being played. Looped sounds will need to be stopped with `edgevn_stop_sound` or by setting the **skippable** parameter to 'true'. A skippable sound will immediately stop if the 'start' string is exceeded while the sound is still playing, otherwise if set to 'false' the sound will continue playing until finished or stopped with `edgevn_stop_sound`.

The **begin** and **end** parameters set the track time in seconds to start and stop playing the sound asset. So for example, in a 30-second audio file, a 'begin' value of `10` and an 'end' value of `20` would start playing the sound at 10 seconds in and stop the sound file at 20 seconds in—a total playing time of only 10 seconds, not 30. The **full** parameter takes this type of duration cutting one step further by playing the file from the beginning *first* and *then* looping within the set range thereafter (assuming the 'loop' parameter is set to true). In this case the sound would play from 0 seconds until it hit 20 seconds and then loop back to 10 seconds and continue doing so until manually stopped.

Note that unlike `edgevn_play_sound`, sounds played with this script **cannot** carry over into other text blocks. This script must be run in the Step event.

6.11.3. edgevn_stop_sound(sound, start, fade)

```
edgevn_stop_sound(my_sound, 20, 500);
```

Stops the specified **sound** asset when the text ID in the **start** parameter has been reached. Also fades audio out before stopping, with the **fade** parameter being the length of time to fade out, in milliseconds (a value of `1000` equals 1 second).

Note that unlike other Edge VN assets, sounds played with `edgevn_play_sound` are not tied to the text block in any way and can continue playing into other text blocks. Therefore, this script can be used to stop sounds played in previous text blocks. This script must be run in the Step event.

6.11.4. edgevn_set_volume(trigger, volume, fade)

```
edgevn_set_volume(“John Doe”, 0.5, 500);

edgevn_set_volume(5, 0.5, 500);
```

Sets the volume of *voiceover* audio played with edgevn_create_text. The affected sound is determined by the **trigger** parameter, which can be either a character name (in which all lines spoken by a certain character will be affected), or a string ID (in which voiceover for the specified string will be affected). The **volume** parameter sets the new volume at which to play voiceover audio, where a value of `0` is silent and `1` is full volume. Lastly, the **fade** parameter sets the length of time to transition between volumes in milliseconds.

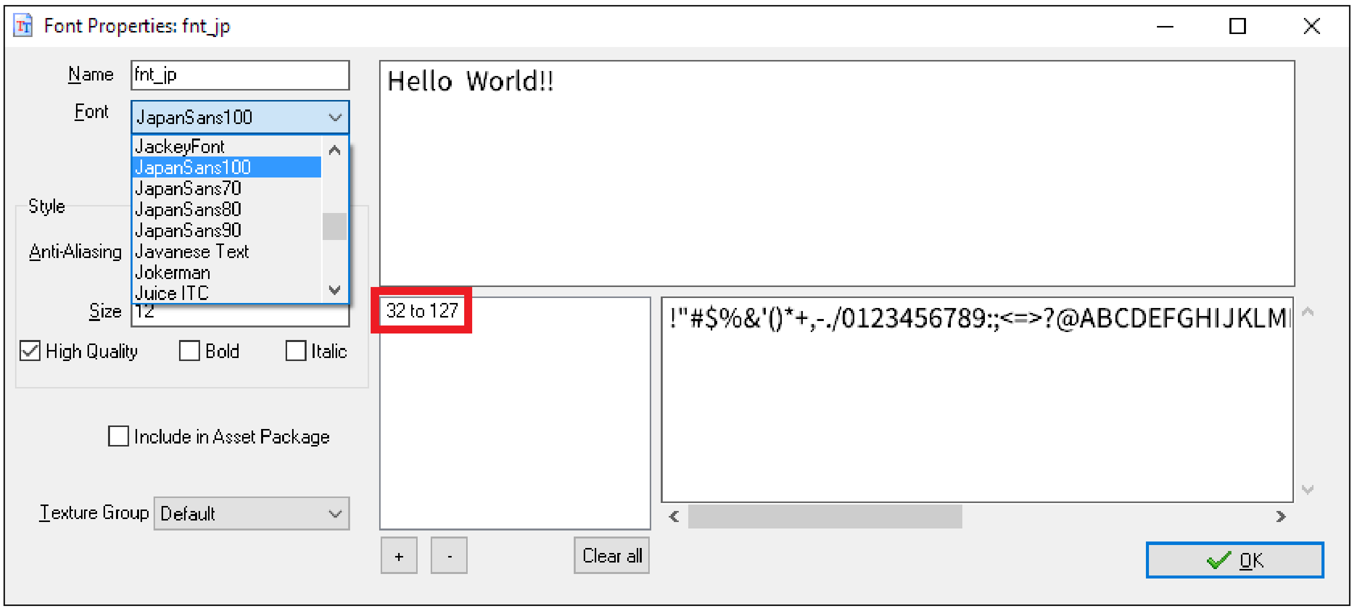
Note that this script must be run *before* `edgevn_draw_text` in order to function.

> Note: Want to change the volume of other sounds besides voiceover audio? Use the `audio_sound_gain` function built into Game Maker Studio!

6.12. Using Additional Languages in Edge VN

While Game Maker Studio and by consequence Edge VN are both 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 we must tell Game Maker Studio to *look* for the characters we need 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 your 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 Game Maker Studio.



Name: fnt_jp
Font: JapanSans100

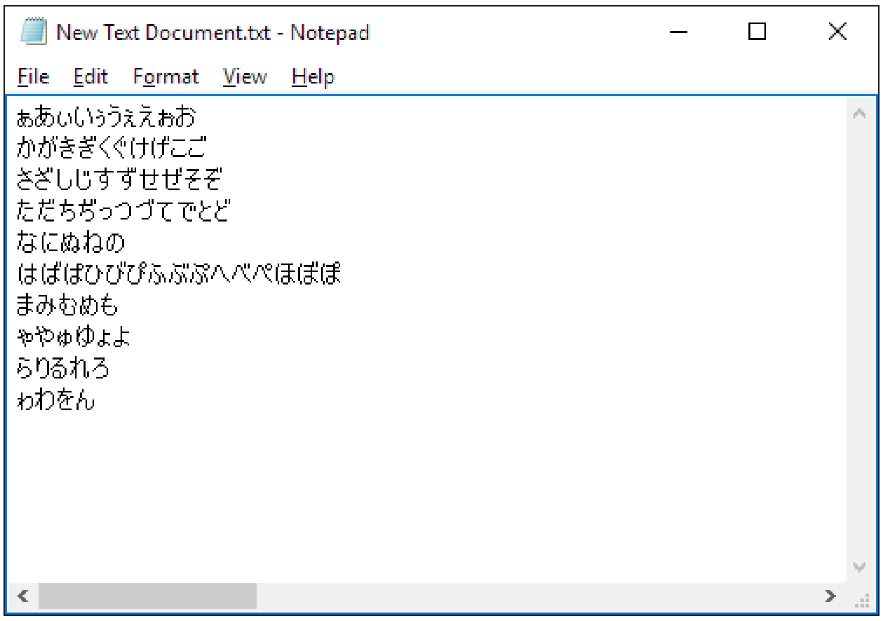
JackeyFont
JapanSans100
JapanSans70
JapanSans80
JapanSans90
Javanese Text
Jokerman
Juice ITC

Size: 12
[✓] High Quality

Hello World!!

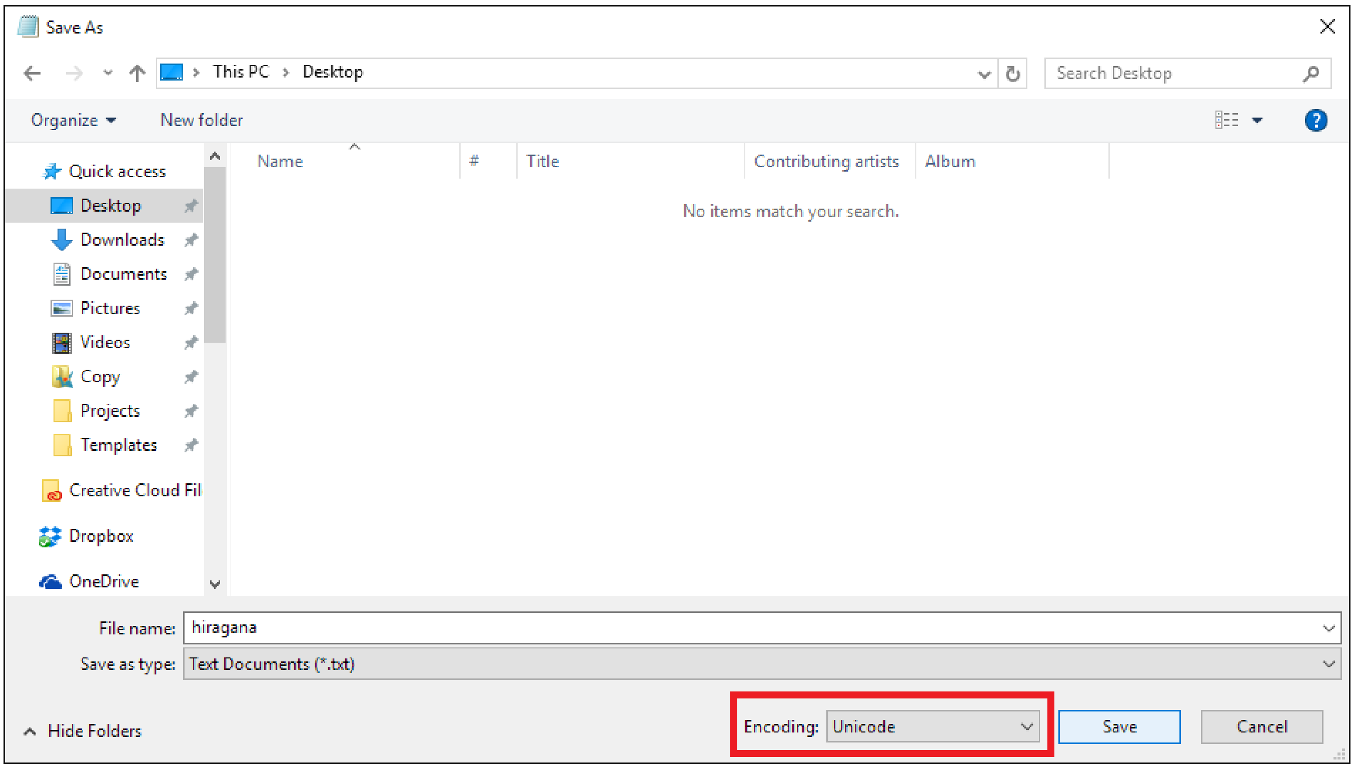
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLM

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 Game Maker 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.



ぁあぃいぅうぇえぉお
かがきぎくぐけげこご
さざしじすずせぜそぞ
ただちぢっつづてでとど
なにぬねの
はばぱひびぴふぶぷへべぺほぼぽ
まみむめも
ゃやゅゆょよ
らりるれろ
ゎわをん

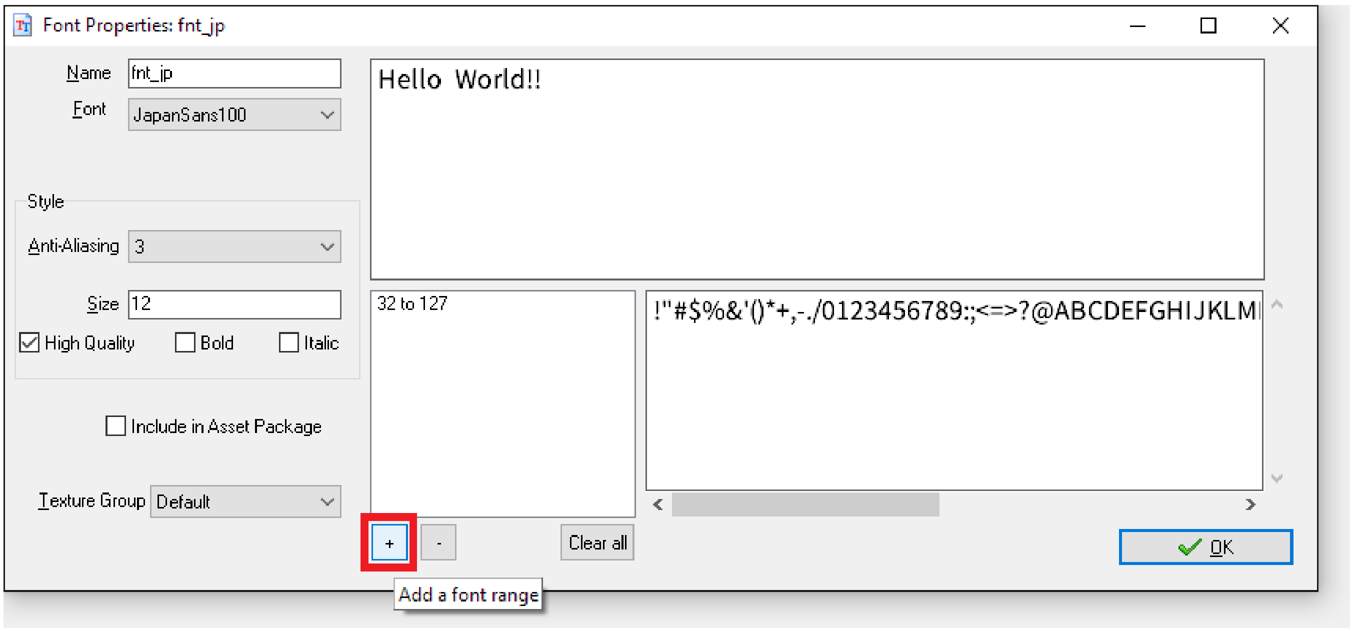
With your text file prepared, save it to your local hard drive as a .txt file with encoding set to 'Unicode'. **This step is important**, as your special characters will be lost if you attempt to save with different encoding.



No items match your search.

File name: hiragana
Save as type: Text Documents (*.txt)
Encoding: Unicode

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



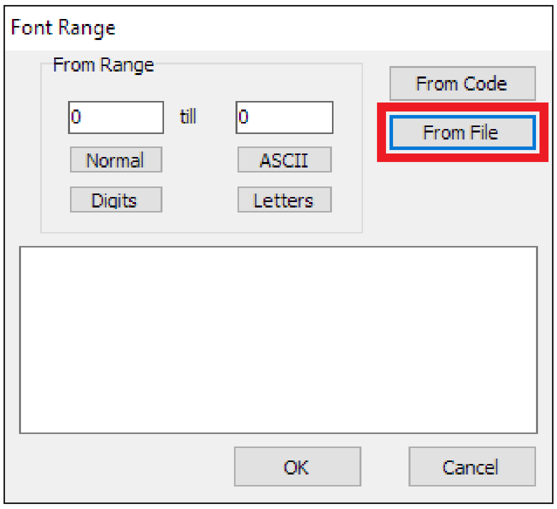
Name: fnt_jp
Font: JapanSans100
Anti-Aliasing: 3
Size: 12
☑ High Quality ☐ Bold ☐ Italic
☐ Include in Asset Package
Texture Group: Default

Hello World!!

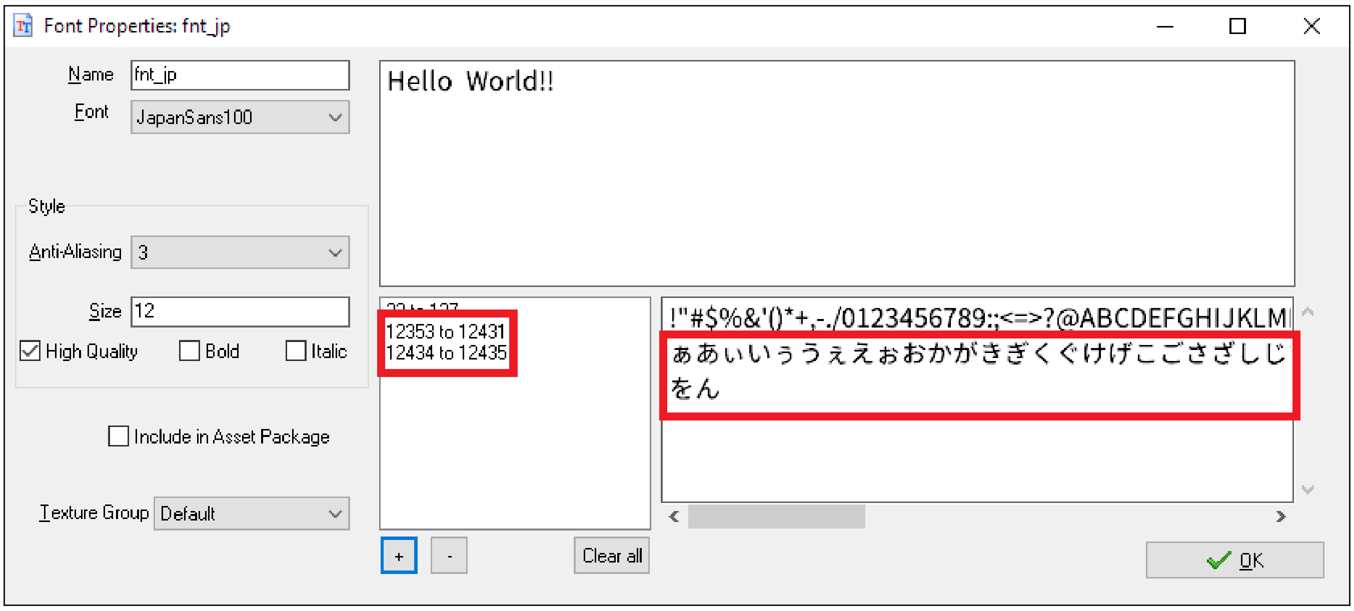
32 to 127

!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLM

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, we can instead import the text file with special characters and have Game Maker Studio detect them automatically by clicking the 'From File' button and choosing the text file saved earlier.



Click ‘OK’ and return to the main font asset window. 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 entry, that is indication that some characters were missed. However, since we can now see the exact values of the font range we are looking for, we can easily adjust the font range to include every character from the smallest value to the largest shown in the font range indicator with another click of the ‘+’ icon.

With that done, you are now ready to use your language in Edge VN!

Note: Using a right-to-left (RTL) language? See documentation for `edgevn_create_text` to display RTL languages in Edge VN!

7. End-User License Agreement ("EULA")

LAST UPDATED: 02/14/2019

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.

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.