Ariah's Menu Library

This written documentation for Ariah's Menu Library is intended to be your guide to using the library. If you would prefer to learn through video documentation, I will be uploading a video to my YouTube channel in the near future where I detail how to use the library and give a tour of the code base. This file will be uploaded with a link to the video when it is released.

Contents

License

Ariah's Menu Library is licensed under CC0. You have the right to use this library in your own game, even a commercial one, without providing attribution.

Overview

Ariah's Menu Library aims to create an api for creating menus in *GameMaker* easily instead of coding them from the ground up.

You can make all sorts of menus using this library, from pause menus to inventory screens, battle menus to menus for rebinding controls. This library is here for you to make the actual menu portion of the task easy so you can focus on the code you want your menu to do.

Importing

After downloading the library from the official download page, open your *GameMaker* project and select **Tools** -> **Import Local Package** from the toolbar.

Select the file ending in **.yymps**, click "*Add All*" and then click "*Import*".

Settings

One you have successfully imported the library, you will see a new folder in your Asset Browser called "*Menu*". It is recommended you create a folder named something along the lines of "*Libraries*" where you can drag your "*Menu*" folder (and the folder for any other libraries you might install) into.

Inside the "*Menu*" folder, you will find an object named **ui_menu**, and a script named **ariah_menu_lib_settings**. You do ***not*** need to modify the **ui_menu** object in any way.

Inside **ariah_menu_lib_settings**, you will find a few functions prefixed with "*__AriahMenuLib__*". These are the settings that you can modify.

Input Settings

These settings deal with how the player can interact with the menu.

`__AriahMenuLibrary__UP`

Versions 1.0.0+

This function by default returns `keyboard_check_pressed(vk_up)`, but you can change this to whatever logic you'd like that you want to result in going upwards on a menu.

`__AriahMenuLibrary__DOWN`

Versions 1.0.0+

This function by default returns `keyboard_check_pressed(vk_down)`, but you can change this to whatever logic you'd like that you want to result in going downwards on a menu.

`__AriahMenuLibrary__LEFT`

Versions 1.0.0+

This function by default returns `keyboard_check_pressed(vk_left)`, but you can change this to whatever logic you'd like that you want to result in going left on a menu.

`__AriahMenuLibrary__RIGHT`

Versions 1.0.0+

This function by default returns `keyboard_check_pressed(vk_right)`, but you can change this to whatever logic you'd like that you want to result in going right on a menu.

`__AriahMenuLibrary__CONFIRM`

Versions 1.0.0+

This function by default returns `keyboard_check_pressed(ord("Z"))`, but you can change this to whatever logic you'd like that you want to result in choosing the selected menu option.

`__AriahMenuLibrary__CANCEL`

Versions 1.0.0+

This function by default returns `keyboard_check_pressed(ord("X"))`, but you can change this to whatever logic you'd like that you want to result in going back a menu.

Game Size Settings

These settings deal with calculating the position and size of a menu based on the width and height of your game window.

`__AriahMenuLibrary__WIDTH`

Versions 1.0.0+

This function by default returns **320**. Change this to either a macro you have that contains your game width, or a variable you have elsewhere if your width changes.

`__AriahMenuLibrary__HEIGHT`

Versions 1.0.0+

This function by default returns **240**. Change this to either a macro you have that contains your game height, or a variable you have elsewhere if your height changes.

Menu Defaults

These settings give the defaults for various customizable aspects of a menu. You may leave these as is if you'd like, but you'll be required to set the font and the sprite for each menu you create.

You can instead change these settings and have every menu use them unless explicitly changed for that menu.

`__AriahMenuLibrary__DEFAULT_FONT`

Versions 1.0.0+

This function by default returns **undefined**. Change this to a 'font' asset you'd like to use as a default font for your menus.

`__AriahMenuLibrary__DEFAULT_SPRITE`

Versions 1.0.0+

This function by default returns **undefined**. Change this to a 'sprite' asset you'd like to use as a default container for your menus.

`__AriahMenuLibrary__DEFAULT_CONFIRM_BLIP`

Versions 1.0.0+

This function by default returns **undefined**. Change this to a 'sound' asset you'd like to use as a default confirm blip. If not undefined, this blip will play whenever `__AriahMenuLibrary__CONFIRM` returns true.

`__AriahMenuLibrary__DEFAULT_CANCEL_BLIP`

Versions 1.0.0+

This function by default returns **undefined**. Change this to a 'sound' asset you'd like to use as a default cancel blip. If not undefined, this blip will play whenever `__AriahMenuLibrary__CANCEL` returns true.

`__AriahMenuLibrary__DEFAULT_MOVE_BLIP`

Versions 1.0.0+

This function by default returns **undefined**. Change this to a 'sound' asset you'd like to use as a default move blip. If not undefined, this blip will play whenever `__AriahMenuLibrary__UP`, `__AriahMenuLibrary__DOWN`, `__AriahMenuLibrary__LEFT`, or `__AriahMenuLibrary__RIGHT` returns true.

`__AriahMenuLibrary__DEFAULT_HORIZONTAL_ANCHOR`

Versions 1.0.0+

This function by default returns "**center**". You may change this to either "**left**" or "**right**". This dictates where on the screen horizontally the menu is anchored.

If this function returns "**center**", the menu will be positioned initially in the center of your screen horizontally.

If this function returns "**left**", the menu will be positioned initially on the left side of your screen.

If this function returns "**right**", the menu will be positioned initially on the right side of your screen.

`__AriahMenuLibrary__DEFAULT_VERTICAL_ANCHOR`

Versions 1.0.0+

This function by default returns "**middle**". You may change this to either "**top**" or "**bottom**". This dictates where on the screen vertically the menu is anchored.

If this function returns "**middle**", the menu will be positioned initially in the middle of your screen vertically.

If this function returns "**top**", the menu will be positioned initially on the top side of your screen.

If this function returns "**bottom**", the menu will be positioned initially on the bottom side of your screen.

`__AriahMenuLibrary__DEFAULT_X_OFFSET`

Versions 1.0.0+

This function by default returns **0**. This number is the x offset for where the menu is drawn.

Basically, the menu is first anchored according to the previous two settings, and then moved right the amount of pixels equal to this value.

If this value is negative, it will simply move left instead.

`__AriahMenuLibrary__DEFAULT_Y_OFFSET`

Versions 1.0.0+

This function by default returns **0**. This number is the y offset for where the menu is drawn.

Basically, the menu is first anchored according to the previous two settings, and then moved down the amount of pixels equal to this value.

If this value is negative, it will simply move up instead.

`__AriahMenuLibrary__DEFAULT_COLORS`

Versions 1.0.0+

This function by default returns:

```
{
  c1: #ffffff,
  c2: #aaaaaa,

  s1: #ffff00,
  s2: #aaaa00,
}
```

You must leave the variable names (`c1`, `c2`, `s1`, `s2`) the same, but you may modify the hex codes assigned to each color.

`c1` and `c2` are the colors used to create unselected text. They create a gradient from top to bottom where the top color is `c1` and the bottom color is `c2`.

Similarly, `s1` and `s2` are the colors used to create selected text. They create a gradient from top to bottom where the top color is `s1` and the bottom color is `s2`.

Getting Started

Once you've modified the settings to your liking, you can start using the library.

Basic Structure

Whenever you need a menu, you can create one using the following basic structure:

```
var _menu = instance_create_depth(0, 0, 0, ui_menu);
with(_menu) {

}
```

All of the code for designing the menu will go inside that `with()` statement.

Adding a Page

There are three types of pages currently recognized by the library. The types are named after how pages using that type are organized.

> ### MenuPageRows
>
> This page type is organized vertically into rows, and each row can have multiple options within.
>
> ### MenuPageColumns
>
> This page type is organized horizontally into columns, and each column can have multiple options within.
>
> ### MenuPageFree
>
> This page type is unorganized. Each option must also give an ***x*** and a ***y*** for its location.

To add the page, add this block of code to the end of your `with()` statement:

```
add_page(new MenuPageRows("Name")

);
```

You can of course replace `MenuPageRows` with either `MenuPageColumns` or `MenuPageFree`. You can also replace the "**Name**" string with whatever you wish the name of this page to be.

The name *is* required, as its used in navigation. You must use the name to reference which page you would like to navigate to.

You can repeat that basic block of code as many times as you wish to have pages. You can also have each page use a different page type.

Adding an Option

Within a page with type `MenuPageRows` or `MenuPageColumns`, you can include the block of code:

```
add_page(new MenuPageRows("Name"))

    .add_option(
        new MenuOption("Name")
    )

);
```

Of course again, you can replace the "**Name**" string with whatever you want. This string will both be the name of this option as well as the text the option displays. If you wish these to be different from each other, it is as simple as adding a second argument:

```
new MenuOption("Name", "Text")
```

If your page has type `MenuPageFree`, you must add more information to the `add_option` function. Here is an example:

```
add_page(new MenuPageFree("Name")

    .add_option(
        new MenuOption("Option 1")
        ,
        x, y, [halign], [valign]
    )

);
```

Additional Arguments for MenuPageFree

> ***x*** ~ a number, the x coordinate of the option on the screen.
>
> ***y*** ~ a number, the y coordinate of the option on the screen.
>
> ***halign*** *(optional)* ~ either "***left***", "***right***", or "***center***", the horizontal position of the option's text relative to its coordinate *(default: "**center**")*.
>
> ***valign*** *(optional)* ~ either "***top***", "***bottom***", or "***middle***", the vertical position of the option's text relative to its coordinate *(default: "**middle**")*.

You can, again, repeat this basic block for as many options as you would like on your page.

```
// An example of a page with multiple options
add_page(new MenuPageRows("Main")
    .add_option(
        new MenuOption("Play")
    )
    .add_option(
        new MenuOption("Settings")
    )
    .add_option(
        new MenuOption("Quit")
    )
);
```

If you run your game, this menu will work. You will see each of your options and you can navigate between them. Obviously we haven't given these options any functionality yet though.

Adding Functionality to an Option

You can add functionality to your options by giving them various functions that are called when different things happen. You can find a full list of these over in the Function Index.

```
set_function_z(function() {
  // Code
})
```

Versions 1.0.0+

Description

The code within this function will run when the player chooses this option.

For example, if we wanted the quit button from the previous example to end the game when chosen, we can do this:

```
add_page(new MenuPageRows("Main")
    .add_option(
        new MenuOption("Play")
    )
    .add_option(
        new MenuOption("Settings")
    )
    .add_option(
        new MenuOption("Quit")
          .set_function_z(function() {
            game_end();
          })
    )
);
```

Linking a Second Page

We can also use this method to link to a second page. You can use the following function in a `.set_function_z()` block.

```
set_page(page);
```

Versions 1.0.0+

Arguments

> ***page*** ~ a string, the name of the page you wish to set.

Description

Calling this function will change the currently active page to the one with the given name.

Let's update our example to make the *Settings* option take us to another page, titled "*Settings*".

```
add_page(new MenuPageRows("Main")
    .add_option(
        new MenuOption("Play")
    )
    .add_option(
        new MenuOption("Settings")
          .set_function_z(function() {
            set_page("Settings");
          })
    )
    .add_option(
        new MenuOption("Quit")
          .set_function_z(function() {
            game_end();
          })
    )
);
```

Of course we would need to add a "*Settings*" page in order for this to work.

For a complete list of what you can have an option do, check the Function Index

Function Index

The functions provided by this library are divided into three types: **Menu Functions**, **Page Functions**, and **Option Functions**.

Menu Functions

The following functions should be used either...

1. Within the initial `with()` statement:

```
var _menu = instance_create_depth(0, 0, 0, ui_menu);
with(_menu) {

  // These functions go here.
  example_function();

  add_page(
    ...
  );
}
```

2. Within a `.set_function_*()` function.

```
new MenuOption("Option")
  .set_function_z(function() {

    // These functions go here.
    example_function();

  })
```

3. From an *outside object* using the `menu.` prefix (where **menu** is a variable containing your menu instance).

```
var _menu = instance_create_depth(0, 0, 0, ui_menu);
with(_menu) {
  ...
}

// These functions go here.
_menu.example_function();
```

Menu Management

```
set_name(name);
```

Versions 1.0.0+

Arguments

> ***name*** ~ a string, the name you wish to give your menu.

Description

You can use this function to name your menu. This is useful if you plan on having multiple menus open at once or if you plan to check if a specific menu is open.

```
get_name();
```

Versions 1.0.0+

Description

This function returns the name of this menu. If the name was never set with `get_name()`, it returns **undefined**.

```
destroy();
```

Versions 1.0.0+

Description

You can use this function to close the menu.

```
request_focus();
```

Versions 1.0.0+

Description

Calling this function will remove *focus* from all other open menus and give focus to this one.

For more information on what *focus* means, check out the documentation for `has_focus()` below.

```
has_focus();
```

Versions 1.0.0+

Description

This function returns whether or not this menu has *focus*.

Focus determines which menu can be interacted with by the player at a given moment.

If there are no other menus (or if none of the other menus currently have *focus*) when you create one, it is given *focus*.

If another menu already has *focus* when this one is created, it does not have *focus*, but you can give it *focus* by calling `request_focus()`.

```
set_visible(visible);
```

Versions 1.0.0+

Arguments

> ***visible*** ~ a boolean, whether or not the menu should be visible.

Description

You can use this function to make the menu *invisible* (or make a previously invisible menu *visible*).

```
is_visible();
```

Versions 1.0.0+

Description

This function returns whether or not the menu is currently *visible*.

```
set_fade(fade);
```

Versions 1.0.0+

Arguments

> ***fade*** ~ a boolean, whether or not the menu should fade in and out.

Description

You can use this function to set whether or not this menu should fade *in* when it's *created* and *out* when it's *destroyed* (or closed).

```
set_darken_background(dark);
```

Versions 1.0.0+

Arguments

> ***dark*** ~ a boolean, whether or not the menu should darken the background.

Description

You can use this function to set whether or not this menu should draw a semi-transparent black background underneath the menu.

```
set_function_step(function() {
  // Code
});
```

Versions 1.0.0+

Description

You can use this function to provide code that runs every step that this menu is open.

Setting Defaults

```
set_default_blips(blip_z, blip_x, blip_move);
```

Versions 1.0.0+

Arguments

> ***blip_z*** ~ a 'sound' asset, the sound to use as the default confirm blip for this menu.
>
> ***blip_x*** ~ a 'sound' asset, the sound to use as the default cancel blip for this menu.
>
> ***blip_move*** ~ a 'sound' asset, the sound to use as the default move blip for this menu.

Description

You can use this function to set the default blips for this menu. If this function is not set, the menu will by default use the blips defined in the settings `__AriahMenuLibrary__DEFAULT_CONFIRM_BLIP`, `__AriahMenuLibrary__DEFAULT_CANCEL_BLIP`, and `__AriahMenuLibrary__DEFAULT_MOVE_BLIP`.

```
set_default_anchor(anchor_h, anchor_v);
```

Versions 1.0.0+

Arguments

> ***anchor_h*** ~ either "**left**", "**center**", or "**right**", the horizontal anchor to use as the default for this menu.
>
> ***anchor_v*** ~ either "**top**", "**middle**", or "**bottom**", the vertical anchor to use as the default for this menu.

Description

You can use this function to set the default anchor locations for this menu. If this function is not set, the menu will by default use the locations defined in the settings `__AriahMenuLibrary__DEFAULT_HORIZONTAL_ANCHOR` and `__AriahMenuLibrary__DEFAULT_VERTICAL_ANCHOR`.

```
set_default_position(x_offset, y_offset);
```

Versions 1.0.0+

Arguments

> ***x_offset*** ~ a number, the x offset to use as the default for this menu.
>
> ***y_offset*** ~ a number, the y offset to use as the default for this menu.

Description

You can use this function to set the default positional offsets for this menu. If this function is not set, the menu will by default use the offsets defined in the settings `__AriahMenuLibrary__DEFAULT_X_OFFSET` and `__AriahMenuLibrary__DEFAULT_Y_OFFSET`.

```
set_default_font(font);
```

Versions 1.0.0+

Arguments

> ***font*** ~ a 'font' asset, the font to use as the default font for this menu.

Description

You can use this function to set a default font for this menu. If this function is not set, the menu will by default use the font defined in the setting `__AriahMenuLibrary__DEFAULT_FONT`.

```
set_default_sprite(sprite);
```

Versions 1.0.0+

Arguments

> ***sprite*** ~ a 'sprite' asset, the sprite to use as the default container for this menu.

Description

You can use this function to set a default sprite for this menu. If this function is not set, the menu will by default use the sprite defined in the setting `__AriahMenuLibrary__DEFAULT_SPRITE`.

```
set_default_colors(color1, [color2], [color3], [color4]);
```

Versions 1.0.0+

Arguments

> ***color1*** ~ a hex code, the first color to set as a default color for this menu.
>
> ***color2*** *(optional)* ~ a hex code, the second color to set as a default color for this menu.
>
> ***color3*** *(optional)* ~ a hex code, the third color to set as a default color for this menu.
>
> ***color4*** *(optional)* ~ a hex code, the fourth color to set as a default color for this menu.

Description

You can use this function to set the default colors for this menu. This function works a little differently in that depending on the amount of colors provided, they are used in slightly different ways.

If **1** color is provided, an ***unselected*** option will by default be this color solidly, and a ***selected*** option will be the opposite color (calculated as `#ffffff - color`).

If **2** colors are provided, an ***unselected*** option will by default be the *first color* solidly, and a ***selected*** option will be the *second color* solidly.

If **3** colors are provided, an ***unselected*** option will by default be a vertical gradient with the *first color* on top and the *second color* on bottom, and a ***selected*** option will be the *third color* solidly.

If **4** colors are provided, an ***unselected*** option will by default be a vertical gradient with the *first color* on top and the *second color* on bottom, and a ***selected*** option will be a vertical gradient with the *third color* on top and the *fourth color* on bottom.

Page Manipulation

```
add_page(page);
```

Versions 1.0.0+

Arguments

> ***page*** ~ an instance of either **MenuPageRows**, **MenuPageColumns**, or **MenuPageFree**, the page to add.

Description

You can use this function to add a page to your menu.

```
set_page(page);
```

Versions 1.0.0+

Arguments

> ***page*** ~ a string, the name of the page you wish to navigate to.

Description

You can use this function to change the active page for a menu.

```
get_page(page);
```

Versions 1.0.0+

Arguments

> ***page*** ~ a string, the name of the page you wish to get.

Description

This function returns the page instance with a name that matches the one provided. If no pages have a matching name, **undefined** is returned.

```
select_option(option, [page]);
```

Versions 1.0.0+

Arguments

> ***option*** ~ a string, the name of the option you wish to select.
>
> ***page*** *(optional)* ~ a string, the name of the page you wish to select the option on *(this will **not** navigate to this page; default: **undefined**)*.

Description

You can use this function to select an option on a given page. If the **page** argument isn't given, it will select the given option on the currently active page. If no option matching the given name is found, nothing happens.

If you call this function for a page that isn't currently active, the selected option on that page will still update. This results in the new option being already selected when you navigate to the given page.

```
get_current_page();
```

Versions 1.0.0+

Description

You can use this function to get the instance of the page that is currently active.

```
get_current_option();
```

Versions 1.0.0+

Description

You can use this function to get the instance of the currently selected option.

Page Functions

The following functions should be used within a ***page*** *declaration*, should be prefixed with a `.`, and should ***never*** end with a semicolon (`;`). An example of this is shown below.

```
add_page(new MenuPageRows("Page")

    // These functions go here.
    .example_function()

);
```

Common

```
.set_anchor(anchor_h, anchor_v)
```

Versions 1.0.0+

Arguments

> ***achor_h*** ~ either "**left**", "**center**", or "**right**", the horizontal anchor to use for this page.
>
> ***anchor_v*** ~ either "**top**", "**middle**", or "**bottom**", the vertical anchor to use for this page.

Description

You can use this function to set the anchor location for this page (whether it's anchored to the left side, bottom side, center of the screen, etc.). If this is not set, it first tries to use the defaults for this menu set by `set_default_anchor()`, and if that isn't found it will use the defaults set by the settings `__AriahMenuLibrary__DEFAULT_HORIZONTAL_ANCHOR` and `__AriahMenuLibrary__DEFAULT_VERTICAL_ANCHOR`.

```
.set_position(x_offset, y_offset)
```

Versions 1.0.0+

Arguments

> ***x_offset*** ~ a number, the x offset to use for this page.
>
> ***y_offset*** ~ a number, the y offset to use for this page.

Description

You can use this function to set the positional offsets for this page (how many pixels it moves from the initial anchor position). If this is not set, it first tries to use the defaults for this menu set by `set_default_position()`, and if that isn't found it will use the defaults set by the settings `__AriahMenuLibrary__DEFAULT_X_OFFSET` and `__AriahMenuLibrary__DEFAULT_Y_OFFSET`.

```
.set_sprite(sprite)
```

Versions 1.0.0+

Arguments

> ***sprite*** ~ a 'sprite' asset, the sprite to use as the container for this page.

Description

You can use this function to set the sprite used for the container of this page. If this is not set, it first tries to use the defaults for this menu set by `set_default_sprite()`, and if that isn't found it will use the defaults set by the setting `__AriahMenuLibrary__DEFAULT_SPRITE`.

```
.get_option(option);
```

Versions 1.0.0+

Arguments

> ***option*** ~ a string, the name of the option you wish to get.

Description

This function returns the option on this page that has a name matching the one that's given. If there are no options on this page matching the given name, **undefined** is returned.[1]

MenuPageRows

```
.add_option(option, ... )
```

Versions 1.0.0+

Arguments

> ***option*** ~ an instance of `MenuOption`, the option to add to the page.
>
> You can include as many options as arguments as you wish, though they must all be instances of `MenuOption`.

Description

You can use this function to add an option to the page. Since this page is organized into rows, each call of this function will create a *new* row. Including multiple options in this function call will put them all on the *same* row.

MenuPageColumns

```
.add_option(option, ... )
```

Versions 1.0.0+

Arguments

> ***option*** ~ an instance of `MenuOption`, the option to add to the page.
>
> You can include as many options as arguments as you wish, though they must all be instances of `MenuOption`.

Description

You can use this function to add an option to the page. Since this page is organized into columns, each call of this function will create a *new* column. Including multiple options in this function call will put them all on the *same* column.

MenuPageFree

```
.add_option(option, x, y, [halign], [valign])
```

Versions 1.0.1+

Arguments

> ***option*** ~ an instance of `MenuOption`, the option to add to the page.
>
> ***x*** ~ a number, the x position for the newly added option.
>
> ***y*** ~ a number, the y position for the newly added option.
>
> ***halign*** *(optional)* ~ a string, either "**left**", "**center**", or "**right**", the horizontal alignment of this option's text *(default: "**center**")*.
>
> ***valign*** *(optional)* ~ a string, either "**top**", "**middle**", or "**bottom**", the vertical alignment of this option's text *(default: "**middle**")*.

Description

You can use this function to add an option to the page. Since this page is not organized, each option needs to also have a coordinate and an alignment. The coordinate is absolute on the screen, however once the menu is drawn, the menu will automatically shrink to fit all of its options, anchor itself to its given anchors, and then adjust itself based on its x and y offsets.

```
.set_mapping(from, direction, to)
```

Versions 1.0.1+

Arguments

> ***from*** ~ a string, the name of the option you wish top set a mapping for.
>
> ***direction*** ~ a string, the direction to map *(either "**up**", "**down**", "**left**", or "**right**").*
>
> ***to*** ~ a string, the name of the option you wish the first option to map to.

Description

You can use this function to set a mapping from one option to another. For example, if you call...

```
.set_mapping("Play", "right", "Settings")
```

...then the player can navigate ***right*** from the "Play" option to get to the "Settings" option.

Option Functions

The following functions should be used within an ***option*** *declaration*, should be prefixed with a `.`, and should ***never*** end with a semicolon (`;`). An example of this is shown below.

```
add_page(new MenuPageRows("Page")
    .add_option(
        new MenuOption("Option")

          // These functions go here.
          .example_function()
    )
);
```

Attributes

```
set_text(text);
```

Versions 1.0.0+

Arguments

> ***text*** ~ a string, the text you wish this option to display.

Description

You can use the function to set the text you wish this option to display. Please note that the option's **text** is *not* the same as the option's **name**. You use its **name** to get the option's instance from a function like `get_option()`, while its **text** is displayed on the menu. If you created your option with one argument like...

```
new MenuOption("Option")
```

...then you've set both its **name** *and* its **text** to be "Option". If you wish to set them to separate values either call `set_text()` elsewhere or add the **text** as a second argument like...

```
new MenuOption("Name", "Text")
```

```
.set_blip_z(blip)
```

Versions 1.0.0+

Argument

> ***blip*** ~ a 'sound' asset, the sound you wish set as this option's confirm blip.

Description

You can use this function to set the sound you wish to play when the player chooses this option. If this is not set, it first tries to use the defaults for this menu set by `set_default_blips()`, and if that isn't found it will use the defaults set by the setting `__AriahMenuLibrary__DEFAULT_CONFIRM_BLIP`.

You can also pass in **undefined** for ***blip***. If you do this, even if there are blips set as default there won't be a confirm blip for this option.

```
.set_blip_x(blip)
```

Versions 1.0.0+

Argument

> ***blip*** ~ a 'sound' asset, the sound you wish set as this option's cancel blip.

Description

You can use this function to set the sound you wish to play when the player backs out while selecting this option. If this is not set, it first tries to use the defaults for this menu set by `set_default_blips()`, and if that isn't found it will use the defaults set by the setting `__AriahMenuLibrary__DEFAULT_CANCEL_BLIP`.

You can also pass in **undefined** for ***blip***. If you do this, even if there are blips set as default there won't be a cancel blip for this option.

```
.set_blip_move(blip)
```

Versions 1.0.0+

Argument

> ***blip*** ~ a 'sound' asset, the sound you wish set as this option's move blip.

Description

You can use this function to set the sound you wish to play when the player selects a different option when this option is selected. If this is not set, it first tries to use the defaults for this menu set by `set_default_blips()`, and if that isn't found it will use the defaults set by the setting `__AriahMenuLibrary__DEFAULT_MOVE_BLIP`.

You can also pass in **undefined** for ***blip***. If you do this, even if there are blips set as default there won't be a move blip for this option.

```
.set_font(font)
```

Versions 1.0.0+

Argument

> ***font*** ~ a 'font' asset, the font you wish to use for this option.

Description

You can use this function to set the font you wish this option to use. If this is not set, it first tries to use the defaults for this menu set by `set_default_font()`, and if that isn't found it will use the defaults set by the setting `__AriahMenuLibrary__DEFAULT_FONT`.

```
.set_color(color1, [color2])
```

Versions 1.0.0+

Argument

> ***color1*** ~ a hex code, the first color you wish the option to have when it is *unselected*.
>
> ***color2*** *(optional)* ~ a hex code, the second color you wish to have when it is *unselected*.

Description

You can use this function to set the color of the option to have when it's ***not selected***. If only one color is given, the option will be the first color solidly while unselected. If both colors are given, the option will have a vertical gradient with the first color on top and the second color on bottom while unselected. If this is not set, it first tries to use the defaults for this menu set by `set_default_colors()`, and if that isn't found it will use the defaults set by the setting `__AriahMenuLibrary__DEFAULT_COLORS`.

```
.set_color_selected(color1, [color2])
```

Versions 1.0.0+

Argument

> ***color1*** ~ a hex code, the first color you wish the option to have when it is *selected*.
>
> ***color2*** *(optional)* ~ a hex code, the second color you wish to have when it is *selected*.

Description

You can use this function to set the color of the option to have when it ***is selected***. If only one color is given, the option will be the first color solidly while selected. If both colors are given, the option will have a vertical gradient with the first color on top and the second color on bottom while selected. If this is not set, it first tries to use the defaults for this menu set by `set_default_colors()`, and if that isn't found it will use the defaults set by the setting `__AriahMenuLibrary__DEFAULT_COLORS`.

```
.set_selectable(selectable)
```

Versions 1.0.0+

Argument

> ***selectable*** ~ a boolean, whether or not you wish this option to be selectable.

Description

You can use this function to set whether you wish the player to be able to select this option. If the option is not selectable, the player will skip past the option when trying to navigate.

> Note that if the first option on a page is not selectable, it will still be selected by default. To avoid this, include the function
>
> ```
> .set_function_init(function() {
>   select_option("Option2", "Page");
> })
> ```
>
> where **"Option2"** is the name of the *second* (or the option you want selected by default) option on the page, and **"Page"** is the name of the page you are adding this option to.

Navigation

```
.set_up_enabled(enabled)
```

Versions 1.0.0+

Argument

> ***enabled*** ~ a boolean, whether or not the player can navigate *up* from this option.

Description

You can use this function to set whether or not the player should be able to navigate *upwards* from this option.

```
.set_down_enabled(enabled)
```

Versions 1.0.0+

Argument

> ***enabled*** ~ a boolean, whether or not the player can navigate *down* from this option.

Description

You can use this function to set whether or not the player should be able to navigate *downwards* from this option.

```
.set_left_enabled(enabled)
```

Versions 1.0.0+

Argument

> ***enabled*** ~ a boolean, whether or not the player can navigate *left* from this option.

Description

You can use this function to set whether or not the player should be able to navigate *leftwards* from this option.

```
.set_right_enabled(enabled)
```

Versions 1.0.0+

Argument

> ***enabled*** ~ a boolean, whether or not the player can navigate *right* from this option.

Description

You can use this function to set whether or not the player should be able to navigate *rightwards* from this option.

Functions

```
.set_function_init(function() {
  // Code
})
```

Versions 1.0.0+

Description

The code within this function runs ***once*** when the menu is *first created*. These functions are called in the order each option is added.

```
.set_function_enter(function() {
  // Code
})
```

Versions 1.0.0+

Description

The code within this function runs ***each time*** this option *becomes selected*.

```
.set_function_selected(function() {
  // Code
})
```

Versions 1.0.0+

Description

The code within this function runs ***every step*** while the option *is selected*.

```
.set_function_leave(function() {
  // Code
})
```

Versions 1.0.0+

Description

The code within this function runs ***each time*** this option *stops being selected*.

```
.set_function_z(function() {
  // Code
})
```

Versions 1.0.0+

Description

The code within this function runs ***each time*** this option is *chosen* (the player is selecting this option and pressed the *confirm* button).

```
.set_function_x(function() {
  // Code
})
```

Versions 1.0.0+

Description

The code within this function runs ***each time*** the player attempts to *back out* while this option is selected (the player is selecting this option and pressed the *cancel* button).

```
.set_function_up(function() {
  // Code
})
```

Versions 1.0.0+

Description

The code within this function runs ***each time*** the player attempts to navigate upwards from this option.

> Note that including this function disables the default code allowing the player to navigate upwards. When using this function make sure you don't make the menu impossible to navigate.

```
.set_function_down(function() {
  // Code
})
```

Versions 1.0.0+

Description

The code within this function runs ***each time*** the player attempts to navigate downwards from this option.

> Note that including this function disables the default code allowing the player to navigate downwards. When using this function make sure you don't make the menu impossible to navigate.

```
.set_function_left(function() {
  // Code
})
```

Versions 1.0.0+

Description

The code within this function runs ***each time*** the player attempts to navigate leftwards from this option.

> Note that including this function disables the default code allowing the player to navigate leftwards. When using this function make sure you don't make the menu impossible to navigate.

```
.set_function_right(function() {
  // Code
})
```

Versions 1.0.0+

Description

The code within this function runs ***each time*** the player attempts to navigate rightwards from this option.

> Note that including this function disables the default code allowing the player to navigate rightwards. When using this function make sure you don't make the menu impossible to navigate.

Examples

Here are some examples using this menu library. Feel free to look through them, copy them, or just create your own from scratch.

Simple Title Screen Menu

This code should be perfectly compatible with an empty project. Use this if you just want a simple menu for your game, or if you want to create your own menu from a small template.

```
var _menu = instance_create_depth(0, 0, 0, ui_menu);
with(_menu) {
  add_page(new MenuPageRows("Main")
      .add_option(
          new MenuOption("New Game")
              .set_function_z(function() {
                if(file_exists("game.sav")) {
                  file_delete("game.sav");
                }
                room_goto_next();
              })
              .set_blip_x(undefined)
      )
      .add_option(
          new MenuOption("Continue")
              .set_function_init(function() {
                if(!file_exists("game.sav")) {
                  get_page("Main").get_option("Continue")
                      .set_color(#888888, #444444)
                      .set_selectable(false);
                }
              })
              .set_function_z(function() {
                game_load("game.sav");
              })
              .set_blip_x(undefined)
      )
      .add_option(
          new MenuOption("Quit")
              .set_function_z(function() {
                game_end();
              })
              .set_blip_x(undefined)
      )
  );
}
```

Complex Title Screen Menu

This is an example of the code that creates a title screen menu. This example was loosely pulled from episode 3 of my YouTube series, *Project Pixel*.

> Note that all of the functions and variables used below beginning with `game.`, the macros `GAME_WIDTH` and `GAME_HEIGHT`, as well as the 'sound' asset `sfx_menu_blip_success`, as are ***not*** part of this library and will not work for you unless you've created them yourself. Check out *Project Pixel* if you want to learn how. In addition, this code block references my **Audio Library** as well (functions beginning with `audio.`). You will need to install that as well for those functions to work for you.

```
var _menu = instance_create_depth(0, 0, 0, ui_menu);
with(_menu) {
  add_page(new MenuPageRows("Main")
      .add_option(
          new MenuOption("Play")
              .set_function_init(function() {
                var _option = get_page("Main").get_option("Play");
                var _started = game.flag.get("game_has_started");
                if(_started) _option.set_text("Continue");
                else _option.set_text("New Game");
              })
              .set_function_z(function() {
                game.flag.set("game_has_started", true);
                game.menu_locked = false;
                instance_create_depth(game.data.x, game.data.y, 0, player);
                room_goto(game.data.room);
              })
              .set_blip_x(undefined);
      )
      .add_option(
          new MenuOption("Settings")
              .set_function_z(function() {
                set_page("Settings");
              })
              .set_blip_x(undefined);
      )
      .add_option(
          new MenuOption("Quit")
              .set_function_selected(function() {
                var _x = window_get_x() + irandom_range(-2, 2);
                var _y = window_get_y() + irandom_range(-2, 2);
                window_set_position(_x, _y);
              })
              .set_function_z(function() {
                game_end();
              })
              .set_blip_z(undefined);
              .set_blip_x(undefined);
```

```
    )
);

add_page(new MenuPageRows("Settings")
    .add_option(
        new MenuOption("Game")
            .set_function_z(function() {
              set_page("Game");
            })
            .set_function_x(function() {
              set_page("Main");
            })
        ,
        new MenuOption("Video")
            .set_function_z(function() {
              set_page("Video");
            })
            .set_function_x(function() {
              set_page("Main");
            })
    )
    .add_option(
        new MenuOption("Controls")
            .set_function_z(function() {
              set_page("Controls");
            })
            .set_function_x(function() {
              set_page("Main");
            })
        ,
        new MenuOption("Audio")
            .set_function_z(function() {
              set_page("Audio");
            })
            .set_function_x(function() {
              set_page("Main");
            })
    )
    .add_option(
        new MenuOption("Back")
            .set_function_z(function() {
              set_page("Main");
            })
            .set_function_x(function() {
              set_page("Main");
            })
            .set_blip_z(blip_x_default)
    )
);

add_page(new MenuPageRows("Game")
    .add_option(
        new MenuOption("Reset Save Data")
```

```
            .set_color(#bb0000, #880000)
            .set_color_selected(#ff0000)
            .set_function_z(function() {
              game.reset_save_data();
              game.save();
              get_page("Main").get_option("Play").set_text("New Game");
              set_page("Reset");
            })
            .set_function_x(function() {
              set_page("Settings");
            })
            .set_blip_z(sfx_menu_blip_success)
    )
    .add_option(
        new MenuOption("Back")
            .set_function_z(function() {
              set_page("Settings");
            })
            .set_function_x(function() {
              set_page("Settings");
            })
            .set_blip_z(blip_x_default)
    )
);

add_page(new MenuPageRows("Reset")
    .add_option(
        new MenuOption("Save Data has been Reset!")
            .set_color_selected(#ffffff, #aaaaaa)
            .set_function_z(function() {
              set_page("Settings");
            })
            .set_function_x(function() {
              set_page("Settings");
            })
            .set_blip_z(undefined)
            .set_blip_x(undefined)
    )
);

add_page(new MenuPageRows("Video")
    .add_option(
        new MenuOption("Scale")
            .set_function_init(function() {
              var _text = "";
              if(game.settings.fullscreen) {
                var _max = min(display_get_width() / GAME_WIDTH,
                               display_get_height() / GAME_HEIGHT);
                _text = string(_max * GAME_WIDTH) + " x " +
                        string(_max * GAME_HEIGHT);
                get_page("Video").get_option("Scale")
                    .set_color(#888888, #444444)
                    .set_selectable(false);
```

```
      select_option("Fullscreen", "Video");
    } else {
      var _max = floor(min(display_get_width() / GAME_WIDTH,
                           display_get_height() / GAME_HEIGHT));
      _text = game.settings.scale > 1 ? "< " : "  ";
      _text += string(game.settings.scale * GAME_WIDTH) + " x " +
               string(game.settings.scale * GAME_HEIGHT);
      _text += game.settings.scale < _max ? " >" : "  ";
    }
    get_page("Video").get_option("Scale").set_text(_text);
  })
  .set_function_left(function() {
    if(game.settings.scale == 1) return;

    audio.sfx_play(get_current_option().blip_move);
    game.settings.scale--;
    if(game.settings.scale < 1) game.settings.scale = 1;
    window_set_size(GAME_WIDTH * game.settings.scale,
                    GAME_HEIGHT * game.settings.scale);
    window_set_position(
        (display_get_width() - GAME_WIDTH*game.settings.scale)/2,
        (display_get_height() - GAME_HEIGHT*game.settings.scale)/2);
    var _text = game.settings.scale > 1 ? "< " : "  ";
    _text += string(game.settings.scale * GAME_WIDTH) + " x " +
             string(game.settings.scale * GAME_HEIGHT);
    _text += " >";
    get_current_option().set_text(_text);
    game.settings_save();
  })
  .set_function_right(function() {
    var _max = floor(min(display_get_width() / GAME_WIDTH,
                         display_get_height() / GAME_HEIGHT));
    if(game.settings.scale == _max) return;

    audio.sfx_play(get_current_option().blip_move);
    game.settings.scale++;
    if(game.settings.scale > _max) game.settings.scale = _max;
    window_set_size(GAME_WIDTH * game.settings.scale,
                    GAME_HEIGHT * game.settings.scale);
    window_set_position(
        (display_get_width() - GAME_WIDTH*game.settings.scale)/2,
        (display_get_height() - GAME_HEIGHT*game.settings.scale)/2);
    var _text = "< ";
    _text += string(game.settings.scale * GAME_WIDTH) + " x " +
             string(game.settings.scale * GAME_HEIGHT);
    _text += game.settings.scale < _max ? " >" : "  ";
    get_current_option().set_text(_text);
    game.settings_save();
  })
  .set_function_x(function() {
    set_page("Settings");
  })
  .set_blip_z(undefined)
```

```
    )
    .add_option(
        new MenuOption("Fullscreen")
            .set_function_init(function() {
              var _text = game.settings.fullscreen ?
                          "< Fullscreen  " : "  Windowed >";
              get_page("Video").get_option("Fullscreen").set_text(_text);
            })
            .set_function_left(function() {
              if(!game.settings.fullscreen) return;

              audio.sfx_play(get_current_option().blip_move);
              var _max = floor(min(display_get_width() / GAME_WIDTH,
                                   display_get_height() / GAME_HEIGHT));
              var _text = game.settings.scale > 1 ? "< " : "  ";
              _text += string(game.settings.scale * GAME_WIDTH) + " x " +
                       string(game.settings.scale * GAME_HEIGHT);
              _text += game.settings.scale < _max ? " >" : "  ";
              get_current_page().get_option("Scale")
                  .set_color(#ffffff, #aaaaaa)
                  .set_selectable(true)
                  .set_text(_text)

              game.settings.fullscreen = false;
              window_set_fullscreen(false);
              window_set_size(GAME_WIDTH * game.settings.scale,
                              GAME_HEIGHT * game.settings.scale);
              window_set_position(
                  (display_get_width() - GAME_WIDTH*game.settings.scale)/2,
                  (display_get_height() - GAME_HEIGHT*game.settings.scale)/2);
              get_current_option().set_text("  Windowed >");
              game.settings_save();
            })
            .set_function_right(function() {
              if(game.settings.fullscreen) return;

              audio.sfx_play(get_current_option().blip_move);
              var _max = min(display_get_width() / GAME_WIDTH,
                             display_get_height() / GAME_HEIGHT);
              var _text = string(_max * GAME_WIDTH) + " x " +
                          string(_max * GAME_HEIGHT);
              get_current_page().get_option("Scale")
                  .set_color(#888888, #444444)
                  .set_selectable(false)
                  .set_text(_text)

              game.settings.fullscreen = true;
              window_set_fullscreen(true);
              get_current_option().set_text("< Fullscreen  ");
              game.settings_save();
            })
            .set_function_x(function() {
              set_page("Settings");
```

```
            })
            .set_blip_z(undefined)
    )
    .add_option(
        new MenuOption("Back")
            .set_function_z(function() {
              set_page("Settings");
            })
            .set_function_x(function() {
              set_page("Settings");
            })
            .set_blip_z(blip_x_default)
    )
);

// This page is incomplete, as it was both much more
// complicated and not included in episode 3.
add_page(new MenuPageRows("Controls")
    .add_option(
        new MenuOption("Keyboard")
            .set_function_x(function() {
              set_page("Settings");
            })
    )
    .add_option(
        new MenuOption("Gamepad")
            .set_function_x(function() {
              set_page("Settings");
            })
    )
    .add_option(
        new MenuOption("Back")
          .set_function_z(function() {
            set_page("Settings");
          })
          .set_function_x(function() {
            set_page("Settings");
          })
          .set_blip_z(blip_x_default)
    )
);

add_page(new MenuPageRows("Audio")
    .add_option(
        new MenuOption("BGM_Label", "Music")
            .set_selectable(false)
            .set_color(#1CC9E1, #18A9BC)
            .set_function_init(function() {
              select_option("BGM", "Audio")
            })
    )
    .add_option(
        new MenuOption("BGM")
```

```
            .set_function_init(function() {
              var _text = game.settings.vol_bgm > 0 ? "< " : "   ";
              _text += string(game.settings.vol_bgm);
              _text += game.settings.vol_bgm < 100 ? " >" : "   ";
              get_page("Audio").get_option("BGM").set_text(_text);
            })
            .set_function_left(function() {
              if(game.settings.vol_bgm == 0) return;

              audio.sfx_play(get_current_option().blip_move);
              game.settings.vol_bgm -= 10;
              if(game.settings.vol_bgm < 0) game.settings.vol_bgm = 0;
              var _text = game.settings.vol_bgm > 0 ? "< " : "   ";
              _text += string(game.settings.vol_bgm);
              _text += " >";
              get_current_option().set_text(_text);
              game.settings_save();
            })
            .set_function_right(function() {
              if(game.settings.vol_bgm == 100) return;

              audio.sfx_play(get_current_option().blip_move);
              game.settings.vol_bgm += 10;
              if(game.settings.vol_bgm > 100) game.settings.vol_bgm = 100;
              var _text = "< ";
              _text += string(game.settings.vol_bgm);
              _text += game.settings.vol_bgm < 100 ? " >" : "   ";
              get_current_option().set_text(_text);
              game.settings_save();
            })
            .set_function_x(function() {
              set_page("Settings");
            })
    )
    .add_option(
        new MenuOption("SFX_Label", "Sound Effects")
            .set_selectable(false)
            .set_color(#1CC9E1, #18A9BC)
            .set_function_init(function() {
              select_option("SFX", "Audio")
            })
    )
    .add_option(
        new MenuOption("SFX")
            .set_function_init(function() {
              var _text = game.settings.vol_sfx > 0 ? "< " : "   ";
              _text += string(game.settings.vol_sfx);
              _text += game.settings.vol_sfx < 100 ? " >" : "   ";
              get_page("Audio").get_option("SFX").set_text(_text);
            })
            .set_function_left(function() {
              if(game.settings.vol_sfx == 0) return;
```

```
            audio.sfx_play(get_current_option().blip_move);
            game.settings.vol_sfx -= 10;
            if(game.settings.vol_sfx < 0) game.settings.vol_sfx = 0;
            var _text = game.settings.vol_sfx > 0 ? "< " : "  ";
            _text += string(game.settings.vol_sfx);
            _text += " >";
            get_current_option().set_text(_text);
            game.settings_save();
          })
          .set_function_right(function() {
            if(game.settings.vol_sfx == 100) return;

            audio.sfx_play(get_current_option().blip_move);
            game.settings.vol_sfx += 10;
            if(game.settings.vol_sfx > 100) game.settings.vol_sfx = 100;
            var _text = "< ";
            _text += string(game.settings.vol_sfx);
            _text += game.settings.vol_sfx < 100 ? " >" : "  ";
            get_current_option().set_text(_text);
            game.settings_save();
          })
          .set_function_x(function() {
            set_page("Settings");
          })
    )
    .add_option(
        new MenuOption("BGS_Label", "Background Sounds")
            .set_selectable(false)
            .set_color(#1CC9E1, #18A9BC)
            .set_function_init(function() {
              select_option("BGS", "Audio")
            })
    )
    .add_option(
        new MenuOption("BGS")
            .set_function_init(function() {
              var _text = game.settings.vol_bgs > 0 ? "< " : "  ";
              _text += string(game.settings.vol_bgs);
              _text += game.settings.vol_bgs < 100 ? " >" : "  ";
              get_page("Audio").get_option("BGS").set_text(_text);
            })
            .set_function_left(function() {
              if(game.settings.vol_bgs == 0) return;

              audio.sfx_play(get_current_option().blip_move);
              game.settings.vol_bgs -= 10;
              if(game.settings.vol_bgs < 0) game.settings.vol_bgs = 0;
              var _text = game.settings.vol_bgs > 0 ? "< " : "  ";
              _text += string(game.settings.vol_bgs);
              _text += " >";
              get_current_option().set_text(_text);
              game.settings_save();
            })
```

```
            .set_function_right(function() {
              if(game.settings.vol_bgs == 100) return;

              audio.sfx_play(get_current_option().blip_move);
              game.settings.vol_bgs += 10;
              if(game.settings.vol_bgs > 100) game.settings.vol_bgs = 100;
              var _text = "< ";
              _text += string(game.settings.vol_bgs);
              _text += game.settings.vol_bgs < 100 ? " >" : "  ";
              get_current_option().set_text(_text);
              game.settings_save();
            })
            .set_function_x(function() {
              set_page("Settings");
            })
      )
      .add_option(
          new MenuOption("Back")
            .set_function_z(function() {
              set_page("Settings");
            })
            .set_function_x(function() {
              set_page("Settings");
            })
            .set_blip_z(blip_x_default)
      )
  );
}
```

1. *This works normally as of version 1.0.1*

> *Version 1.0.0*:
>
> **undefined** should be returned here, but isn't for pages of type `MenuPageRows`. Be weary when using this function as (for `MenuPageRows`) nothing is returned if there is no matching option.

↩