Layout Manager

V 0.1 – Phase 1

Contents

[1.0 Component/Module Overview 2](#_Toc89415593)

[2.0 How to Use 2](#_Toc89415594)

[3.0 Allowable Options 2](#_Toc89415595)

[3.1 Grid Layout 2](#_Toc89415596)

[3.2 Fill Layout (Unfinished) 3](#_Toc89415597)

[3.3 Best Fit (Unfinished) 3](#_Toc89415598)

[3.4 Weighted Sizes (Unfinished) 3](#_Toc89415599)

[3.5 Circle Layout (Unfinished) 3](#_Toc89415600)

[3.6 User Defined (Unfinished) 3](#_Toc89415601)

[4.0 Setting Defaults 3](#_Toc89415602)

[Appendix A 3](#_Toc89415603)

[A.1 Grid Layout API 3](#_Toc89415604)

# 1.0 Component/Module Overview

This module allows the user to set up a series of parameters to get a layout that the user desires for the application. This will only give a table of positions back to the user. The positions can then be used for many things like cloning controls, judging touchzones, and more.

# 2.0 How to Use

To set up and trigger a creation of a layout the module must be brought in full to the project. The main file (layout\_manager/layout\_manager.lua) must be required. From there the user can many options to get the layout that will be the best for the situation.

# 3.0 Allowable Options

Each of the below options for a layout will have to be called through different functions. However, with mixing and matching some of the layouts and parameters the entire look can change.

## 3.1 Grid Layout

**[setup\_grid\_layout()]**

The grid layout is the basic layout that allows for a user to have a table of controls with a header and a footer. The below are the parameters that can be included with a grid layout.

View Appendix A.1 for setting up parameters and API

**max\_col:** Allows the user to set a maximum number of columns for the layout

-Must be a number

-Overwritten if there is a dynamic resize option set up

**max\_row:** Allows the user to set a maximum number of rows for the layout

-Must be a number

-Overwritten if there is a dynamic resize option set up

**alignment:** Adjusts the layout based on the layer given below in the parameters. If there is no layer, then it will default to the screen.

-4 options (left, center, justify, top)

**orientation:** Adjusts the orientation of the layer to be setup to scroll horizontal or Vertical

-2 options (vertical, horizontal)

-Be aware to setup the scrolling of the layer that is set with this layout to the same orientation.

**control\_size:** Adjusts the size of the control

-Can be a number for a square

-Table of width/height for a rectangle

-Will be the smallest size for the dynamic\_resize

**padding:** Adjusts the padding of the layout. This gives space between each of the positions given back

-Can be a number for equal padding everywhere

-Other options are vertical, horizontal, left, right, top, bot.

-If using the other options, they must be within a table, and cannot use the basic number for equal padding anymore.

**layer:** The name of the layer that allows for the alignment and dynamic resize to work properly.

-If not given, defaults to screen size and offsets. Can give odd results.

**item\_amount:** The amount of items to layout, a number. Will have a default.

**dynamic\_resize:** allows the cells to resize to best fit the screen using a certain multiplier and as many steps as given

**header:** A table of parameters on how the user wants the header to look and be sized

-nil gives no header.

-returned separately then the rest of the positions

**footer:** same as header, but below the table

## 3.2 Fill Layout (Unfinished)

## 3.3 Best Fit (Unfinished)

## 3.4 Weighted Sizes (Unfinished)

## 3.5 Circle Layout (Unfinished)

## 3.6 User Defined (Unfinished)

# 4.0 Setting Defaults (Unfinished)

# Appendix A

## A.1 Grid Layout API

Multiple types show discrete different parameter entries that can be allowed in a specific parameter

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Parameter | Type | Options | | Examples/Notes | |
| **max\_col** | *Number* | nil | |  | |
|  |  | Any number | | 15 | |
|  | Dynamic resolution will overwrite the max column. For infinite max columns use nil. (useful for horizontal orientation) | | | | |
| **max\_row** | *Number* | Nil | |  | |
|  |  | Any number | | 15 | |
|  | Same as above. | | | | |
| **Alignment** | *string* | Left | |  | |
|  |  | Center | |  | |
|  |  | Justify | | Will align the cells to best fit the space. Uses screen if there is no layer | |
|  |  | Top | |  | |
|  |  | Nil | |  | |
|  | To not have to worry about string literal typing errors, use the options given by layout manager options. (layout\_manager\_options.lua). [options.alignment.left] | | | | |
| **Orientation** | String | Vertical | | Makes the table scroll up/down | |
|  |  | Horizontal | | Makes the table scroll left/right | |
|  | Same as above for no string literal typing errors. Use the correct max row/ max column for the correct orientation. | | | | |
| **Control\_size** | number | | Any Number | | Makes a square control |
|  | table | | {vertical = x, horizontal = x} | | Makes a rectangular control |
| **Padding** | number | | Any number | |  |
|  | table | | {vertical = x, horizontal = x} | | Equal padding on the left/right and top/bottom |
|  |  | | {left = x, right = x, top = x, bot = x} | | Unique Padding for every side |
| **Layer** | string | | layer\_name\_here | | Allows the layout to be sized properly to the layer it is on. |
| **Item\_amount** | number | | Any Number | |  |
| **Dynamic\_resize** | table | | {steps = x, multiplier = y.z} | | Resizes items upwards with a certain amount of steps (integer) and a multiplier of the base size before it (float) |
| **Header** | table | | {alignment = x, width = y, col\_width = y(1), height = z} | | Alignment can be left, first column, or right. First column aligns the header with the first entry into the layout.  Width can be a number or a string option. String options include col\_width or fill. col\_width will size the header to that of a number of columns specified by col\_width option if width = col\_width.  As above, col\_width is the amount of colums to size the header to IF the header width is set to col\_width  Height can be a number or base\_size which will match it up. |
| **footer** | table | | Same as above | | Same as above |