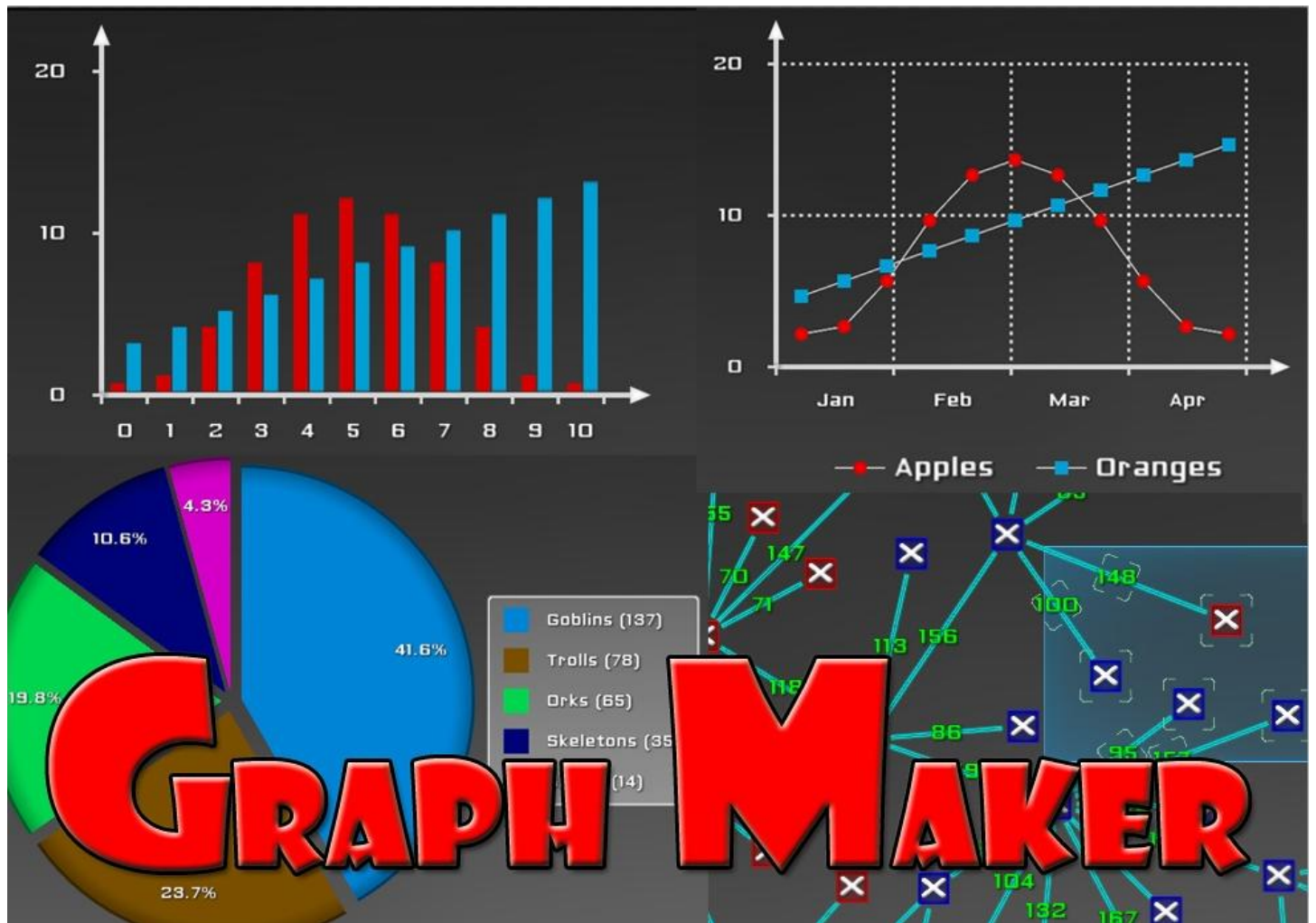


# Graph Maker



<b>I. OVERVIEW</b>	<b>3</b>
1.1 GOALS	3
1.2 FEATURES	3
<b>II. PIE GRAPHS</b>	<b>4</b>
2.1 GETTING STARTED	4
2.2 PARAMETERS	4
<b>III. LINE GRAPHS</b>	<b>10</b>
3.1 GETTING STARTED	10
3.2 AXIS GRAPH PARAMETERS	10
3.3 SERIES PARAMETERS	15
3.4 OTHER NOTES	18
3.5 ANIMATIONS	18
3.6 EVENTS	19
3.7 DYNAMICALLY ADD / DELETE SERIES	19
<b>IV. BAR GRAPHS</b>	<b>20</b>
4.1 GETTING STARTED	20
4.2 AXIS / SERIES GRAPH PARAMETERS	20
<b>V. SQUARE / RECT / HEX GRIDS</b>	<b>21</b>
5.1 GETTING STARTED	21
5.2 PARAMETERS	22
<b>VI. RANDOM GRAPHS</b>	<b>24</b>
6.1 GETTING STARTED	24
6.2 PARAMETERS	24
<b>VII. GRAPH EDITOR</b>	<b>27</b>
7.1 GETTING STARTED	27
7.2 PARAMETERS	28
7.3 EDITOR CONTROLS	30
7.4 SELECTIONS	30
7.5 SAVING AND LOADING	31
7.6 PREFAB SWAPPING	32
7.7 DELETING	33
7.8 ADDING GRAPHS	34

# I. Overview

## 1.1 Goals

---

The primary goal of this package is to make adding quality graph GUIs such as pie graphs, line graphs, and bar graphs to your project very easy. The secondary goal is to allow a way to create graph based GUIs that don't necessarily conform to a specific typical graph. The common use cases of these types of graphs include skill trees and world map GUIs that you see commonly in space games such as Escape Velocity, Space Pirates and Zombies, and Faster than Light.

## 1.2 Features

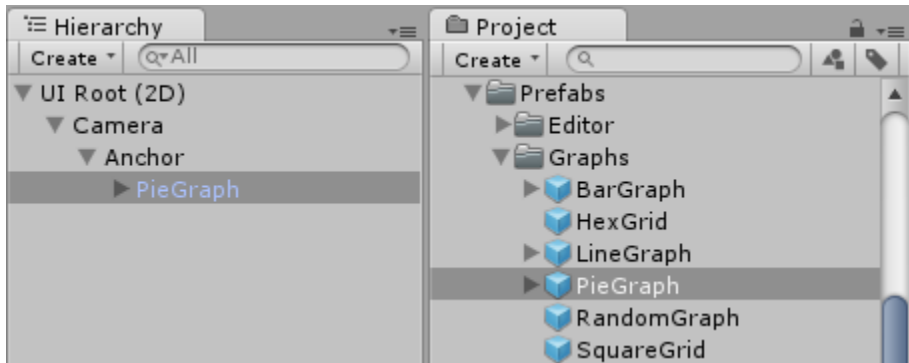
---

- ❖ Pie Graphs
- ❖ Line Graphs
- ❖ Bar Graphs
- ❖ Quadrilateral Grids
- ❖ Hexagonal Grids
- ❖ Data changes update graphs in real-time
- ❖ Customize many other visual aspects in real-time
- ❖ Random Graphs
- ❖ Custom Editor that can zoom and pan
- ❖ Drag select, move, and delete
- ❖ Swap custom prefabs in real-time
- ❖ Save and load your custom graph GUIs
- ❖ World map example
- ❖ Skill tree example hand crafted from the editor

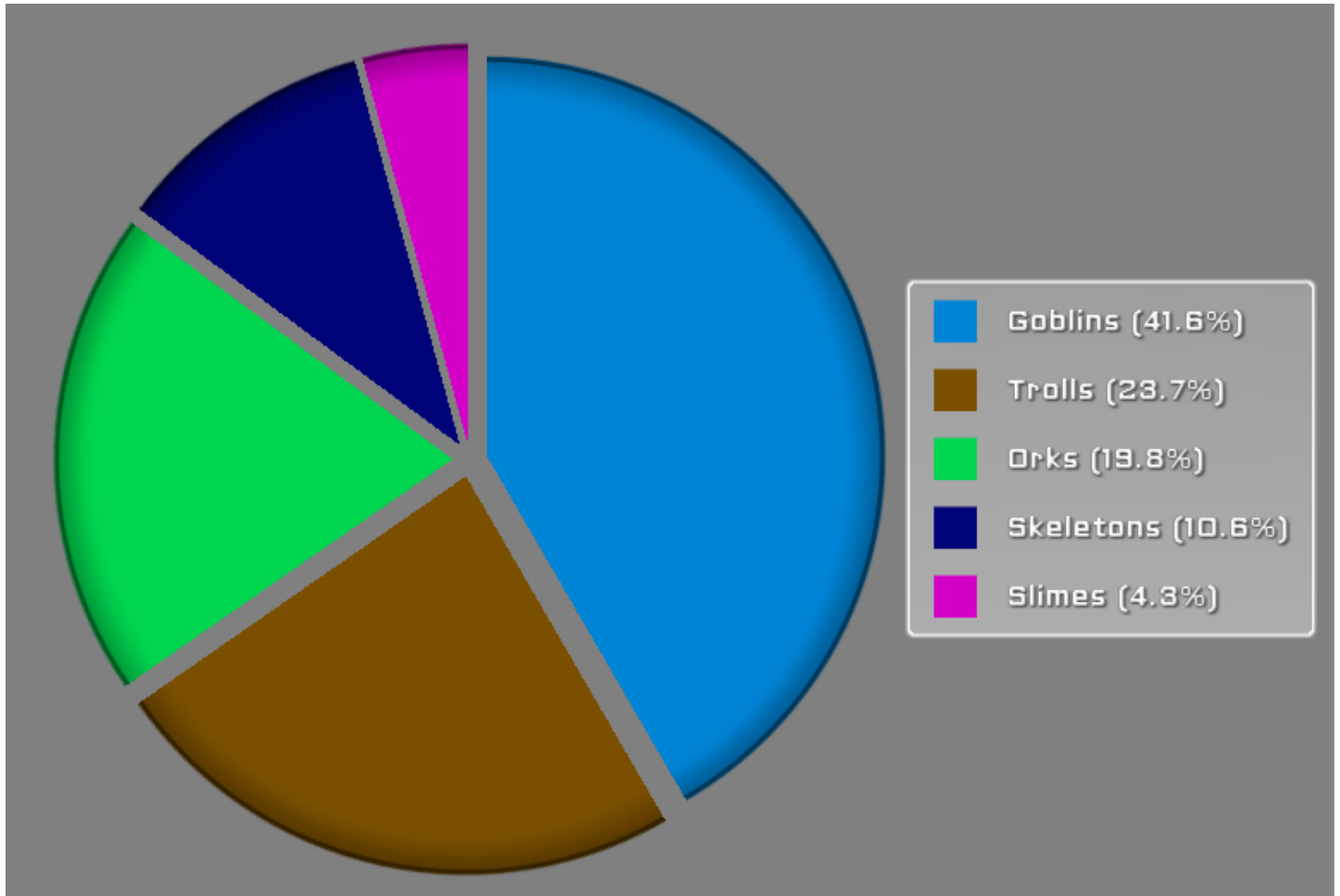
## II. Pie Graphs

### 2.1 Getting Started

Drag and drop the PieGraph prefab from the Prefabs/Graphs folder into your scene:



The graph will appear when you play the scene:



### 2.2 Parameters

Pie graphs can be customized based on these parameters:



#### - Update Graph Every Frame

By default, the graph updates every frame, meaning changing parameters will constantly update the graph. This can be unchecked if you want to control the graph updating yourself, or if you know for sure the graph will not change. In this case you will need to get a reference to the script from your code and call `RefreshGraph()`.

#### - Update Graph / Animation Duration / Sort Animation Duration

These parameters are for doing animated updates instead of constant instant updates. Setting update graph will perform one animated update. The animation for deleting pie slices for changing pie slice data is to expand / contract the affected slices. The sorting animation only applies if the sort by parameter is set to something other than none, and the data was changed such that sorting would rearrange the order of pie slices. The sorting animation shrinks and then expands all of the slices.

#### - Slice Values

This is a list of floats for the actual data of the pie graph. The number of slices is affected by this parameter, so if you entered 6 slice labels, but have only 5 values, then only 5 slices will appear and the 6th label is ignored.

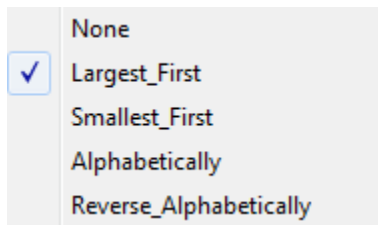
#### - Slice Labels

This is the list of strings to label the pie slices. This can appear in the legend, or in text overlaying the pie slices. If the number of slice values is increased beyond what is specified then extra labels are automatically labeled but default to the empty string. If the number of slice values is decreased beyond what is specified, then the extra labels are not automatically deleted.

#### - Slice Colors

This is the list of colors of the pie slices. This appears in the legend, and obviously the pie slice itself. If the number of slice values is increased beyond what is specified then extra colors are automatically added but default to the white color. If the number of slice values is decreased beyond what is specified, then the extra colors are not automatically deleted. No auto deletion is by design, so that you only need to create 1 large default color set for your pie graph that will not be affected by the number of slices that can appear.

#### - Sort by



This controls the order that the pie slices appear. The default is Largest\_First, meaning that the slice with the highest value will appear in the top right, and subsequent slices go in clockwise order. The other sorting options should be intuitive based on the name. For the "None" sorting, the slice value corresponding to the 0 index appears in the top right corner.

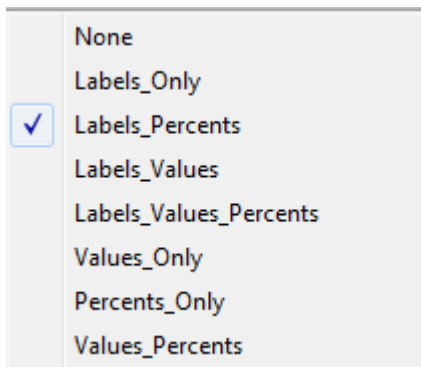
#### - Swap Colors During Sort

By default colors do get swapped, meaning if the 3rd slice is green and gets increased so that it becomes the first slice, the green color stays with that slice so now green will be in the top right slice. If this is set to false, then the colors remain static, meaning green will always be the third slice.

#### - Explode Length

This is the radial distance the pie slices are from the center. A small value here usually enhances the visual appeal of the graph.

#### - Legend Type

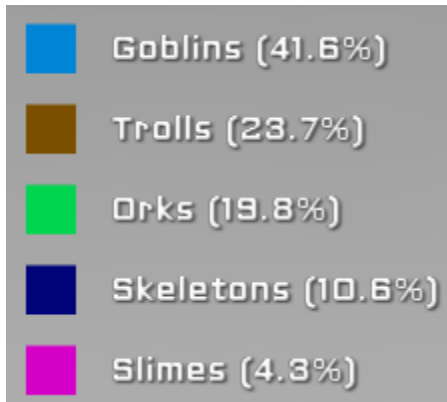


None means no legend will appear. The rest of the options configure the text that is displayed next to the color swatch.

Labels Only:



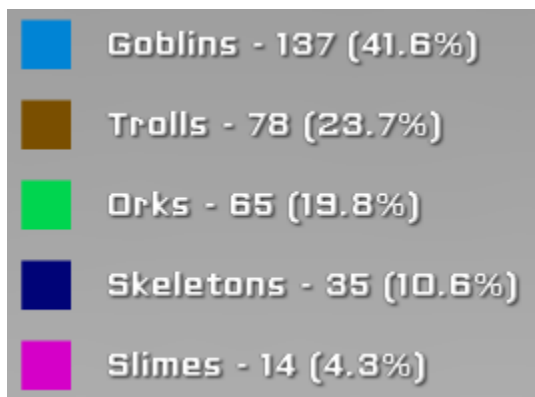
Labels Percents:



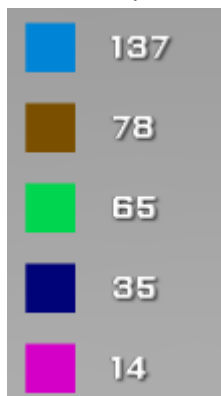
Labels Values



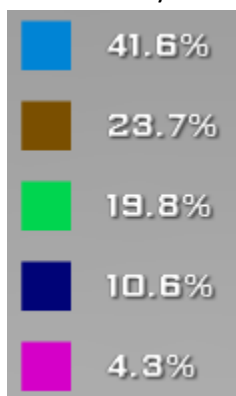
Labels Values Percents



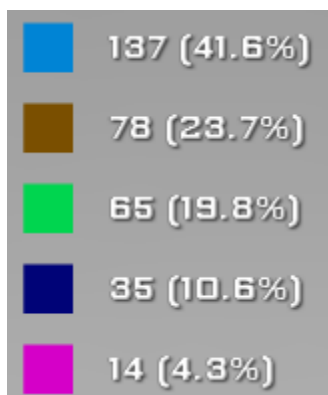
Values Only



Percents Only



Values Percents



- Legend Location / Width / Row Height

These control where the legend appears, how wide the background is, and how much space is between each legend entry.



- Slice Label Type

This is identical to legend type except it applies to the labels that overlay the pie slices themselves.

- Slice Label Explode Length

This controls where the labels that overlay pie slices appear. By default it is 0 so they will appear in the center of the pie slice.

- Number of decimals in percents

This controls the number of decimals displayed in the legend entries and pie slice labels when a percent is displayed.

- Limit number slices / max slices

When limit number slices is checked, the pie graph will limit the number of slices displayed to the maximum number set. So if there are 10 data values, and 3 is set then only 3 of those will be used, the 3 used depends on sorting.

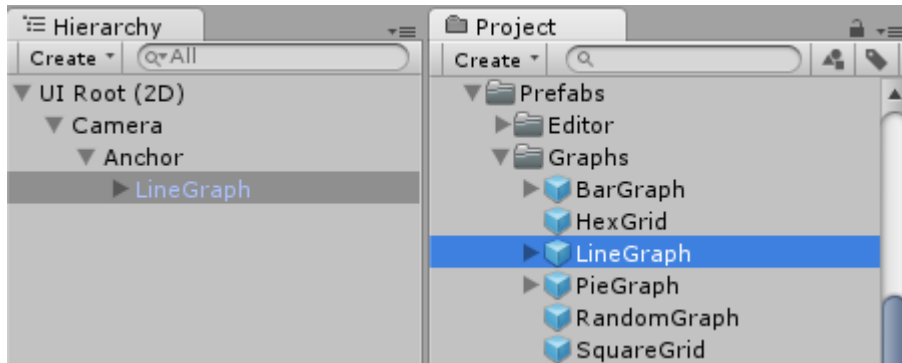
- Include Others / Others Label / Others Color

When include others is checked, the slices that got excluded from limiting the number of slices will be lumped into a single "Others" slice. So if you have 100 monsters in your game, and the player wants to see the top 10 monsters they've slain you could show the top 10 and include others, which could be labeled "Other Monsters" and will include the sum of the other 90 monster data. This also needs to be given its own color defined by the Others Color.

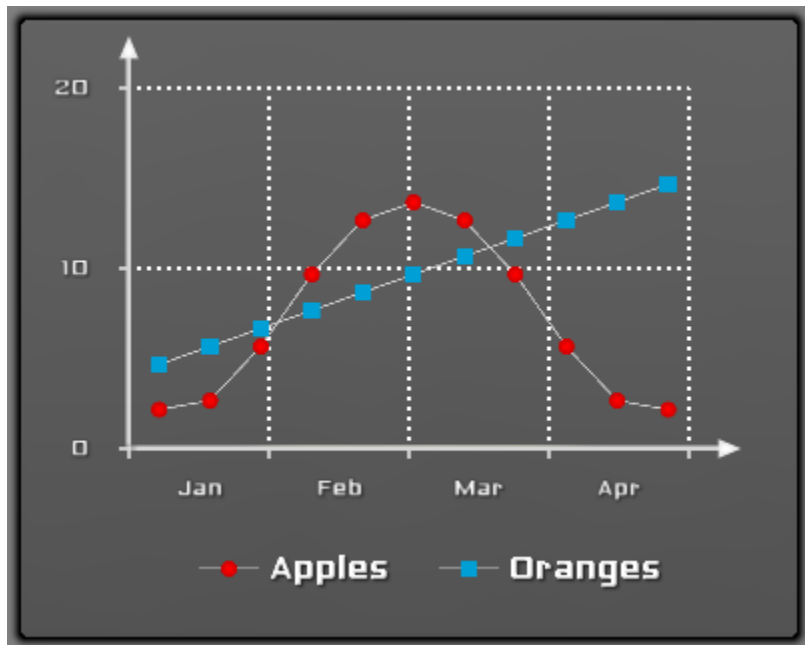
## III. Line Graphs

### 3.1 Getting Started

Drag and drop the LineGraph prefab from the Prefabs/Graphs folder into your scene:



The graph will appear when you play the scene:



### 3.2 Axis Graph Parameters

Line graphs can be customized based on the axis graph parameters and the parameters associated with each series assigned to the graph. Parameters for the axis graph are explained here:

**WMG\_Axis\_Graph (Script)**

Script WMG\_Axis\_Graph

Graph Type line

Orientation Type vertical

Axes Type MANUAL

Line Series

Point Prefabs

Link Prefabs

Bar Prefab Bar

Series Prefab Series

YAxis Max Value 100

YAxis Min Value -100

YAxis Num Ticks 9

YAxis Length 430

XAxis Max Value 10

XAxis Min Value -10

XAxis Num Ticks 9

XAxis Length 865.5

YMin Max Auto Grow

YMin Max Auto Shrink

XMin Max Auto Grow

XMin Max Auto Shrink

Auto Shrink At Percent 0.6

Auto Grow And Shrink By Percent 0.2

Bar Width 25

Axis Width 4

Hide XTicks ☐

Hide YTicks ☐

Hide XLabels ☐

Hide YLabels ☐

YAxis Labels

XAxis Labels

Set YLabels Using Max Min ☒

Num Decimals YAxis Labels 1

Set XLabels Using Max Min ☒

Num Decimals XAxis Labels 1

Auto Center YAxis Labels ☐

Auto Center XAxis Labels ☐

YAxis Label Spacing Y 0

YAxis Label Spacing X 10

XAxis Label Spacing Y 20

XAxis Label Spacing X 0

YAxis Label Size 0.75

XAxis Label Size 0.75

YAxis Ticks Right ☐

XAxis Ticks Above ☐

XAxis Arrows

XAxis Line Padding 20

XAxis YTick 0

Hide YTick ☐

YAxis Arrows

YAxis Line Padding 20

YAxis XTick 0

Hide XTick ☐

Legend Type	Horizontal
Legend Position	Bottom
Hide Legend Labels	<input checked="" type="checkbox"/>
Legend Entry Spacing X	75
Legend Entry Spacing Y	25
Legend Parent Offset X	-37.5
Legend Parent Offset Y	50
Background Padding Left	65
Background Padding Right	60
Background Padding Top	180
Background Padding Bottom	75

### -Graph Type

Changes the graph from the default line graph to various different types of bar graphs:

<input checked="" type="checkbox"/>	line
<input type="checkbox"/>	bar_side
<input type="checkbox"/>	bar_stacked
<input type="checkbox"/>	bar_stacked_percent

### -Orientation Type

Change the graph from vertical to horizontal. This can be useful for horizontal bar charts.

### -Axes Type

Change the position of the axes based on a quadrant system. Can also set it to automatically update its position to be closest to the origin.

### - Line Series

This is the list of gameobjects with a series script attached. If nothing is specified here, then only the graph axes and grids will display.

### - Point prefabs

This is the list of point prefabs with which the series' point prefab index corresponds. A series with a point prefab of 0 means use element 0 in this list.

### - Link prefabs

This is the list of link prefabs with which the series' link prefab index corresponds. A series with a link prefab of 0 means use element 0 in this list.

### - Bar Prefab

For bar graphs, this is the prefab used in drawing the bars for all series.

### - Series Prefab

Dynamically adding series with the `addSeries()` function will use this prefab to create the new series.

### - X/Y Axis Max and Min Values

This determines where each point / bar in a series gets positioned.

#### - X/Y Axis Num Ticks

This determines the number of ticks that appear on each axis. Each tick can also be associated with a tick label and the gridlines are also aligned with the ticks. The minimum value is forced to be 1 to get around divide by 0 errors, so if 0 ticks are required, then simply navigate the hierarchy of the graph and disable the ticks. A parameter may be added to do this later.

#### - X/Y Axis Lengths

Each axis length can be configured, and nearly everything updates based on the axis length. The grid lines, the series points, the ticks and tick labels, the legend, and even the background sprite depend on the axis lengths.

#### - X/Y min/max auto grow booleans

If true, the absolute value of the corresponding axis value will automatically increase if any series data exceeds the boundary. The increase amount is specified in another parameter.

#### - X/Y min/max auto shrink booleans

If true, the absolute value of the corresponding axis value will automatically decrease if any series data is significantly below the boundary. The decrease amount is specified in another parameter. Also, the threshold at which the decrease happens is specified in another parameter.

#### - Auto shrink at percent

This is the threshold at which an auto shrink occurs. It is a percentage of the total axis length. For example, if the y axis min is 0 and max is 100, and this parameter is 60%, then a shrink will occur when the series data has a data point below 60.

#### - Auto grow and shrink by percent

This is the amount by which a grow / shrink increases / decreases an axis max / min value based on the total axis length. For example, if the y axis min is 0 and max is 100, and the series data has a data point exceeding 100, and this parameter is 20%, then the new max to be 120.

#### -Bar Width

This determines the width of the bars for the bar graphs. This is specified here instead of the series script, since the bar widths should never vary across series. However, the same may not be true for line graph point sizing, and so the parameter to control point sizes is on the series script.

#### -Axis Width

This determines the width of both the x and y axes.

#### -Hide X/Y ticks

This determines whether x/y ticks appear.

#### -Hide X/Y labels

This determines whether x/y labels appear.

#### - X/Y axis labels

This is the actual list of X/Y axis labels. This can be set automatically based on axis parameters, or set manually.

- Set X/Y labels using max / min

If this is true, then the labels automatically get set based on the number of ticks and axis max and min values. If, for some reason different labels like non-numeric labels are desired, then this should be false, and the labels set manually.

- Num Decimals in X/Y axis labels

This controls the number of decimals displayed in the axis labels.

- Auto center X/Y axis labels

This offsets each label to be in the center of each tick instead of the actual tick itself. This is fairly common practice for the x-axis, especially non-numerical x-axis labels, but not so much the y-axis.

- x/y-axis label spacing x/y

These control the distances that y-axis and x-axis labels are offset from the actual axis lines. The auto center axis labels parameters simply automatically sets these values based on the axis length and axis number ticks.

- x/y-axis label sizes

These control the size of the text sprites for the axis labels.

- y-axis ticks right, x-axis ticks above

These control where the axes ticks appear in relation to the axes themselves. These are always automatically set if an axes type other than "Manual" is specified. For example, an axes type of quadrant 1, the x-axis ticks will be below the x-axis, but for axes type of quadrant 3, the x-axis is at the top edge of the graph, and the x-axis labels should be above instead of below the x-axis.

- x/y-axis arrows

These control what, if any arrows display on the axes. Element 0 corresponds to the positive (top / right) arrows.

- x/y-axis line padding

This controls how much more space is extended beyond the actual axis length for axis arrows. This will likely depend on the size of your axis arrow sprite. If set to 0 then the arrow sprite would overlap with your maximum axis tick which wouldn't look good.

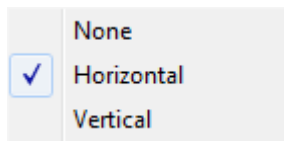
- x/y-axis y/x tick

These control where the axes are actually placed. By default they are set to 0, meaning the axes will be placed at tick 0, which is the bottom left. If these are set to be the middle tick such as 2 if the max tick is 5, then the axes will be in the center, corresponding to a 4 quadrant graph.

- Hide x/y tick

This will hide the tick label corresponding to the above parameter. Generally you will not need to hide the labels if the axes are at the edge, but when they are in the middle, the axis labels may overlap the actual axis if this is not set to true.

- Legend Type



This controls how the legend entries are arranged. The default is horizontal, but they could be vertical. If you don't want any legend at all then set this to None.

- Legend Position / legend parent offset x / y



This controls where the legend is positioned. By default it appears on the bottom.

- Hide legend labels

This will hide the legend labels, this should likely be used if using hover events to display series information.

- Legend Entry spacing x/y

This controls the space between each legend entry. This will likely depend on the text size of the legend entries and number of series used.

- Background Padding left / right / top / bottom

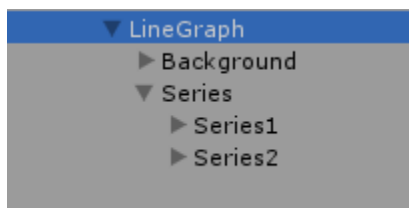
This controls how the graph background is padded. If these were set to 0, the background would be the same as the axis lengths + the axis padding lengths. If the legend is on the bottom, then you will need a larger bottom padding if you want the legend to be on the background.

## 3.3 Series Parameters

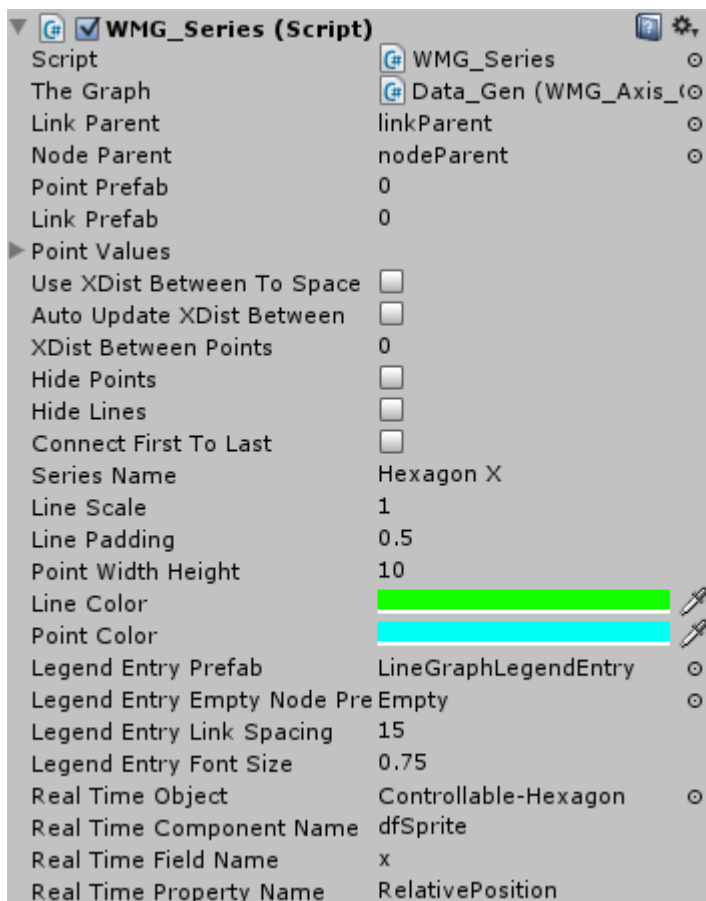
---

Line graphs can be customized for each series based on the series parameters.

Note that each series exists under the series parent under the graph parent:



The parameters:



#### - The Graph

This is a reference to the object that has the WMG\_Axis\_Graph script. An axis graph script is required to render a series.

#### - Point Prefab

For line graphs, this is the index of the prefab used in drawing the points. The list of possible point prefabs is on the graph script.

#### - Link prefab

For line graphs, this is the index of the prefab used in drawing the lines. The list of possible link prefabs is on the graph script.

#### - Point Values

This is the list of Vector2 float data used for this series. This data controls how the points are positioned. If only y-data is needed then set anything (like 0) to the x-value for each vector 2 and use the below parameters.

#### - XDist Between Points / Auto Update XDist Between / Use XDist Between to Space

These parameters can be used to automatically space data for the x-axis. In this case, it is not necessary to set series' x-axis data to be anything meaningful. "XDist Between Points" controls the x-spacing between points for each line series. If "auto update xDist Between" is true, then "XDist Between Points" auto updates based on the number of points that exist in the series and the length of the x-axis. Lastly, if "Use XDist Between to Space" is false, then the x-values specified in the line series data will be used.



- Hide points / lines

These can be used if you want your line graph to only have points or only have lines for example.

- Connect first to last

If true, this will create a line between the first and last points. This can be used to create a circle or other shapes. See the circle example in the data gen example scene.

- Series Name

This is the name of the series. This can appear in a graph legend.

- Line Scale

For line graphs, this is the thickness of the lines as defined by the object's transform local scale.

- Line Padding

This is the amount of space between a line ending and the middle of a point. This can be used if using hollow points for your node prefab.

- Point Width Height

For line graphs, this is the width and height of each point sprite.

- Line / Point Color

These can be used to controls the colors of lines / points / bars

- Legend Entry Prefab / Empty Node Prefab

These are prefabs used to create the legend for the graph if a legend is displayed. The empty node prefab is used to create line graph legends where there is a point connected with two lines on the side. This is actually constructed using 3 points with a link between the center points and the other 2 points. The other 2 points need to be empty so they don't display.

- Legend Entry Link Spacing

This is the length of each of the lines appearing on the side of the node for the legend entry for line graphs.

- Legend entry font size

This is the font size of the series name that will display in graph legends.

- Real time object

See the real-time example in the data gen scene for more info. This is the gameobject that will be used when starting a real-time series. The functions `WMG_Series.StartRealTimeUpdate()` and `WMG_Series.StopRealTimeUpdate()` will use this.

- Real Time Component Name

This is the string for the component to use. For example if graphing a transform's local position's x, then this would be "Transform". This can be any arbitrary script.

- Real Time Field Name

This is the string representing the field to use. For example if graphing a transform's local position's x, then this would be "x". This can also be a standalone property.

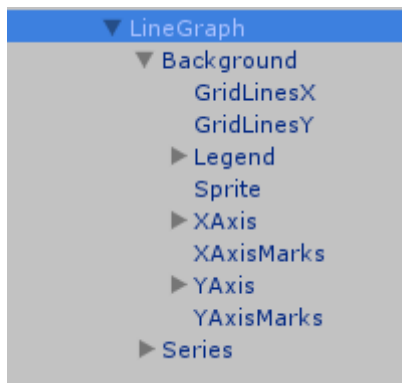
#### - Real Time Property Name

This is the string representing a property parent to use. For example if graphing a transform's local position's x, then this would be "localPosition".

## 3.4 Other Notes

---

The grid lines are implementations of grids. Grid parameters will be explained in a later section, but the grids can be turned off by simply disabling them in the hierarchy since the parent objects for them exist before running the scene:



The horizontal grid is controlled by GridLinesX.

The vertical grid is controlled by GridLinesY.

The Legend is just an empty gameobject that will be used as a parent for the legend entries when they are created.

The x-axis and y-axis each are just three sprites, one for the line and two for the arrows.

The x / y axis marks (ticks) are also grid implementations where the node prefab is a label, and the link object is the axis tick. The grids would normally produce 2 text labels for each tick, but half the text labels are disabled by calling some functions to get them that are public in the grid script.

## 3.5 Animations

---

The graph script contains functions that can be used in your own code to do animations. There are currently three main functions:

#### - animScaleAllAtOnce

This animates everything at once.

#### - animScaleBySeries

This animates each series consecutively, one after the other.

#### - animScaleOneByOne

This animates points or lines based on their position in the List<Vector2> , and attempts to animate across multiple series at the same time. If each series has the same number of points, then each point should animate at the same time across all the series. If there are 50 points in one series and 10 points in another series, then 5 points will animate from the 50 point series in the same time it takes to animate 1 point from the 10 point series.

Another useful function to get different looking animations involves changing the pivots of lines:

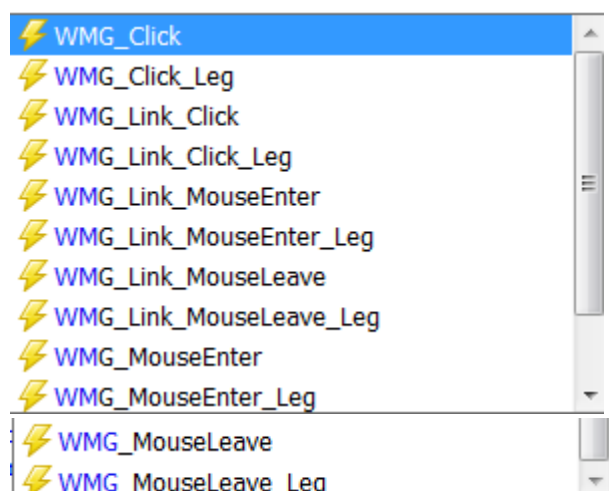
- changeAllLinePivots

Refer to the Data Gen example scene and WMG\_Data\_Gen script for more details on how to use these in your code.

## 3.6 Events

---

The graph script contains events that can be used in combination with your custom code to add interactivity to your graphs. These are currently 12 available events (note that "Leg" refers to legend):



Refer to the example scene for examples of them being used practically. Note that for NGUI, there are no "MouseLeave" events because for NGUI there is only an OnHover() event, but for Daikon, there is both MouseEnter and MouseLeave events. Instead, the OnHover() event passes a boolean to represent whether it was an enter or leave.

## 3.7 Dynamically Add / Delete Series

---

The graph script contains functions to add and delete series:

```
public WMG_Series addSeries()
```

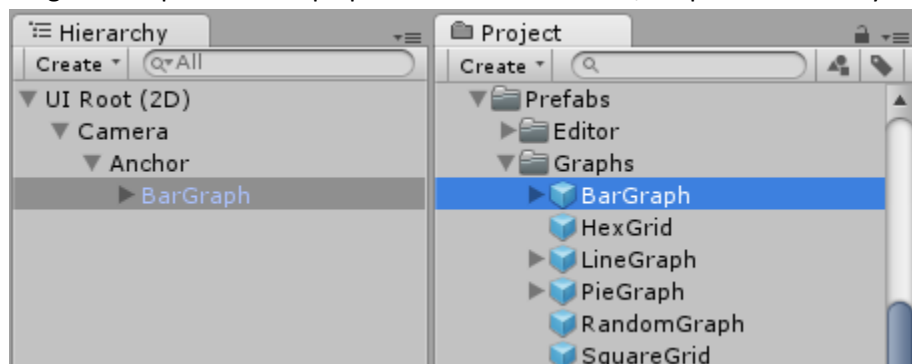
```
public void deleteSeries()
```

Simple call these functions and a series is added onto the end or deleted from the end.

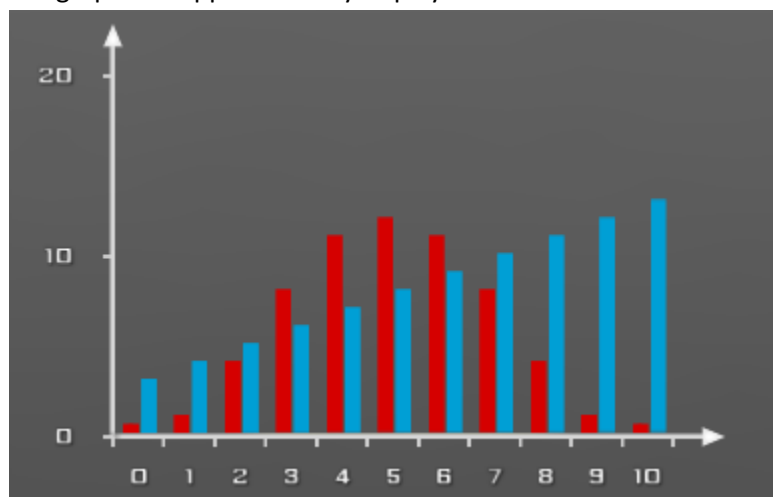
## IV. Bar Graphs

### 4.1 Getting Started

Drag and drop the BarGraph prefab from the Prefabs/Graphs folder into your scene:



The graph will appear when you play the scene:



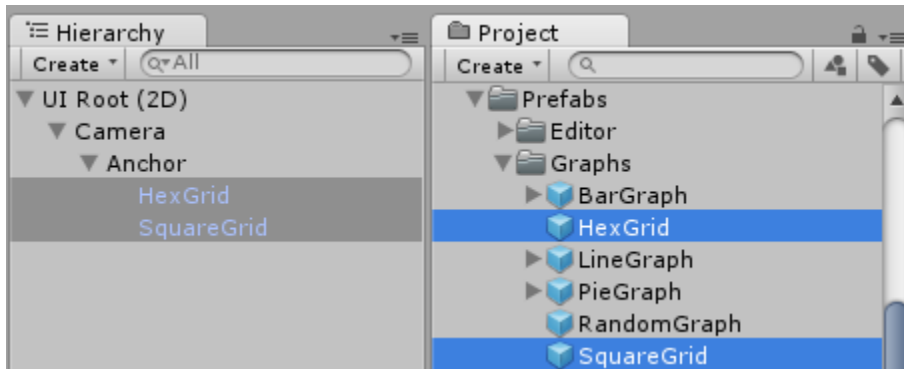
### 4.2 Axis / Series Graph Parameters

Bar graphs can be customized based on the axis graph parameters and the parameters associated with each series assigned to the graph. Parameters for the axis graph are the same for both line and bar graphs. There are only a couple parameters that are specific to bar / line. Refer to section 3.2 for the axis graph parameters. Simply use the graph type parameter to change between line and bar graphs.

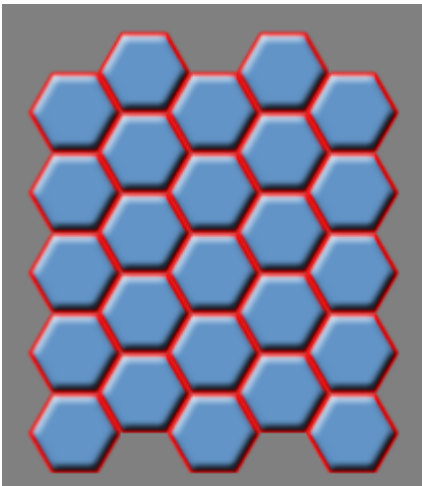
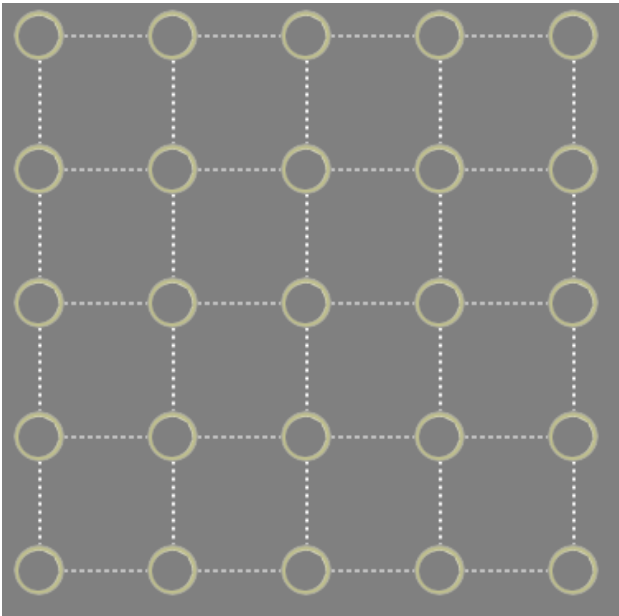
## V. Square / Rect / Hex Grids

### 5.1 Getting Started

Drag and drop the SquareGrid / HexGrid prefab from the Prefabs/Graphs folder into your scene:



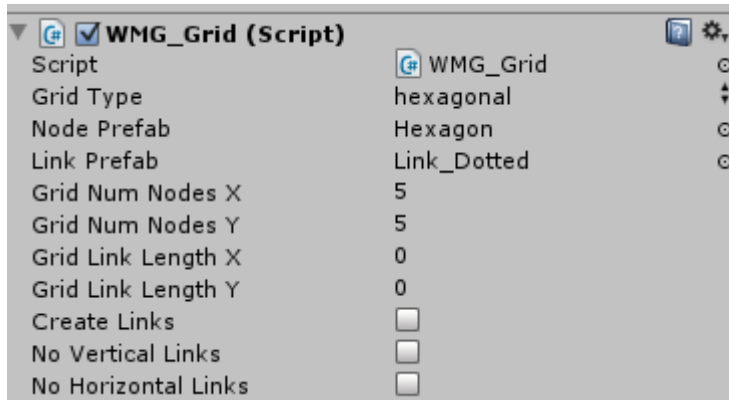
The grids will appear when you play the scene:



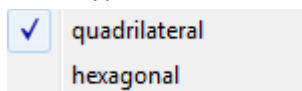
## 5.2 Parameters

---

Grids can be customized based on following parameters:



### - Grid Type



This determines whether the grid will be a square / rectangular grid vs. a hexagonal grid

### - Node Prefab

This is the prefab used for the nodes.

### - Link Prefab

This is the prefab used for the links. For quadrilateral grids, each node has 4 links. For hexagonal grids, each node has 6 links.

### - Grid num nodes x / y

This determines how many nodes are in the x and y directions.

### - Grid Link Length x / y

This determines the length of the links in the x and y directions.

### - Create Links

This determines whether links are created.

### - No vertical / horizontal links

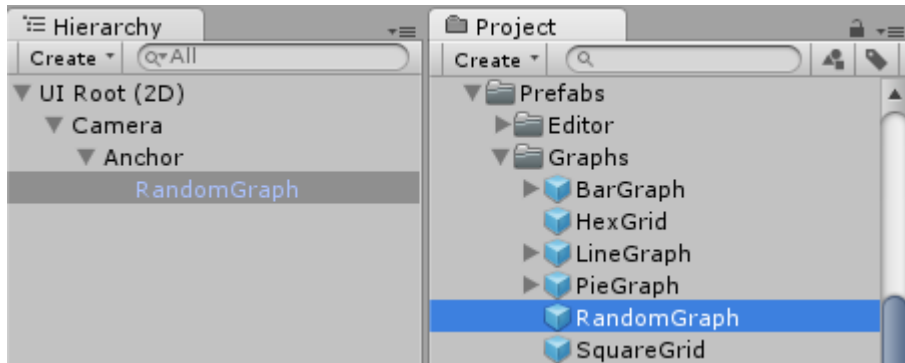
This determines whether links are created in certain directions. This is primarily used for the grid implementations in the line graph. The horizontal grid lines have no vertical links checked, and the vertical grid lines have not horizontal links checked.



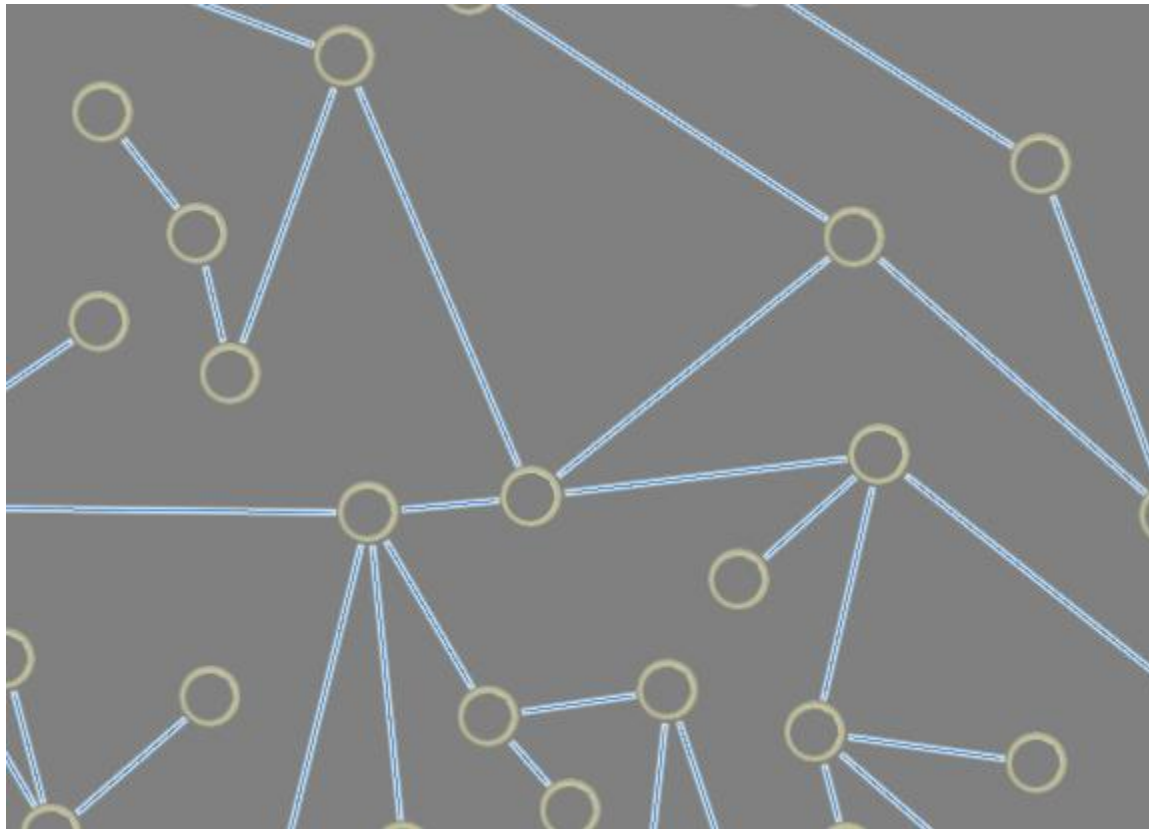
## VI. Random Graphs

### 6.1 Getting Started

Drag and drop the RandomGraph prefab from the Prefabs/Graphs folder into your scene:



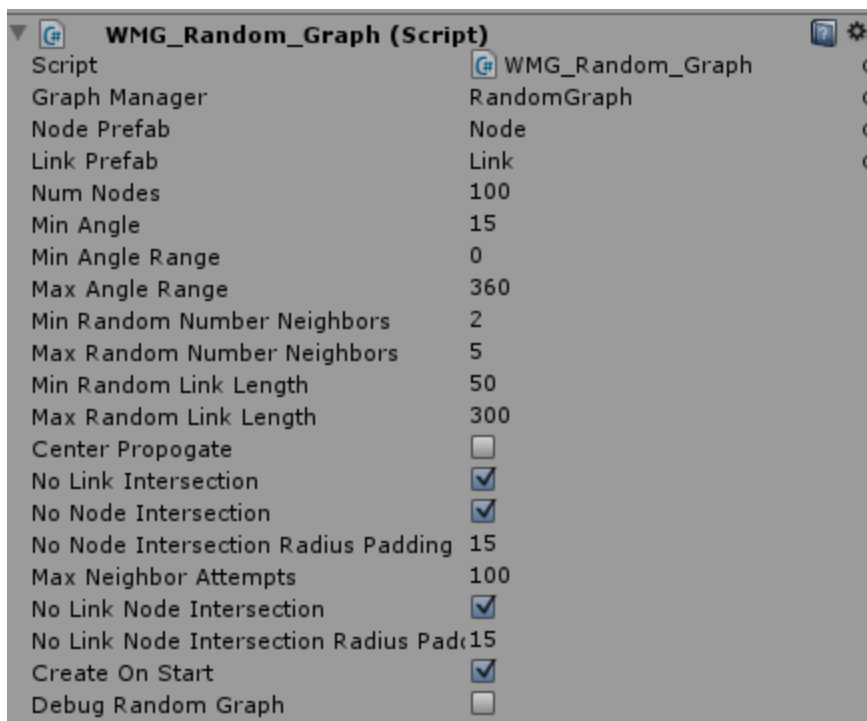
The graph will appear when you play the scene:



### 6.2 Parameters

Random Graphs can be customized based on following parameters:





#### - Graph Manager

This is a reference to the game object with the graph manager script required for all graphs.

#### - Node / link prefabs

The prefabs used for each node and link

#### - Num Nodes

This is the total number of nodes that will appear in the resulting graph

#### - Min Angle

This is the minimum possible angle between neighbor nodes. So if this is 15, then there should not exist any 2 neighbors that are less than 15 degrees apart from one another.

#### - Min / Max Angle Range

By default the range is 0 - 360, meaning that any randomly generated node can appear at any angle around a given node. This can be used to control the direction in which the graph propagates. For example, to create procedural lightning looking graphs you would want to set this to a narrow range like 0 - 45.

#### - Min / Max random number neighbors

This controls how many neighbors there are for each node. For example, if every node should have 3 neighbors, then set the min and max to 3.

#### - Min / Max random link length

This determines the distance between nodes. Setting a high range will create more sporadic looking graphs, while setting the values equal will generate grid like graphs.

#### - Center propagate

This determines how the propagation proceeds. If it is unchecked, then a node is randomly picked from the set of unprocessed nodes to process. Processing a node randomly generates neighbors for that node, marks the node processed, and moves on to another node process. If this is set to true, then the next node processed will be the oldest one that was created.

#### - No link intersection

This ensures that a randomly generated link does not intersect with any existing links. This should generally always be set to true unless you want to create some strange overlapping graph.

#### - No node intersection

This ensures that a randomly generated node will not intersect with any existing nodes. Circle intersection checks are done using the radii of the nodes. This should generally always be set to true unless you want nodes to possibly overlap.

#### - No node intersection radius padding

This adds onto the radii used in the circle intersection checks used for the node intersection checks. Increasing this value will ensure that nodes are more spaced apart from one another.

#### - Max neighbor attempts

Sometimes highly depending on the parameters used, the graph will fail to produce any results, or fail to produce all the nodes specified. If this happens a warning is logged to the console saying how many nodes were produced which was less than the number of nodes you specified. This generally means your parameters were too specific. If you still feel your parameters are accurate, you can increase this number to try and fully complete the graph. The default is 100, meaning while processing a neighbor, up to 100 random angles and link lengths are generated. Failing any of the checks such as the min neighbor angle or intersection checks will increase the attempt number and generate a new possibility.

#### - No link node intersection

This ensures that creating a new node does not intersect an existing link, or that creating a new link does not intersect an existing node. This performs circle-line intersection checks with the creating link / node with all existing nodes / links.

#### - No links node intersection radius padding

This increases the radius of the node used in the circle-line intersection checks. A higher value will ensure a graph that has links and nodes that are more spaced apart from each other.

#### - Create on start

If this is true, then the `GenerateGraph()` function is called in the `OnStart()` function. If you want to change parameters at run-time and then generate the graph yourself then you would set this to false, get a reference to the script and call the public function `GenerateGraph()`.

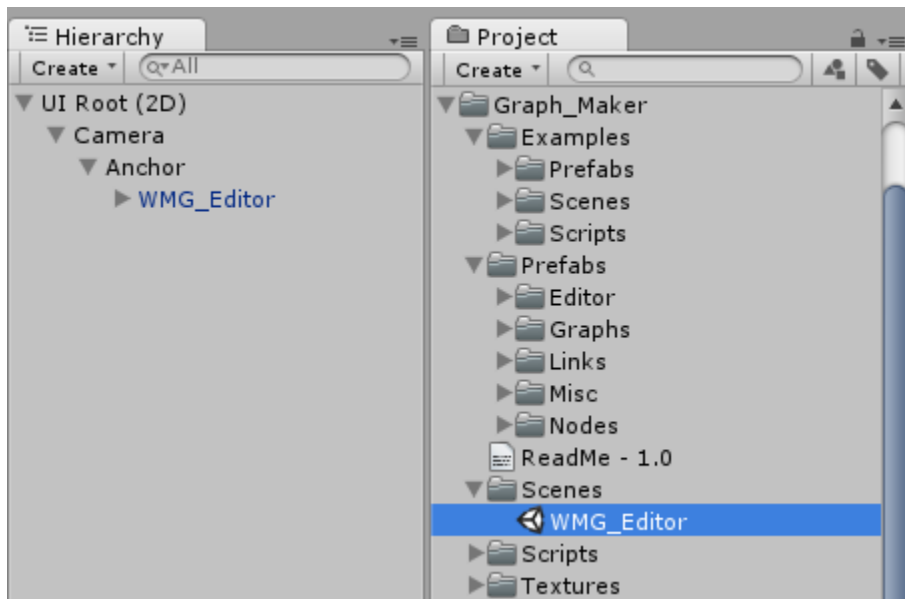
#### - Debug Random Graph

This can be useful to troubleshoot exactly what is happening during the random graph generation process. I resolved many bugs, and added new functionality using this parameter.

## VII. Graph Editor

### 7.1 Getting Started

Double click the WMG\_Editor scene in the Scenes folder:

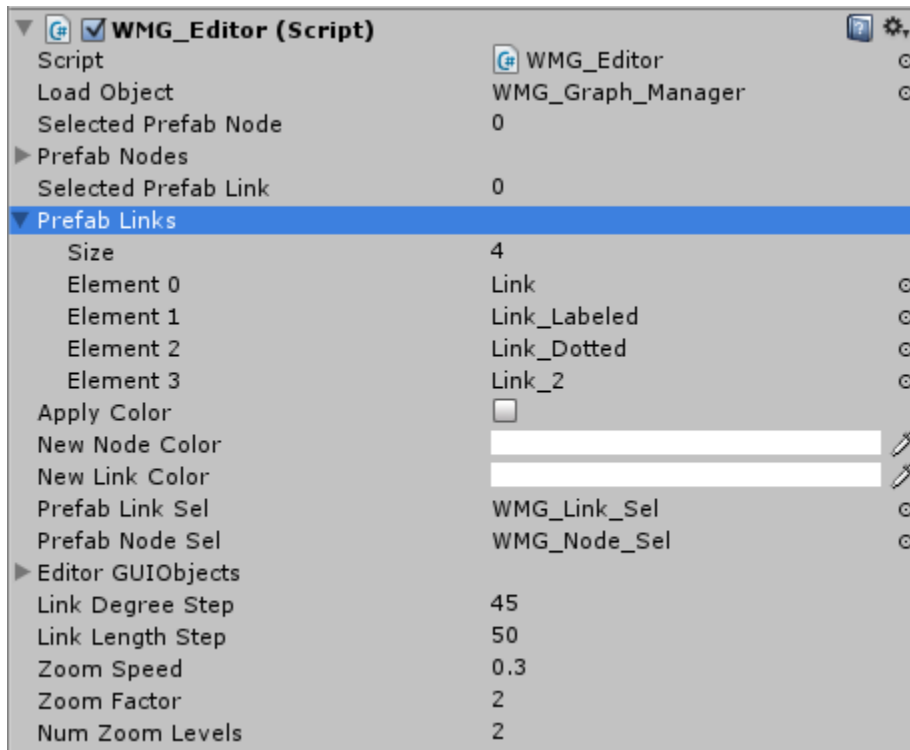


The editor GUI will appear:



## 7.2 Parameters

The editor can be configured based on the following parameters:



### - Load Object

This object is loaded when you hit play. The load object just needs the graph manager script attached and it should load. This can be used in combination with the Save button to save and load your own custom creations.

### - Prefab Nodes / Selected Prefab Node:

The prefab nodes is the list of node prefabs you can use in the editor. Each node just needs a WMG\_Node script attached for it to work in the editor. This list of prefabs automatically populates this drop-down list for use in the editor:



#### - Prefab Links / Selected Prefab Link:

Similar to nodes, this is the list of link prefabs that can be used in the editor. Each link just needs a WMG\_Link script attached for it to work in the editor. The list of prefabs automatically populates this drop-down list for use in the editor:



#### - Apply Color / New Node / Link Colors:

When apply color is checked, the color object referenced in WMG\_Node / WMG\_Link will automatically update to the colors specified here if the node / link is selected or new nodes / links are created.

#### - Prefab link / node sel:

These are the objects used to display the selection objects when selecting nodes / links in the editor. These should generally not change, unless you specifically need different looking editor selection objects.

#### - Editor GUI objects:

This is just the list of object references used in the editor script. This should not change, unless you want to add additional editor GUI options.

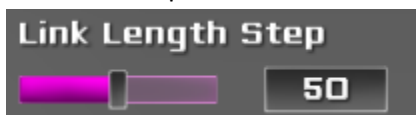
#### - Link Degree Step:

You can place nodes relative to another node with a specific degree step using this parameter and by holding left shift and right click dragging from an existing node, or by left click dragging a node with a neighboring link selected. The value set here should correspond to this GUI, but can also be manually changed in the Editor script:



#### - Link Length Step:

You can place nodes relative to another node with a specific link length step by using this parameter and by holding left control and right click dragging from an existing node, or by left click dragging a node with a neighboring link selected. The value set here should correspond to this GUI, but can also be manually changed in the Editor script:



#### - Zoom speed / zoom factor / num zoom levels:

These can be configured to change how zooming works in the editor.

## 7.3 Editor Controls

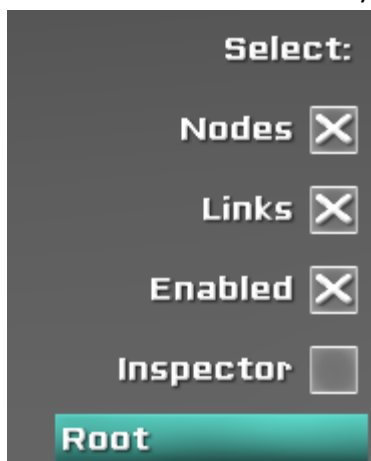
---

General	
<b>Mouse R:</b>	Create new node with currently selected prefab at mouse location
<b>Mouse R + Drag From Node:</b>	Create a new node and link, or just a link between nodes if released on another node
<b>Mouse R + Drag From Background:</b>	Pan
<b>Mouse Wheel:</b>	Zoom in / out
<b>Mouse L:</b>	Select node or link mouse is over
<b>Mouse L + Drag From Node:</b>	Move current selection
<b>Mouse L + Drag From Background:</b>	Multi-select, hold Left Shift to add to selection, hold left control to inverse selection

## 7.4 Selections

---

There are some additional ways to work with selections in the editor.



The Nodes and Links checkboxes control whether nodes / links are selected. For example, if nodes is checked and links is unchecked and you drag select a bunch of objects, only node objects will be selected.

The Enabled checkbox controls whether selections in general are enabled in the editor. This can be used to preview your GUI as if it were not in the editor (editor selection objects are disabled).

The Inspector checkbox controls whether your selections also apply to the hierarchy window in the inspector. This can be used to mass select objects in the hierarchy and then take actions on those objects from the project windows.

The pop-up list that says "Root" by default allows selecting other objects in the inspector when inspector is checked. For example, if your node prefab has an "Object to Label" reference you can change root to "Label Object" which will select that node's referenced label object. The default is root, meaning the node with the WMG\_Node / WMG\_Link script is selected in the inspector.

## 7.5 Saving and Loading

---

It is possible to save your custom creation and then load it at a later time in the editor. Since the editor has editor specific objects such as the selections, these are automatically deleted during the saving process and automatically re-added during the load process.

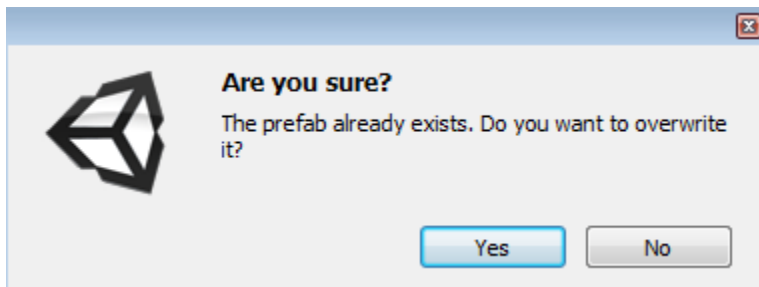
To save, simply click the save button:



This will create a new prefab at Resources/Prefabs with a name equal to the prefab name in the "Load Object" variable in the Editor script:



If a prefab with this name already exists, then you will be prompted to overwrite the existing prefab:



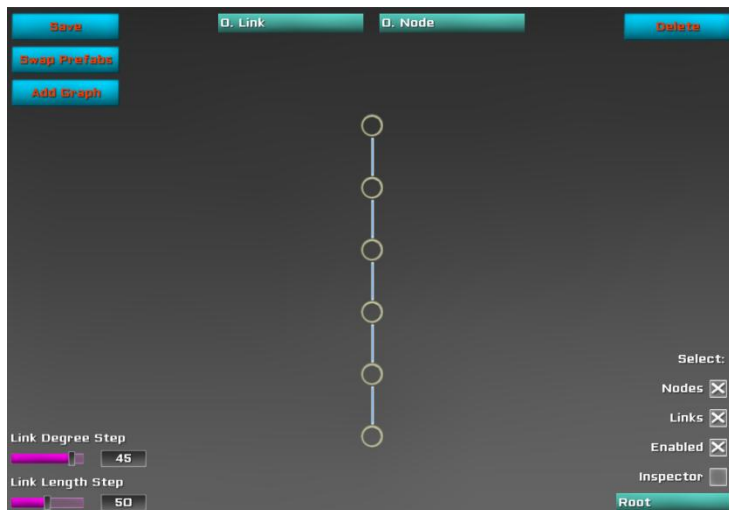
To load an object, stop the WMG\_Editor scene and drag and drop the prefab that was created from the Save process into the "Load Object" variable. Now when you hit play, the editor will load your previously saved object.

## 7.6 Prefab Swapping

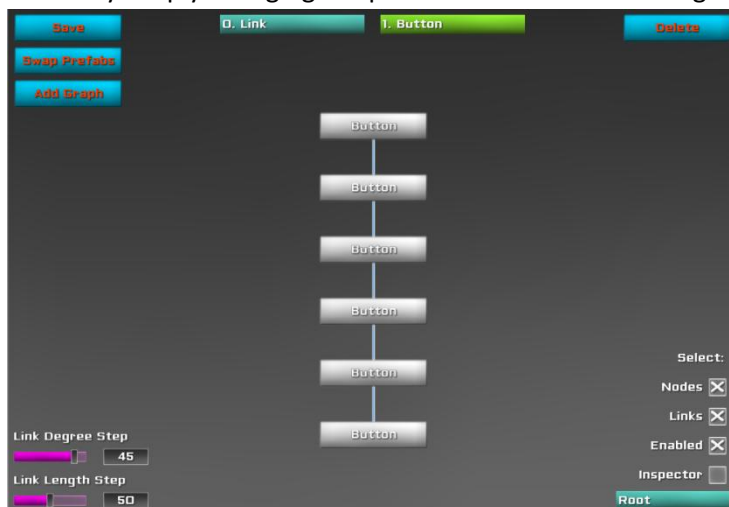
It is possible to swap prefabs in the editor at any time based on your current selections using the Swap Prefabs button:



You can change this:



To this by simply changing the prefab selection and clicking swap prefabs:



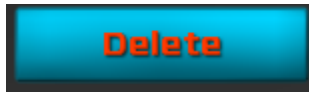
How it works-

It destroys the existing selected objects / links and creates new objects and links with the same WMG\_Node / WMG\_Link data as the destroyed objects. The names of the gameobjects are also preserved. If you have custom scripts with custom data on your prefab those will not survive the process unless of course the custom scripts are the same / have the same default values across the prefabs, then it doesn't matter in that case.

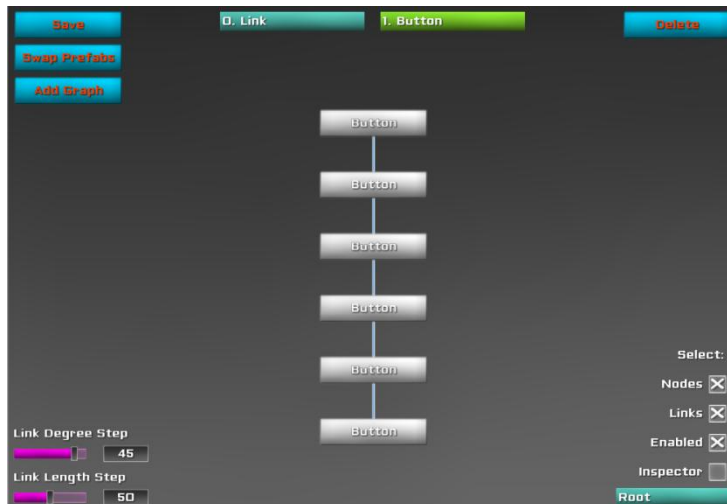


## 7.7 Deleting

You can delete based on your current selections with the delete button:



You can change this:



To this by simply selecting the links and clicking the delete button:



There is no undo action at this time, so delete responsibly!

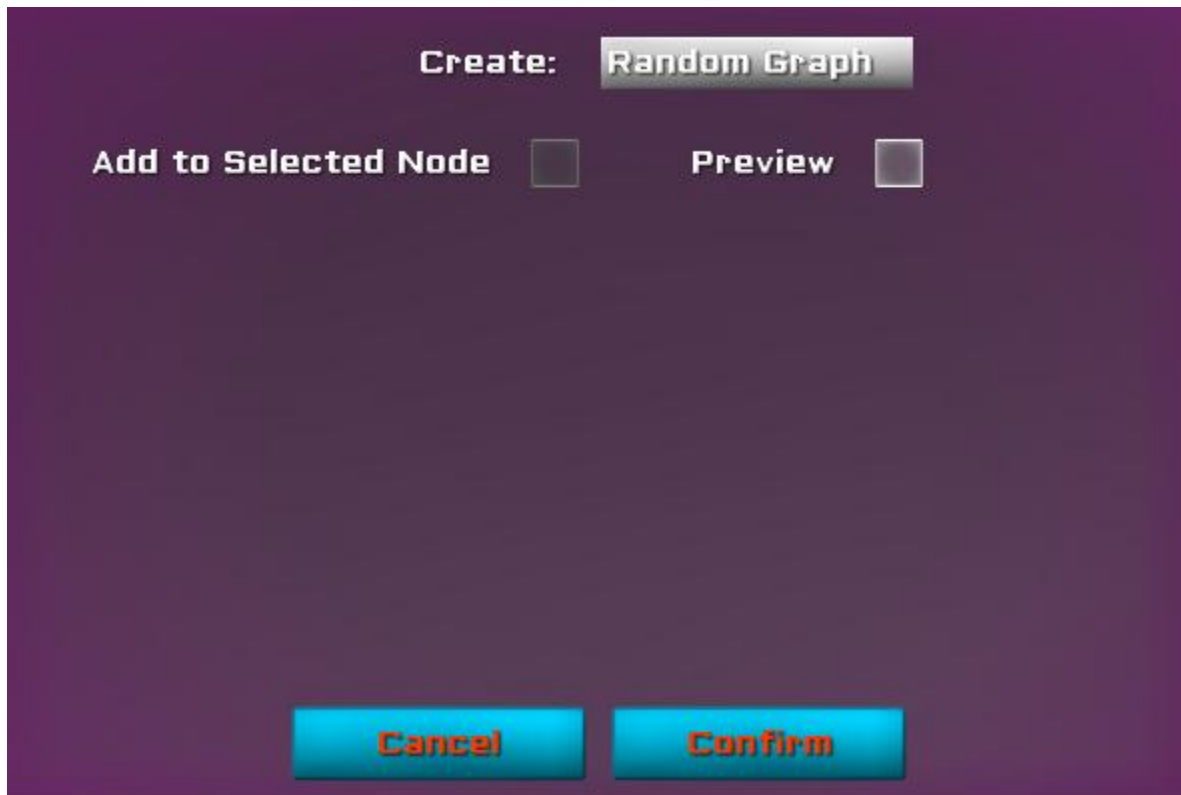
## 7.8 Adding Graphs

---

You can add random graphs and grids to be used in the editor with the Add Graph button:



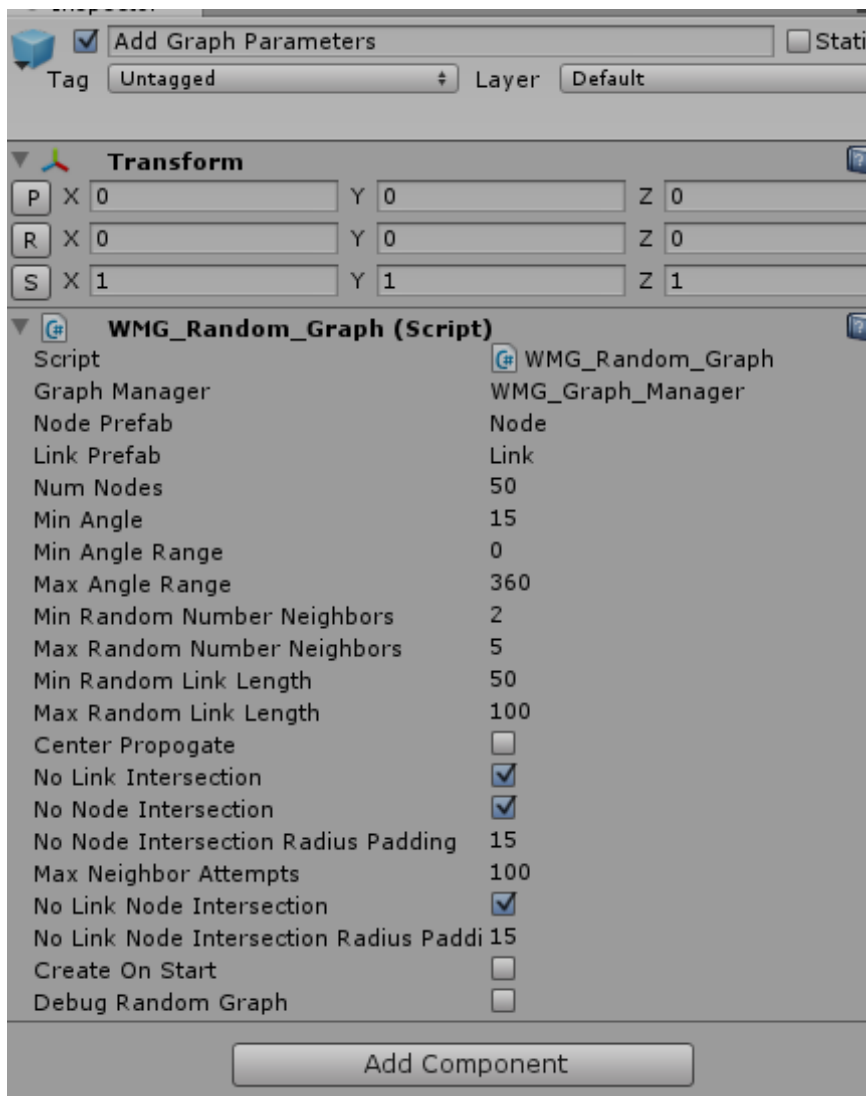
A popup window will appear with some additional options:



Choose the graph type with the pop-up list. This controls what script is attached to the "Add Graph Parameters" object in the hierarchy. This object is automatically selected when adding a graph, and is the place you go to edit the parameters corresponding with the graph. The default is random graph, so this object will be selected:



and this object will have only this script when Random Graph is selected in the drop down:



After editing the parameters you can preview the result by clicking the preview checkbox. This will create the random graph. If you uncheck preview it will delete the results, allowing you to preview a different result by checking preview again. With this, you can refresh until you get a graph you want.

Note that preview is disabled for grids, because they are created via a refresh function.

You can also click the add to selected node for random graphs if you have a single node selected. This will create the random graph from the selected node.

Once you confirm the graph will be created in the editor and all the editor objects will be applied, allowing you to manipulate the results within the editor.