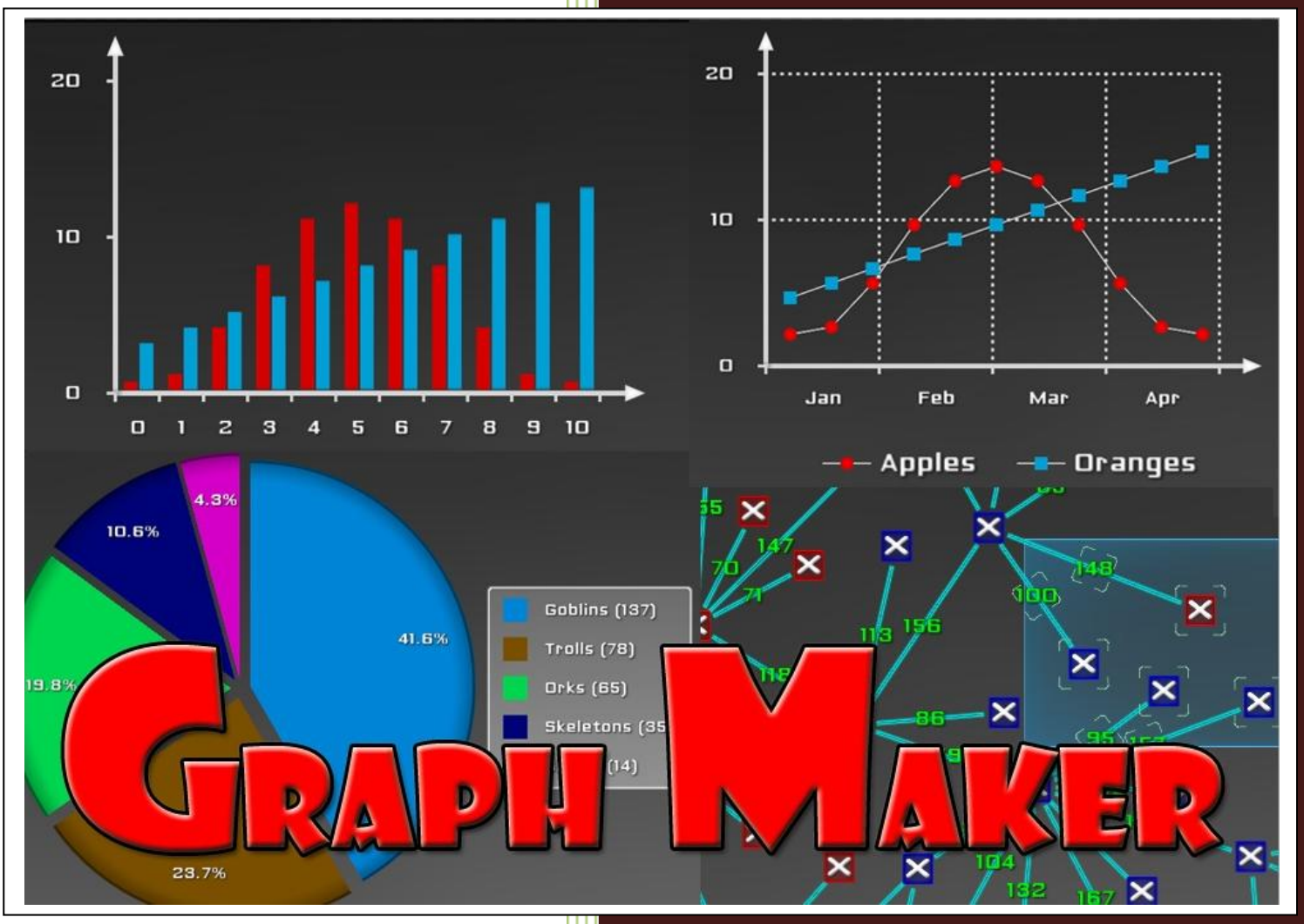
Graph Maker


20
10
0
0 1 2 3 4 5 6 7 8 9 10
20
10
0
Jan
Feb
Mar
Apr
Apples
Oranges
4.3%
10.6%
41.6%
19.8%
28.7%
Goblins (137)
Trolls (78)
Orks (65)
Skeletons (35
(14)
GRAPH MAKER

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.

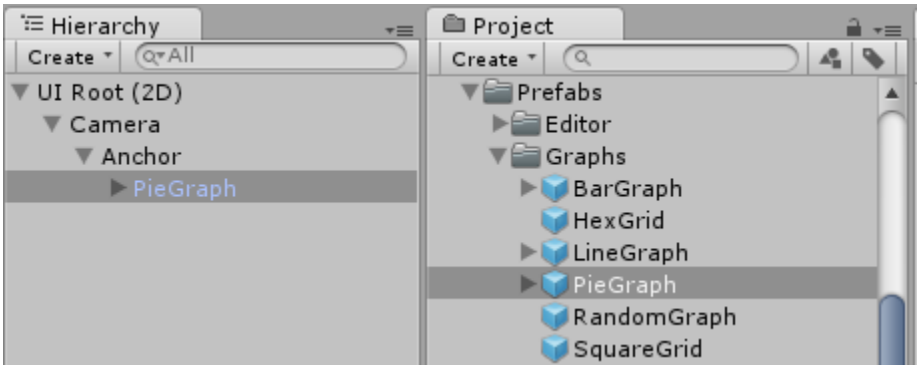
1.2 Features

- 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

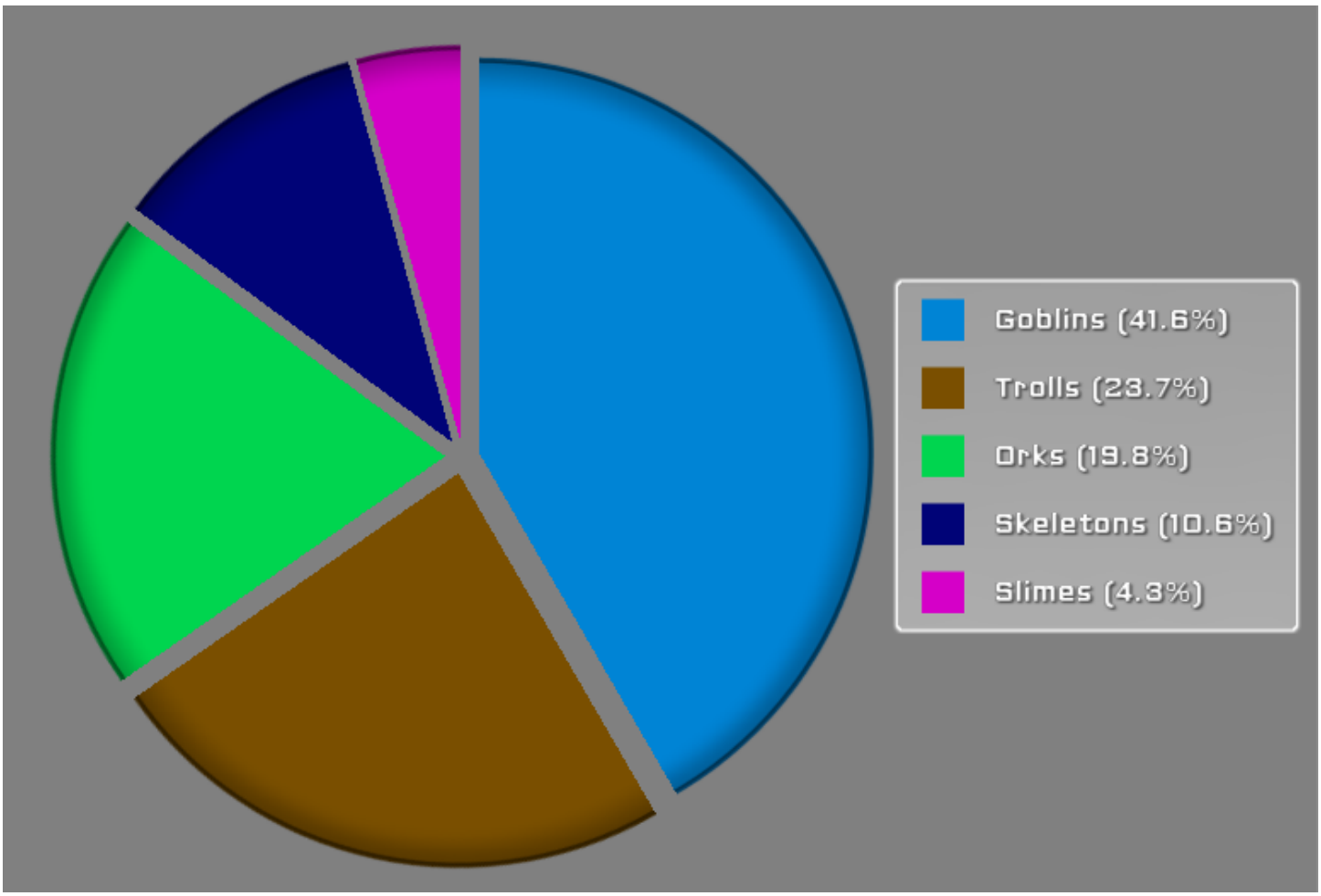
II. Pie Graphs

2.1 Getting Started

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

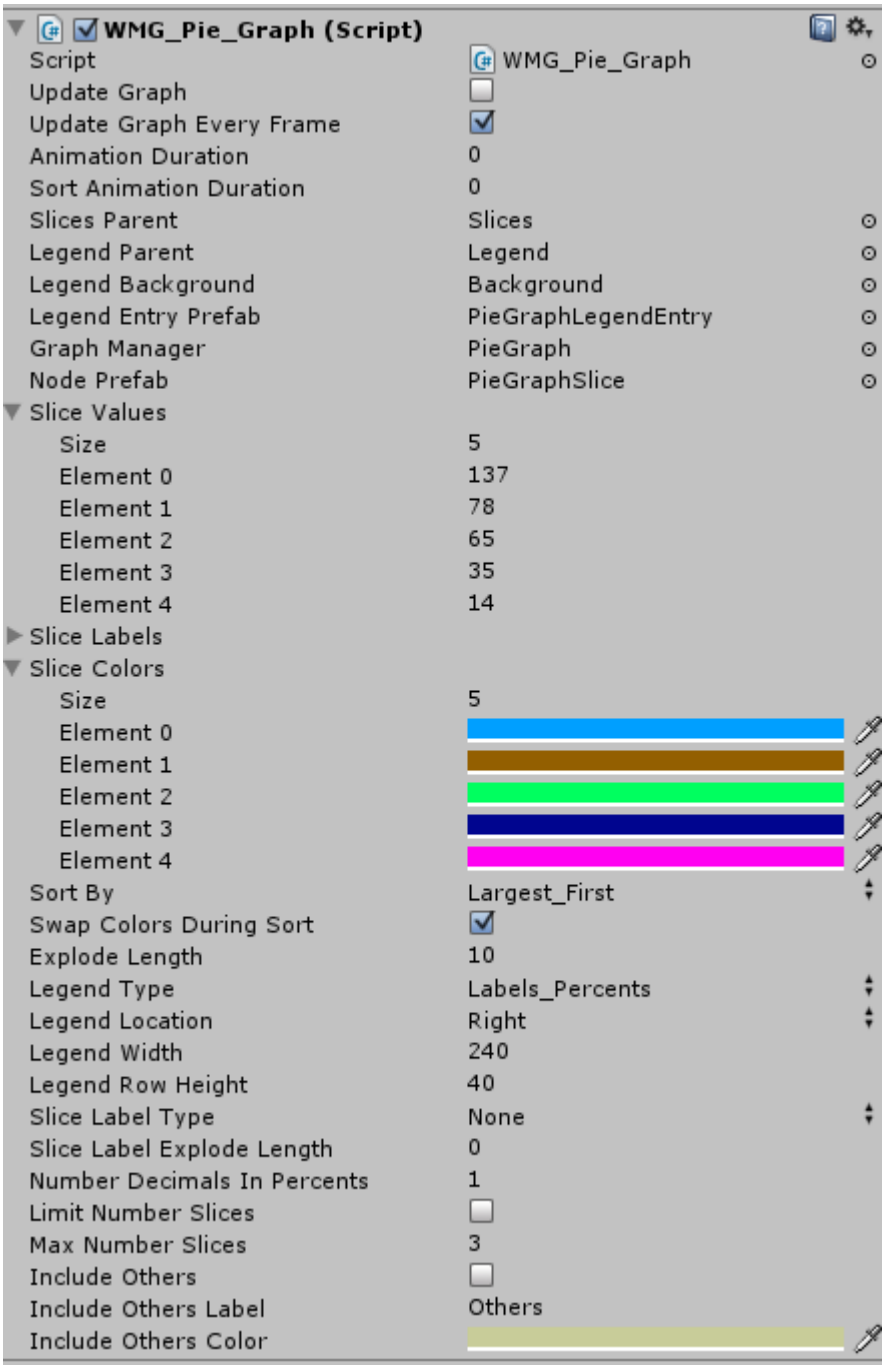


The graph will appear when you play the scene:



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.

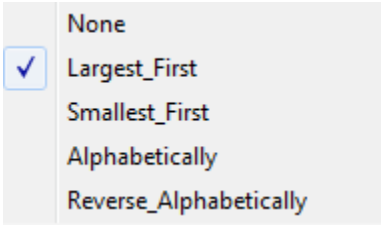
- Slice Values

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

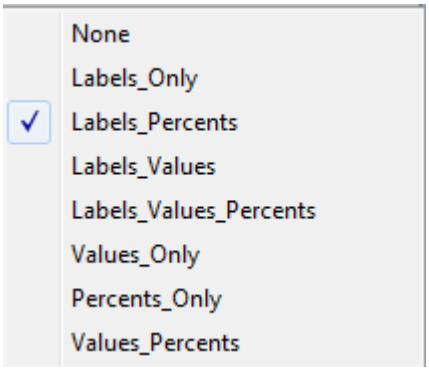


This controls the order that the pie slices appear. The default is Largest_First, meaning that the slice with the highest value will appear in the top right, and subsequent slices go in clockwise order. The other sorting options should be intuitive based on the name. For the "None" sorting, the slice 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.

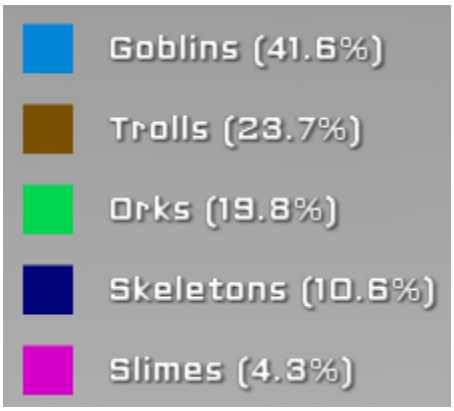
- Legend Type



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

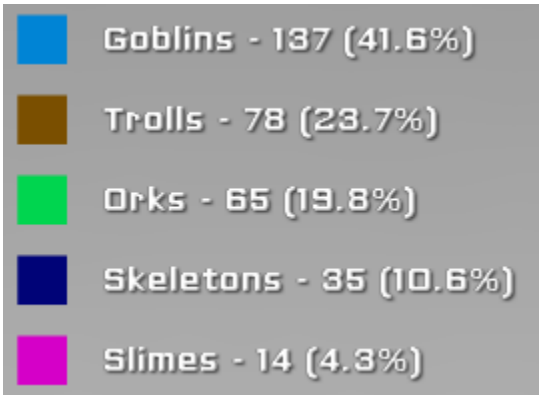
Labels Only:

Labels Percents:

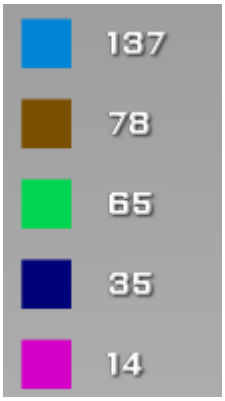


Labels Values

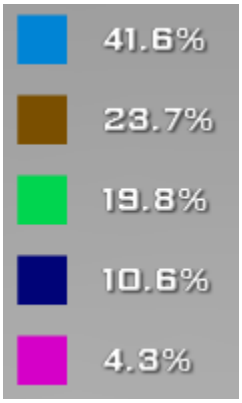
Labels Values Percents



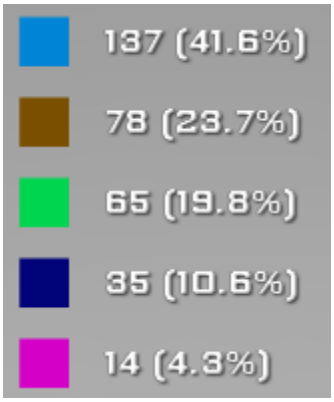
Values Only



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.

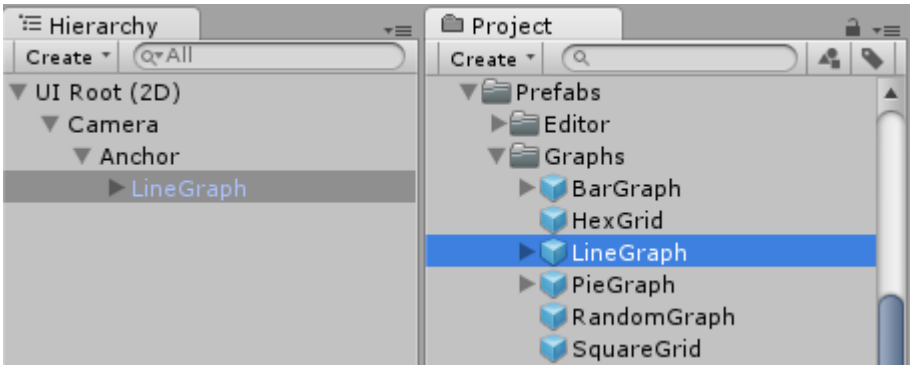
- Limit number slices / max slices
When limit number slices is checked, the pie graph will limit the number of slices displayed to the maximum number set. So if there are 10 data values, and 3 is set then only 3 of those will be used, the 3 used depends on sorting.

- 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.

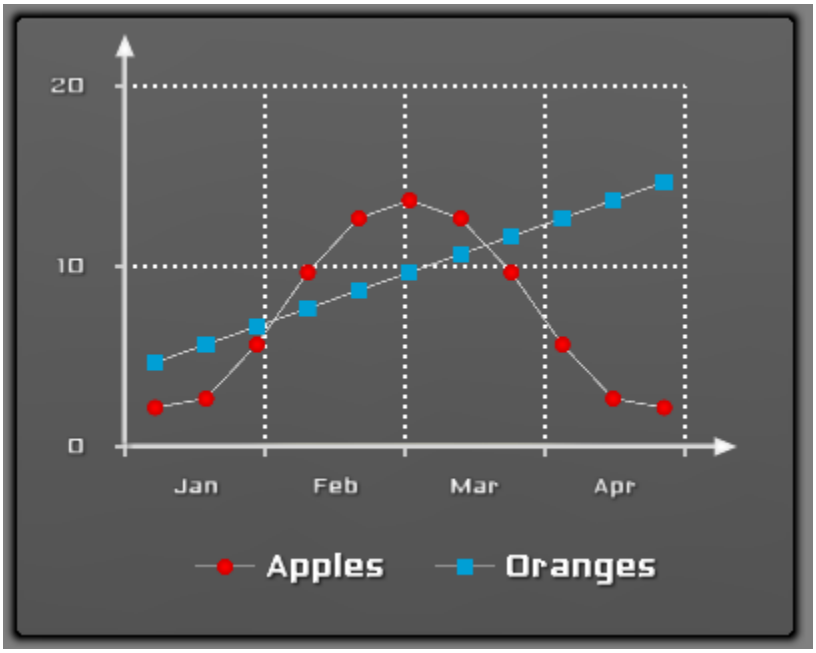
III. Line Graphs

3.1 Getting Started

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



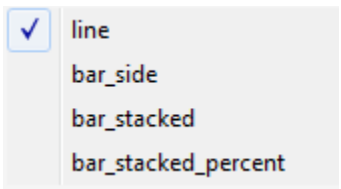
The graph will appear when you play the scene:



3.2 Axis Graph Core Parameters

-Graph Type

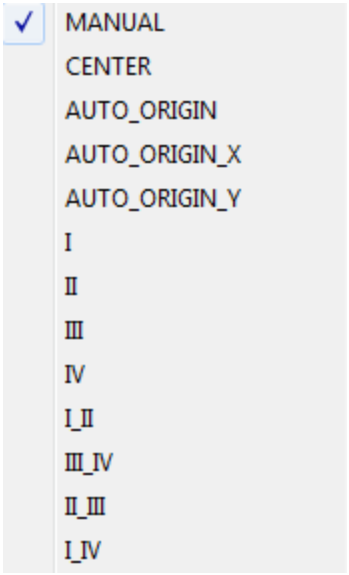
Changes the graph from the default line graph to various different types of bar graphs:



-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.



Manual means Graph Maker will not do anything with regards to moving around the axes.
Roman numerals refer to the various quadrant possibilities.
Auto origin will automatically position the axes to be closest to the origin. So if you change the axes min and max values such that the axis would change its position, then the axes will automatically adjust to be closest to the origin. The origin is also configurable via another parameter.

- 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.

- Bar Axis Value
This controls the starting point for bar charts. For example, if the y-axis min is 0 and y-axis max is 20, and this is set to 10, then the base will start from 10 and either go up or down depending on the data set for each bar. So, a bar representing a value of 5 will start from 10 and go down to 5, and a bar with a value of 15 will start from 10 and go up to 15.

- The Origin
This is the graph's origin. This will affect the behavior of the auto origin axes type options.
-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.

3.3 Axis Graph Tooltip Parameters

- Tooltip Enabled
This will show a tooltip based on where your mouse hovers for points, bars, and legends. The tooltip will display a single x or y value depending on the graph orientation, and line graphs will display (x,y). Legends will display the series name.

- Tooltip Offset
This is the number of pixels to offset the tooltip from the mouse.

- Tooltip Number Decimals
This is the number of decimals to display in the tooltip's x and y data.

- Tooltip Display Series Name
Determines whether or not the series name displays inside the tooltip. Useful if you have many series, otherwise recommend disabling.

- Tooltip Animations Enabled
Determines whether or not the gameobject underneath the mouse for the tooltip plays an animation.

- Tooltip Animations Easetype
The easetype for the tooltip animations.

- Tooltip Animations Duration
The duration for the tooltip animations.

3.4 Axis Graph Animation Parameters

- Auto Animations Enabled
Determines whether automatic animations play. Automatic animations will happen for orientation change, and for data changes such that data points are not added or deleted. Automatic animations currently does not work for stacked bar charts.

- Auto Animations Easetype
The easetype for auto animations.

- Auto Animations Duration
The duration for auto animations.

3.5 Axis Graph Axis Parameters

- 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.

-Hide X/Y ticks
This determines whether x/y ticks appear.

3.6 Axis Graph Label Parameters

-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.

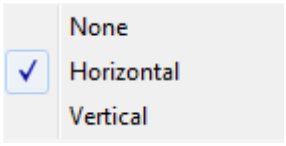
- Auto center X/Y axis labels
This offsets each label to be in the center of each tick instead of the actual tick itself. This is fairly common practice for the x-axis, especially non-numerical x-axis labels, but not so much the y-axis.

- 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.

3.7 Axis Graph Legend Parameters

- 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.

-Legend Num Rows or Columns
This controls how many rows will appear for horizontal legends, and how many columns will appear for vertical legends. If the number of series does not divide evenly into the number of rows / columns, then the first row(s) / column(s) will have the extras. For example, for a horizontal legend, if there are 10 series, and this is set to 4, then the first 2 rows will have 3, and the second 2 rows will have 2.

3.8 Axis Graph Misc Parameters

-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.

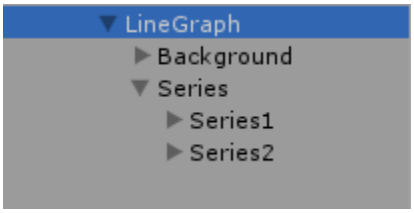
- x/y-axis Use Non Tick Percent
This positions the axes based on percentages rather than fixed gridline ticks. This should be enabled if you want the auto origin axes type options to move the axes around freely rather than at fixed gridlines.

- x/y-axis Non Tick Percent
This is the location of the axes relative to the other axis based on a percentage. This is only used if the "Use Non Tick Percent" is enabled. This is automatically calculated for the auto origin axes type options.

- 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.

3.9 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 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.

- Extra X Space
This can be used to add extra space between series. This can be useful to add spacing between bars for side-by-side bar charts.

- 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 Spacing
This controls the spacing between the icon and the text for the legend entry of this series.

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

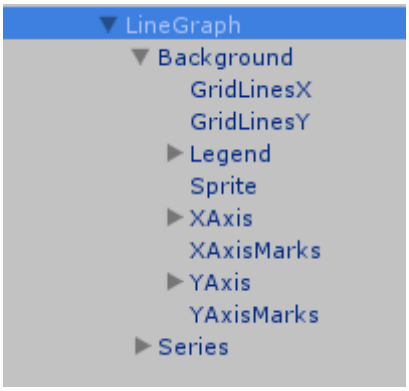
- Area Shading Type Parameters
Refer to the separate section about area shading.

- Real-Time Parameters
Refer to the separate section about real-time updating.

- Data Label Parameters
Refer to the separate section about data labels.

3.10 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.

3.11 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.

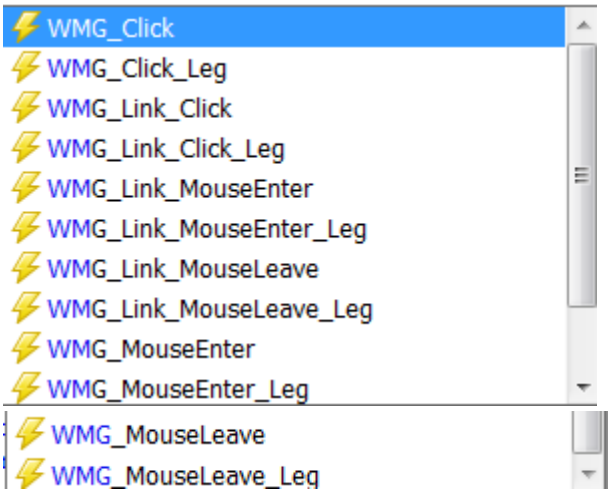
- animScaleOneByOne
This animates points or lines based on their position in the List<Vector2> , and attempts to animate across multiple series at the same time. If each series has the same number of points, then each point should animate at the same time across all the series. If there are 50 points 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.

3.12 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.

3.13 Dynamically Add / Delete Series

The graph script contains functions to add and delete series:

public WMG_Series addSeries()

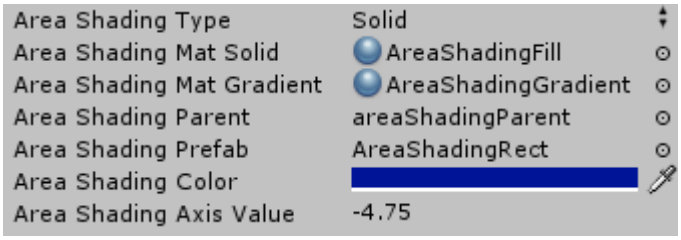
public void deleteSeries()

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

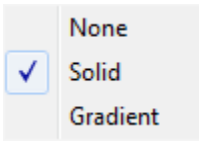
3.14 Stacked Line Graphs / Area Shading

There are several parameters on WMG_Series that allows the ability to add custom area shading for the series. Note that each rectangle takes an additional draw call due to the use of the custom shaders. There is a rectangle created between each 2 points in the series when area shading is enabled (Area Shading Type != None).

So, if a series has 10 data points, then 9 rectangles will be created which have an applied custom material. A new material is instantiated and various parameters are set on the material for each rectangle in order to set the correct alpha clipping based on the series point data.



The area shading type can be changed from None (default) to either solid or gradient:



Solid will use the material assigned for Area Shading Mat Solid, and Gradient will use the material assigned for Area Shading Mat Gradient.

The solid material has no alpha transparency, and will be the better option for performance if alpha transparency is not needed. The color picker will still let you set an alpha, but it will not do anything.

The Area Shading Parent is just the empty gameobject parent for the rectangles created for area shading.

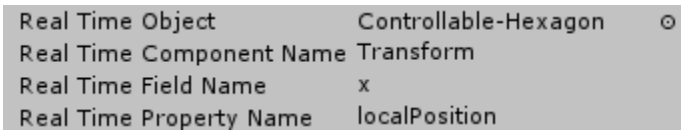
The Area Shading Prefab is what is instantiated for each area shading rectangle.

The Area Shading Color changes the color for all area shading rectangles for the series. The alpha value will have an effect for the gradient type shader.

The Area Shading Axis Value controls the minimum y-value (or x-value for horizontal orientation graphs), that is used for each area shading rectangle.

3.15 Real-Time Updating

There are several parameters on WMG_Series that allow the ability to do real-time data updating for the series.



- 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.

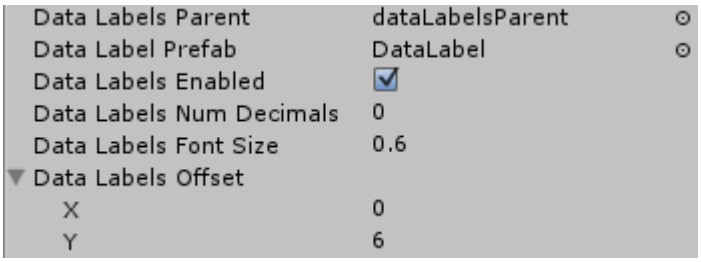
- Real Time Component Name
This is the string for the component to use. For example if graphing a transform's local position's x, then this would be "Transform". This can be any arbitrary script.

- 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".

3.16 Data Labels

There are several parameters on WMG_Series that allow the ability to add data labels for the series.



- Data Labels Parent
This is just a reference to the gameobject that is the parent for all data labels created.

- Data Label Prefab
This is the prefab used to create each data label. This could be changed if there is a need to for example change the font color or change the font.

- Data Labels Enabled
This determines whether or not data labels are created.

- Data Labels Num Decimals
If the data has decimals, then this determines how many decimals will display in the data labels.

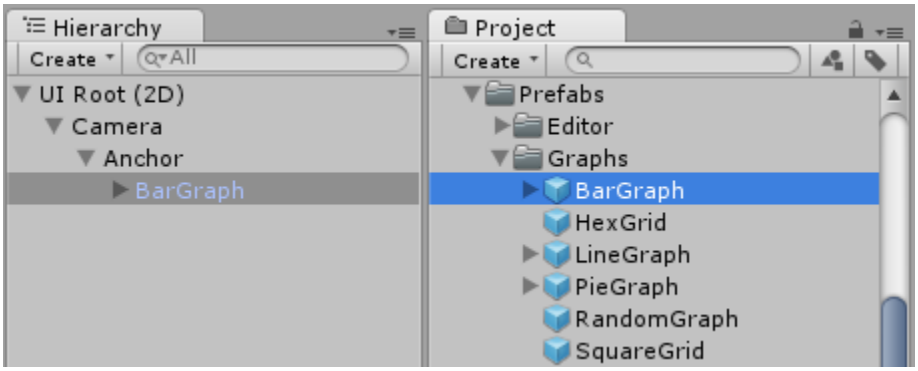
- Data Labels Font Size
Controls the font size for the data labels.

- Data Labels Offset
This can be used to further control the positioning of the data labels.

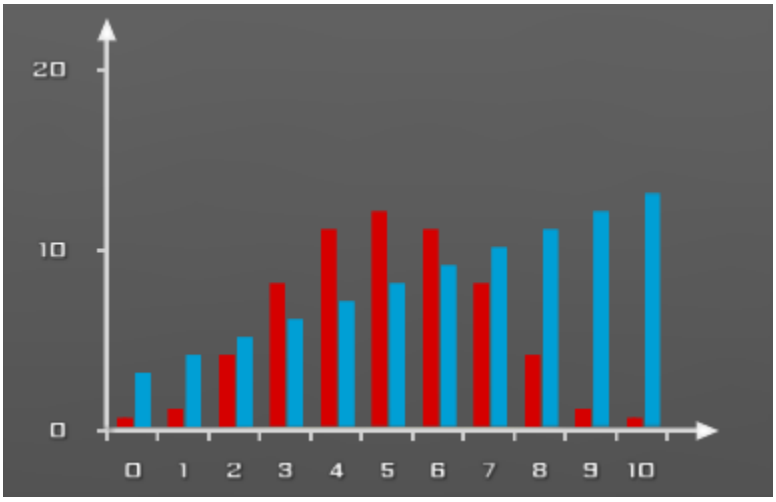
IV. Bar Graphs

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. Useful Functions

5.1 Axis Graph Functions

- public WMG_Series addSeries()
Use this function to create a new series on the graph. This will append it to the end of the list of series.

- public void deleteSeries()
Use this function to delete the last series on the graph.

- public List<WMG_Node> getYAxisLabels()
- public List<WMG_Node> getXAxisLabels()
These return the list of nodes which are the x/y-axis labels.

- public List<GameObject> getXAxisTicks()
- public List<GameObject> getYAxisTicks()
These return the list of gameobjects which are the x/y-axis ticks.

5.2 Series Functions

- public List<GameObject> getPoints()
Returns the list of gameobjects which represent the nodes of this series. Each gameobject will have WMG_Node script. Works the same for line graphs / bar graphs.

- public List<GameObject> getLines()
Returns the list of gameobjects which represent the links of this series. Each gameobject will have WMG_Link script. Works for line graphs.

- public Vector2 getNodeValue(WMG_Node aNode)
Returns the x and y data that corresponds with the given WMG_Node for this series.

- public GameObject getLegendParent()
Returns the gameobject that is the parent of the legend for this series.

- public void StartRealTimeUpdate()
If you have setup the graph to do real-time updating with the public variables related to real-time updating, then this will begin plotting data in real-time.

- public void StopRealTimeUpdate()
This will stop real-time update data plotting.

5.3 Graph Manager Functions

If you are interested in creating your own complex graphs, then you can use the graph manager as the basis for your own custom graph. All graph maker graphs inherit from the graph manager (even pie graphs and grids). So you can use this to create closed loop graphs, or any other more complex graphs.

- public GameObject CreateNode(Object prefabNode, GameObject parent)
This creates a node. The prefab just needs to have a WMG_Node script attached. If parent is NULL, then the gameobject that has the graph manager script is used as the parent.

- public GameObject CreateLink(WMG_Node fromNode, GameObject toNode, Object prefabLink, GameObject parent)
This creates a link between nodes. Note that both nodes need a WMG_Node script. The prefab link needs to have a WMG_Link script attached. If parent is null then the to node's parent is used.

- public void DeleteNode(WMG_Node theNode)
This deletes the given node gameobject, as well as all links associated with this node.

- public void DeleteLink(WMG_Link theLink)
This deletes the given link gameobject.

- public List<GameObject> NodesParent
This is the list of all the gameobject nodes in the graph.

- public List<GameObject> LinksParent
This is the list of all the gameobject links in the graph.

- public GameObject ReplaceNodeWithNewPrefab(WMG_Node theNode, Object prefabNode)
You can use this to dynamically replace all nodes in a graph with a different prefab node.

- public List<Vector2> GenLinear(int numPoints, float minX, float maxX, float a, float b)
- public List<Vector2> GenQuadratic(int numPoints, float minX, float maxX, float a, float b, float c)
- public List<Vector2> GenExponential(int numPoints, float minX, float maxX, float a, float b, float c)
- public List<Vector2> GenLogarithmic(int numPoints, float minX, float maxX, float a, float b, float c)
- public List<Vector2> GenCircular(int numPoints, float a, float b, float c)
- public List<Vector2> GenRandomXY(int numPoints, float minX, float maxX, float minY, float maxY)
- public List<Vector2> GenRandomY(int numPoints, float minX, float maxX, float minY, float maxY)
These functions are mainly useful for generating data used in line or bar graphs, however they may be useful in other graphs, so they are on the graph manager script.

- public List<WMG_Link> FindShortestPathBetweenNodes(WMG_Node fromNode, WMG_Node toNode)
Given two nodes return one or more shortest paths between the nodes based on the number of links. There can be multiple shortest paths in closed loop graphs or grids.

- public List<WMG_Link> FindShortestPathBetweenNodesWeighted(WMG_Node fromNode, WMG_Node toNode, bool includeRadii)
Given two nodes return one or more shortest paths between the nodes based on the link weights, and also node radii if include radii is true. Every WMG_Link has a weight variable which you can use to specify weights, and every WMG_Node has a radius value, which can also be used in the calculation of shortest paths.

5.4 Node Functions

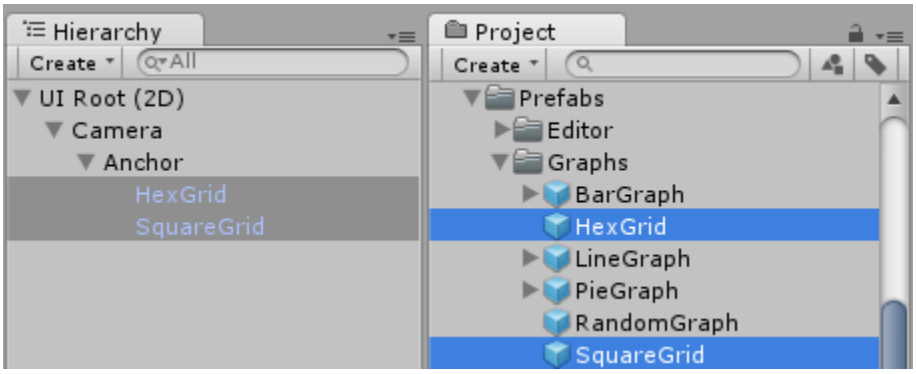
- public void Reposition (float x, float y)
Repositions the node to the newly specified local (NGUI) / relative (Daikon) position. This also repositions all associated links.

- public void RepositionRelativeToNode (WMG_Node fromNode, bool fixAngle, int degreeStep, float lengthStep)
Repositions the node relative to another node based on the degree and length steps. Refer to the WMG_Editor example scene in NGUI package / web-player demo posted on the first page of the Unity forum for an example of this in use. Hold control and / or shift while creating a node from another node.

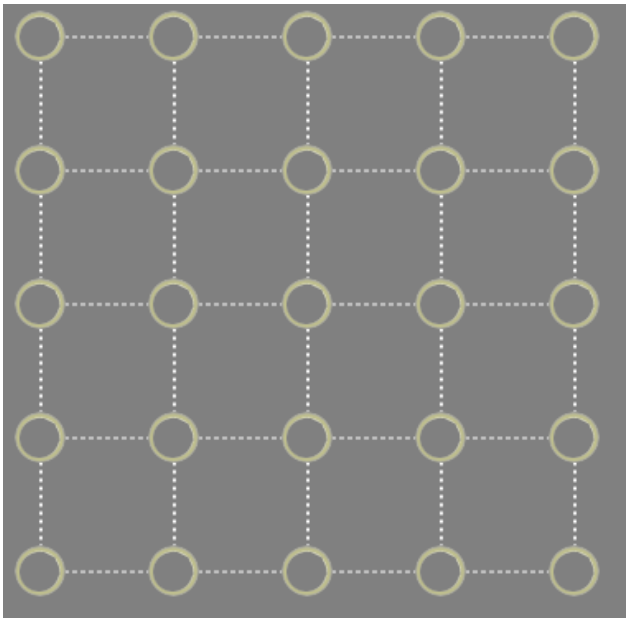
VI. Square / Rect / Hex Grids

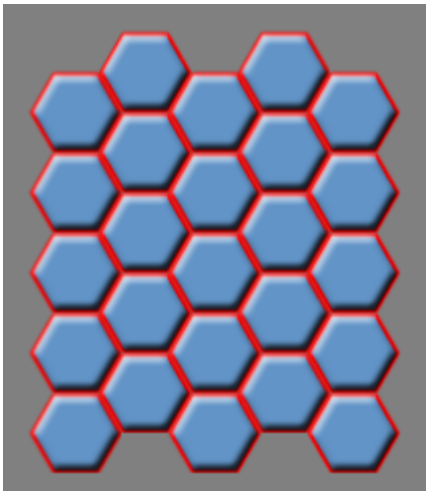
6.1 Getting Started

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



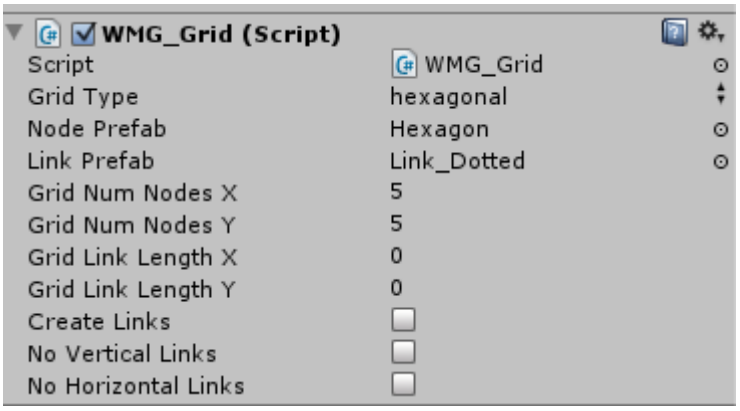
The grids will appear when you play the scene:



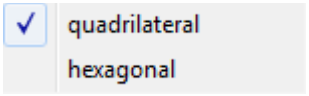


6.2 Parameters

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.

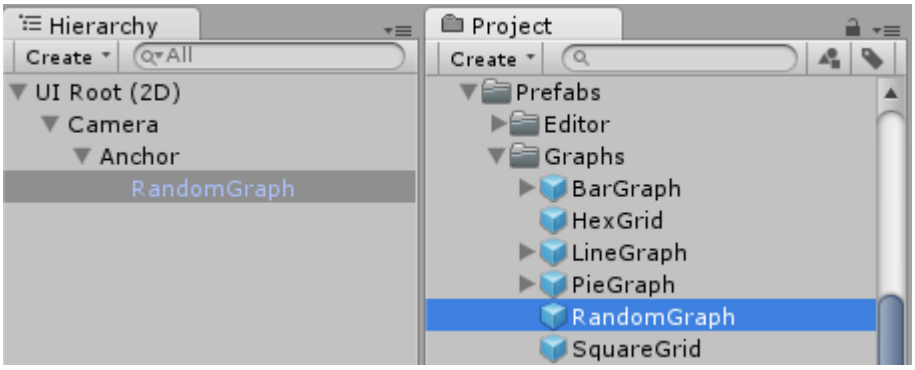
- No vertical / horizontal links

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.

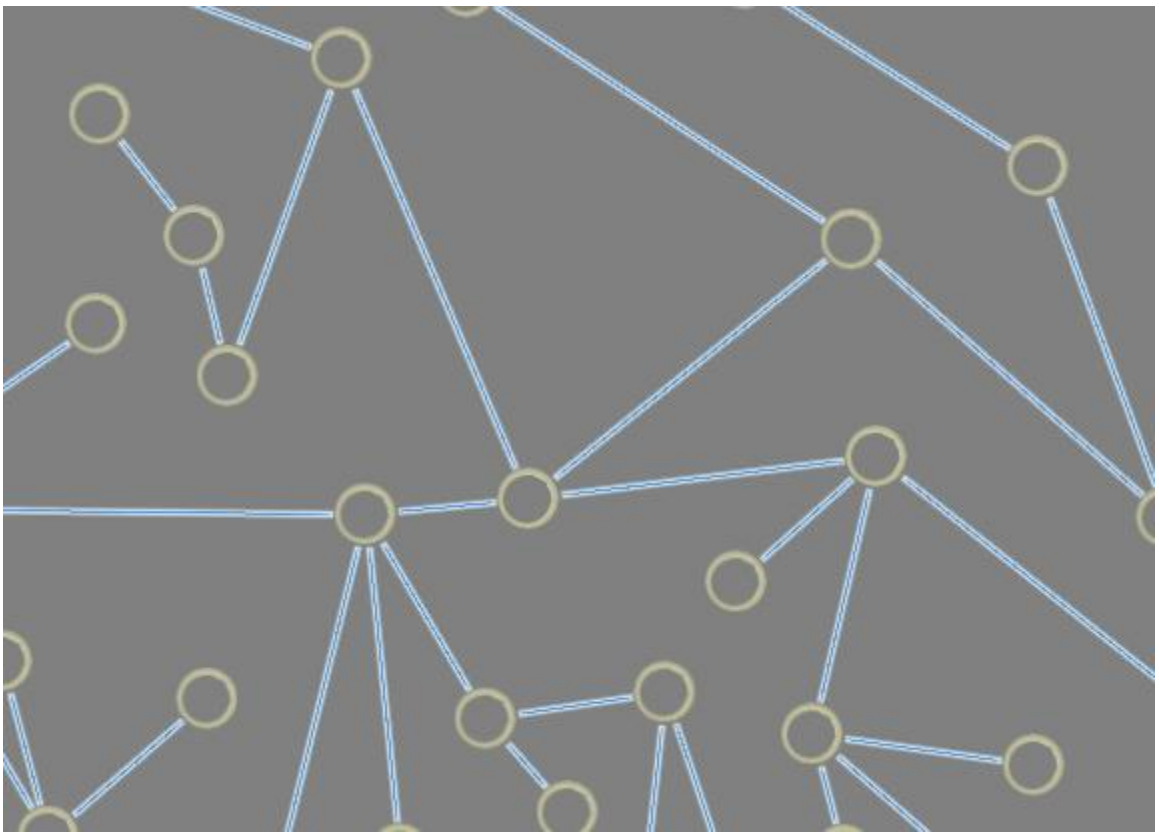
VII. Random Graphs

7.1 Getting Started

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

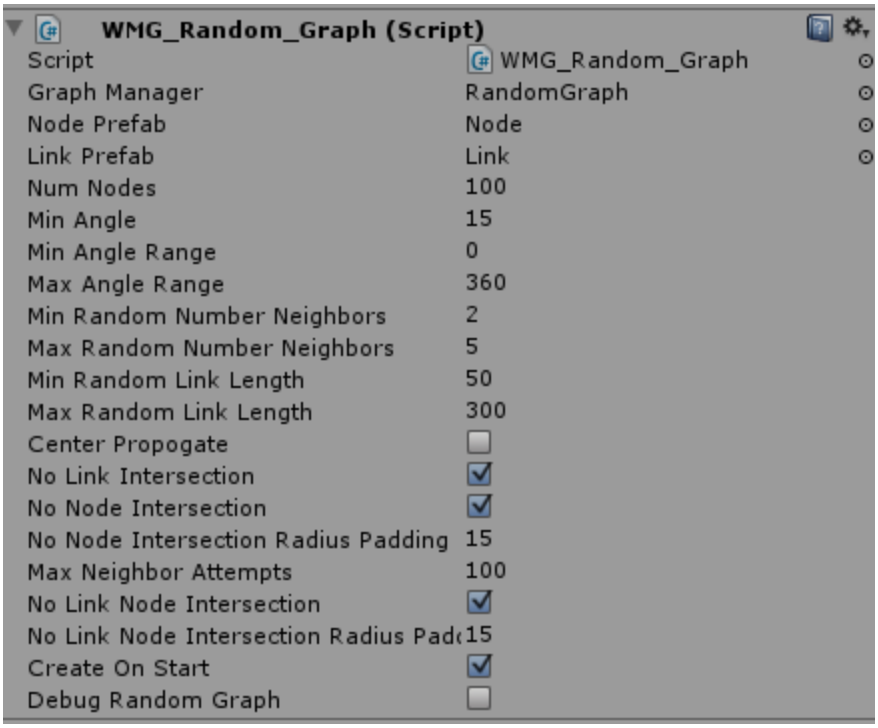


The graph will appear when you play the scene:



7.2 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().

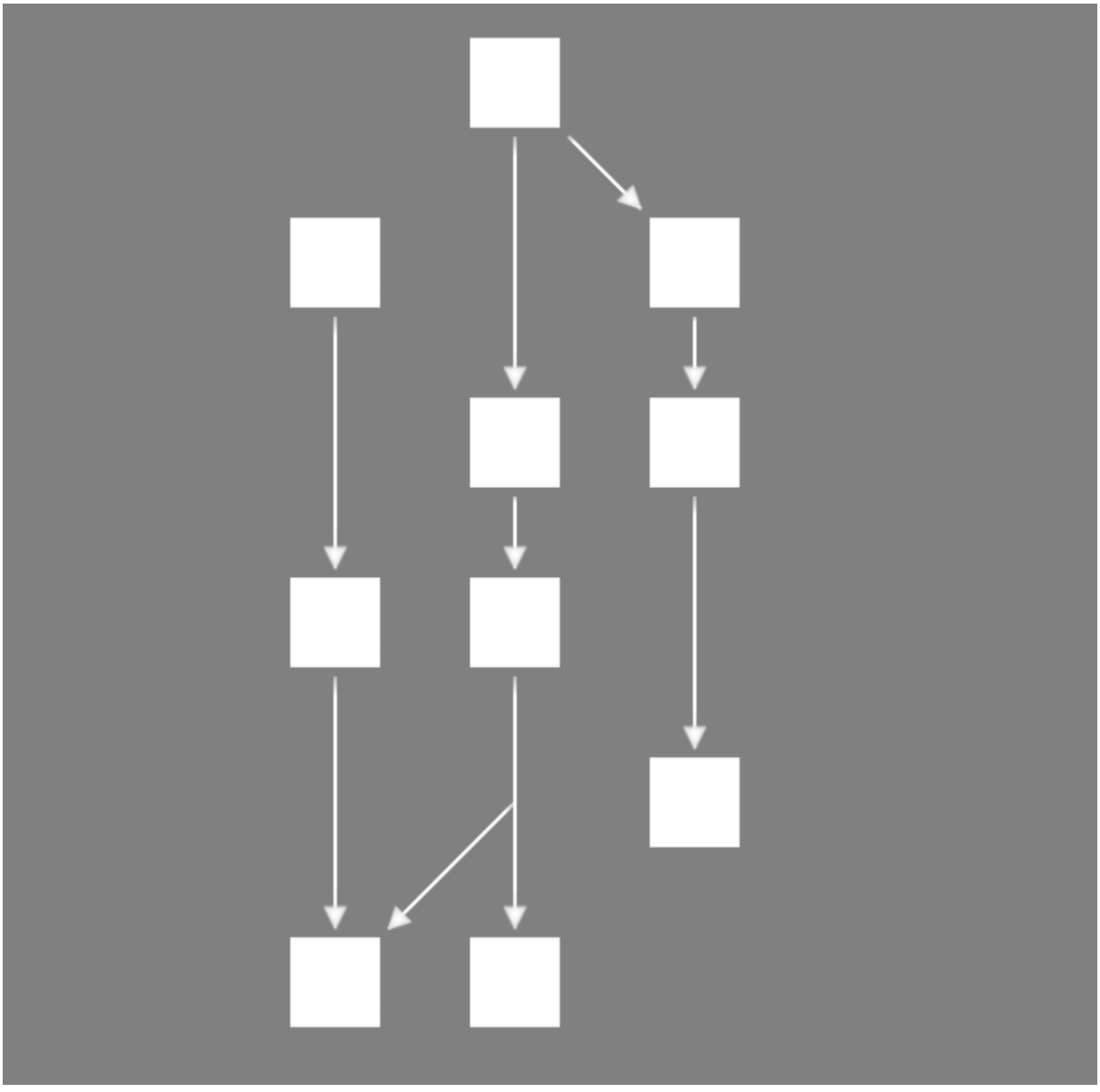
- Debug Random Graph
This can be useful to troubleshoot exactly what is happening during the random graph generation process. I resolved many bugs, and added new functionality using this parameter.

VIII. Hierarchical Skill Trees

8.1 Getting Started

Drag and drop the HierarchicalTree prefab into your scene:

The example tree will then appear when you play the scene:



8.2 Summary

Overall, the tree is a collection of nodes, "invisible nodes", and links. The "invisible nodes" are necessary to create links that do not appear to start from an actual node.

Node positions are defined by a column and a row position. Row height and column width are entirely configurable, so the columns and rows could be represented as a number of pixels.

The width / height, and radius of all nodes can also be set at the tree level. In this example the radius is set to a be a little more than half the width / height of the nodes so that the links appear to have a little bit of space instead of directly touching the node. You could also just set a radius of 0, to have the links go behind the nodes.

You can also set whether all the nodes represent circles or squares. Square is the default, and the effect is that the link end and start position will be based on a square edge instead of a circle edge. The radius for a square means half the width / height of a square, and for a circle, well it means the radius :)

Lastly, to replace the default white squares there is a prefab list. Each position in the prefab list corresponds to the node in the lists that define the node's position. You can also replace the default white square with your own custom default prefab by changing the default node prefab parameter.

8.3 Parameters

- Note Parent
This is the parent game object for all the nodes.

- Link Parent
This is the parent game object for all the links.

- Default Node Prefab
The default node prefab is a white square, which can be overridden by the node prefab list.

- Link Prefab
This is the prefab for all the links.

- Num Nodes
This is the number of nodes that will be in the tree (excluding invisible nodes).

- Num Links
This is the number of links that will be in the tree.

- Node Prefabs
These prefabs override the default node prefab. Each element in the list corresponds to the nodes in the column and row position lists.

- Node Columns
This is each node's column position.

- Node Rows
This is each node's row position.

- Link Node From IDs
This is the list of links and each link's from node. The ID here corresponds to the element in the list for the node columns / rows. For example node column / row element 0 is node ID 1. This list also applies for invisible nodes, however the node IDs will be denoted by a negative number (explained more in the invisible nodes section).

- Link Node To IDs
Same as above, except this is the end point for the link instead of the start point for the link.

- Invisible Node Prefab
This is just the prefab for the invisible nodes, and you will probably never need to change this.

- Num Invisible Nodes
This is the number of invisible nodes that will be in the tree. Invisible nodes are used to create links that do not necessarily have to start or end from an actual node. In the link To / From IDs list, invisible nodes are represented by negative numbers. For example -1 means the invisible node at element 0 for the invisible node columns / rows lists.

- Invisible Node Columns
This is each invisible node's column position.

- Invisible Node Rows
This is each invisible node's row position.

- Grid Length X
This is the column width. Essentially, this number multiplied by the column position determines the node's x position.

- Grid Length Y
This is the row height. Essentially, this number multiplied by the row position determines the node's y position.

- Node Width Height
This is the height and width of every node's sprite.

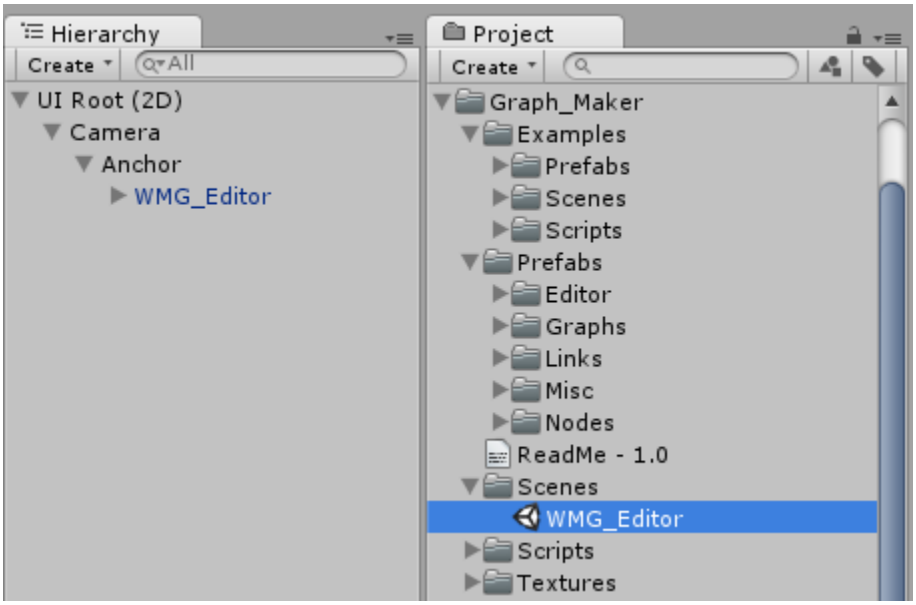
- Node Radius
This is the radius of all circle nodes or half the width / height of all square nodes. This determine the starting and ending points for the links.

- Square Nodes
This sets a boolean on all nodes to tell whether or not the node is represented as a square instead of a circle. Square nodes will have the effect of making the link start and end points be based on the square's edge.

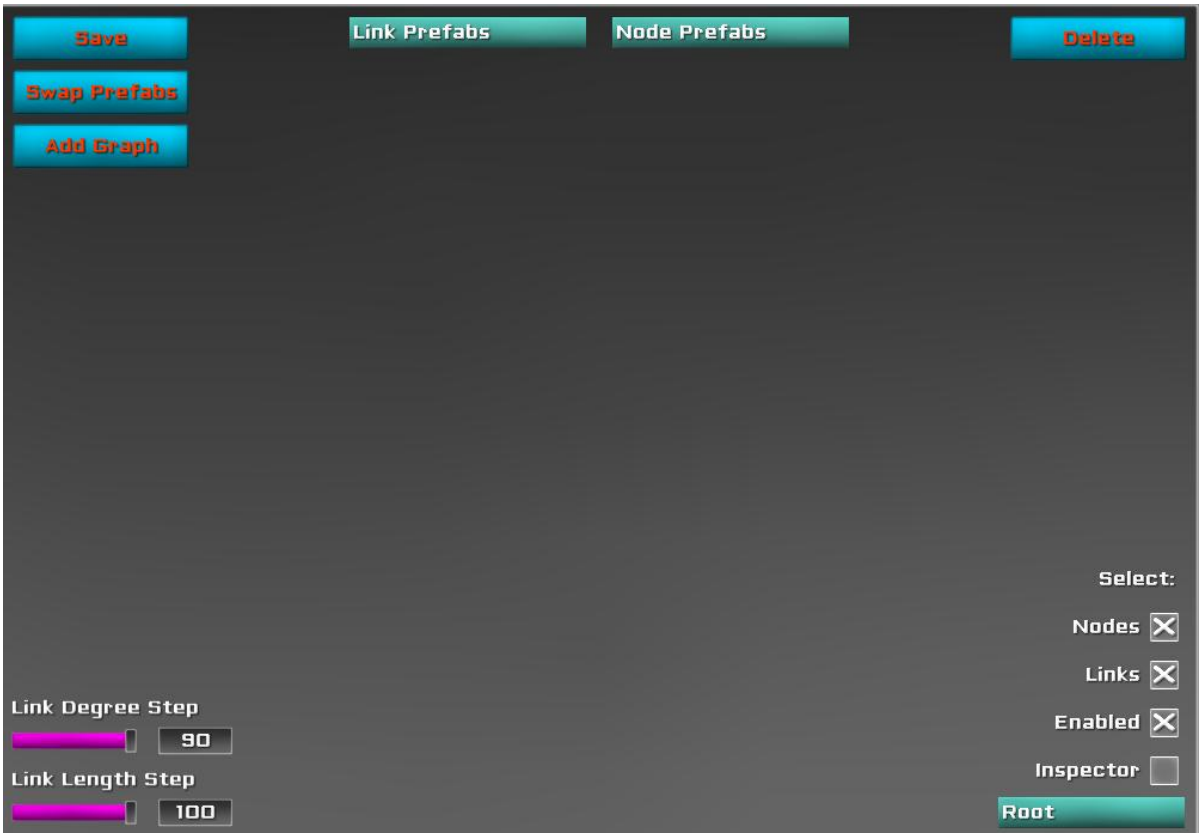
IX. Graph Editor (NGUI / Deprecated)

9.1 Getting Started

Double click the WMG_Editor scene in the Scenes folder:

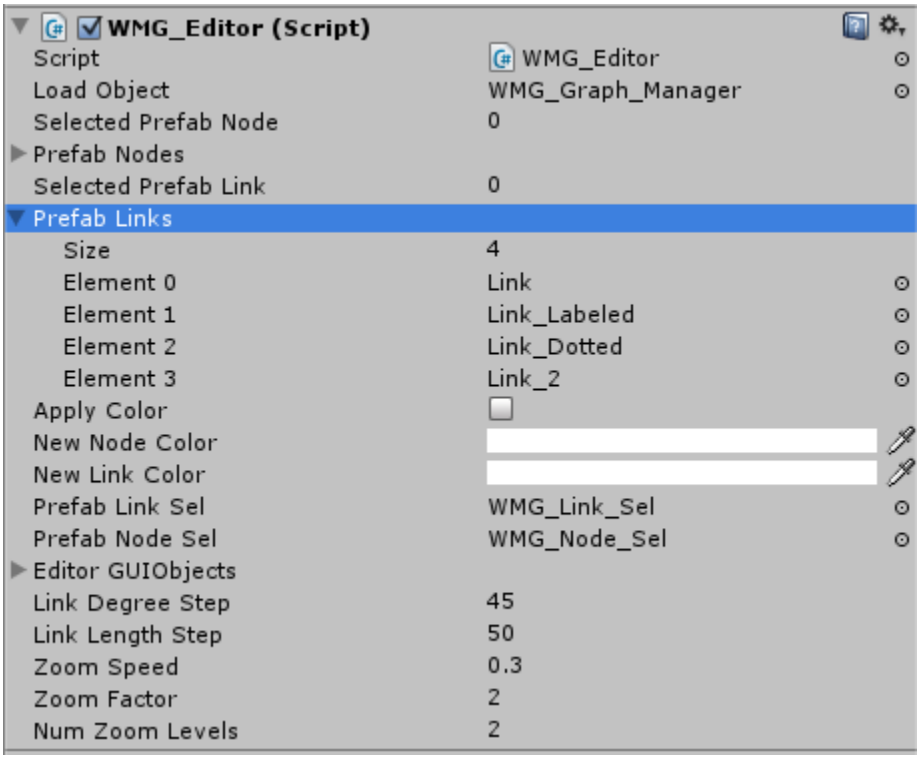


The editor GUI will appear:



9.2 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:

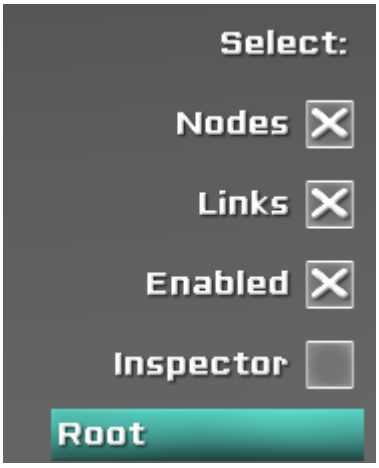
- Zoom speed / zoom factor / num zoom levels:
These can be configured to change how zooming works in the editor.

9.3 Editor Controls

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

9.4 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 check 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.

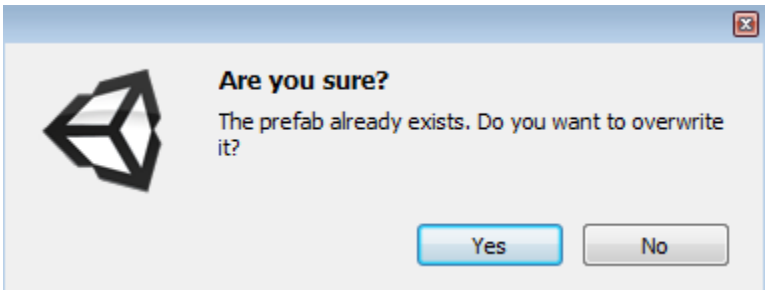
The pop-up list that says "Root" by default allows selecting other objects in the inspector when inspector is checked. For example, if your node prefab has an "Object to Label" reference you can change root to "Label Object" which will select that node's referenced label object. The default is root,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

9.5 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:

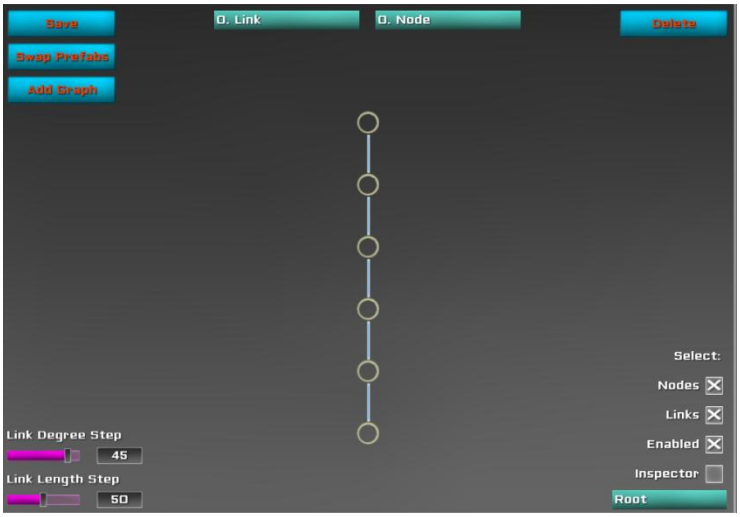


To load an object, stop the WMG_Editor scene and drag and drop the prefab that was created from the Save process into the "Load Object" variable. Now when you hit play, the editor will load your previously saved object.

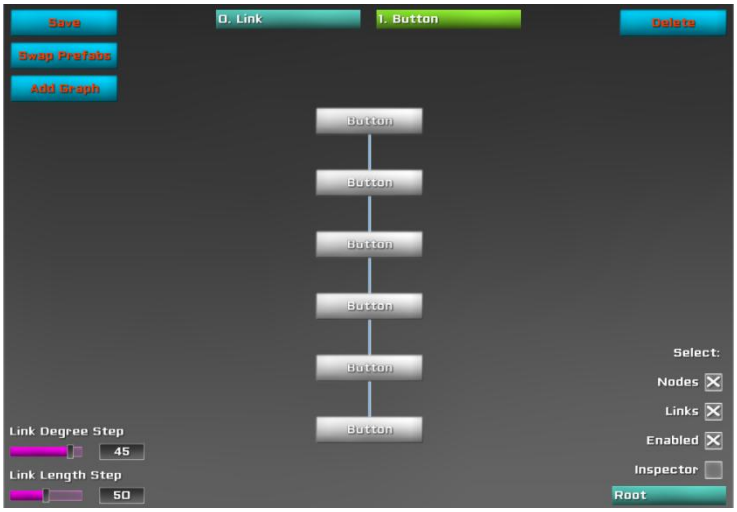
9.6 Prefab Swapping

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:

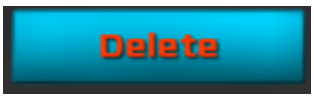


How it works-

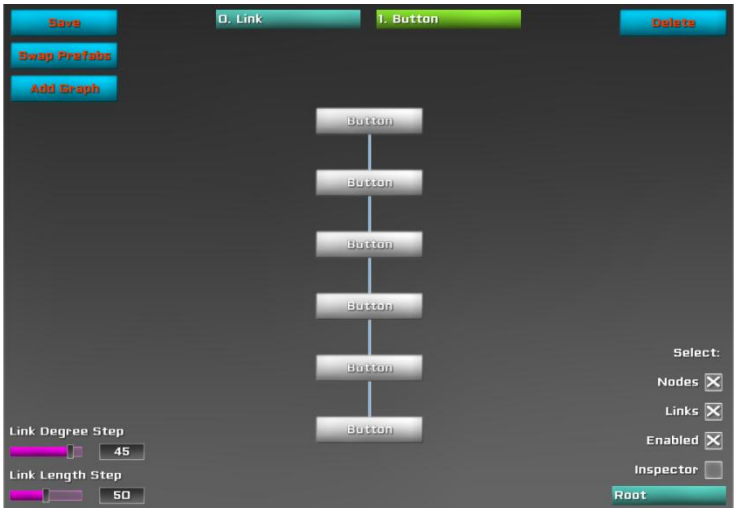
It destroys the existing selected objects / links and creates new objects and links with the same WMG_Node / WMG_Link data as the destroyed objects. The names of the gameobjects are also preserved. If you have custom scripts with custom data on your prefab those will not survive the process unless of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

9.7 Deleting

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



You can change this:



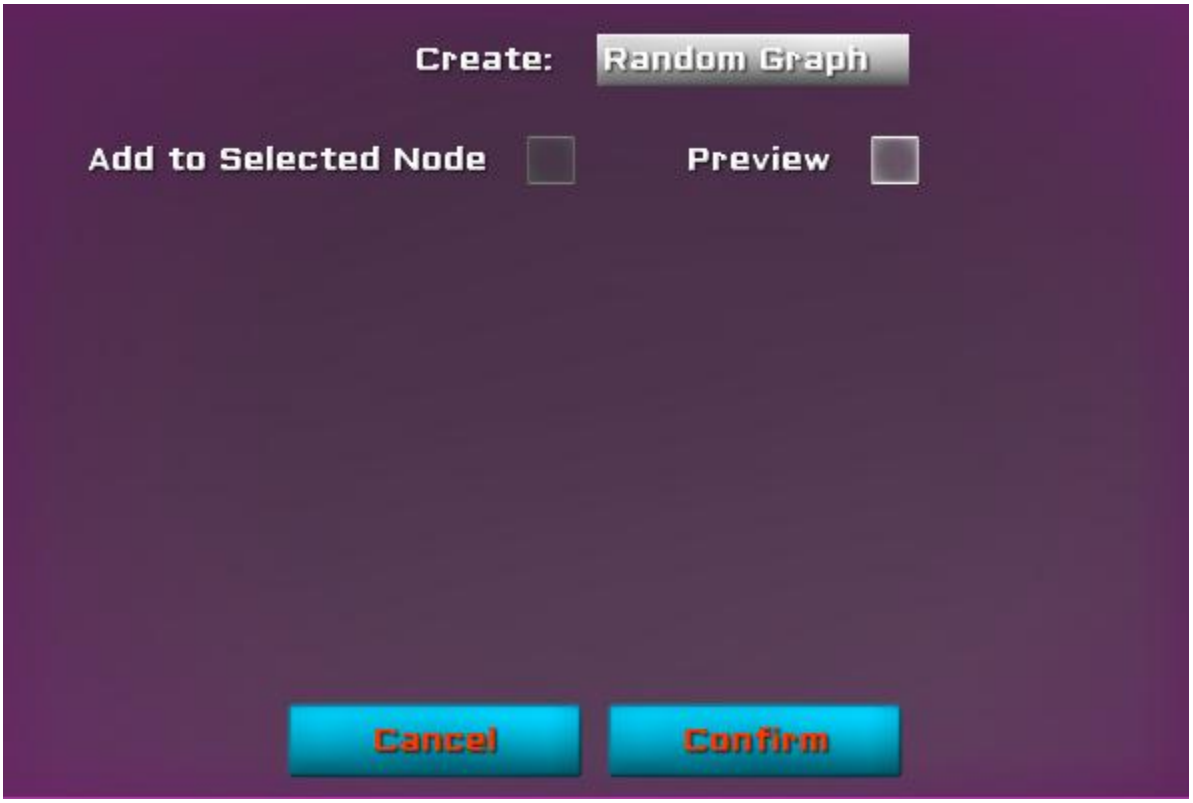
To this by simply selecting the links and clicking the delete button:

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

9.8 Adding Graphs

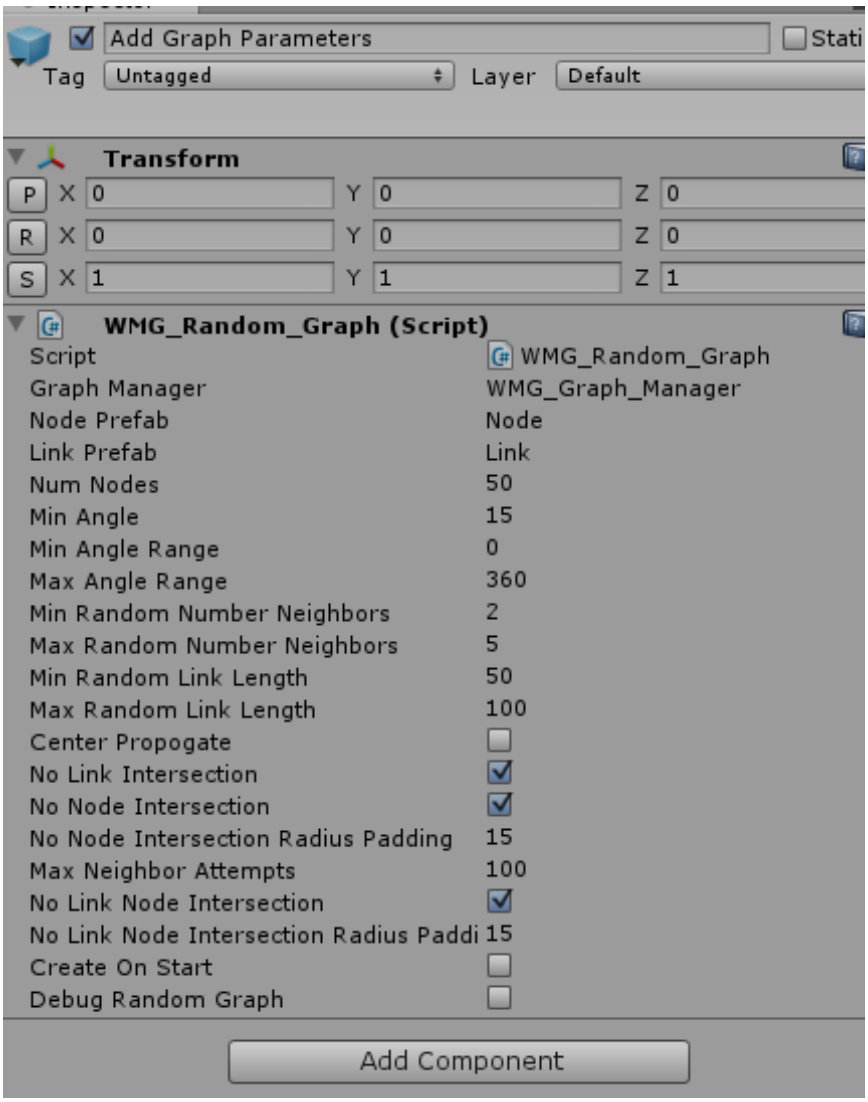
You can add random graphs and grids to be used in the editor with the Add Graph button:

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.