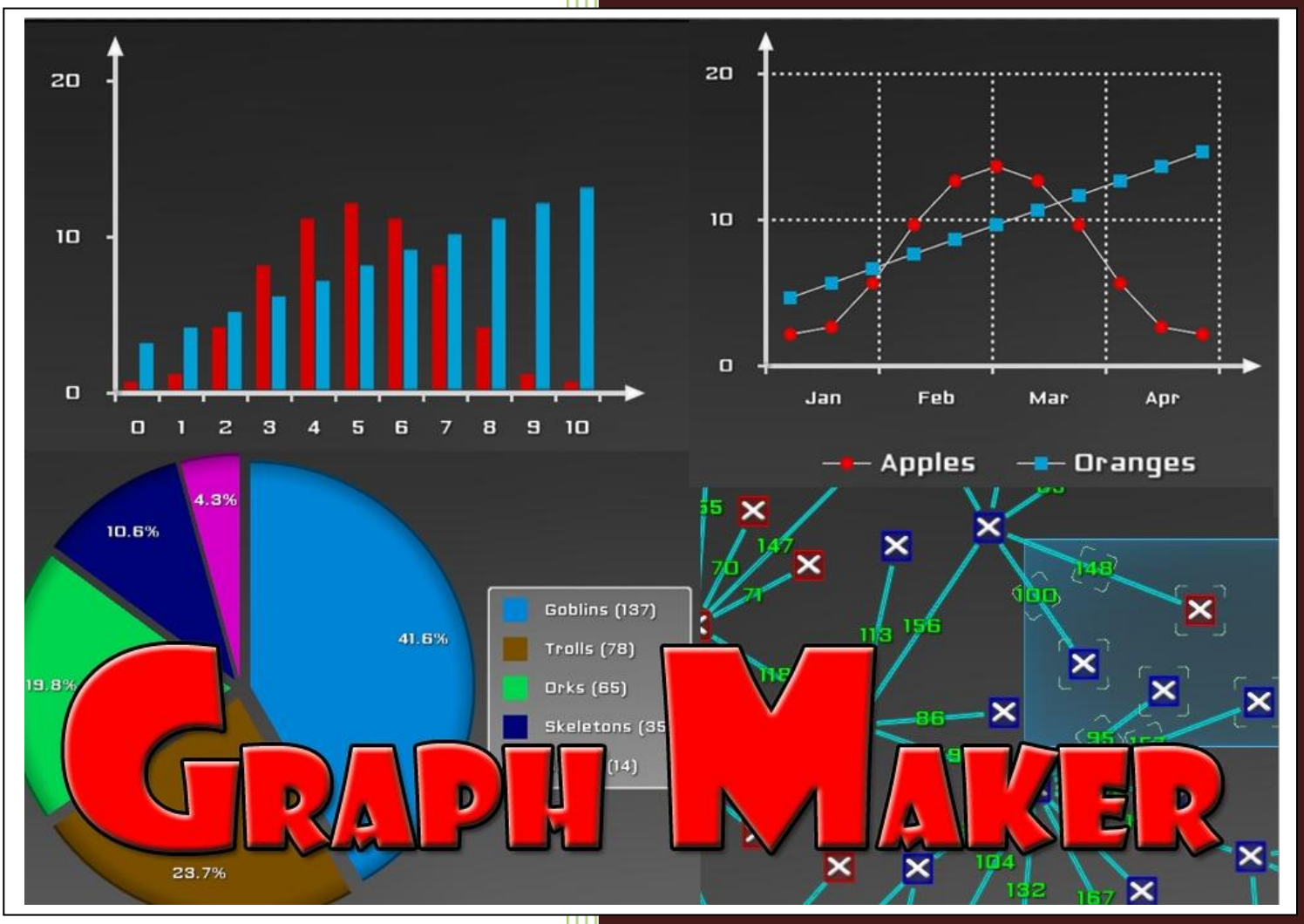
Graph Maker


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%
23.7%
Goblins (137)
Trolls (78)
Orks (65)
Skeletons (35
(14)
GRAPH MAKER

I. OVERVIEW 4

1.1 Goals 4
1.2 Features 4

II. PIE GRAPHS 5

2.1 Getting Started 5
2.2 Core Parameters 6
2.3 Other Slice Parameters 7
2.4 Animation Parameters 7
2.5 Label Parameters 8
2.6 Misc Parameters 8

III. LINE GRAPHS 9

3.1 Getting Started 9
3.2 Axis Graph Core Parameters 9
3.3 Axis Graph Axis Parameters 12
3.4 Axis Graph Tooltip Parameters 14
3.5 Axis Graph Animation Parameters 15
3.6 Axis Graph Misc Parameters 15
3.7 Series Parameters 16
3.8 Other Notes 19
3.9 Animations 19
3.10 Events 20
3.11 Dynamically Add / Delete Series 20
3.12 Area Shading 20
3.13 Real-Time Updating 21
3.14 Data Labels 22
3.15 Dynamically Resizing 23

IV. BAR GRAPHS 24

4.1 Getting Started 24
4.2 Axis / Series Graph Parameters 24

V. FUNCTIONS & MISC INFO 25

5.1 Axis Graph Functions 25
5.2 Series Functions 25
5.3 Graph Manager Functions 26
5.4 Node Functions 27
5.5 Populating Data Dynamically via Reflection 27
5.6 Legends 28
5.7 TextMesh Pro 30
5.8 Custom ToolTips and DataLabels 30

VI. SQUARE / RECT / HEX GRIDS 32

6.1 Getting Started 32
6.2 Parameters 33

VII. RANDOM GRAPHS 34

7.1 Getting Started 34
7.2 Parameters 34

VIII. HIERARCHICAL SKILL TREES 37

8.1 Getting Started 37
8.2 Summary 38
8.3 Parameters 38

IX. RADAR GRAPHS 41

9.1 Getting Started 41
9.2 Summary 41
9.3 Parameters 42

X. RING GRAPHS 44

10.1 Getting Started 44
10.2 Summary 45

10.3 Parameters **45**

XI. BEZIER BAND GRAPHS **48**

11.1 Getting Started **48**
11.2 Summary **49**
11.3 Parameters **49**

XII. GRAPH EDITOR (DEPRECATED) **51**

12.1 Getting Started **51**
12.2 Parameters **52**
12.3 Editor Controls **54**
12.4 Selections **54**
12.5 Saving and Loading **55**
12.6 Prefab Swapping **56**
12.7 Deleting **57**
12.8 Adding Graphs **58**

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. Common use cases of these types of graphs include skill trees, and world maps.

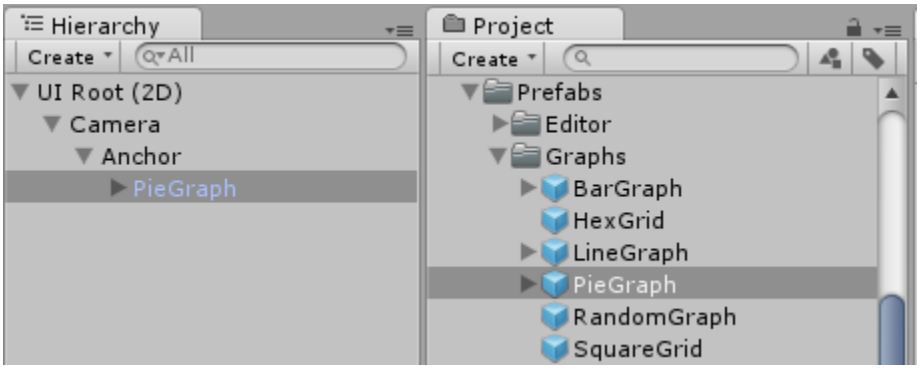
1.2 Features

- Pie Graphs
- Line / Bar Graphs
- Radar Graphs
- Ring Graphs
- Random Graphs
- Quadrilateral / Hexagonal Grids
- Hierarchical Trees
- Bezier Band Graphs
- Customize visual aspects at run-time

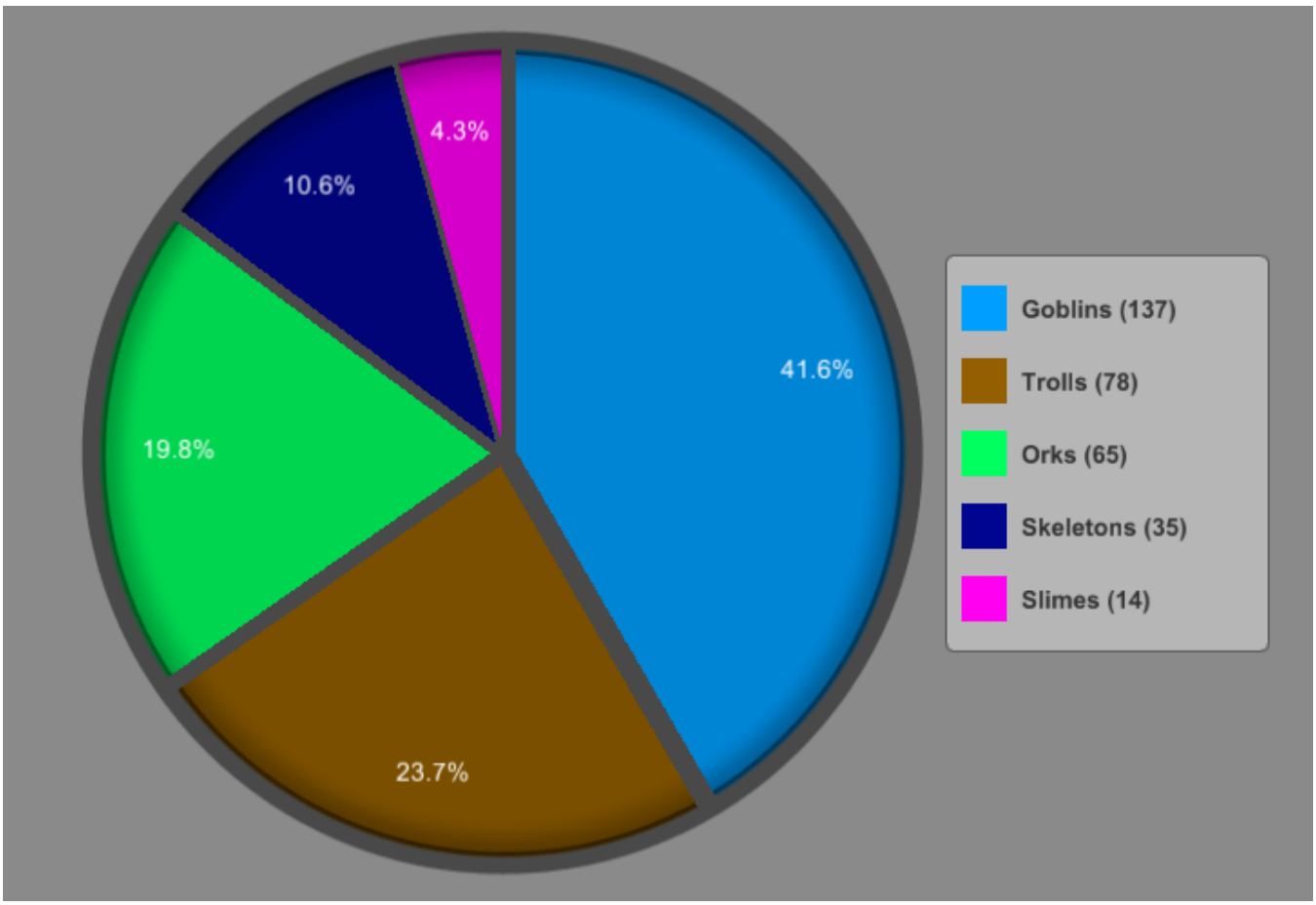
II. Pie Graphs

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 Core Parameters

- Resize Enabled / Resize Properties
When the rect transform width / height changes, this determines which pie graph elements resize as a result.

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

- 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.

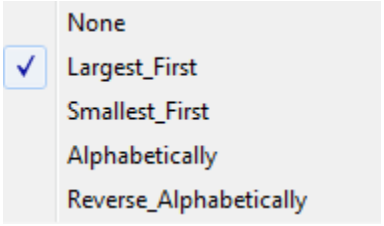
- Colors
This is the list of colors of the pie slices. This appears in the legend, and the pie slice itself. If the number of slice values is increased beyond what is specified then extra colors are automatically added but default to the white color. If the number of slice values is decreased beyond 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.

- Left / Right / Top / Bot Padding
Controls the padding of the background relative to the pie graph circle.

- Auto Center / Auto Center Min Padding
This ensure the pie graph and legend both stay in the center of its rect transform. This will automatically adjust the background padding as well, using the Min Padding value as the absolute minimum for the padding.

- Bg Circle Offset
This controls the size of the background circle (behind the pie graph). For example, if this is 10, then the background circle will exceed the borders of the pie graph slices by 10 pixels.

- 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 of the Values list appears in the top right corner.

- Swap Colors During Sort
When set to true, if the 3rd slice is green and the 3rd slice value gets increased so that it becomes the first slice after sorting, then the green color will swap to 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.

- Slice Label Type
Controls how the labels on the slices display.

- 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.

- Explode Symmetrical
When the explode length is > 0, this determines whether the outer edge of the slices align.

- Doughnut Radius
This cuts out a circle of the specified radius from each slice. This can be useful for making doughnut graphs.

- Hide Zero Value Legend Entry
When set to true and the value of a slice is 0, this will hide that legend entry (and re-arrange the other legend entries). When set to false, the legend entry will remain even though no slice will be visible.

2.3 Other Slice Parameters

- Limit number slices / max slices
When limit number slices is checked, the pie graph will limit the number of slices displayed to this specified value. So if there are 10 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.

2.4 Animation Parameters

- Animation Duration / Sort Animation Duration
These parameters are for doing animated updates instead of instant updates. 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.

2.5 Label Parameters

- Slice Label Explode Length
This controls where the labels that overlay pie slices appear. A value of 0 corresponds with the outer edge of the pie graph slices.

- Slice Label Font Size
Controls the font size of the pie slice labels.

- Number of decimals in percents
This controls the number of decimals displayed in the pie slice labels when a percent is displayed.

- Slice Label Color
The color of the slice labels

2.6 Misc Parameters

- Values / Labels / Colors Data Source
These reference a WMG_Data_Source script. This script can pull data dynamically via reflection to populate data used in the graph.

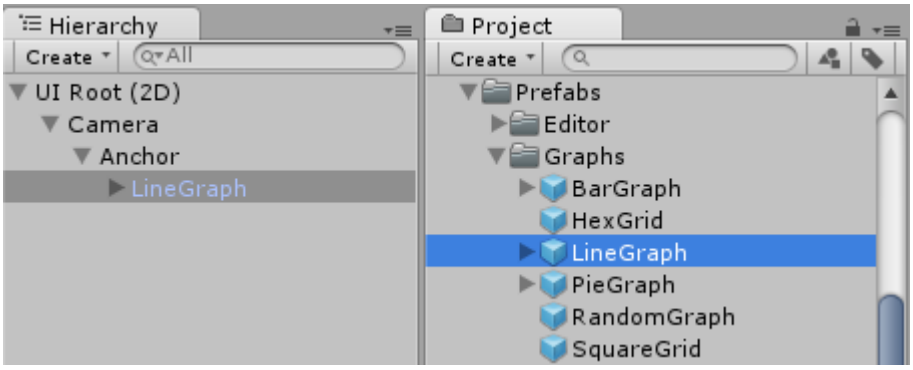
- Legend
A reference to the legend. Refer to the legend section for info about customizing the legend.

- Slice Prefab
The prefab used to generate the slices. If you want a different look you can change out the sprite used here.

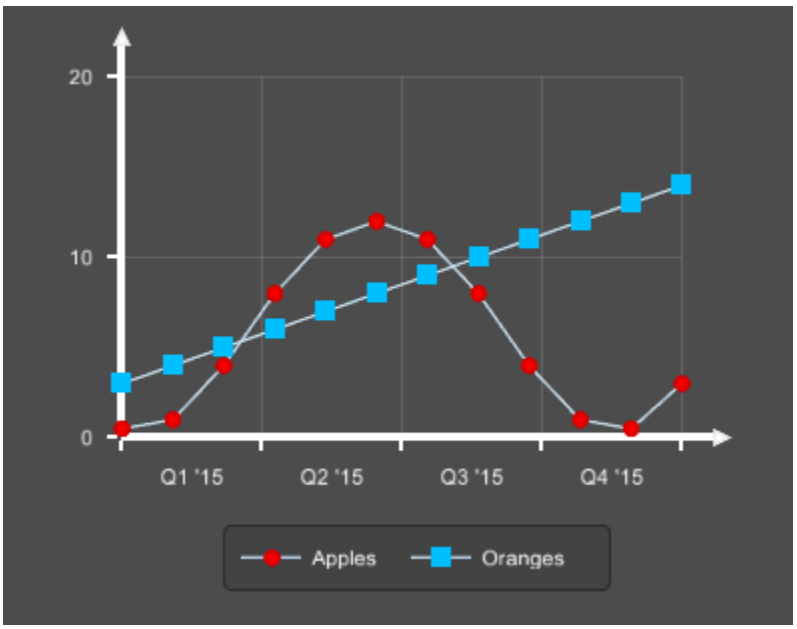
III. Line Graphs

3.1 Getting Started

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



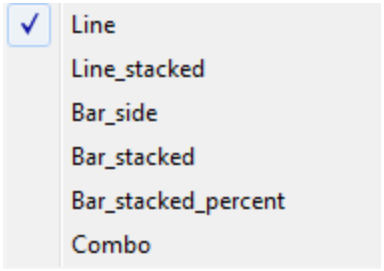
The graph will appear when you play the scene:



3.2 Axis Graph Core Parameters

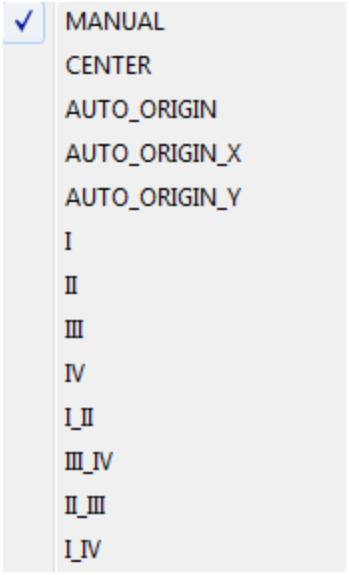
-Graph Type

Changes the graph type with one of the following options:



-Orientation Type
Change the graph from vertical to horizontal. This is mostly useful for creating 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.
To get axes to not stick to a particular tick set the "Axis Use Non Tick Percent" to true.

- Resize Enabled / Resize Properties
Refer to the dynamic resize section.

- Use Groups / Groups
This option should be used if it is important to graph nulls (e.g. broken line segments), and / or to have axis labels aligned with groups of bars or line points.
The list of groups is defined by the List<string> groups variable. The number of groups defined here controls how many Vector2 values must be present for every series. The x value in each Vector2 corresponds with the index of the group in the List<string> groups.
A function is run whenever the Vector2 list changes for a series that sanitizes the data when grouping is enabled. This will, for example automatically insert nulls as necessary, and also combine duplicate groups. Note that nulls are defined by a negative x value in each Vector2.

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

- 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, which should be the same as the width and height of the root gameobject. If, for example, the legend is on the bottom, then you will likely want a larger bottom padding for the legend, unless you want the legend to not be on the background.

- 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.

- 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.

- Auto Update Origin
This automatically sets the origin based on the axes type. For example, if axes type is quadrant I, and the min X axis value is -100, and min Y value is 50, then the origin will be (-100, 50).

- Auto Update Bar Width
This updates the bar width when orientation changes based on the ratio of x and y axis lengths. For example if the x axis length is twice the y axis length and the orientation is changed to horizontal, then the bar width will be divided by 2. This also ensures bars don't overlap when dynamically adding series by reducing the bar width as needed.

- Auto Update Bar Width Spacing
When auto update bar width is enabled, this automatically updates the bar width to be based on a certain percentage of the graph's axis length. This ensures the total amount of space not occupied by bars is equal to this percent. For example at 0.3, 30% of the space occupied by bars is empty space.

- Auto Update Series Axis Spacing
This automatically updates each series "Extra X Space" which is just a padding of space from the axis. For example for line charts this will set it the extraXSpace to 0, so that the points stay on top of the axes, and for bar graphs it is the amount of space between bars, so that the bars are evenly spaced from each other and from the axes.

- Auto Update Bar Axis Value
This automatically sets the bar axis value to the origin x or y component depending on the graph's orientation.

- Auto Fit Labels / Auto Fit Padding
This is a work in progress feature, that will ensure labels do not cross the graph background border.

3.3 Axis Graph Axis Parameters

- Axis Max and Min Values
This determines where each point / bar in a series gets positioned.

- 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.

- Min / Max Auto Grow
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 in the Misc parameters.

- Min / Max Auto Shrink
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 in the Misc parameters. Also, the threshold at which the decrease happens is specified in another parameter in the Misc parameters.

- 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.

-Hide Grids
This determines whether grid lines for this axis appear.

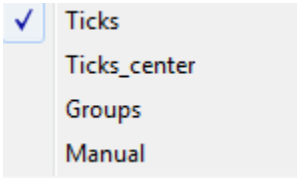
-Hide Ticks
This determines whether tick marks for this axis appear.

- Axis title string / offset / font size
The string displayed for x / y axis titles. The offsets control the position in relation to the axes. The font size controls the font size of the axis titles.

- Axis Use Non Tick Percent
This positions the axes based on percentages rather than on fixed grid ticks. This should be enabled for the auto origin axes types or the manual axis type. For auto origin axes type, this will move the axes around freely based on the origin value.

Label Parameters

- Label Type
Determines how many labels there are, and how they are positioned.



Ticks - There is exactly 1 label per axis tick, and the label is positioned next to the tick.
Ticks_center - There is exactly N - 1 labels where N is the number of ticks. Each label is centered between 2 ticks.
Groups - The number of labels matches the number of groups. The labels are positioned next to the points / bars in this mode. Also, the labels pull from the List<string> groups variable.
Manual - The number of labels is based off the X/Y labels list, and positioning is based off the label spacing and dist between variables.

- Labels
This is the actual list of axis labels. This is usually set automatically, or can be set manually in Manual Label Type mode or manually in Ticks or Ticks_center Label Type mode when "Set Labels Using Max Min" is disabled.

- Axis Label Rotation
Rotates the labels with the specified degrees.

- Set 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, different labels like non-numeric labels are desired, then this should be false, and the labels set manually.

- Axis Label Size
The font size of the axis labels.

- Num Decimals Axis labels
This controls the number of decimals displayed in the axis labels.

-Hide Labels
This determines whether x/y axis labels appear.

- Axis Label Space Offset
These control the distance the labels are offset from the axes. For example, for y-axis labels this is the amount of space in the x direction the labels are offset from the y-axis.

Manual Axes Type Parameters

- Axis Non Tick Percentage
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.

- Axis Arrows
These control which, if any arrows display on the axes.

-Axis ticks right 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.

- Axis 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 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.

Manual Label Type Parameters

- Axis Label Spacing
These control the distances labels are offset from the axes lines along the same direction as the axis. This is generally set automatically based on the X/Y label type, but could be set manually for the manual axis label type.

- Axis Label Dist Between
This is the distance between each label. This is generally set automatically based on the Label type, but could be set manually for the manual axis label type.

3.4 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.

3.5 Axis Graph Animation Parameters

- 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.

- 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.

- Auto Animations Easetype
The easetype for auto animations.

- Auto Animations Duration
The duration for auto animations.

3.6 Axis Graph Misc Parameters

-Axis Width
This determines the width of both the x and y axes.

- Auto shrink at percent
Only used when Min / Max Auto Shrink on an axis is enabled. 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
Only used when Min / Max Auto Shrink or Grow on an axis is enabled. 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.

- 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.

- Tick size
The width and height of the axis ticks.

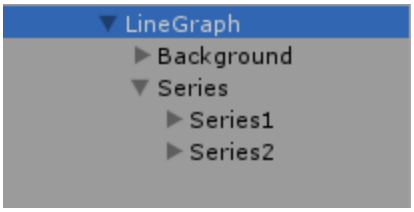
- Graph title string / offset
The string displayed for the graph title. The offsets control the position in relation to the top of the graph.

- Y / X Axis
A reference to the Axes. Refer to the Axis Parameters section for info about customizing the axes.

- Legend
A reference to the legend. Refer to the legend section for info about customizing the legend.

3.7 Series Parameters

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:



Core Parameters

- Point Values
This is the list of Vector2 float data used for this series. This data controls how the points are positioned. In certain cases the x-value of this data is completely ignored.

- Combo Type
Applies when the graph's Graph Type is set to Combo. Determines whether the series displays as a line or as a set of bars.

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

- Point Width Height
For line graphs, this is the width and height of each point sprite.

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

- Point Color
This is the colors of all the points / bars, unless use point colors is enabled and point colors are specified.

- Use Point Colors / Point Colors
This can be enabled and individual colors can be specified to colorize individual points / bars.

- Line Color
The color of the lines for line graphs.

- Use X Dist Between to Space / Auto Update X Dist Between / X Dist Between Points
These parameters can be used to automatically space data for the x-axis (or y-axis depending on the graph orientation). If Use X Dist Between To Space is enabled, it is not necessary to set series' x data to be anything meaningful.
X Dist Between Points controls the spacing between points / bars. If "auto update x Dist Between" is true, then "X Dist Between Points" auto updates based on the number of points that exist in the series and the length of the axis.

- Extra X Space
This adds space between the series and the axis (or shifts the entire series by this amount). This can be used to add extra space between the series and the axis. It can also be useful to add more spacing between bars for side-by-side bar charts. Note that this is set automatically if the "Auto Update Series Axis Spacing" is enabled on the graph.

- 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.

- Line Padding
This is the amount of space between a line ending and the middle of a point. This can be used for a different look for line graphs.

Label Parameters

Refer to the separate section about data labels.

Shading Parameters

Refer to the separate section about area shading.

Misc 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.

- Real-Time / Point Values Data Source
These reference a WMG_Data_Source script. This script can pull data dynamically via reflection to populate data used in the graph.

- 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.

- Legend Entry Prefab
The prefab used to create the legend entry for this series.

3.8 Other Notes

Note that the grid lines, axis ticks, and axis labels are all implementations of WMG_Grid. Grid parameters are explained in a later section.

Horizontal grid is GridLinesX.
Vertical grid is GridLinesY.

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

3.9 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.

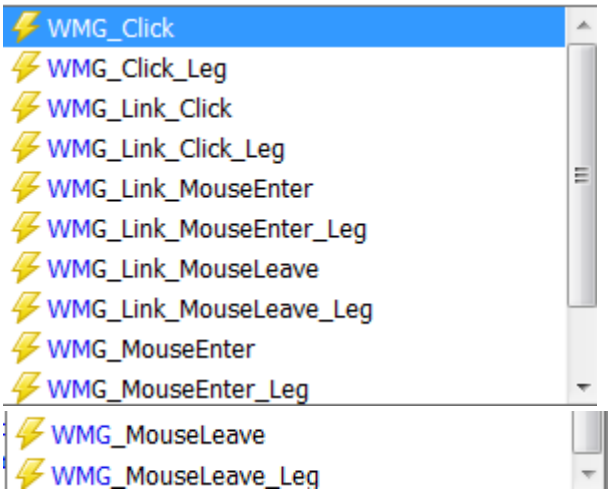
- animScaleBySeries
This animates each series consecutively, one after the other.

- 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

3.10 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):



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.

3.11 Dynamically Add / Delete Series

The graph script contains functions to add and delete series:

public WMG_Series addSeries()

public void deleteSeries()

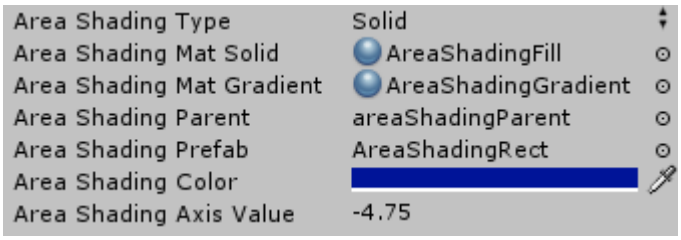
Simple call these functions and a series is added onto the end or deleted from the end.

There are also functions to add / delete a series at a specified indexed position.

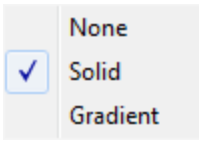
3.12 Area Shading

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.

3.13 Real-Time Updating

Setting up real-time updating from an arbitrary variable is pretty easy. Simply drag and drop a WMG_Data_Source script to any object in the scene. Drag and drop this component to the Real Time Data Source on the WMG_Series component.

Refer to the Populating Data Dynamically via Reflection section for info on how to setup the WMG_Data_Source component.

Then use the following function on the WMG_Series to start and stop plotting data.

- public void StartRealTimeUpdate()

- public void StopRealTimeUpdate()

3.14 Data Labels

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

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

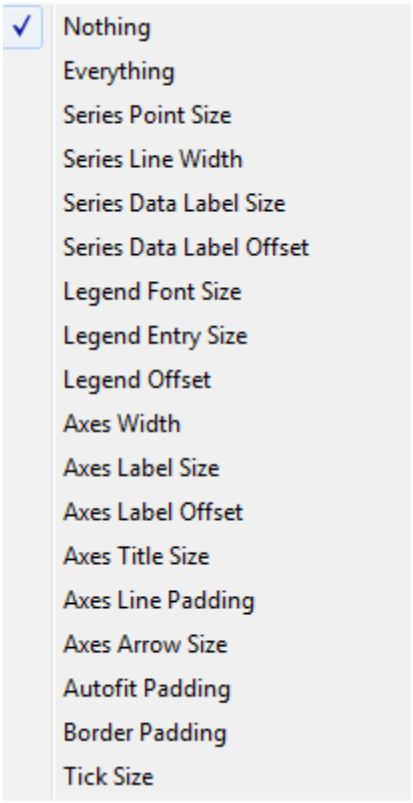
- Data Labels Offset
This can be used to further control the positioning of the data labels.

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

3.15 Dynamically Resizing

There are several options on WMG_Axis_Graph that allow the ability to dynamically resize content in a graph based on the container dimensions of the graph. This means that when this functionality is enabled, changing the width or height of a RectTransform (UGUI), UIWidget (NGUI), or dfControl (DFGUI) will automatically resize the graph as well as its content.

Resize Properties

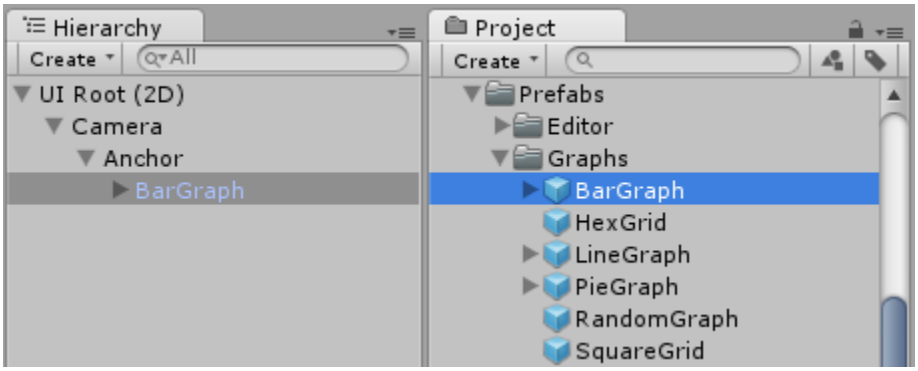


The resize properties control what resizes when a graph's width / height changes.

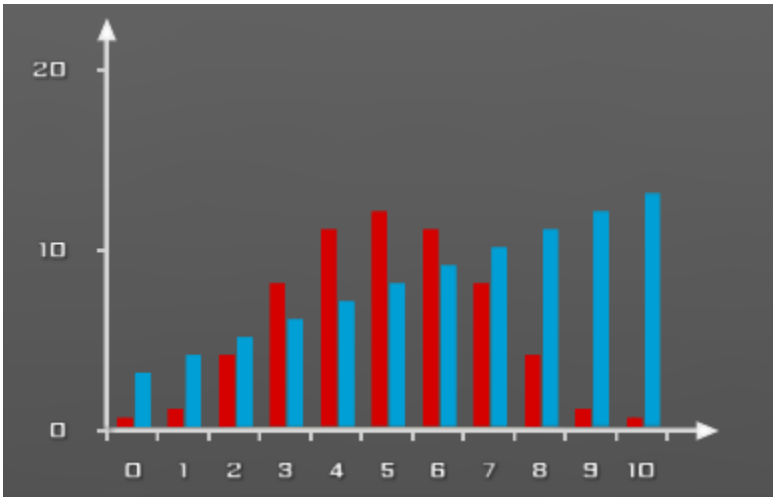
IV. Bar Graphs

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. Simply use the graph type parameter to change between line and bar graphs.

V. Functions & Misc Info

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.

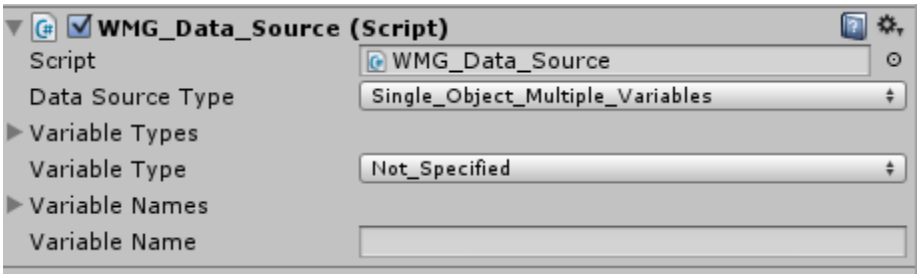
5.4 Node Functions

- 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.

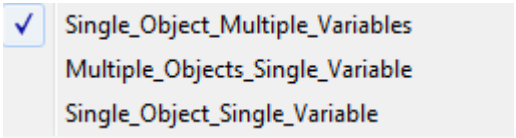
5.5 Populating Data Dynamically via Reflection

Use the WMG_Data_Source component to populate data dynamically via reflection



Any Graph Maker script that has a "Data Source" reference will automatically populate data based on the referenced WMG_Data_Source component.

There are 3 types of data sources:



There are also public functions that set data provider

public void setDataProviders<T> (List<T> dataProviderList)
public void setDataProvider<T> (T dataProvider)
public void addDataProviderToList<T> (T dataProvider)
public bool removeDataProviderFromList<T> (T dataProvider)

Let's say you want to populate the List<float> for a WMG_Pie_Graph using this component.

For single object multiple variables, and single object single variable, you must set the data provider using setDataProvider().

For multiple objects single variable, you must set the data providers using the setDataProviders / add / remove data provider functions.

Lastly, specify the variable names that will be used from those objects either through the editor, or also via script.

You can optionally set the variable type to improve performance. If no variable type is specified, the code will search for a field, then a property, then a field of a property, and then a field of a field.

You can specify a field of a property, for example, you can specify a variable name of "localPosition.x"

Populating data via Play Maker variables

Note that playmaker doesn't allow calling a function with a generic (T) argument. To pull in data via Play Maker variables, open the WMG_Data_Source script and uncomment the top portion of the script by deleting the 2 lines that say this:

/* // DELETE THIS LINE FOR USE WITH PLAYMAKER

Ensure that the data source type is "Multiple_Objects_Single_Variable". Set the variable name to "Value" (this is the name of the variable where the data is stored for all PlayMaker FSM variables). Then call the following functions on WMG_Data_Source in PlayMaker:

addPlaymakerVar(PlaymakerFSM, string)

removePlaymakerVar(PlaymakerFSM, string)

The first parameter is the PlaymakerFSM object, and the second parameter is the string name of the Playmaker variable.

5.6 Legends

WMG_Pie_Graph and WMG_Axis_Graph reference a WMG_Legend. In order to customize the legend appearance, first find the legend in the hierarchy:

Core Parameters

- Hide Legend
Controls whether the legend is displayed.

- Legend Type
This controls where the legend is positioned (bottom or right), as well as the default arrangement of the legend entries. Bottom arranges the elements horizontally, but vertically for right.

- Show Background
Controls whether the legend background is displayed.

- Opposite side legend
This positions the legend on the opposite side than normal / defined by the legend type. For example for a right legend, the legend will be placed on the left side if this is enabled.

- Offset
Controls how far the legend is offset from the graph.

- Set Width From Labels
When enabled, the legend entry width is automatically set based on the font size and the largest amount of text in the legend entries.

- Legend Entry Width / Height
The width / height of every legend entry.

- 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.

- 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.

- Pie Swatch Size
The size of the swatches for pie graph legends

- Background Padding
The number of pixels of the border of the legend relative to the entries.

- Autofit Enabled
Changes the number of rows or columns to best fit the graph's width / height.

Label Parameters

- Label Type
Controls how the labels appears for the legend entries.

- Num Decimals
The number of decimals for the legend text entries.

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

- Label Color
The color of the legend entry labels.

Misc Parameters

References to legend objects.

5.7 TextMesh Pro

You can use TextMesh Pro instead of UGUI for all text objects with these steps:
1. Import Graph_Maker/TMP/UGUItoTMP.unitypackage
2. Click menu option Assets/Graph Maker/UGUI -> TMP Prefabs
3. Change the code in WMG_GUI_Functions.cs to inherit from WMG_TMP_Text_Functions instead of WMG_Text_Functions

5.8 Custom ToolTips and DataLabels

You can customize tooltips by creating your own custom function with the following signature:

string myCustomFunction (WMG_Series series, WMG_Node node) {}

Your function just needs to return the string that displays for a given node that is hovered over. To get the x and y values corresponding to the node you can use series.getNodeValue(node)

To use your function just set the publicly exposed delegate like so (where graph is a WMG_Axis_Graph):
graph.theTooltip. tooltipLabeler = myCustomFunction;

Similarly for data labels (the labels that appear over individual bars or points if you have them enabled can also be customized). To set the delegate for this, you need a reference to the series (it is series specific) like so:

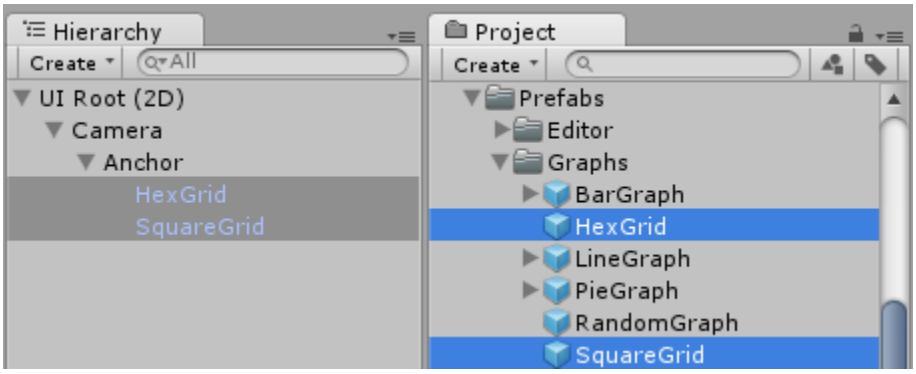
series. seriesDataLabeler = myCustomFunction;
where myCustomFunction has the signature:
string myCustomFunction(WMG_Series series, float val);

Note that the default function that graph maker uses for labeling is set in the WMG_Axis_Graph.Init() which is called automatically in Start(). But, if you instantiate a graph and set your function in your own custom script in Start() then it will not get set because the Start() function does not happen immediately. To workaround this, just called graph.Init() after instantiating the graph and before assigning the delegate.

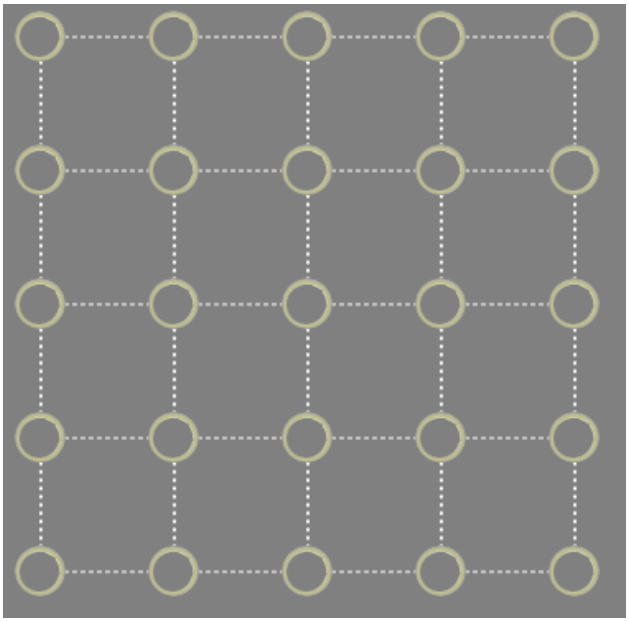
VI. Square / Rect / Hex Grids

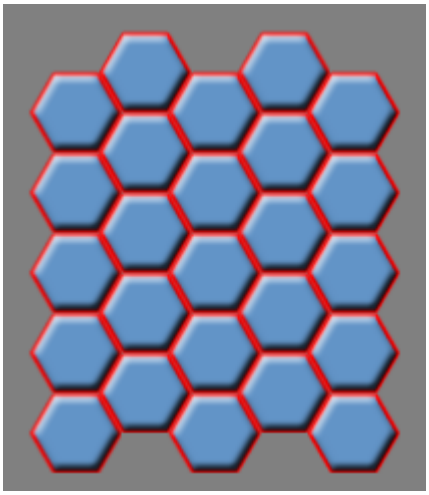
6.1 Getting Started

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



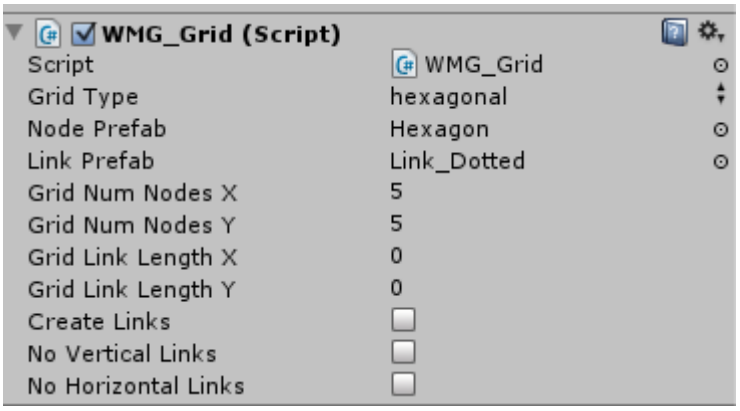
The grids will appear when you play the scene:





6.2 Parameters

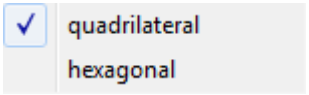
Grids can be customized based on following parameters:



- Auto Refresh

Automatically refreshes the grid based on changes to grid 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.

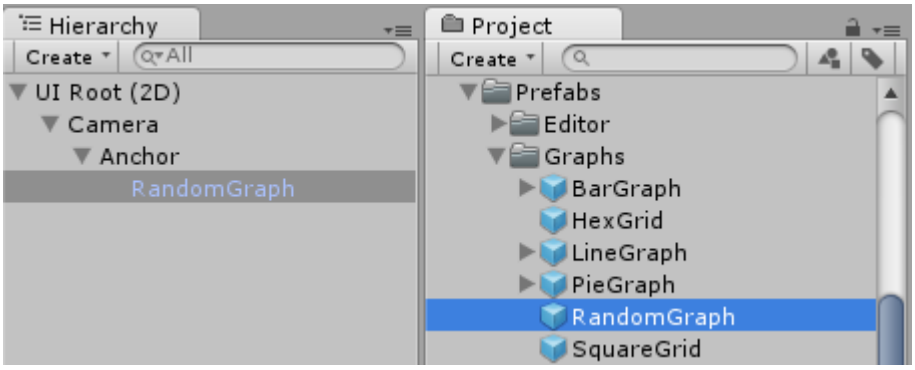
- No vertical / horizontal links

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.

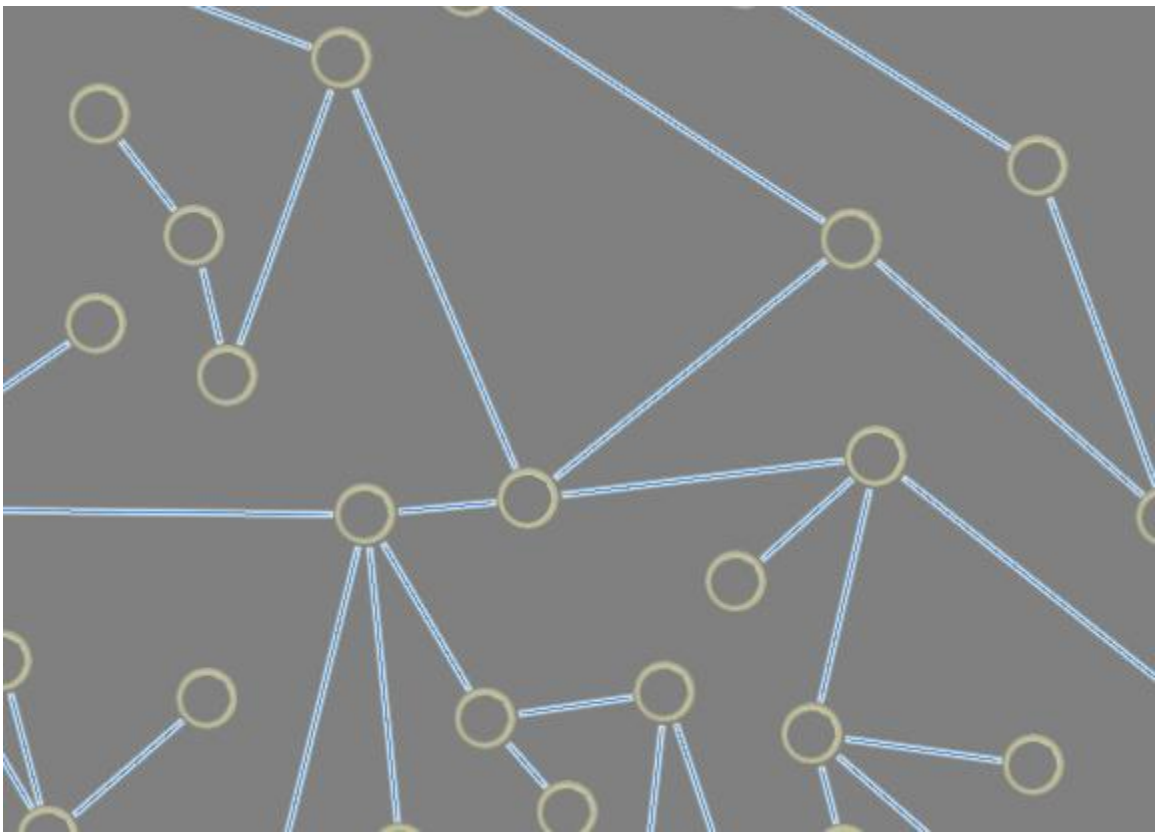
VII. Random Graphs

7.1 Getting Started

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

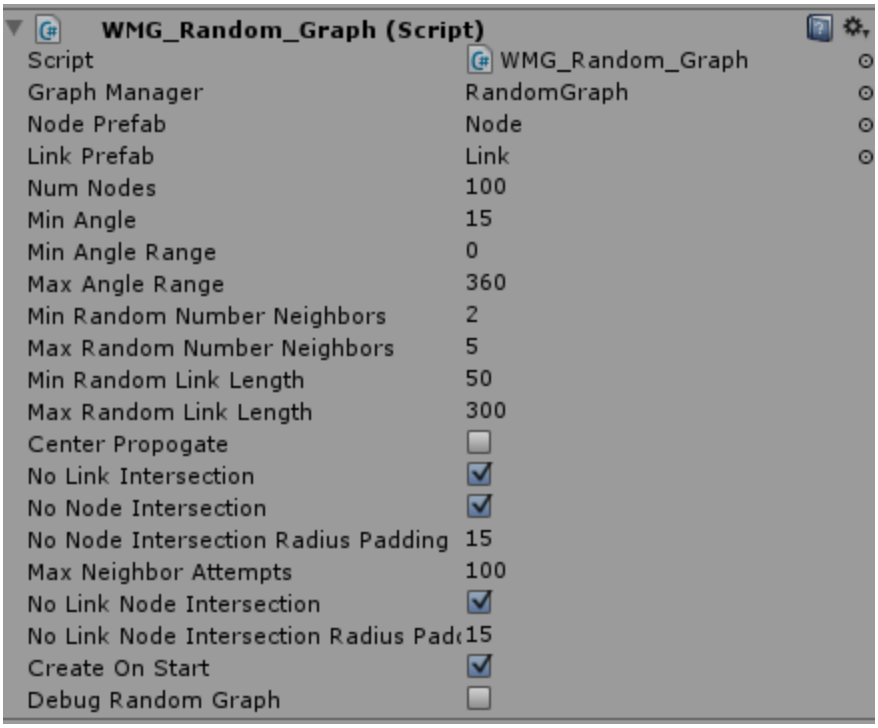


The graph will appear when you play the scene:



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().

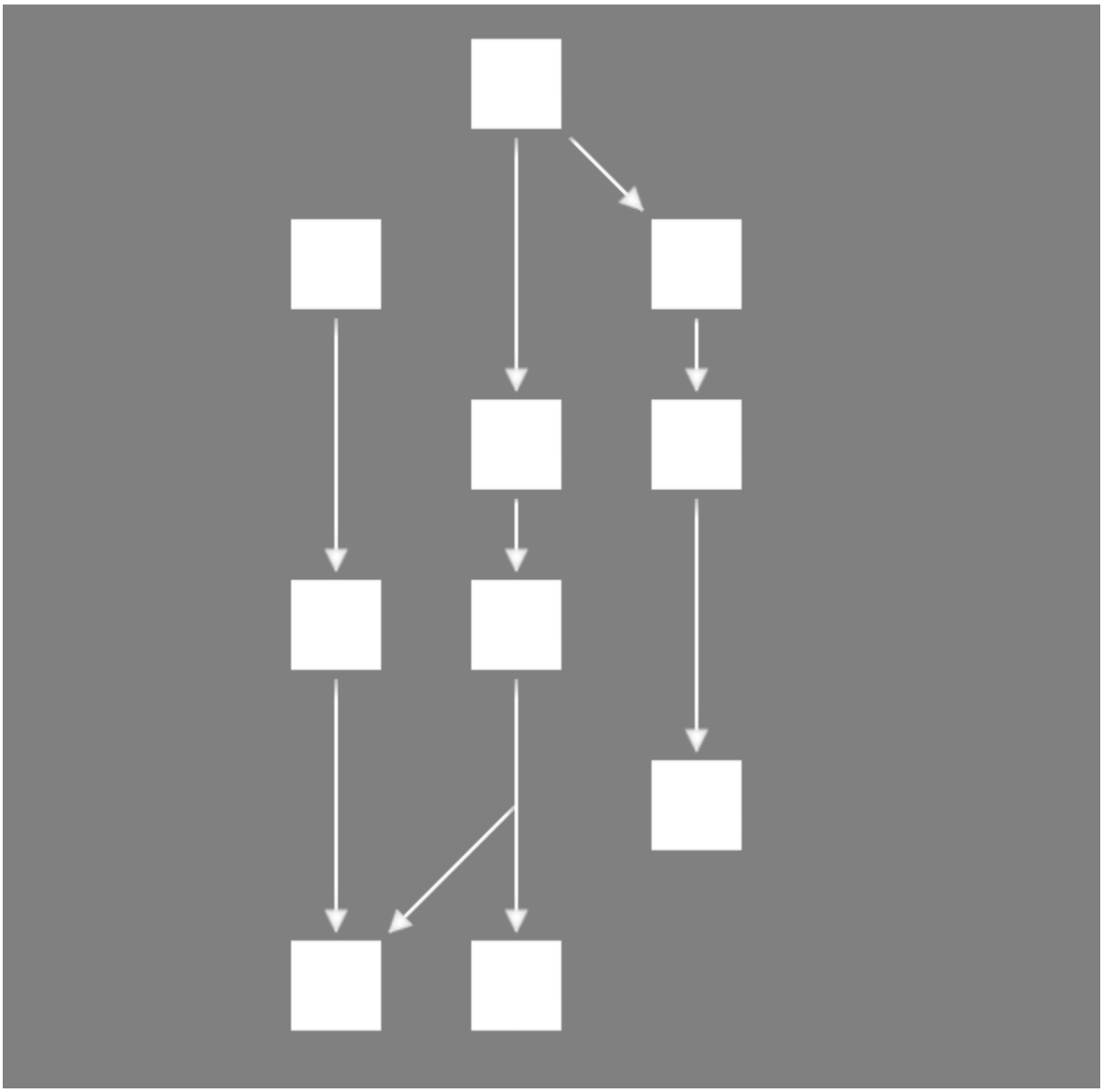
- Debug Random Graph
This can be useful to troubleshoot exactly what is happening during the random graph generation process. I resolved many bugs, and added new functionality using this parameter.

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

Core Parameters

- 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.

- 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.

Misc Parameters

- Node 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.

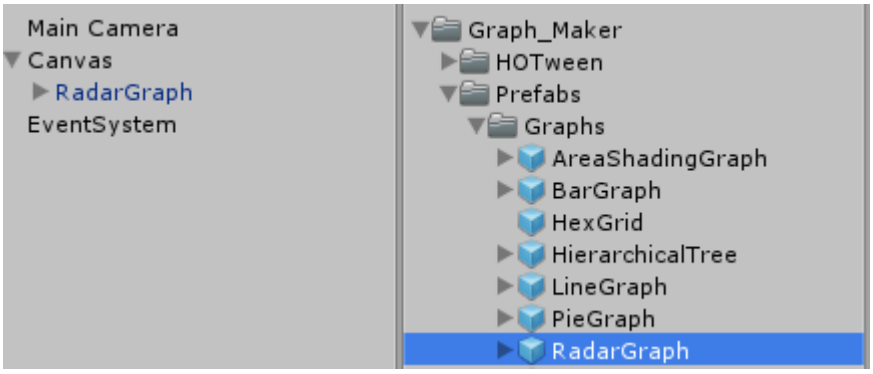
- Invisible Node Prefab

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

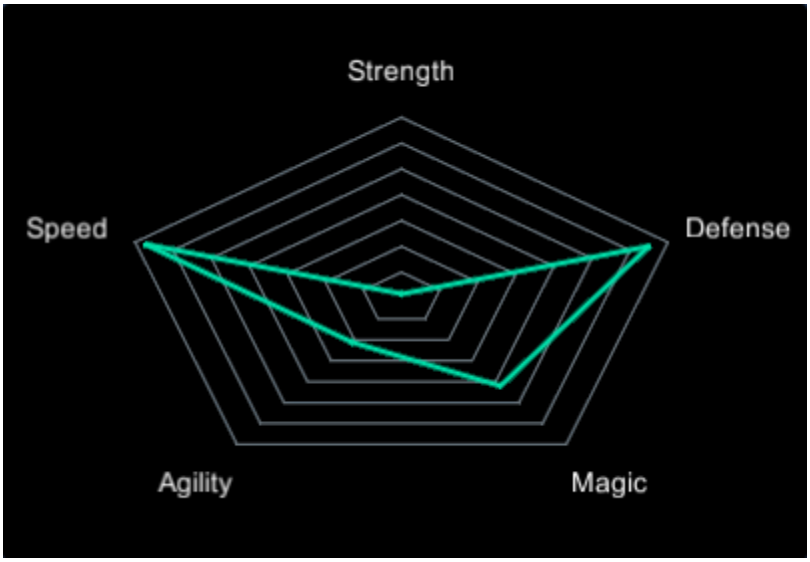
IX. Radar Graphs

9.1 Getting Started

Drag and drop the RadarGraph prefab into your scene:



The example radar graph will then appear when you play the scene:



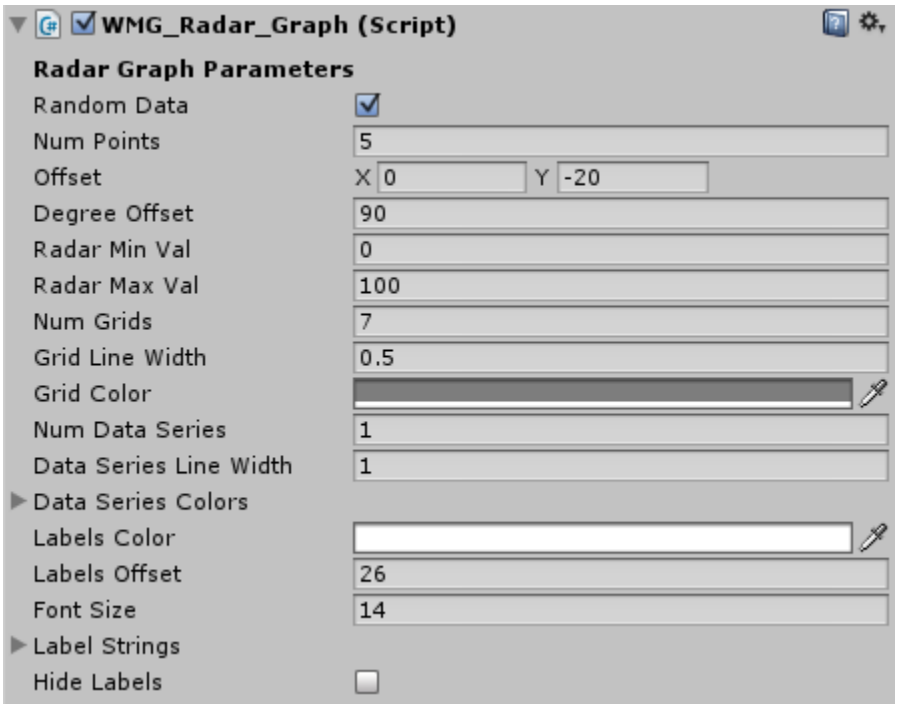
9.2 Summary

Overall, the radar graph is a collection of series from a normal Axis graph.

Each series has the "Connect First to Last" set to True, which is what allows creating a closed loop line graph. The points are also all disabled, though they could be enabled if you want points to appear.

The text labels are also created from a series. The lines are disabled and the points are created from a text node prefab which is a text label with the WMG_Node script.

9.3 Parameters



- Random Data

This generates random data for the data series of the radar graph, and should typically only be used for demonstration purposes. To use your own data you would disabled this and use your own List<float>.

- Num Points

This control how many points or edges there are for the radar graph. So setting 5 here will set all the grids to be pentagons.

- Offset

This can be used for moving around the radar graph without moving the graph as a whole (e.g. without moving the background sprite).

- Degree offset

This allows rotating the content of the radar graph such as the grids.

- Radar min / max value

This controls the radius of the radar graph.

- Num Grids

This is the number of grids that appear. Each grid is evenly spaced.

- Grid line width / color

Controls visual aspects of the grids.

- Num data series
This controls how many data series there are. Typically 1 or maybe 2 can be used. Anything more and the graph will be difficult to read.

- Data series line width / colors
Controls visual aspects of the data series.

- Labels color / offset / font size
Controls visual aspects of the labels.

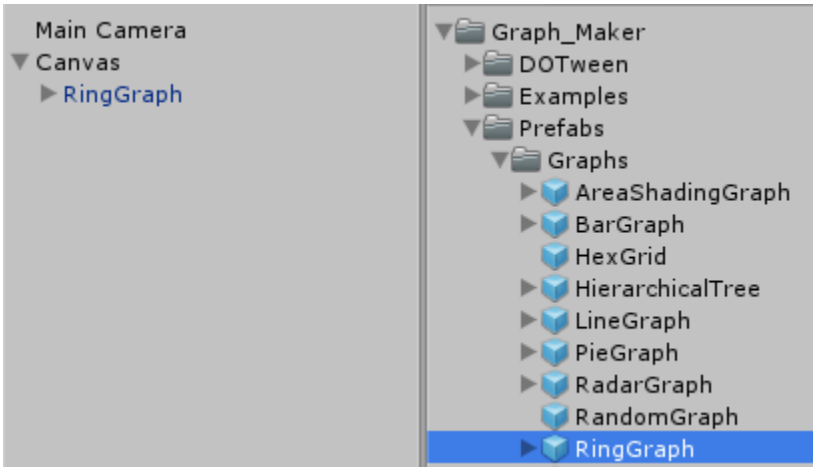
- Label strings
The text values of the labels.

- Hide Labels
Whether to hide the labels.

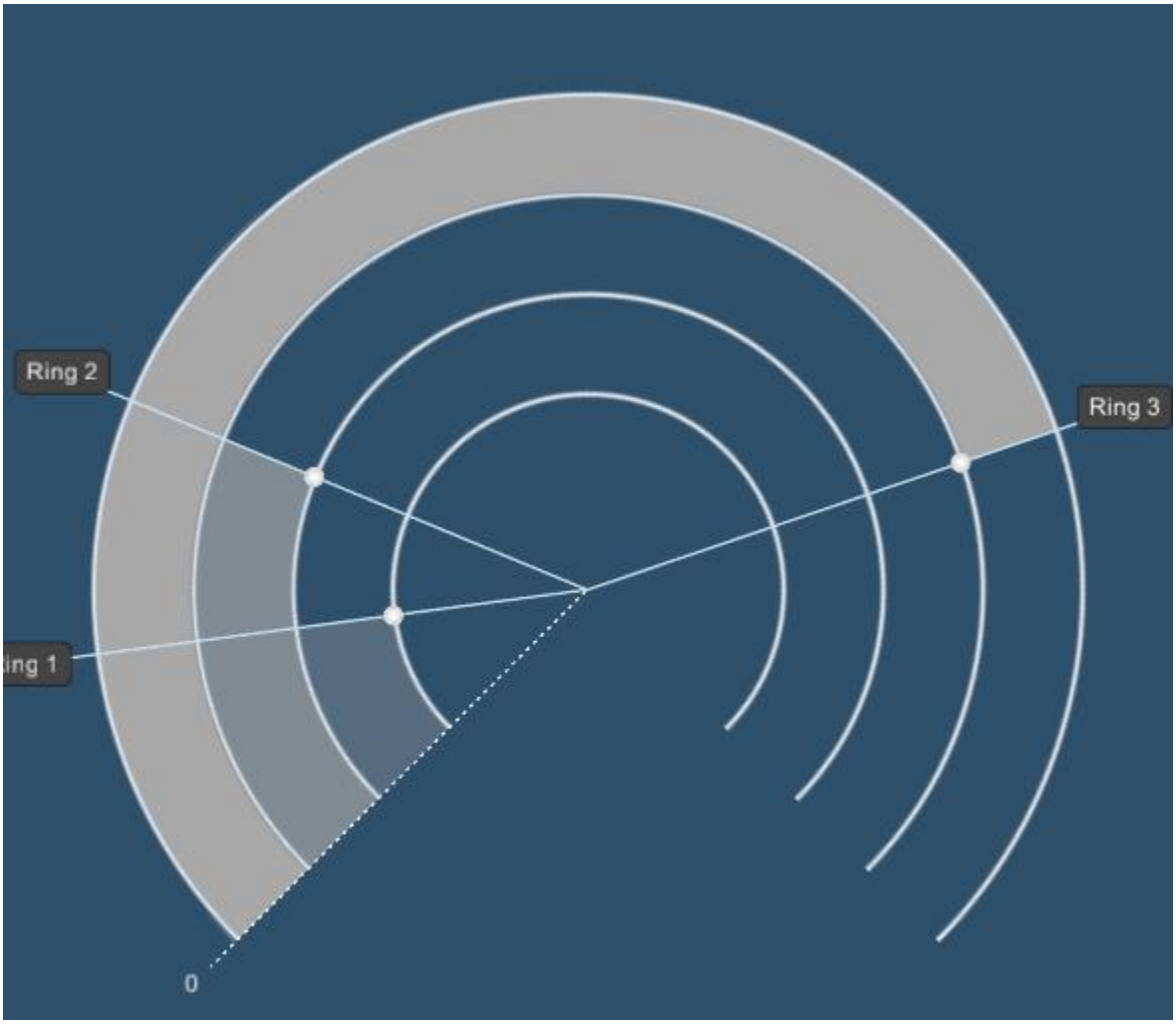
X. Ring Graphs

10.1 Getting Started

Drag and drop the RingGraph prefab into your scene:



The example ring graph will appear when you play the scene:

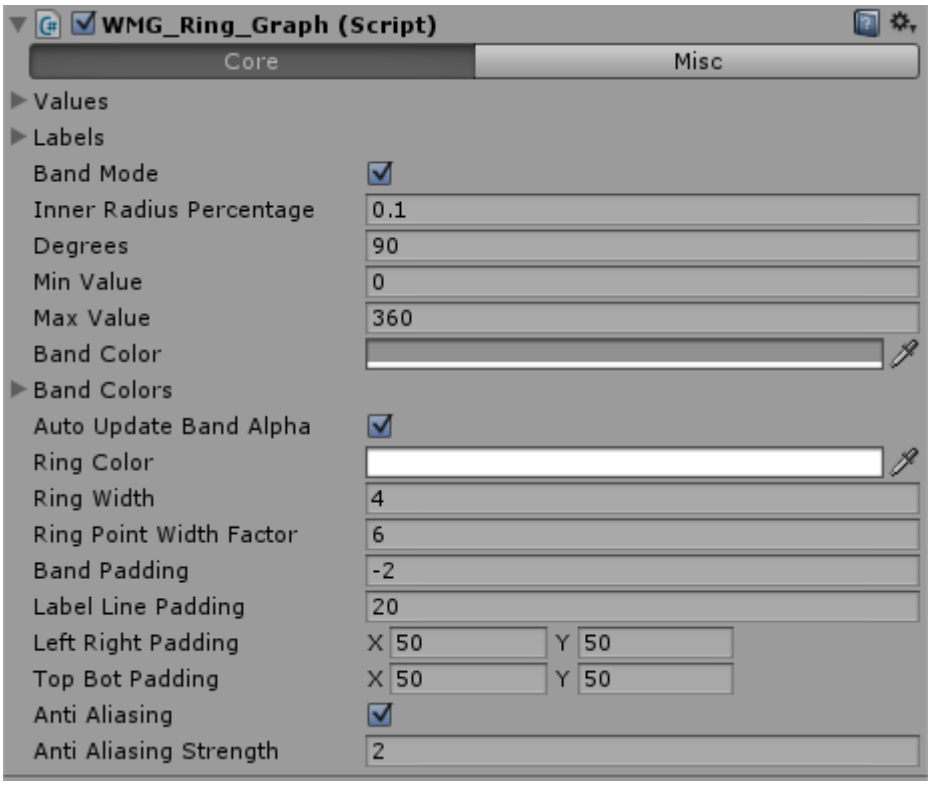


10.2 Summary

Overall, the ring graph is a collection of rings and optionally bands. Each ring and band is a radial sprite, that has an alpha cutout which happens via manipulation of the texture in memory.

The ring graph can also have an arbitrary number of degrees specified, which controls how many degrees are cutout from the circle. For example at 180, a half-circle appears for all the rings and bands.

10.3 Parameters



Core Parameters

- Values
This determines where the bands / labels appear. For example, if the min were 100, and the max 200, and a value of 150 were specified, then the band / label will appear in the center of the graph.

- Labels
This determines the text displayed in the label for each ring.

- Band Mode
When band mode is enabled, each ring has a corresponding band, except for the outer most ring. When disabled, only rings will appear.

- Inner Radius Percentage
This is the radius of the innermost ring relative to the outermost ring. The radius of the outermost ring is determined by the width / height of the graph rect transform.

- Degrees
This is the number of degrees cut out from all the rings and bands. For example if 90 is specified, then 3/4 of a circle will appear for all rings and bands.

- Min Value
This is the value that represents the minimum of the ring graph (the left-most side).

- Max Value
This is the value that represents the maximum of the ring graph (the right-most side).

- Band Color
This is the base color given to all bands.

- Band Colors
This can be used to override colors of individual bands if it is needed that the color be different than the base band color.

- Auto Update Band Alpha
This automatically adjusts the alpha of the band colors, such that the bands gradually fade out, the closer the band is to the center of the graph.

- Ring Color
This is the color given to all the rings.

- Ring Width
This is the width of all of the rings.

- Ring Point Width Factor
Determines the size of the points as a factor of the ring width.

- Band Padding
This is the padding between a ring and a band.

- Label line padding
The number of pixels the label lines and the zero line of the graph extend beyond the outer most ring.

- Left / Right / Top / Bot Padding
The number of pixels of padding for the background sprite in relation to the outer most ring sprite.

- Anti Aliasing / Strength

Because the rings and bands are constructed dynamically via texture manipulation, we can have higher control over the anti-aliasing strength applied to the ring / band sprites. The strength represents a number of pixels to which alpha fading is applied in order to simulate a perfect circle.

Misc Parameters

- Animate Data / Anim Duration / Anim Ease Type

When enabled, any updates to the data will animate the band and label line towards the newly specified value over the specified animation duration using the specified ease type.

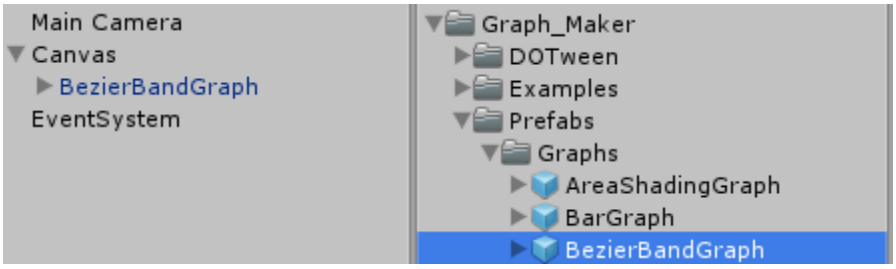
-Ring IDs

This associates an ID to each ring, so that an API to query individual rings can be used. Also note that an example API function exists to highlight a particular ring using the ring ID.

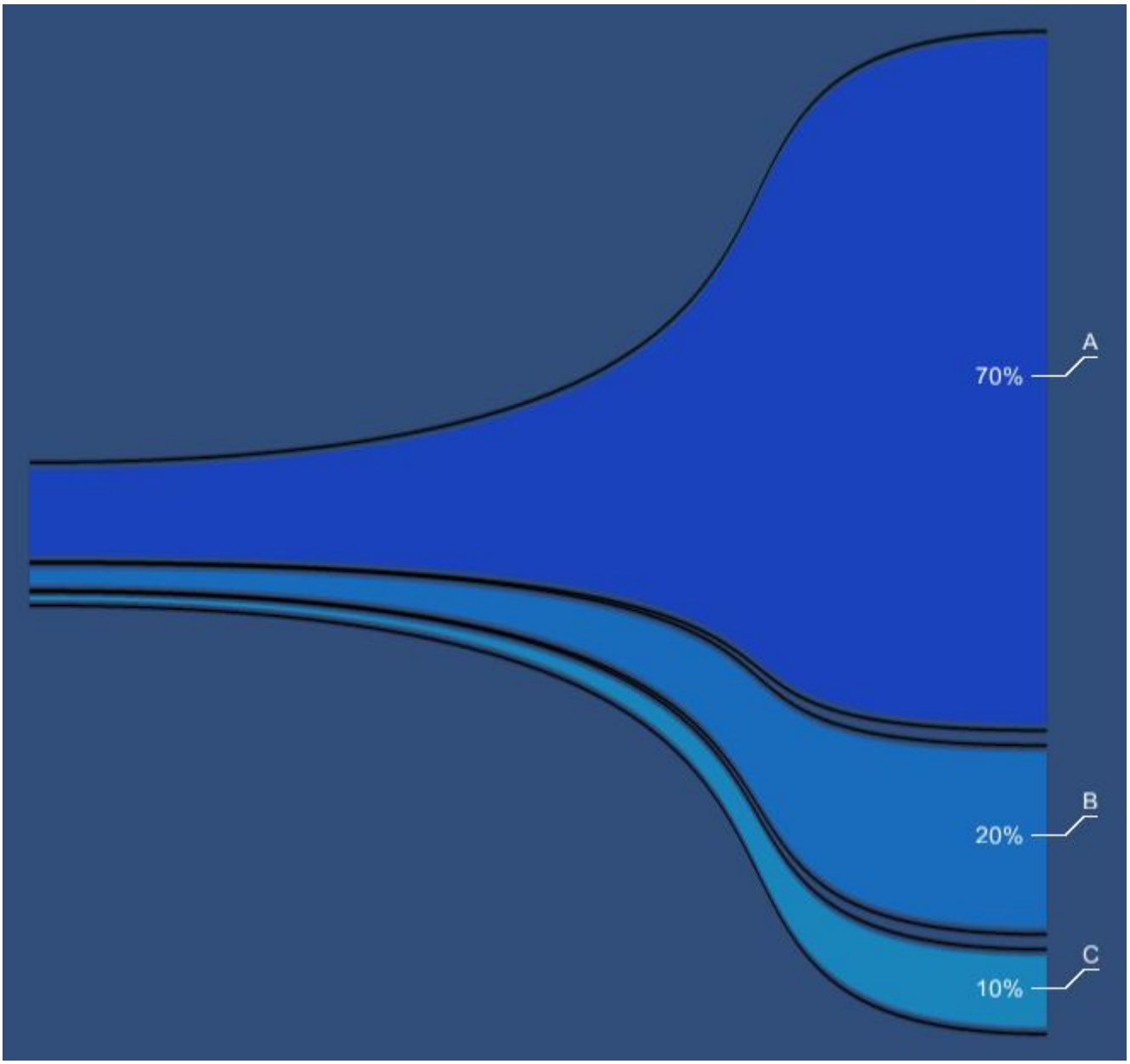
XI. Bezier Band Graphs

11.1 Getting Started

Drag and drop the BezierBandGraph prefab into your scene:



The example will appear when you play the scene:

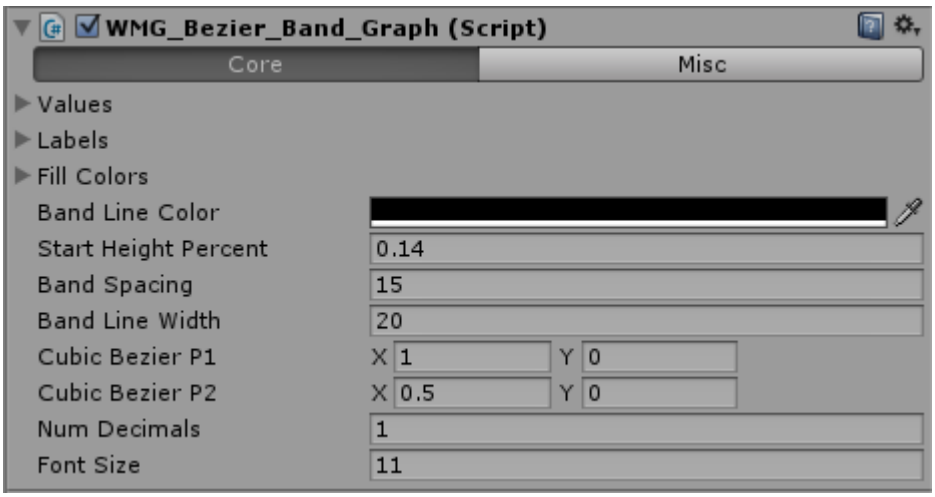


11.2 Summary

Overall, the bezier band graph is just a fancy pie chart / way of visualizing percentages. It is a collection of bands defined by bezier curves. Each band and its borders are white textures whose colors were manipulated using the SetPixels() function.

Note that setting pixels of textures is a slow operation and thus updating this graph in real-time is not really possible. A lower resolution texture on the band prefab (default 2k x 2k), will greatly increase the speed, but not look so good.

11.3 Parameters



Core Parameters

- Values
This is the List<float> values used to determine how many, where, and how wide each of the bands are.

- Labels
This determines the text displayed in the label for each band.

- Fill Colors
The colors used for each of the bands.

- Band Line Color
Each band is bordered by 2 lines, this is the color of all of the border lines.

- Start Height Percent
This is the height in terms of a percentage of the rect transform height that the collection of bands starts at.

- Band Spacing
The number of pixels between each band.

- Band Line Width
The number of pixels in each band line border.

Cubic Bezier P1 / P2
- Controls the overall shape of the graph. Play around with these for a different look. Refer to wiki on cubic bezier formula for more information for what these really control.

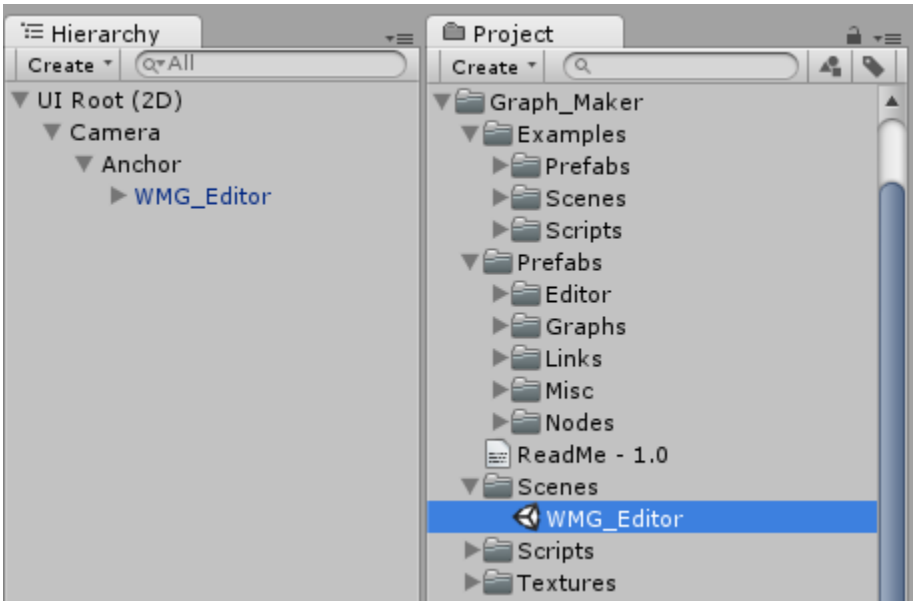
Num Decimals
- The number of decimals used in the labels.

Font Size
- The font size of the labels.

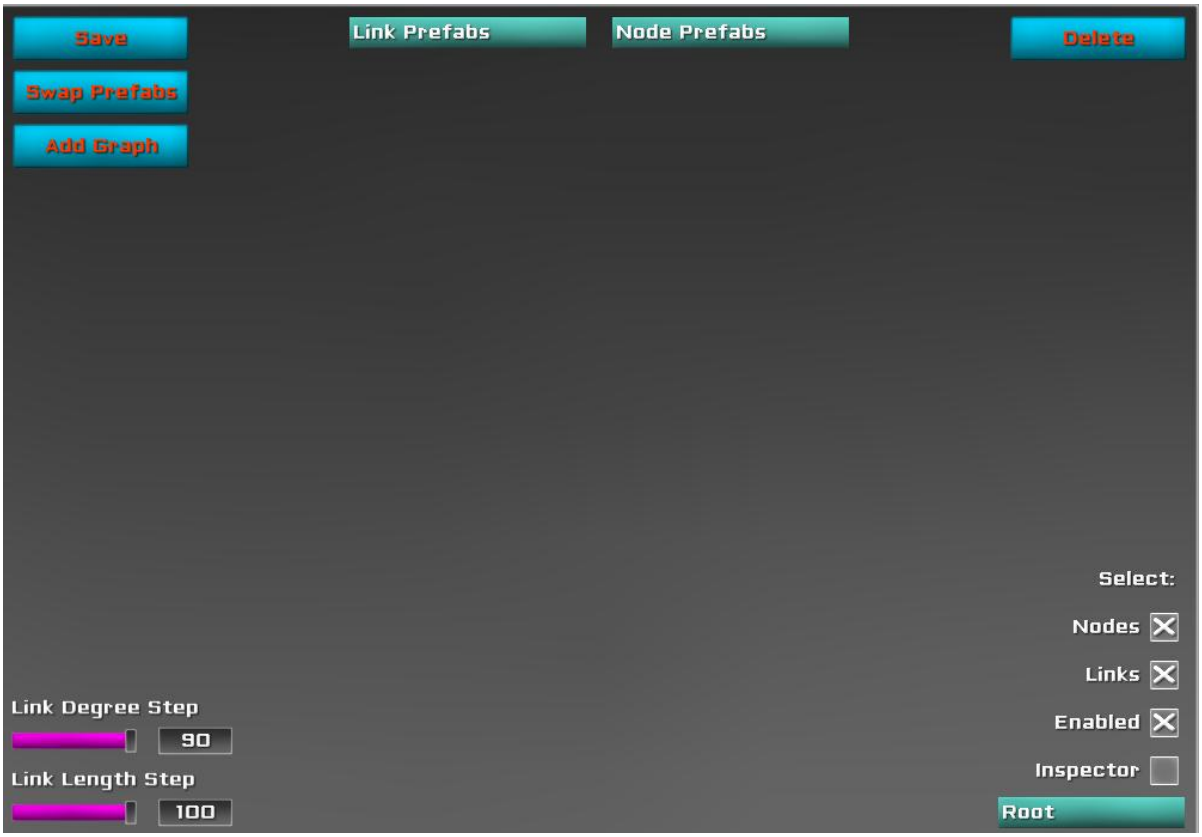
XII. Graph Editor (Deprecated)

12.1 Getting Started

Double click the WMG_Editor scene in the Scenes folder:

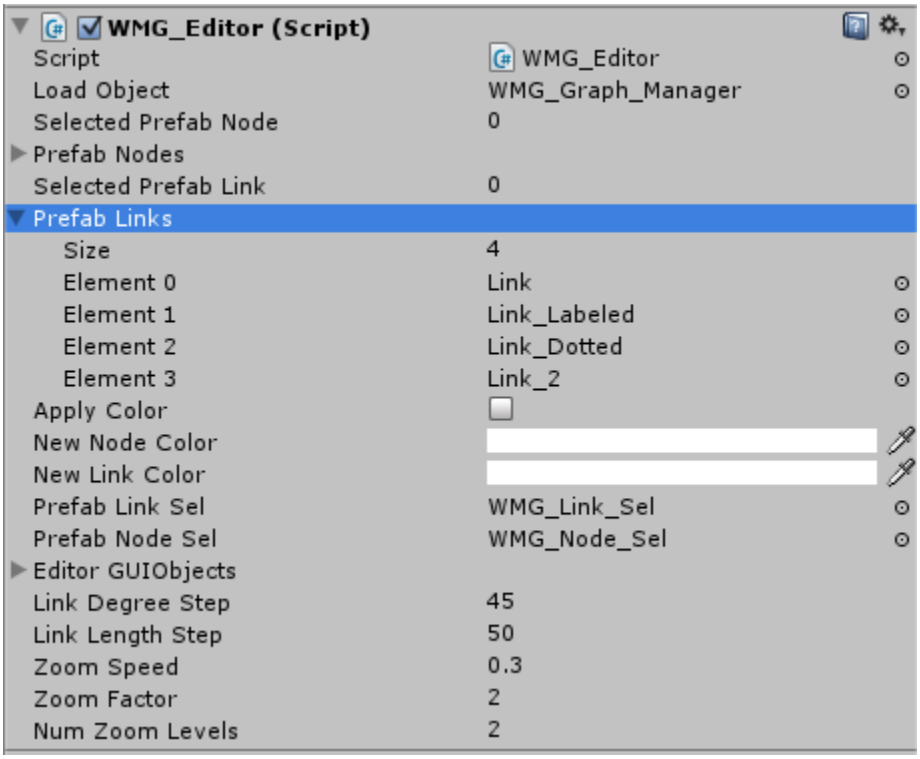


The editor GUI will appear:



12.2 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:

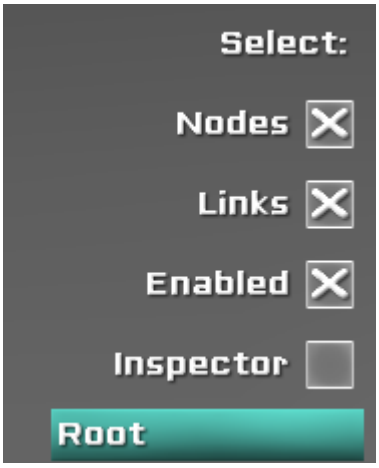
- Zoom speed / zoom factor / num zoom levels:
These can be configured to change how zooming works in the editor.

12.3 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

12.4 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.

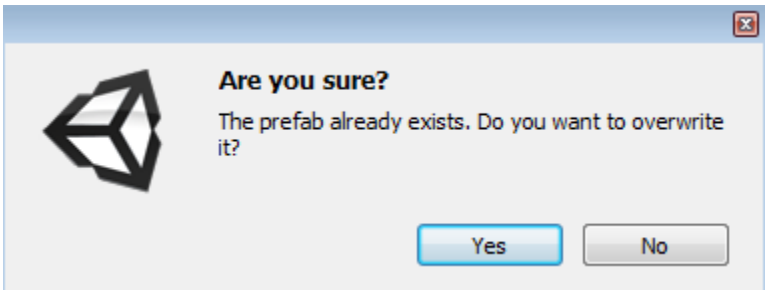
The pop-up list that says "Root" by default allows selecting other objects in the inspector when inspector is checked. For example, if your node prefab has an "Object to Label" reference you can change root to "Label Object" which will select that node's referenced label object. The default is root,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

12.5 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:

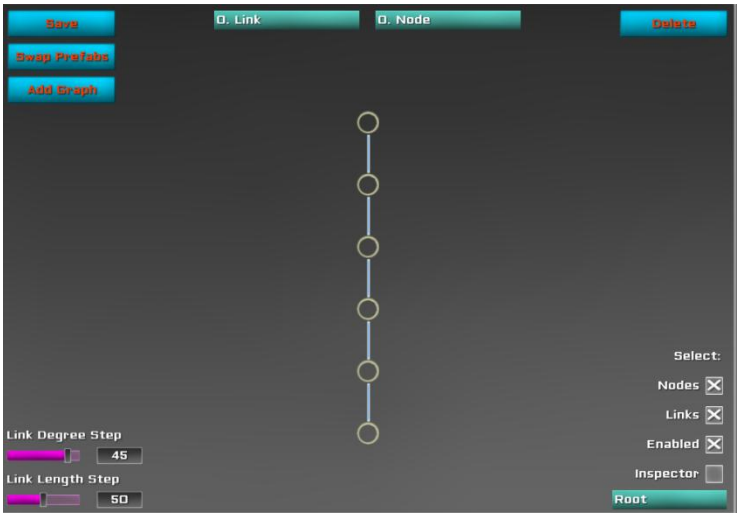


To load an object, stop the WMG_Editor scene and drag and drop the prefab that was created from the Save process into the "Load Object" variable. Now when you hit play, the editor will load your previously saved object.

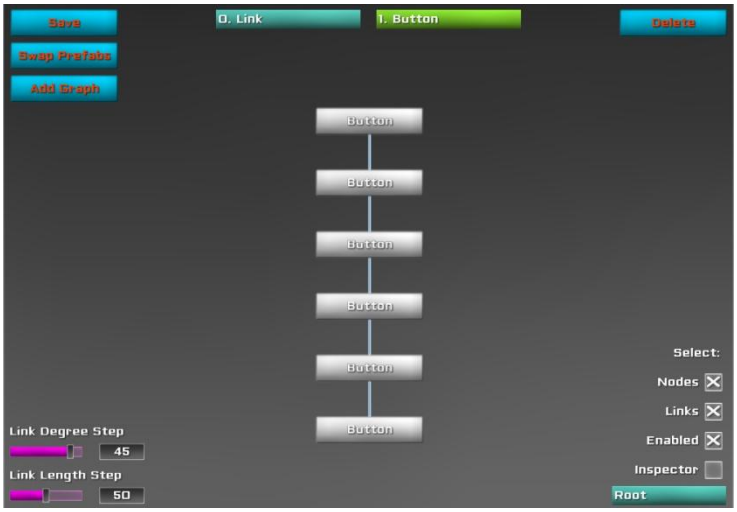
12.6 Prefab Swapping

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:

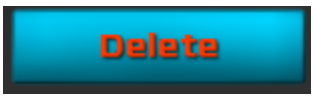


How it works-

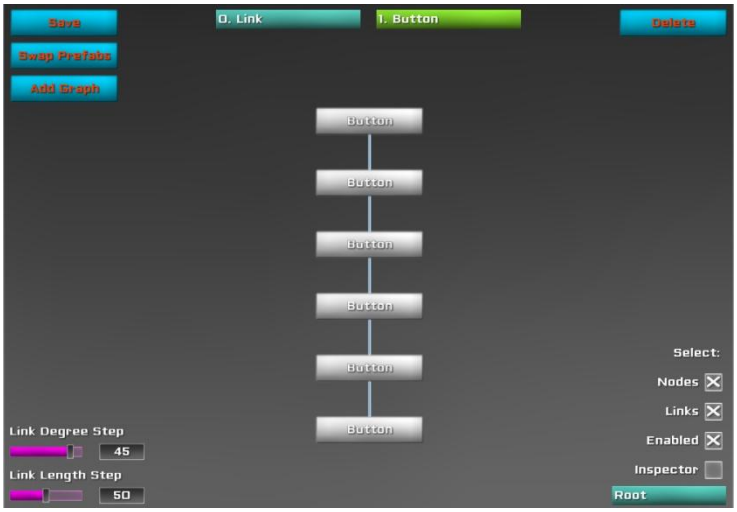
It destroys the existing selected objects / links and creates new objects and links with the same WMG_Node / WMG_Link data as the destroyed objects. The names of the gameobjects are also preserved. If you have custom scripts with custom data on your prefab those will not survive the process unless of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

12.7 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!

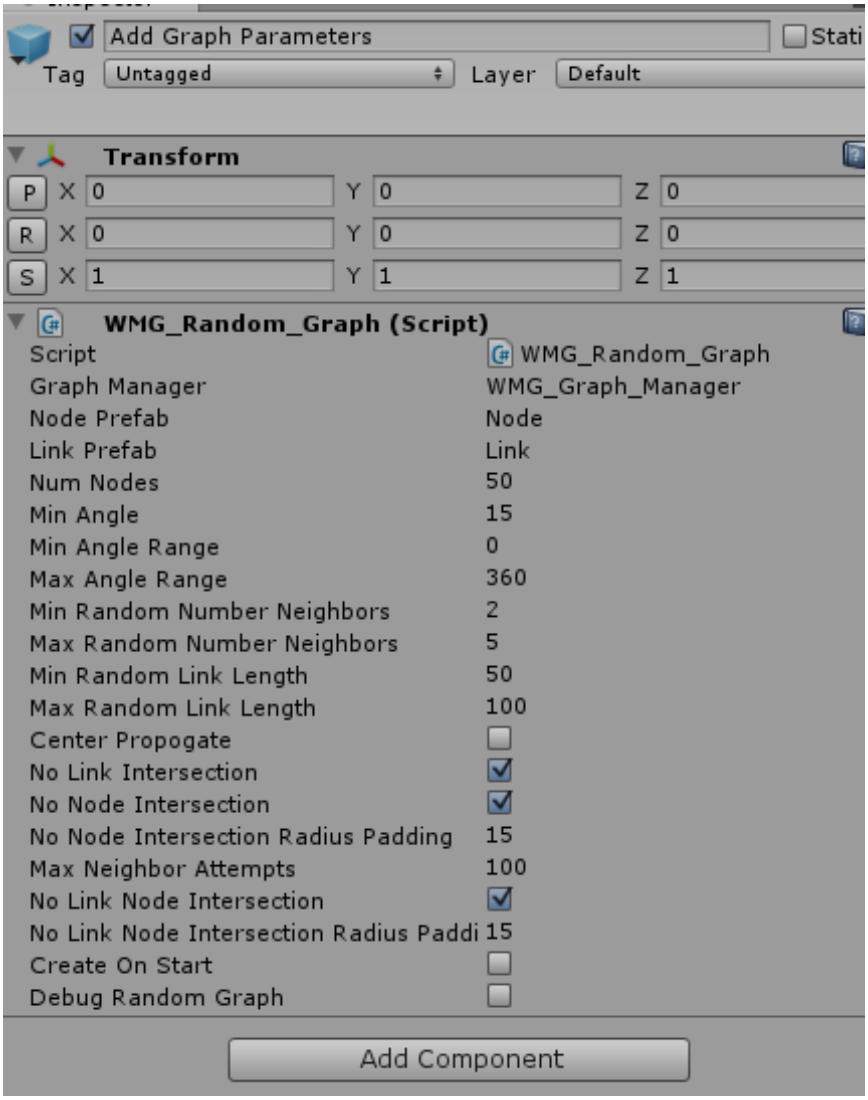
12.8 Adding Graphs

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.