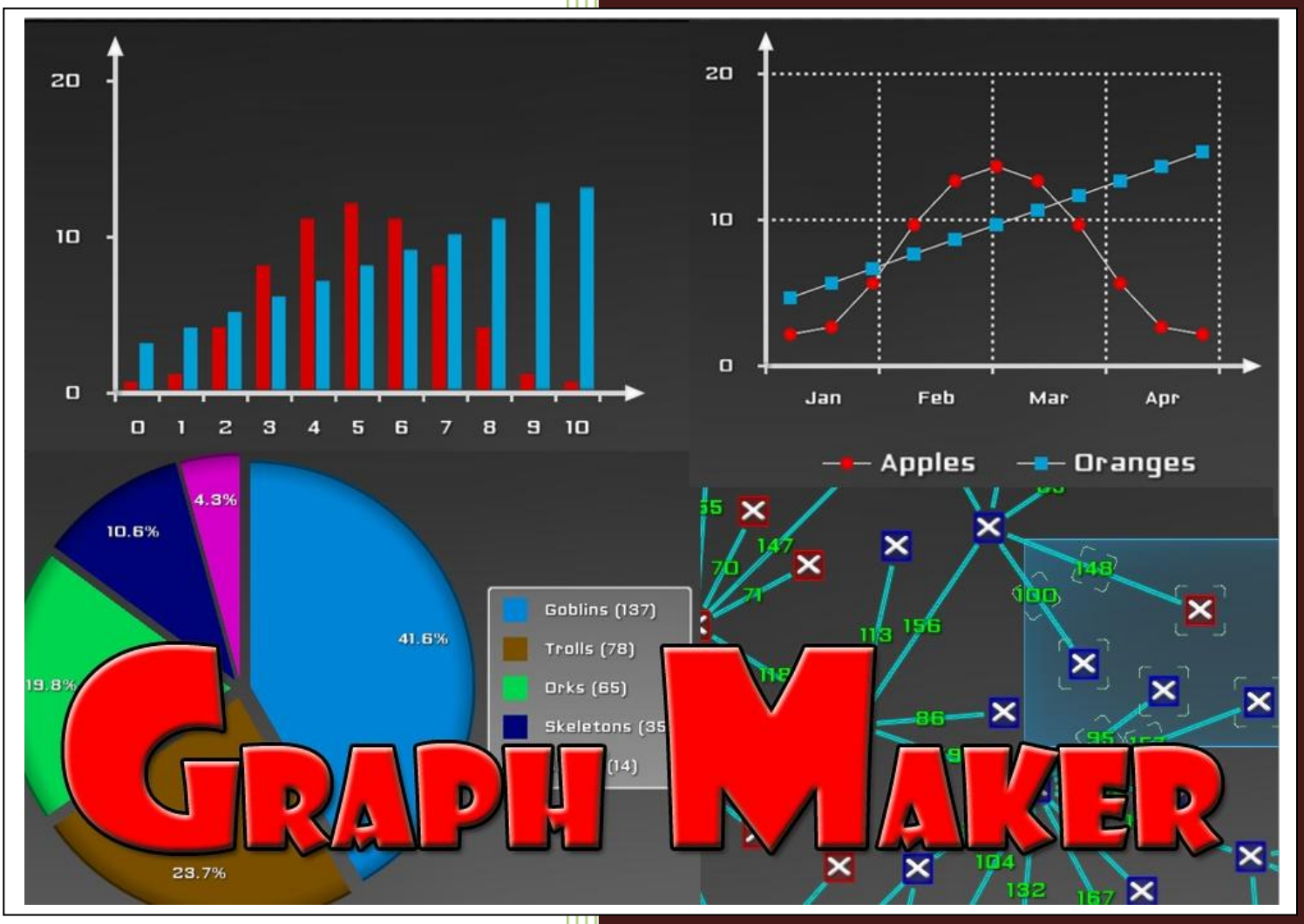
Graph Maker


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. Common use cases of these types of graphs include GUI objects based on grids, trees, and 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

1.3 FAQ

Q: Does Graph Maker work well with just Unity GUI system? I don't use NGUI, Text Mesh PRO, or other 3rd party systems.
A: Yes! Graph Maker is designed and developed primarily for the Unity GUI system. Using 3rd party systems is optional.

Q: I dragged and dropped a Graph Maker prefab into an empty scene, and hit play, but I don't see anything?
A: Since Graph Maker uses the Unity UI system, all graphs (as well as other UI objects), must be a child of a canvas. To create a canvas in your scene, go to Gameobject -> UI -> Canvas.

Q: I have some code that instantiates a graph and sets some graph properties. The graph gets created, but my other code that set the graph properties did nothing?
A: Graph Maker relies on Start() being called, and Instantiate() doesn't call Start() immediately. For Graph Maker, Start() calls a public Init() function, so just write some code to call Init() immediately after you instantiate the graph, e.g. myGraph.Init(). Don't worry about Init() being called more than once, Graph Maker handles that.

Q: How do I do my own custom stuff when I click or hover over a point or bar in a graph?
A: Write some code to subscribe to the appropriate Graph Maker event. For example, for clicks:

```
1.
2.  public class GraphPointInteraction : MonoBehaviour {
3.
4.      public WMG_Axis_Graph myGraph;
5.
6.      void MyCustomFunction(WMG_Series series, WMG_Node node) {
7.          Debug.Log("Node: " + node.name + " on series: " + series.name + " was clicked!");
8.      }
9.
10.     void Start() {
11.         myGraph.WMG_Click += MyCustomFunction;
12.     }
13.
14. }
```

Q: I set a list of vector2 for a series, and then looped through the gameobjects of the series using WMG_Series.getPoints() to add my own custom script to the newly created point gameobjects, but it doesn't look like my custom script got attached to the newly created gameobjects?
A: Short answer - call Refresh() on the graph immediately after setting the list of vector2. Long answer - Graph Maker's refresh system looks at all the properties that changed (including lists), and applies the changes that occurred in the Update() function. Since the changes get applied each frame in the Update() function, the changes are not immediate, however you can make the changes apply immediately at anytime in your custom code by calling the Refresh() function.

Q: My graph dimensions are driven based on Layout groups, and I notice a jitter when the graph resizes. Any way to fix this?
A: Yes, add this code to WMG_Axis_Graph.cs

```
1.  void OnRectTransformDimensionsChange () {
2.          if (!hasInit) return;
3.          updateFromResize();
4.          Refresh();
5.      }
```

Q: When I add 1,000 points I notice a big lag, any way to fix this?
A: First thing to check is whether "Area Shading" is enabled on your series. This functionality creates N-1 (where N is your number of points) separate instanced materials with different shader properties set for each, and will significantly slow things down. It is not recommended for graphs of a large number of points.
Second thing you can do is use a Coroutine to load your data overtime rather than all at once to avoid an initial FPS hit.
Third thing is to consider reducing the number of points shown in your graph to have at least 5 pixels available per point, otherwise you won't be able to see much. For example, if the graph has 800 pixels on the x-dimension then 5 pixels per point will get you 800/5 = 160 points.

Q: I'm not sure how to use WMG_Data_Source / how do I do real-time update graphs?
A: So, for the WMG_Data_Source component, there are a couple main uses:

The example at the end of the X_Dynamic example scene is the "real-time" updating. In this case a WMG_Data_Source component is linked with a variable "Transform.localPosition.x", and the graph pulls data from this variable constantly over time. In this case the x-axis is always time or "real-time" and the y-axis is the one variable being continuously graphed over time. This essentially means the source variable must be a single float (no control over the x-axis). Example of how to set this up / use it can be found in the function realTimeTests() in X_Dynamic.cs. This method is useful if you want to graph a single variable continuously, such as a Player's ammunition count in a game. There is some fairly complicated logic that determines when a data point is actually plotted with this method. Basically, the method calculates slope differentials from previous points, and if the slopes are significantly different from the point that is about to be plotted, then a new point is plotted. This is to avoid plotting excessive amounts of data which would otherwise lead to performance issues.

The other example using WMG_Data_Source component is also in the X_Dynamic example scene. It is the "Dynamic data population via reflection". With this method the x and y axes must both be specified and can be whatever you want them to be. However in this case your data source must correspond to either a List<Vector2> or just a Vector2 if you go with the "multi-objects single variable" setting on the data source. Once the data source has been setup, then the graph's data will automatically change based on any changes made to the source data (List<Vector2> variable from an arbitrary script object, or a Vector2 on a collection of arbitrary script objects). Take a look at function dynamicDataPopulationViaReflectionTests() in X_Dynamic.cs if this is what you want to do. There isn't really anything special with setting things up with this, it just bypasses the need to write your own code to update a Series List<Vector> of pointValues. This methodology was originally created so Graph Maker could be used with PlayMaker, to essentially make a graph that automatically updates corresponding with a PlayMaker variable.

So, that about covers WMG_Data_Source component. Another way to plot data in "real-time" is to instead use your own custom Coroutine function. The method of using Coroutine is shown in the "Plot Overtime" example scene. In this example a random value is plotted overtime continuously, however it is not too much work to modify the example to work with your own data instead of random data and / or to plot data very fast with no special animations in between points.

1.4 Upgrade Guides

Over the past few years Graph Maker has evolved substantially. It has become quite stable and feature-complete. However, that doesn't mean there won't be more changes on the horizon that break or greatly change existing behavior. On that note, this section serves as a guide to making old Graph Maker objects / prefabs work with the newest version of Graph Maker. The idea is that by following these upgrade guides sequentially from one version to the next it will be possible to reproduce the latest version Graph Maker graph from older versions of Graph Maker.

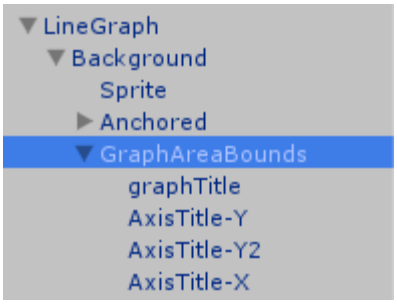
That said, if you plan on making quite a few various different Graph Maker prefabs; I recommend to generate these prefabs from code. If you generate your prefabs from code you won't really even need to use this upgrade guide. Instead, you could use your code that generates prefabs to generate prefabs from Graph Maker's latest version prefab. Graph Maker itself actually creates all axis graph prefabs from one base prefab. That base prefab

is the "LineGraph.prefab". All the other axis graphs (scatter, bar, area shading, stacked line, etc.), are generated dynamically from code from the Unity editor menu. This code is under Scripts/Editor/WMG_Prefab_Gen.cs

Upgrading from 1.5.7 to 1.5.8

A) Create a new empty game object underneath graph / background, and set it to be the Graph Area Bounds object under the WMG_Axis_Graph / Misc tab in the inspector:

B) Move the graph title, and all of the axis title objects to be parented to the new object you created above:



C) For the Series prefab / series under your graphs, add a CanvasGroup component to Series / AreaShadingParent object, and set interactable and blocks raycasts to false.

D) On the tooltip object underneath your graphs, set the min / max anchors and pivots all to 0.

Upgrading from 1.5.x to 1.5.7

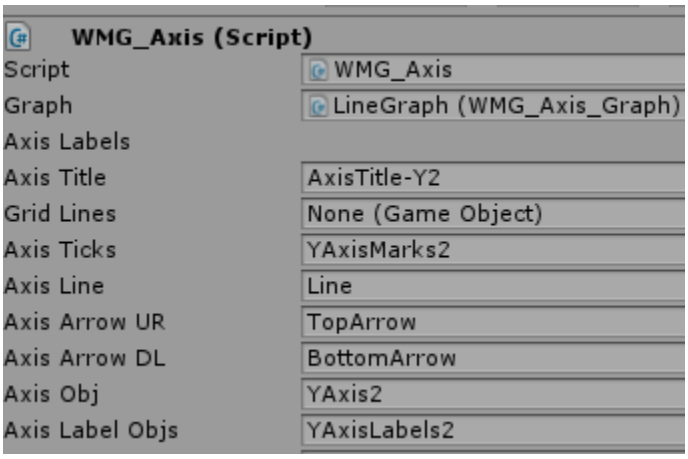
A) (Do this if you use x/y axis titles)

This upgrade changed how x/y-axis titles are positioned. Additionally, the x/y axis titles are parented to different objects in the unity hierarchy. First thing is to move the x/y axis titles from being parented to the graph background to being parented to the x/y-axis lines like so:

Next, the behavior of the x/y-axis title offset has changed to be based on the axis line. For example, the default value of y-axis title offset was -40, but should be changed to something like 40 instead to be positively offset away from the axis line.

B) (Do this if you plan to use the new dual-y axis chart functionality)
This upgrade also added a whole new secondary y-axis. First make a duplicate of the y-axis object, call it yaxis2 for clarity.

Next, references to various objects the axis use need to be set.



Set references to the title, the ticks, the line, the top / bottom arrows, the axis object parent, and the labels. The only reference to not set is the grid lines, grid lines are only controlled by the primary axis.
Note that you also need to create YAxisMarks2, and YAxisLabels2 gameobjects. You can just duplicate the existing ones used for the original y-axis.
Lastly, on the graph itself under the "Misc" tab, set the reference to the secondary y-axis.

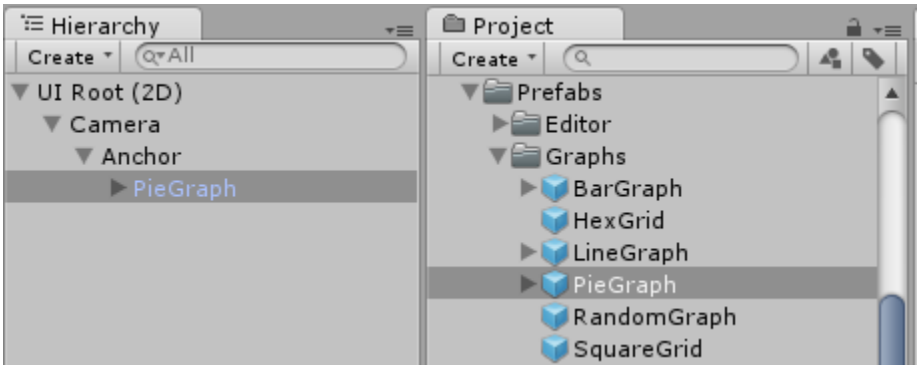
Upgrading from 1.4+ to 1.5+

For this upgrade, Graph Maker created a new separate class for axes. Any custom code you have that changes axes parameters must be updated. For example, if you have code that says "myGraph.xAxisNumTicks = 5", then the new code needs to be "myGraph.xAxis.NumTicks = 5".
Additionally all lists on graphs were changed to use a new custom list class that is observable and fires events when elements are added or removed. You can call list functions as normal on Graph Maker lists, however there is a special case for when setting the list directly to an entirely new list. For example, if you have code such as "mySeries.PointValues = myList", this must now change to be "mySeries.PointValues.SetList(myList)"

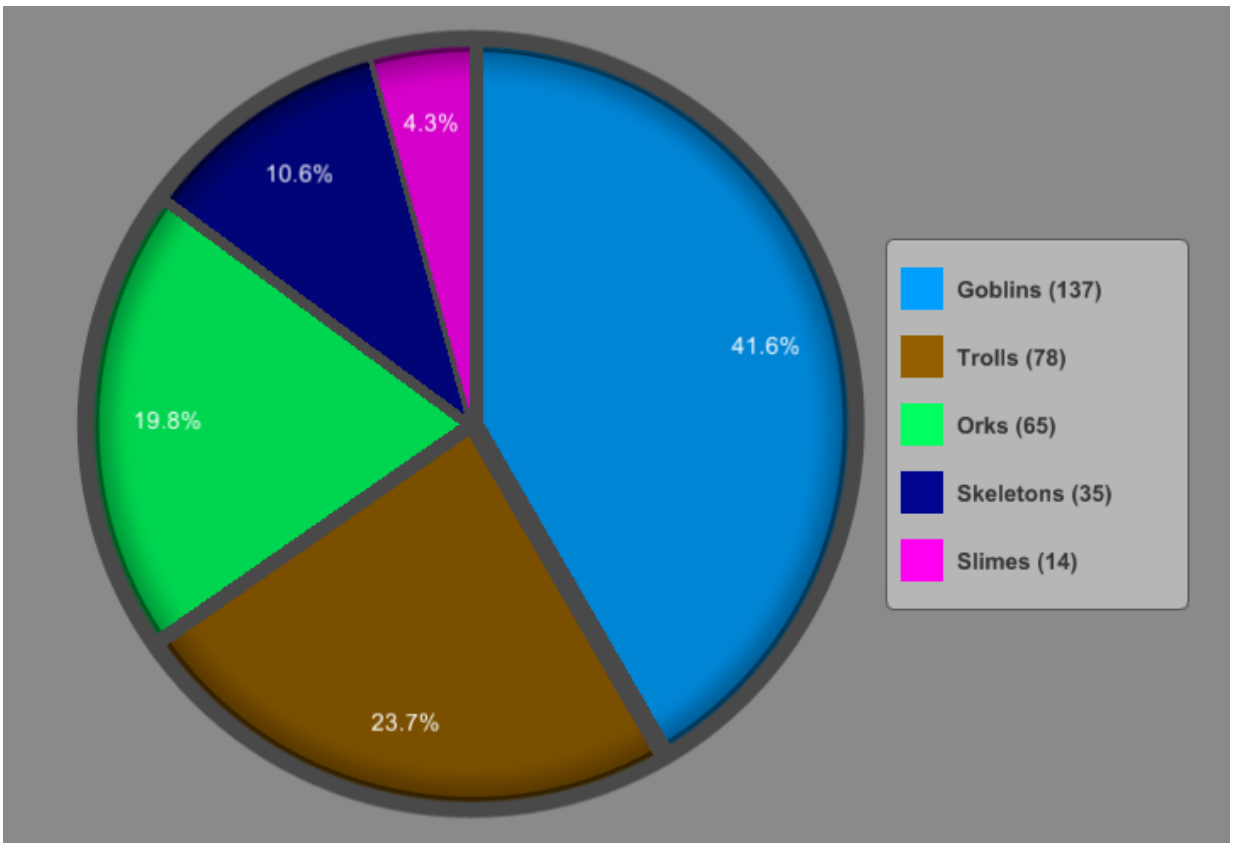
II. Pie Graphs

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:



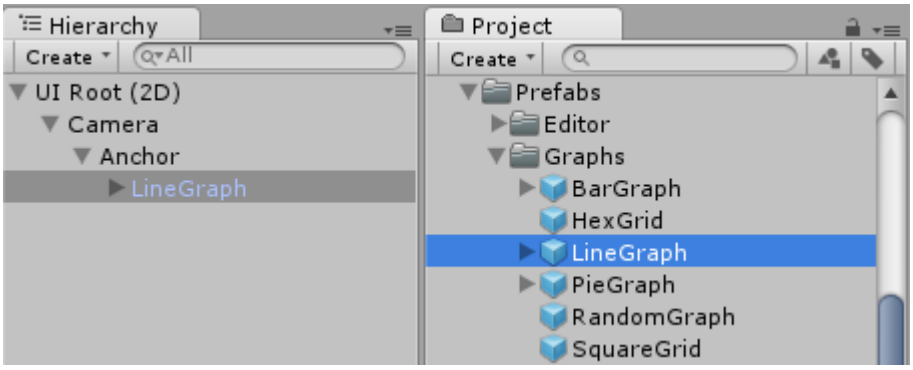
2.2 Parameters

Detailed information for pie graph parameters can be found under "Properties" section here:
http://stew-soft.com/GraphMaker/Docs/html/class_w_m_g___pie___graph.html

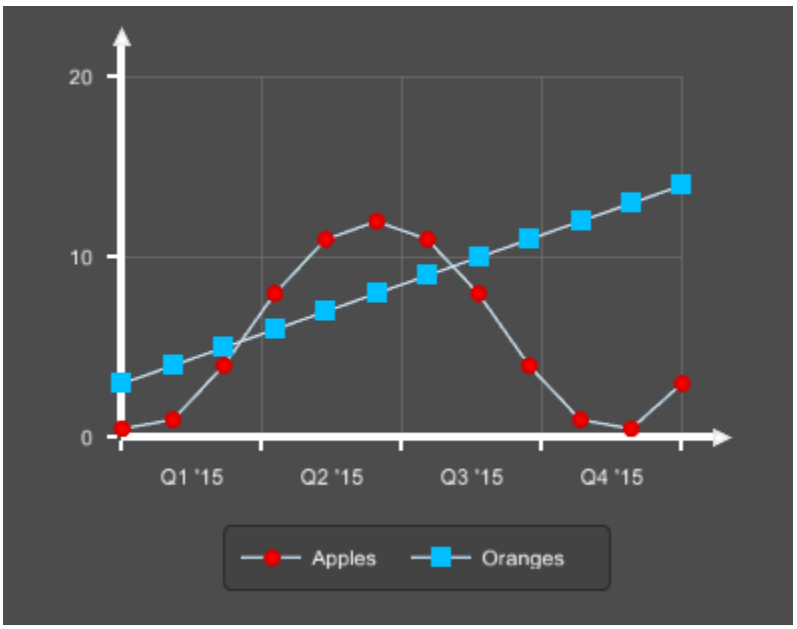
III. Line Graphs

3.1 Getting Started

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



The graph will appear when you play the scene:



3.2 Axis Graph Parameters

Detailed information for axis graph parameters can be found under "Properties" section here:
http://stew-soft.com/GraphMaker/Docs/html/class_w_m_g___axis___graph.html

3.3 Axis Graph Axis Parameters

Detailed information for x / y axis parameters can be found under "Properties" section here:
http://stew-soft.com/GraphMaker/Docs/html/class_w_m_g___axis.html

3.4 Series Parameters

Detailed information for series parameters can be found under "Properties" section here:
http://stew-soft.com/GraphMaker/Docs/html/class_w_m_g___series.html

3.5 Other Notes

Note that the grid lines, axis ticks, and axis labels are all implementations of WMG_Grid.

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.

3.6 Animations

WMG_Axis_Graph also 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

3.7 Events

There are several events to which you can subscribe for performing actions from a user mouse click or hover. Example code and list of all events can be found online in the API docs, here is the event and example code for the series point click event.

3.8 Dynamically Add / Delete Series

The graph script contains functions to add and delete series:

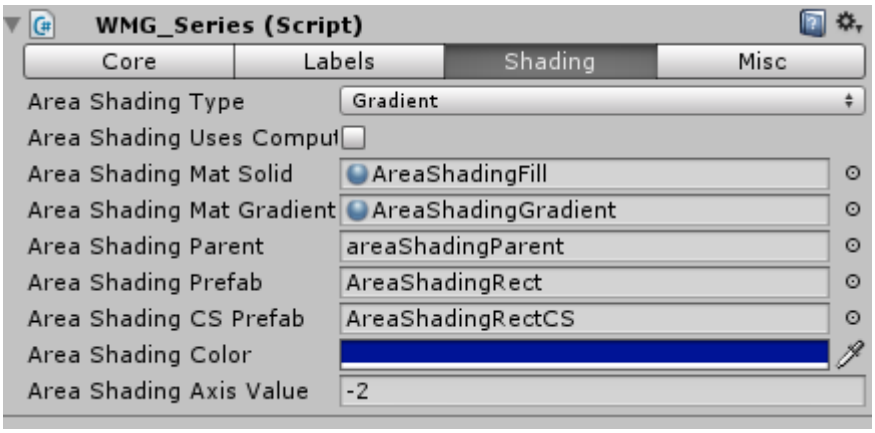
public WMG_Series addSeries()

public void deleteSeries()

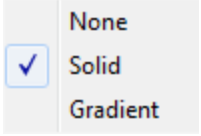
Simple call these functions and a series is added onto the end or deleted from the end.

3.9 Area Shading

There are several parameters on WMG_Series that allow the ability to add area shading for the series.



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



Area Shading Uses Compute Shader - Determines whether or not the area shading uses a compute shader in order to render. If enabled, then only a single rectangle is created and the shading is drawn in one pass via a compute shader, otherwise there is a rectangle created between each 2 points in the series and each rectangle takes an additional draw call due to the use of custom shaders / instanced materials. It is much better for performance to use a compute shader, however it will only work for certain platforms. Refer to the Unity docs for more info - https://docs.unity3d.com/Manual/ComputeShaders.html

Area Shading Mat Solid / Gradient (only used when compute shader is not enabled) - These are the materials used on the area shading rectangles.

Area Shading Parent - The empty gameobject parent for the object(s) created for area shading.

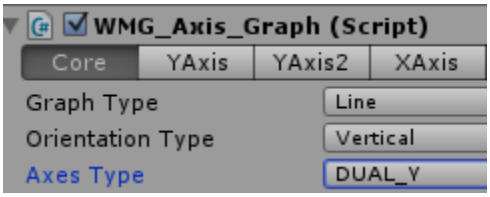
Area Shading Prefab - Instantiated for each area shading rectangle when not using compute shader.
Area Shading CS Prefab - Instantiated as a single rectangle when using compute shader.

The Area Shading Color changes the color for all area shading rectangles for the series.

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.

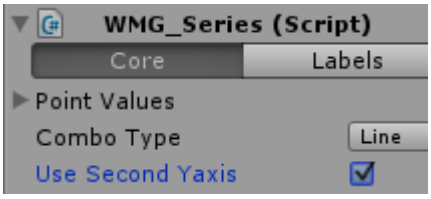
3.10 Dual-Y Axis

To make charts use two different y-axes, set the graph's "Axes Type" to "DUAL_Y". Note that this does not work for charts of horizontal orientation. After setting this, a new "Yaxis2" tab will appear in the inspector, making it possible to configure the secondary y-axis.

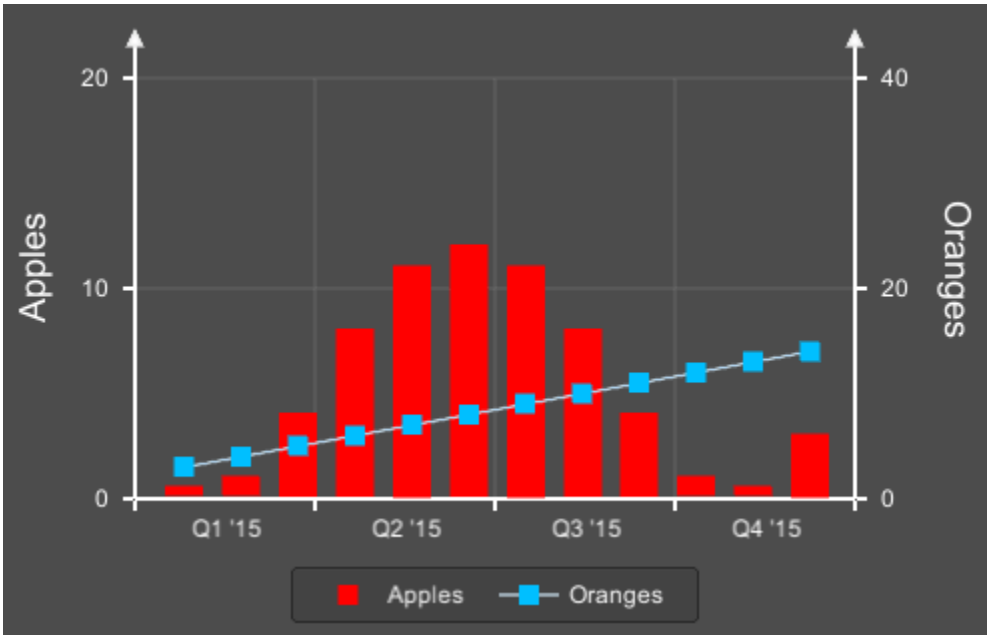


All the settings on the secondary y-axis work the same way as the primary y-axis. The only exception being that the horizontal grid lines are based on the number of ticks of the primary y-axis, and not the secondary y-axis.

Now to have different series use the secondary y-axis instead of the primary y-axis, set "Use Second Yaxis" to true.



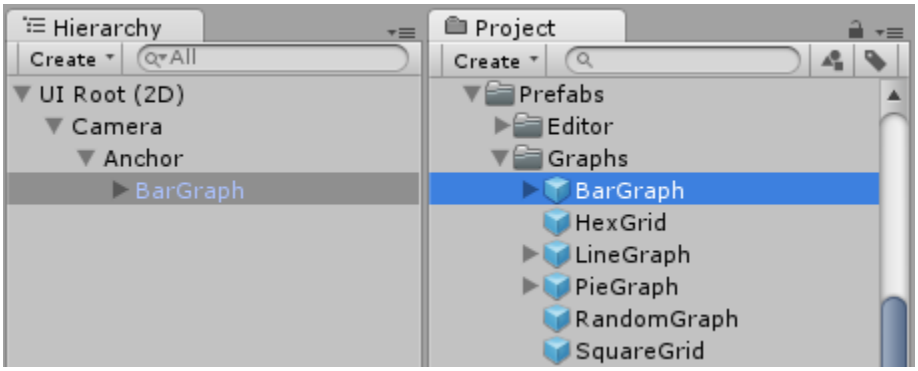
That's pretty much all there is to it. Lastly note that the secondary y-axis title is rotated 180 degrees and anchored on the right side



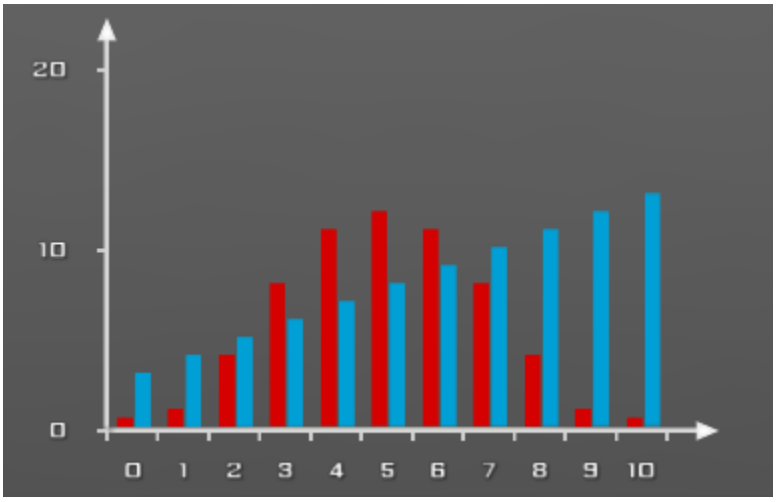
IV. Bar Graphs

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 Useful Series Functions

- public List<GameObject> getPoints()

- public List<GameObject> getLines()

- public Vector2 getNodeValue(WMG_Node aNode)

5.2 Graph Manager Functions

If you are interested in creating your own complex graphs, then you can use WMG_Graph_Manager as the basis for your own custom graph. Most graph maker graphs inherit from the graph manager (even pie graphs and grids).

- Create Node

- Create Link

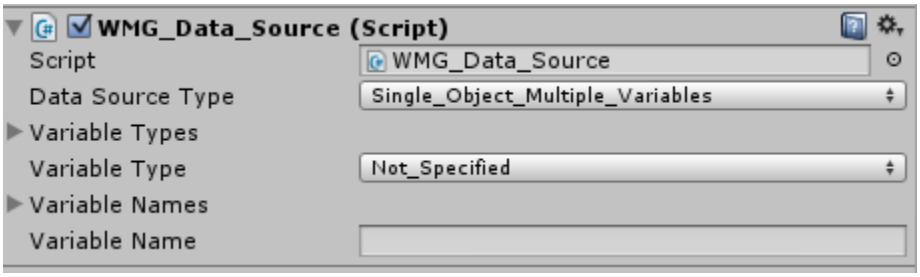
5.3 Data Generator Functions

There are a variety of functions that can be used from the static utility class WMG_Util to create lists of data.

For example you can generate a straight line with GenLinear.

5.4 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.

Check out the documentation for WMG_Data_Source as well as the example scene code X_Dynamic.cs.

Populating data via Play Maker variables

To use data from 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.

5.5 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:

Parameters information for the legend can be found here:
http://stew-soft.com/GraphMaker/Docs/html/class_w_m_g___legend.html

5.6 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

5.7 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)

See here as well as X_Plot_Overtime example scene code for usage example.

Similarly for series 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:

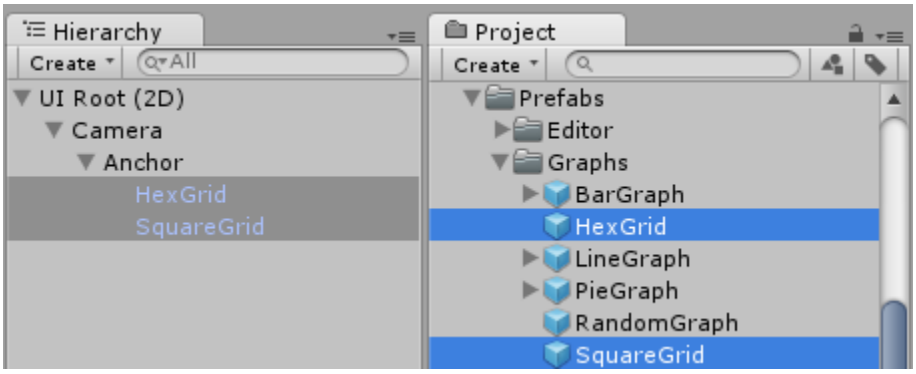
Create a function with the signature:
string myCustomFunction(WMG_Series series, float val);

See here as well as X_Plot_Overtime example scene code for usage example.

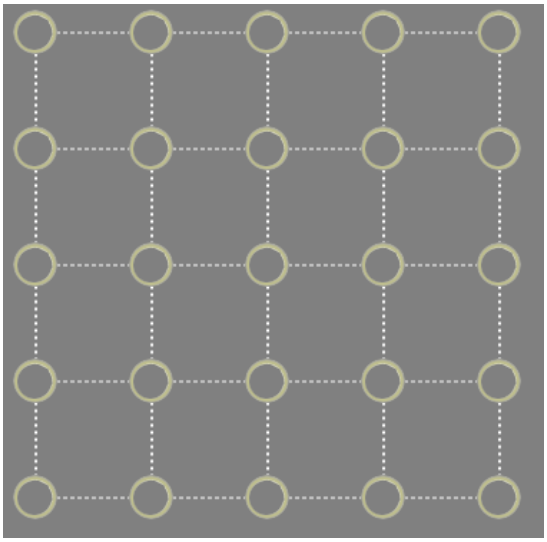
VI. Square / Rect / Hex Grids

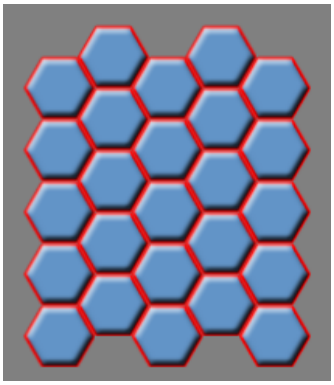
6.1 Getting Started

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



The grids will appear when you play the scene:





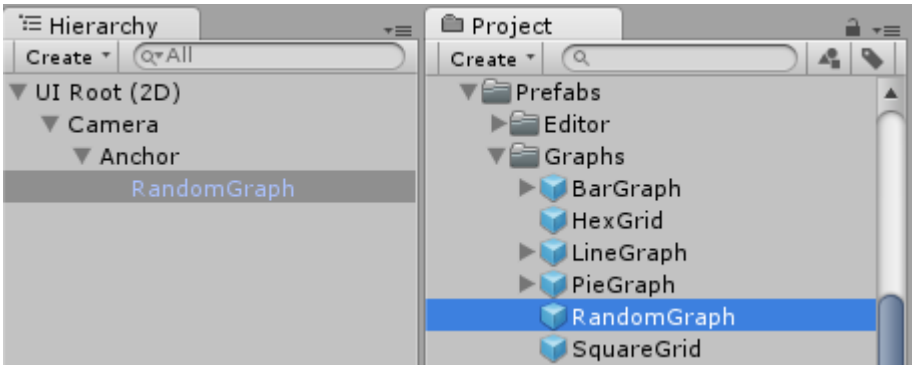
6.2 Parameters

Detailed information for grid parameters can be found under "Properties" section here:
http://stew-soft.com/GraphMaker/Docs/html/class_w_m_g___grid.html

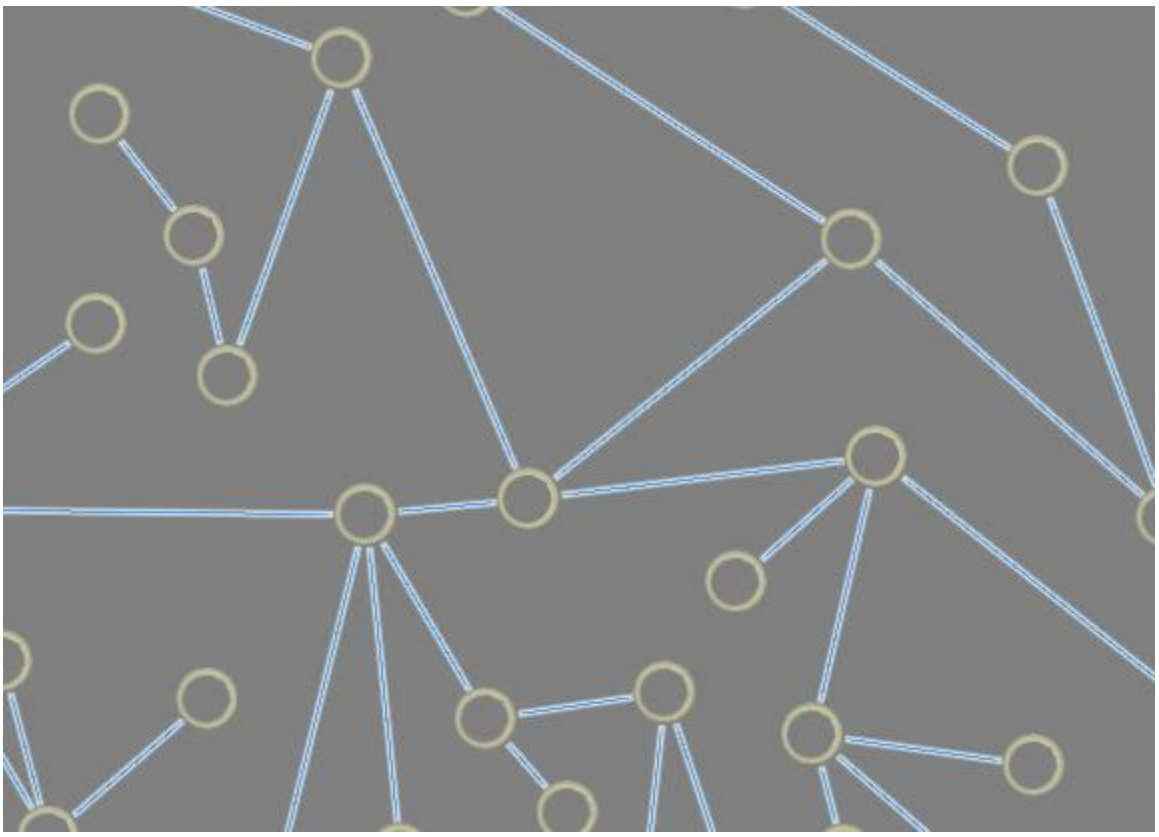
VII. Random Graphs

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

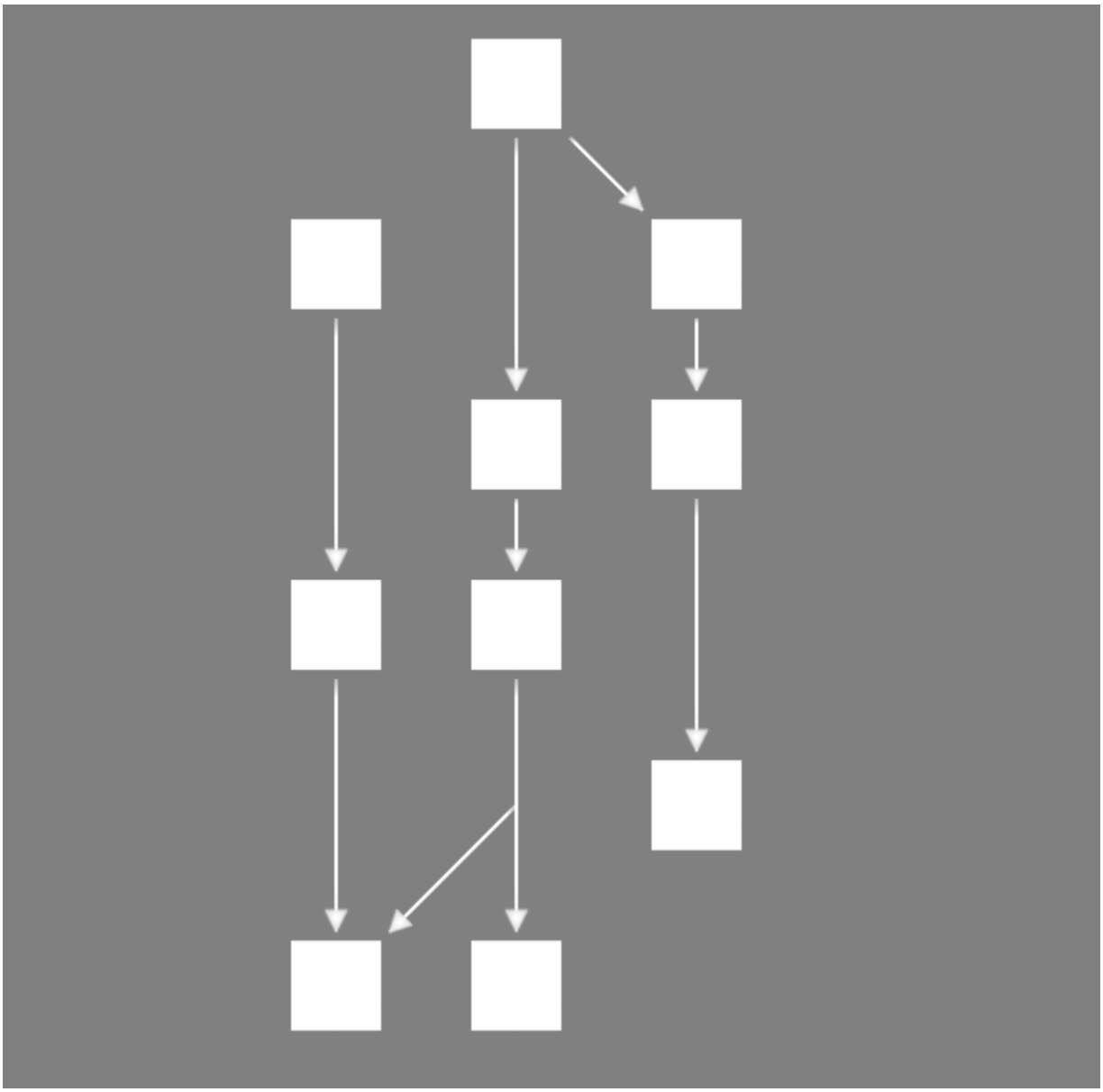
Detailed information for random graph parameters can be found under "Properties" section here:
http://stew-soft.com/GraphMaker/Docs/html/class_w_m_g___random___graph.html

VIII. Hierarchical 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.

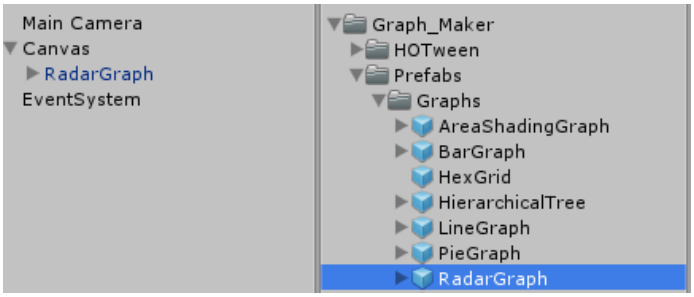
8.3 Parameters

Detailed information for hierarchical tree graph parameters can be found under "Properties" section here: http://stew-soft.com/GraphMaker/Docs/html/class_w_m_g___hierarchical___tree.html

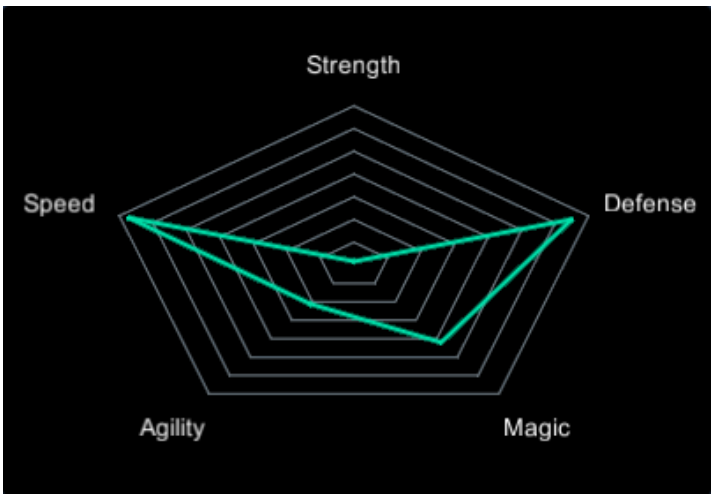
IX. Radar Graphs

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.

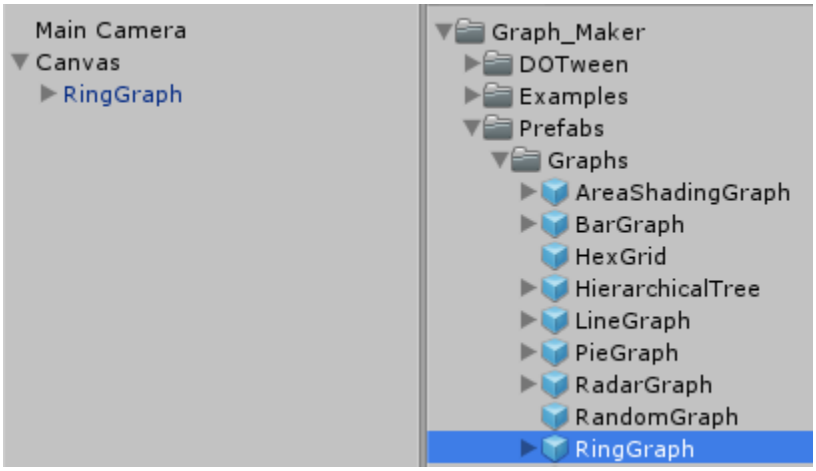
9.3 Parameters

Detailed information for radar graph parameters can be found under "Properties" section here:
http://stew-soft.com/GraphMaker/Docs/html/class_w_m_g___radar___graph.html

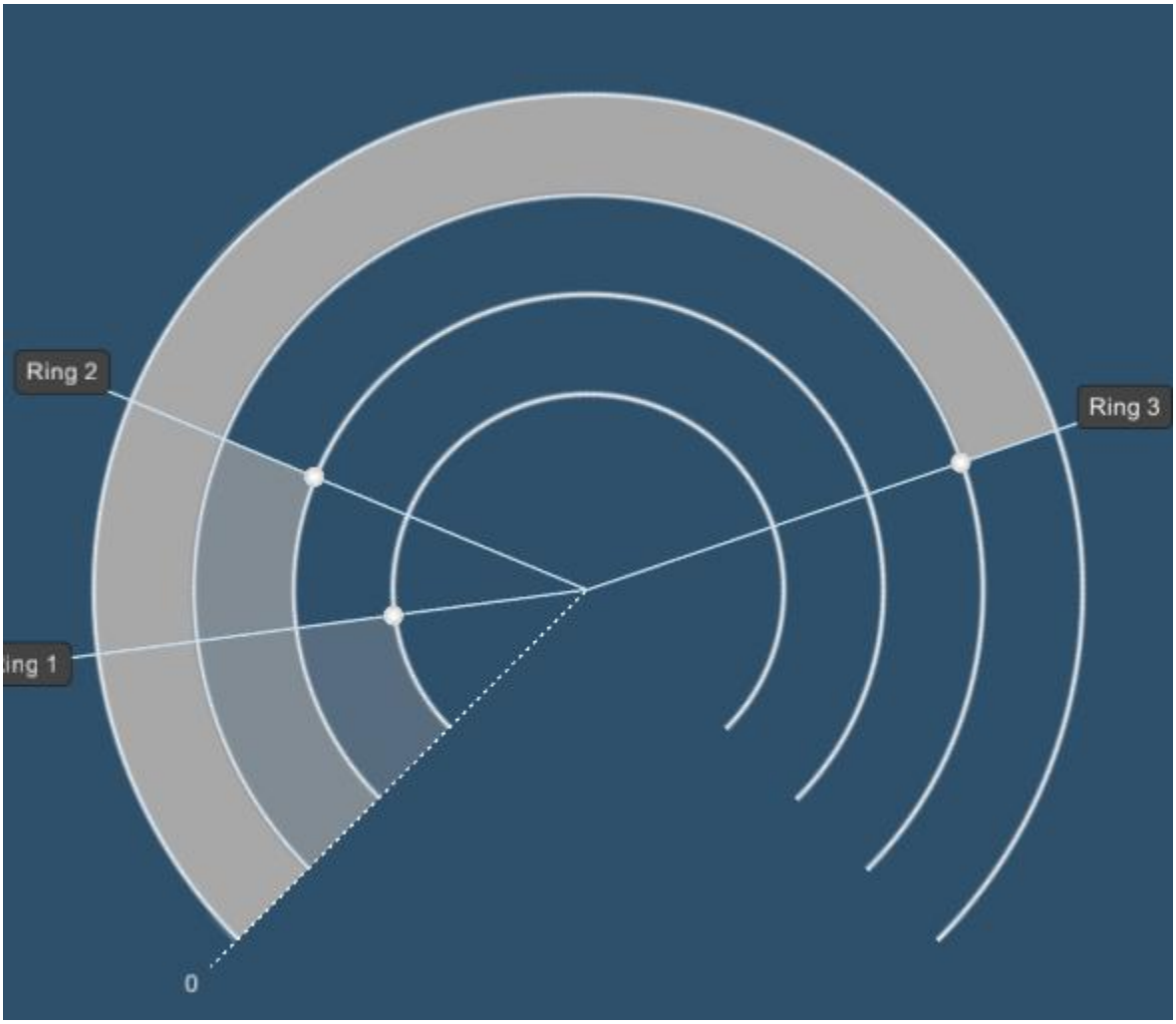
X. Ring Graphs

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.

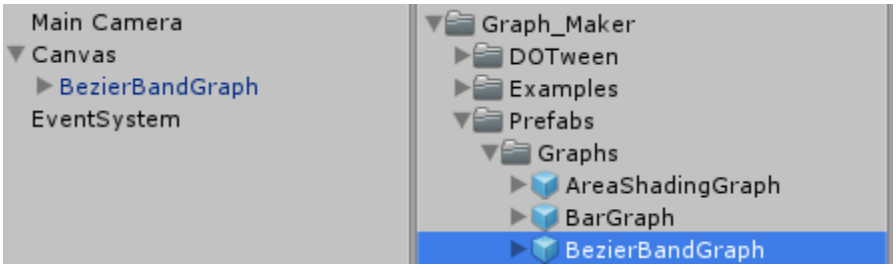
10.3 Parameters

Detailed information for ring graph parameters can be found under "Properties" section here:
http://stew-soft.com/GraphMaker/Docs/html/class_w_m_g___ring___graph.html

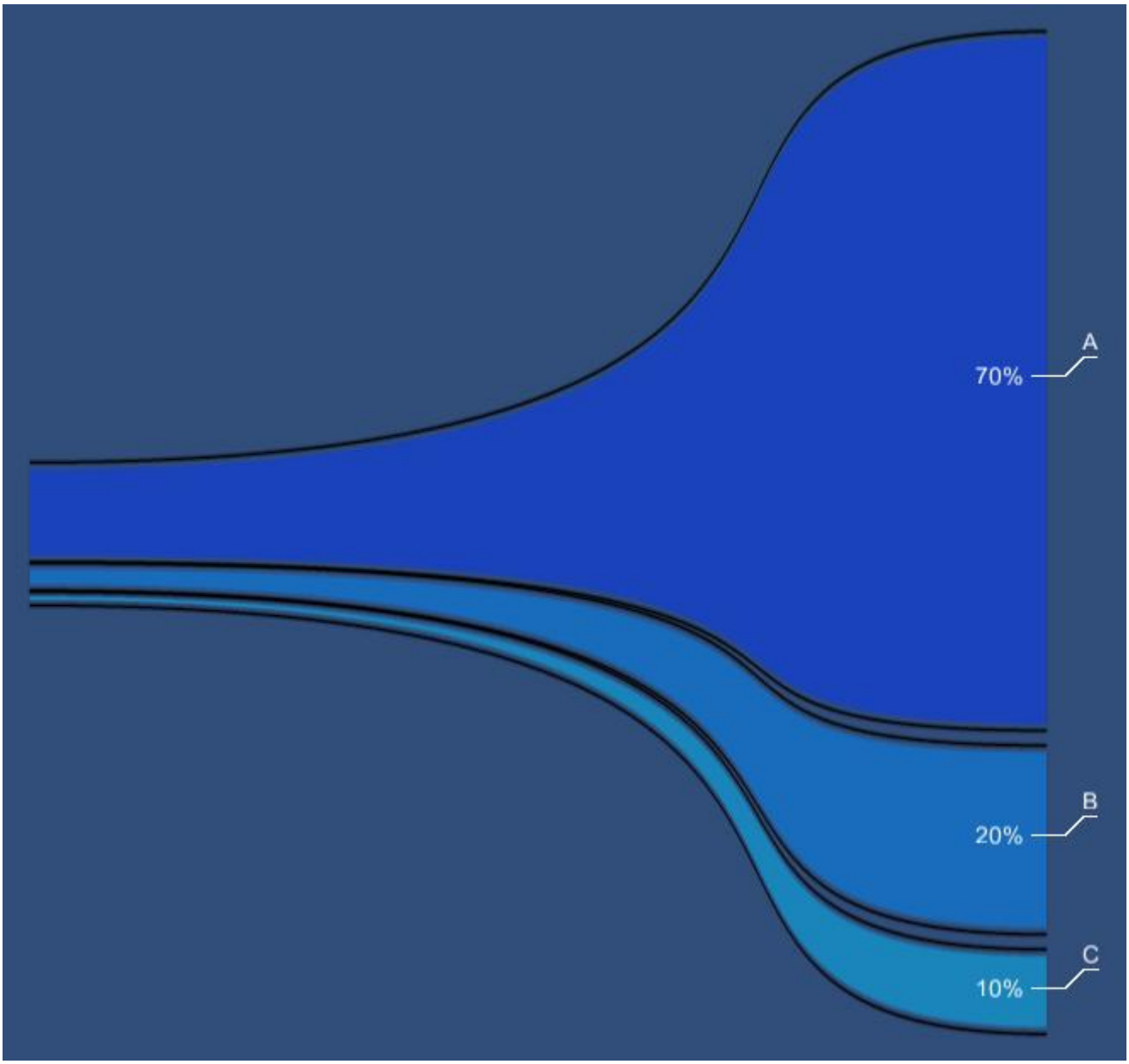
XI. Bezier Band Graphs

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 procedurally generated textures whose colors are set 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 unless a compute shader implementation is added later. A lower resolution setting for the textures (default 2048), will greatly increase the speed, but may not look as good.

11.3 Parameters

Detailed information for bezier band graph parameters can be found under "Properties" section here:
http://stew-soft.com/GraphMaker/Docs/html/class_w_m_g___bezier___band___graph.html