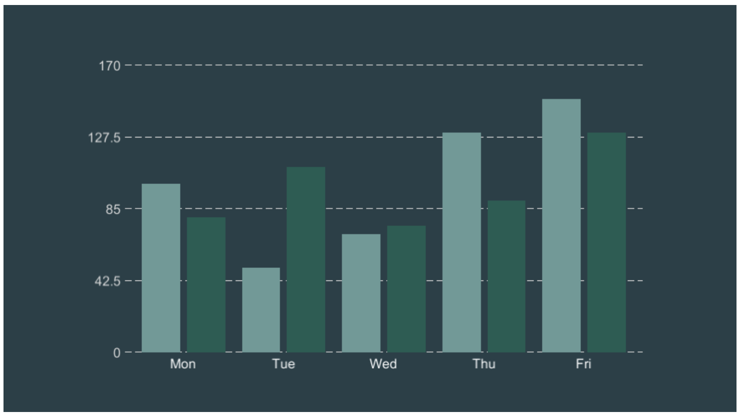
Awesome Charts documentation

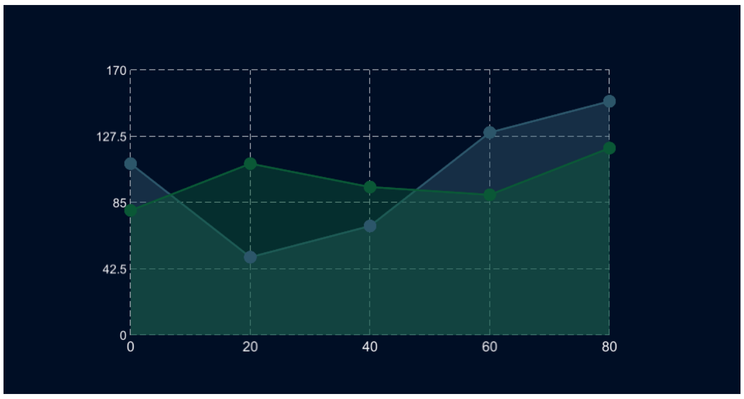
version 1.1.4

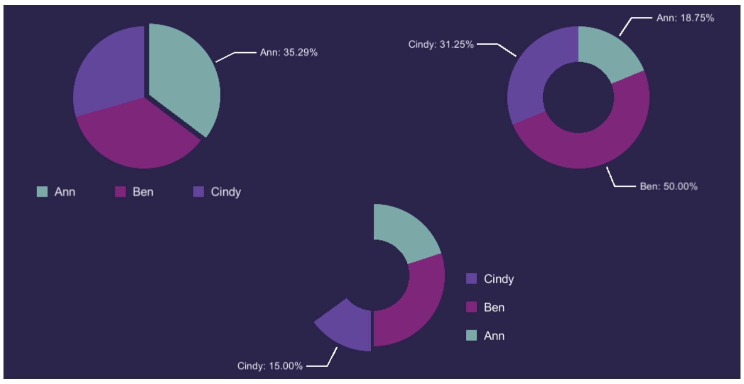
Table of content

1. Introduction

Awesome Charts is a library that aims to help build beautiful charts effortlessly. Main goal is to provide all necessary classes for building 2D, UI canvas charts, that are easy to customize and doesn't require a lot of time to setup. Current version of library supports LineChart, BarChart and PieChart.






Ann: 35.29%
Ann
Ben
Cindy
Cindy: 31.25%
Ann: 18.75%
Ben: 50.00%
Cindy
Ben
Ann
Cindy: 15.00%

2. Requirements

Minimum unity version

5.6.4f1

Dependencies

Library is using some classes from the Unity-UI-Extenstions library. Required files are included and necessary for library proper work. Full source code of Unity-UI-Extensions can by found here https://bitbucket.org/UnityUIExtensions/unity-ui-extensions

3. Axis

The axis classes are responsible for styling and controlling following elements:

- labels (vertical on y axis and horizontal on x axis)
- lines (those lines create grid and a limit lines)

The all configuration is separate for both x and y axis.

Base

Properties mentioned below can be applied for each axis separately.

- **Lines**
    - LineThickness(int value) - width of the lines
    - LineColor(Color color) - color of the lines
    - LinesCount(int count) - number of visible lines (including min and max lines),
      this property affects labels count on given axis as well
- **Bounds**
    - MinAxisValue(int value) - minimum value of chart on given axis
    - MaxAxisValue(int value) - maximum value of chart on given axis
    - AutoAxisMinValue(boolean enabled) - responsible for min axis value calculation settings.
      If disabled, MinAxisValue will be used, otherwise value will be calculated depending on provided data.
    - AutoAxisMaxValue(boolean enabled) - responsible for max axis value calculation settings.
      If disabled, MaxAxisValue will be used, otherwise value will be calculated depending on provided data.
- **Labels**
    - LabelSize(int size) - font size of labels
    - LabelMargin(int margin) - margin between label and axis line
    - LabelColor(Color color) - font color of labels
    - LabelFont(Font font) - font of labels
    - LabelFontStyle(FontStyle style) - style of font used in labels
- **Drawing**
    - ShouldDrawLabels(boolean draw) - if set to true labels will be drawn. Default: true
    - ShouldDrawLines(boolean draw) - if set to true lines will be drawn. Default: true
    - DashedLines(boolean draw) - if set to true lines will be dashed. Default: true

X Axis

X axis specific properties

- **Lines**
    - LineStep(int step) - use this, instead of LinesCount in order to place lines on positions that are multiplication of step value instead of fixed lines count.

Y Axis

Does not have any specific methods at this moment.

4. Charts

Chart classes are responsible for drawing vertical and horizontal axis, together with chart content. This chapter focus on common methods. Data setting instruction is provided on the following chapters, as it differs for line and bar chart.

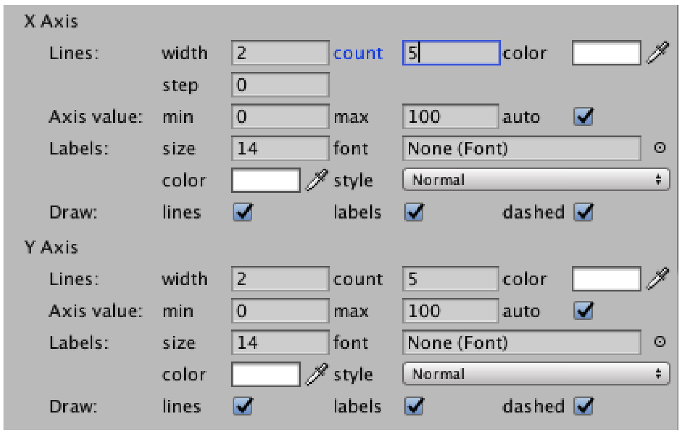
Public properties

- XAxis(XAxis axis) - x axis configuration object, can be edited or replaced with new configuration
- YAxis(YAxis axis) - y axis configuration object, can be edited or replaced with new configuration
- LegendView - allow you to connect LegendView script, so chart will be able to fill it automatically based on data sets.

Examples

Axis can be configured through inspector or by code.

Inspector config



X Axis

Lines:	width	2	count	5	color	
	step	0				
Axis value:	min	0	max	100	auto	✓
Labels:	size	14	font	None (Font)		
	color		style	Normal		
Draw:	lines	✓	labels	✓	dashed	✓

Y Axis

Lines:	width	2	count	5	color	
Axis value:	min	0	max	100	auto	✓
Labels:	size	14	font	None (Font)		
	color		style	Normal		
Draw:	lines	✓	labels	✓	dashed	✓

Naming convention should be consistent with public methods of X and Y axis. Lines width property is equal to LineThickness(int value) property.

Code config

Changing axis settings in code can be done by editing current config or setting new one.

```
// accessing and editing current config
chart.XAxis.DashedLine = true;
chart.XAxis.LineThickness = 2;
chart.XAxis.LabelColor = Color.white;
chart.XAxis.LabelSize = 20;

// setting new config
YAxis axisConfig = new YAxis {
    DashedLine = false,
    LineThickness = 2,
    LabelColor = Color.red,
    LabelSize = 20,
    AutoAxisValue = false,
    MinAxisValue = 0,
    MaxAxisValue = 1000f
};
chart.YAxis = axisConfig;
```

5. BarChart

Bar chart is responsible for showing data in bars. Each entry represents one bar, it consist of value and position. Chart is capable of displaying multiple series of data.

Config

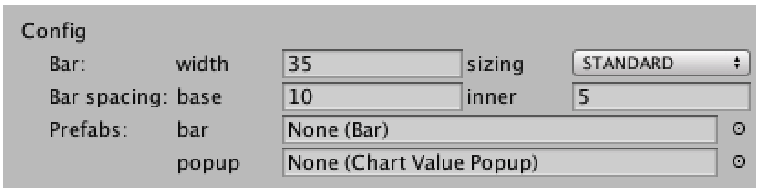
BarChart can be configured through Config property.

Public properties

- BarWidth(int width) - width of each bar
- BarSpacing(int spacing) - space between bars on different position
- InnerBarSpacing(int spacing) - space between bars on the same position in different data sets
- SizingMethod(BarSizingMethod method) - defines how bar width is calculated, available options:
  - *STANDARD* - bar width is never grater than BarWidth property value
  - *SIZE_TO_FIT* - bar width is expanded to fill available space, respecting spacings
- BarPrefab(Bar prefab) - custom prefab of single bar
- PopupPrefab(ChartValuePopup prefab) - custom popup prefab, showing value after user clicks on bar
- BarChartClickAction(BarChartAction action) - callback on bar instance click

Examples
Chart can be configured through inspector or by code.

Inspector config



Config

Bar:	width	35	sizing	STANDARD
Bar spacing:	base	10	inner	5
Prefabs:	bar	None (Bar)		
	popup	None (Chart Value Popup)		

Naming convention should be consistent with public properties. Spacing base property is equal to BarSpacing(int spacing).

Code config
Chart configuration in code can be done by editing current config or setting new one.

```
// accessing and editing current config
chart.Config.BarWidth = 45;
chart.Config.BarSpacing = 15;
chart.Config.SizingMethod = BarSizingMethod.STANDARD;

// setting new config
BarChartConfig config = new BarChartConfig {
    BarWidth = 40,
    BarSpacing = 20,
    InnerBarSpacing = 10,
    SizingMethod = BarSizingMethod.SIZE_TO_FIT
};
chart.Config = config;
```

Data

BarChart can be supplied with data through BarData class.

BarData - represents whole set of data provided to BarChart. It contains list of BarDataSet objects.

Properties

- DataSets(List<BarDataSet> dataSets) - list of DataSets
- CustomLabels(List<String> labels) - list of custom label values, that will be displayed instead of position value on X Axis.

BarDataSet - single data set represents list of entries to display in one group, one DataSet must have entries with different positions

Properties

- Entries(List<BarEntry> entries) - list of entries, each of them represents one bar on chart
- Title(string title) - title of data set, used to create LegendEntry
- BarColors(List<Color> colors) - list of bar colors, each bar can have its own color

BarEntry - holds information about position and value of the bar. Each entry belongs to single data set.

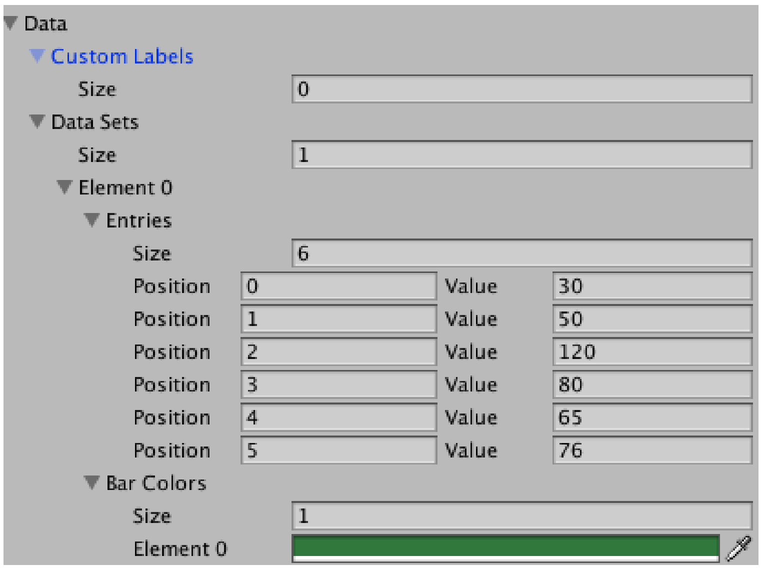
Properties

- Value(float value) - value of bar, it defines the hight of given bar
- Position(int position) - position of bar in chart

Examples

Chart data can be supplied through inspector or by code.

Inspector



Data
Custom Labels
Size: 0
Data Sets
Size: 1
Element 0
Entries
Size: 6

Position	0	Value	30
Position	1	Value	50
Position	2	Value	120
Position	3	Value	80
Position	4	Value	65
Position	5	Value	76

Bar Colors
Size: 1
Element 0: 🟩

Code

```
// Create data set for entries
BarDataSet set = new BarDataSet();

// Add entries to data set
set.AddEntry(new BarEntry(0, 100f));
set.AddEntry(new BarEntry(1, 50));
set.AddEntry(new BarEntry(2, 70));
set.AddEntry(new BarEntry(3, 130));
set.AddEntry(new BarEntry(4, 80));
set.AddEntry(new BarEntry(5, 150));

// Set bars color
set.BarColors.Add(Color.red);

// Add data set to chart data
chart.GetChartData().DataSets.Add(set);

// Refresh chart after data change
chart.SetDirty();
```

Whether you set or change data by code, SetDirty() method **must** be called in order to refresh chart.

6. LineChart

Line chart is responsible for showing linear data connected by lines. Each entry represents one point, it consist of value and position. Chart is capable of displaying multiple series of data.

Config

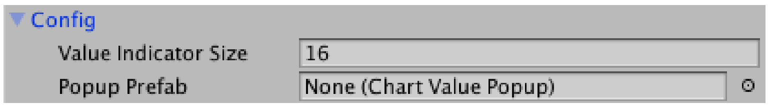
LineChart can be configured through Config property.

Public properties

- ValueIndicatorSize(int size) - size of point indicator on chart
- PopupPrefab(ChartValuePopup prefab) - custom popup prefab, showing value after user clicks on point value indicator
- ShowValueIndicators(bool show) - decides whether line entry indicators (dots) should be draw or not
- OnValueClickAction(LineChartAction action) - callback on value indicator click

Examples
Chart can be configured through inspector or by code.

Inspector config



Config
Value Indicator Size 16
Popup Prefab None (Chart Value Popup)

Naming convention should be consistent with public properties.

Code config
Chart configuration in code can be done by editing current config or setting new one.

```
// accessing and editing current config
chart.Config.ValueIndicatorSize = 16;
chart.Config.PopupPrefab = myCustomPopup;

// setting new config
LineChartConfig config = new LineChartConfig {
    ValueIndicatorSize = 16,
    PopupPrefab = myCustomPopup
};
chart.Config = config;
```

Data

LineChart can be supplied with data through LineData class.

LineData - represents whole set of data provided to LineChart. It contains list of LineDataSet objects.

Properties

- DataSets(List<LineDataSet> dataSets) - list of DataSets
- CustomLabels(List<String> labels) - list of custom label values, that will be displayed instead of position value on X Axis.

LineDataSet - single data set represents list of entries to display in one group, one DataSet must have entries with different positions

Properties

- Entries(List<LineEntry> entries) - list of entries, each of them represents one bar on chart
- Title(string title) - title of data set, used to create LegendEntry
- LineThickness(float thickness) - thickness of line drawn between value points
- LineColor(Color color) - color of line drawn between value points
- FillColor(Color color) - color of background drawn between line and bottom of chart
- FillTexture(Texture texture) - texture of background drawn between line and bottom of chart
- UseBezier(Bool use) - decides whether line between points should be straight or bezier interpolated

LineEntry - holds information about position and value of the bar. Each entry belongs to single data set.

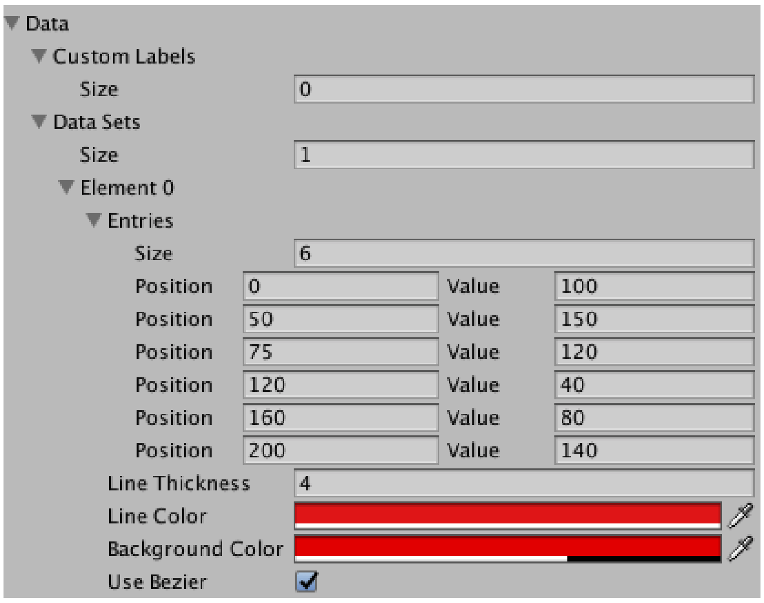
Properties

- Value(float value) - value of point, it defines the y position on chart
- Position(float position) - position of point, it defines x position in chart

Examples

Chart data can be supplied through inspector or by code.

Inspector



```
▼ Data
  ▼ Custom Labels
      Size                0
  ▼ Data Sets
      Size                1
    ▼ Element 0
      ▼ Entries
          Size            6
          Position  0     Value  100
          Position  50    Value  150
          Position  75    Value  120
          Position  120   Value  40
          Position  160   Value  80
          Position  200   Value  140
        Line Thickness    4
        Line Color        🟥
        Background Color  🟥
        Use Bezier        ✓
```

Code

```
    // Create data set for entries
    LineDataSet set = new LineDataSet();

    // Add entries to data set
    set.AddEntry(new LineEntry(0, 100f));
    set.AddEntry(new LineEntry(100, 50));
    set.AddEntry(new LineEntry(150, 70));
    set.AddEntry(new LineEntry(180, 130));
    set.AddEntry(new LineEntry(230, 80));
    set.AddEntry(new LineEntry(300, 150));

    // Configure line
    set.LineColor = Color.red;
    set.LineThickness = 4;
    set.UseBezier = true;

    // Add data set to chart data
    chart.GetChartData().DataSets.Add(set);

    // Refresh chart after data change
    chart.SetDirty();
```

Whether you set or change data by code, SetDirty() method **must** be called in order to refresh chart.

7. PieChart

Pie chart is a circular graphic divided into slices . Each entry represents one slice, it consist of value and it size is proportional to percentage value of sum of all entry values.

Config

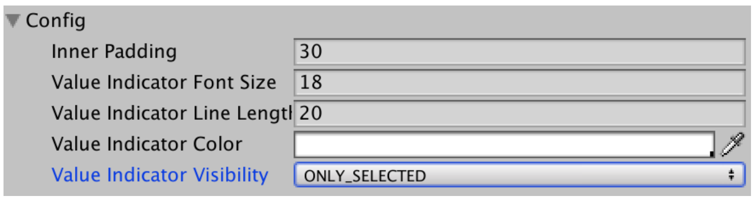
PieChart can be configured through Config property.

Public properties

- InnerPadding(int padding) - size of inner circle, this property can be used to create a hole in chart
- ValueIndicatorFontSize(int size) - size of value indicator label
- ValueIndicatorLineLength(int size) - length of value indicator line connecting chart and value label
- ValueIndicatorColor(Color color) - color of value indicator line and label
- ValueIndicatorVisibility(ValueIndicatorVisibilityMode mode) - defines when value indicator should be visible, available options:
    - ALWAYS - indicator will be visible for all chart entries
    - ONLY_SELECTED - indicator will be visible for selected entries
    - NEVER - indicator will not be visible for any entry

Examples
Chart can be configured through inspector or by code.

Inspector config



Config	
Inner Padding	30
Value Indicator Font Size	18
Value Indicator Line Lengtl	20
Value Indicator Color	
Value Indicator Visibility	ONLY_SELECTED

Naming convention should be consistent with public properties.

Code config
Chart configuration in code can be done by editing current config or setting new one.

```
    // accessing and editing current config
    pieChart.Config.InnerPadding = 10;
    pieChart.Config.ValueIndicatorFontSize = 18;
    pieChart.Config.ValueIndicatorLineLength = 30;
    pieChart.Config.ValueIndicatorColor = Color.white;
    pieChart.Config.ValueIndicatorVisibility = 
PieChartConfig.ValueIndicatorVisibilityMode.ONLY_SELECTED;

    // setting new config
    PieChartConfig config = new PieChartConfig {
        InnerPadding = 10,
        ValueIndicatorFontSize = 10,
        ValueIndicatorLineLength = 10,
        ValueIndicatorColor = Color.white,
        ValueIndicatorVisibility =
PieChartConfig.ValueIndicatorVisibilityMode.ONLY_SELECTED
    };
    pieChart.Config = config;
```

Data

PieChart can be supplied with data through PieData class.

PieData - represents whole set of data provided to PieChart. It contains a PieDataSet object.

Properties

- DataSet(PieDataSet dataSet) - container for pie chart entries

PieDataSet - data set represents list of entries to display in chart

Properties

- Entries(List<PieEntry> entries) - list of entries, each of them represents one slice on chart
- Title(string title) - title of data set
- ValuesAsPercentages(bool valueAsPercentages) -if true, value of entry will be treated as percent value of full chart

PieEntry - holds information about position and value of the bar. Each entry belongs to single data set.

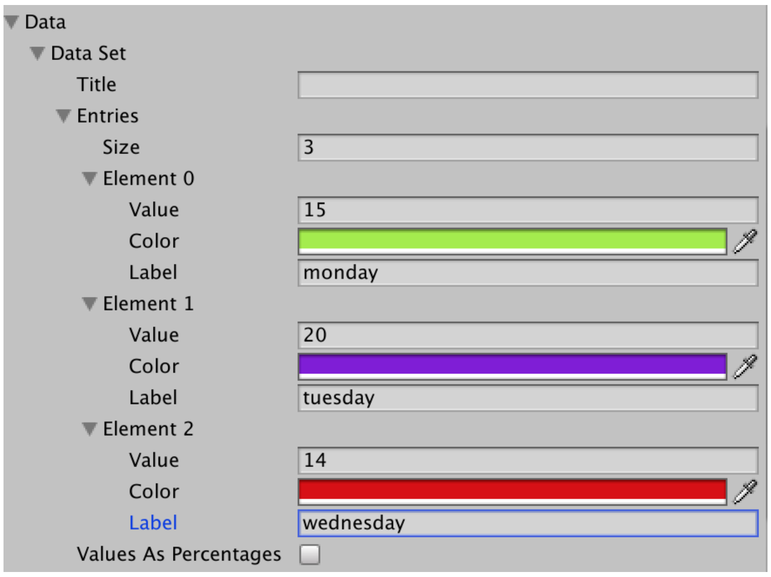
Properties

- Value(float value) - value of chart slice, it defines the proportion taken by the slice on chart
- Label(String label) - label of entry, used by value indicator and LegendView
- Color(Color color) - color of pie entry slice

Examples

Chart data can be supplied through inspector or by code.

Inspector



```
Data
  Data Set
    Title
    Entries
      Size                      3
      Element 0
        Value                   15
        Color                   🟩
        Label                   monday
      Element 1
        Value                   20
        Color                   🟪
        Label                   tuesday
      Element 2
        Value                   14
        Color                   🟥
        Label                   wednesday
    Values As Percentages       ☐
```

Code

```
// Create data set for entries
PieDataSet set = new PieDataSet ();

// Add entries to data set
set.AddEntry (new PieEntry (20, "Entry 1", Color.red));
set.AddEntry (new PieEntry (30, "Entry 2", Color.green));
set.AddEntry (new PieEntry (25, "Entry 3", Color.blue));

// Add data set to chart data
pieChart.GetChartData ().DataSet = set;

// Refresh chart after data change
pieChart.SetDirty ();
```

Whether you set or change data by code, SetDirty() method **must** be called in order to refresh chart.

Public methods

Those methods are available for PieChart only:

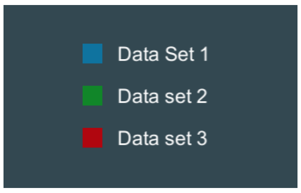
- SelectEntryAtPosition (int position) - makes entry at position selected, that entry slice will be larger and split from other slices
- DeselectEntryAtPosition (int position) - remove selection from entry at position
- IsEntryAtPositionSelected (int position) - returns true, if entry at position is selected
- AddEntryClickDelegate (EntryClickDelegate clickDelegate) - allow to add entry click listener
- RemoveEntryClickDelegate (EntryClickDelegate clickDelegate) - removes entry click listener

8. Legend

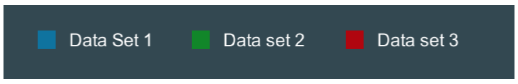
Legend is able to present information about data sets used by chart. Each legend entry is represented by title and icon.

Public properties

- IconSize(int size) - size of an legend entry icon
- IconSpacing(int spacing) - space between legend entry icon and title
- IconImage(Sprite sprite) sprite of an legend entry icon
- TextSize(int size) - font size of legend entry title
- TextColor(Color color) - text color of legend entry title
- TextFont(Font font) - text font of legend entry title
- ItemHeight(int height) - height of each entry item
- ItemWidth(int width) - width of each entry item
- ItemSpacing(int spacing) - horizontal or vertical (dependent on Orientation) space between each entry item
- Alignment(LEGEND_ALIGNMENT alignment) - decides how legend items will align to LegendView. Can be: TOP_LEFT, TOP_RIGHT, BOTTOM_LEFT, BOTTOM_RIGHT
- Orientation(LEGEND_ORIENTATION orientation) - defines if legend items are drawn vertically or horizontally. Can be VERTICAL, HORIZONTAL.
- Entries(List<LegendEntry> entries) - list of legend entries defining titles and colors.



Vertical legend

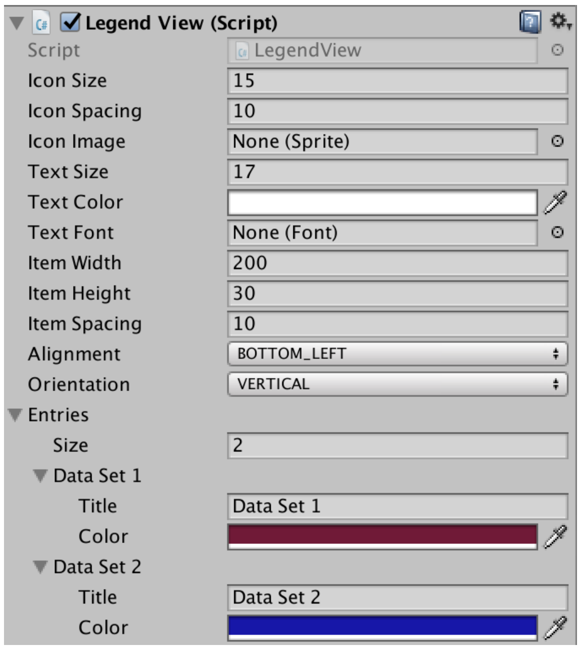


Horizontal legend

Examples

Legend can be configured through inspector or by code.

Inspector config



Legend View (Script)

Script	LegendView
Icon Size	15
Icon Spacing	10
Icon Image	None (Sprite)
Text Size	17
Text Color	⬜
Text Font	None (Font)
Item Width	200
Item Height	30
Item Spacing	10
Alignment	BOTTOM_LEFT
Orientation	VERTICAL
Entries	
Size	2
Data Set 1	
Title	Data Set 1
Color	🟥
Data Set 2	
Title	Data Set 2
Color	🟦

9. Migrations

This section describes breaking changes and steps to migrate to certain library versions.

1.1.4

Axis model property AutoAxisValue has been split into 2 properties: AutoAxisMinValue and AutoAxisMaxValue. After migration those values will be set to false by default. This change can require your setup of this properties according to your needs.

1.0.0 - 1.1.3

No actions needed

10. Example scenes

Source code is provided with example scenes and scripts, those can be found at Awesowe Charts/examples/ folder

Examples

- BarChart 1 - shows how to dynamically manipulate data of BarChart
- BarChart 2 - shows how to configure BarChart by code
- LineChart 1 - shows how to dynamically manipulate data of LineChart
- LineChart 2 - shows how to configure LineChart by code
- PieChart 1 - shows how to dynamically manipulate data of PieChart
- PieChart 2 - shows how to configure PieChart by code