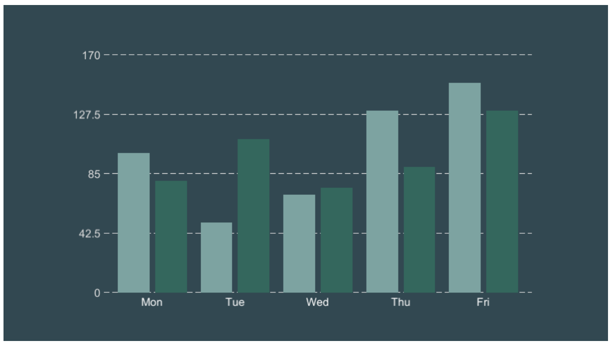
Awesome Charts documentation

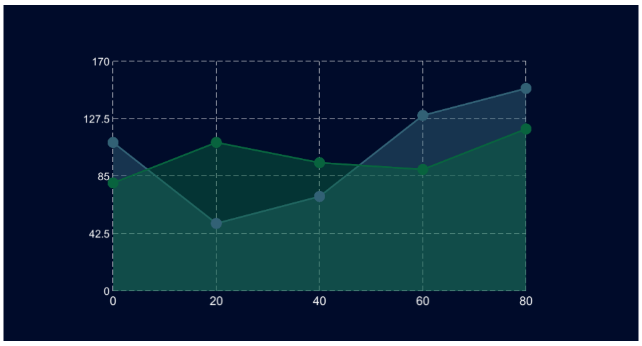
version 1.1.5

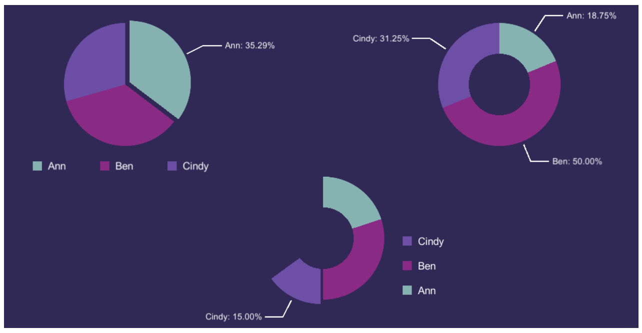
Table of content

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.







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 horizontal and vertical axis.

Axis Config

Public properties

- VerticalAxisConfig(SingleAxisConfig config) - config of vertical axis
- HorizontalAxisConfig(SingleAxisConfig config) - config of horizontal axis

SingleAxisConfig

Base axis config class, holds common properties for all config types.

- LabelsConfig(AxisLabelConfig config) - used to control axis labels styling
- Bounds(AxisValue bounds) - defines minimum and maximum value of given axis
- AxisLabelPrefab(AxisLabel prefab) - allows to provide custom prefab for axis label

AxisLabelConfig

- 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

AxisValue

- Min(int value) - minimum value of chart on given axis
- Max(int value) - maximum value of chart on given axis
- MinAutoValue(boolean enabled) - responsible for min axis value calculation settings.
  If disabled, MinAxisValue will be used, otherwise value will be calculated depending on provided data.
- MaxAutoValue(boolean enabled) - responsible for max axis value calculation settings.
  If disabled, MaxAxisValue will be used, otherwise value will be calculated depending on provided data.

LinearSingleAxisConfig

Config used by LineChart vertical and horizontal axis, as well as BerChart vertical axis.

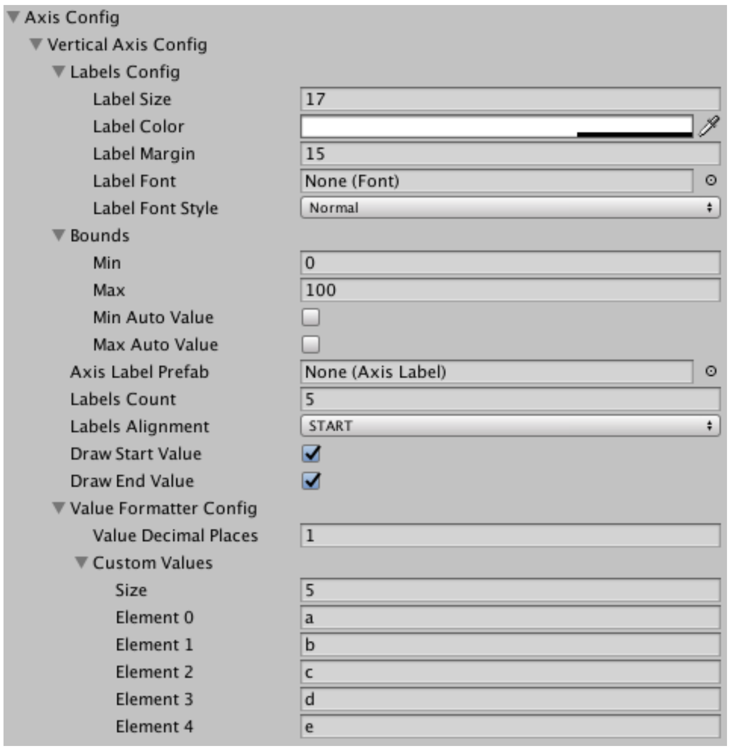
- LabelsCount(int count) - number of labels on given axis
- LabelsAlignment(AxisLabelGravity alignment) - defines how to align labels to the chart, can be either START or END
- DrawStartValue(bool draw) - defines whether to draw label on the beginning of the chart or not
- DrawEndValue(bool draw) - defines whether to draw label on the ending of the chart or not
- ValueFormatterConfig(BasicValueFormatterConfig config) - allows to control label values formatting or define custom texts for labels

BarSingleAxisConfig

Config used by BarChart horizontal axis.

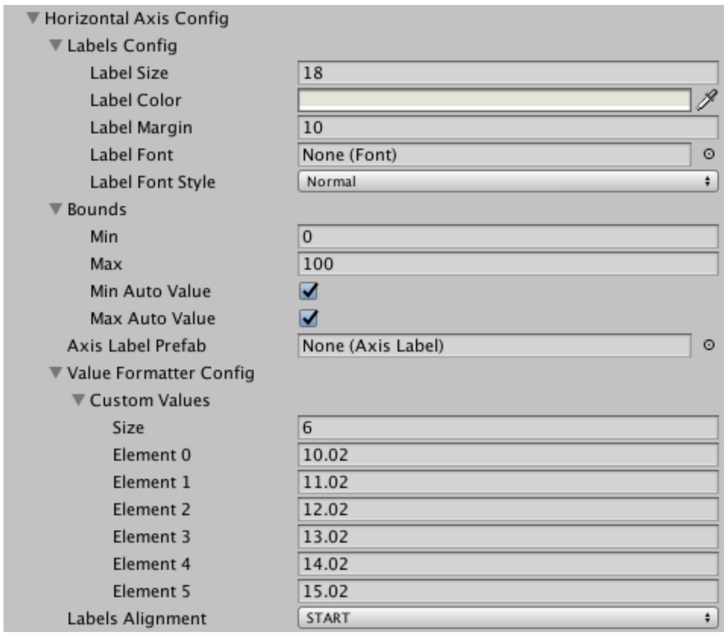
- LabelsAlignment(AxisLabelGravity alignment) - defines how to align labels to the chart, can be either START or END
- ValueFormatterConfig(BarAxisValueFormatterConfig config) - allows to define custom texts for labels

Linear single axis editor



▼ Axis Config	
▼ Vertical Axis Config	
▼ Labels Config	
Label Size	17
Label Color	
Label Margin	15
Label Font	None (Font)
Label Font Style	Normal
▼ Bounds	
Min	0
Max	100
Min Auto Value	☐
Max Auto Value	☐
Axis Label Prefab	None (Axis Label)
Labels Count	5
Labels Alignment	START
Draw Start Value	✓
Draw End Value	✓
▼ Value Formatter Config	
Value Decimal Places	1
▼ Custom Values	
Size	5
Element 0	a
Element 1	b
Element 2	c
Element 3	d
Element 4	e

Bar single axis editor



Horizontal Axis Config
- Labels Config

Label Size	18
Label Color	
Label Margin	10
Label Font	None (Font)
Label Font Style	Normal

- Bounds

Min	0
Max	100
Min Auto Value	✓
Max Auto Value	✓

Axis Label Prefab	None (Axis Label)

- Value Formatter Config
  - Custom Values

Size	6
Element 0	10.02
Element 1	11.02
Element 2	12.02
Element 3	13.02
Element 4	14.02
Element 5	15.02

Labels Alignment	START

4. Grid and Frame

This chapter describes usage of chart grid and frame.

- Grid is represented by crossing lines behind the graph.
- Frame is defined as chart edge aligned lines.

Grid

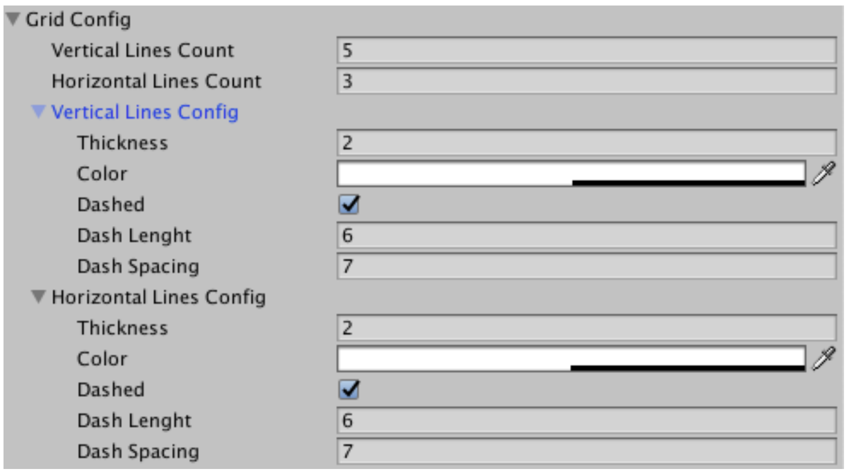
Public properties

- VerticalLinesCount(int count) - number of vertical lines in the grid
- HorizontalLinesCount(int count) - number of horizontal lines in the grid
- VerticalLinesConfig(GridLineConfig config) - config of vertical lines
- HorizontalLinesConfig(GridLineConfig config) - config of vertical lines

GridLineConfig

Public properties

- Thickness(int thickness) - line width
- Color(Color color) - line color
- Dashed(bool dashed) - defines whether grid line should be dashed or not
- DashLength(int length) - if grid line is dashed, this property is used as length of each section
- DashSpacing(int spacing) - if grid line is dashed, this property is used as spacing between sections

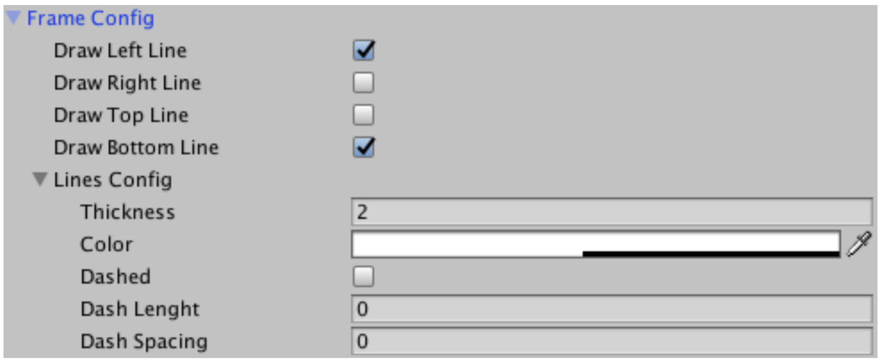


▼ Grid Config	
Vertical Lines Count	5
Horizontal Lines Count	3
▼ Vertical Lines Config	
Thickness	2
Color	
Dashed	✓
Dash Lenght	6
Dash Spacing	7
▼ Horizontal Lines Config	
Thickness	2
Color	
Dashed	✓
Dash Lenght	6
Dash Spacing	7

Frame

Public properties

- DrawLeftLine(bool draw) - if true, line on the left edge of graph will be drawn
- DrawRightLine(bool draw) - if true, line on the right edge of graph will be drawn
- DrawTopLine(bool draw) - if true, line on the top edge of graph will be drawn
- DrawBottomLine(bool draw) - if true, line on the bottom edge of graph will be drawn
- VerticalLinesConfig(GridLineConfig config) - frame lines config, same as for grid lines



Frame Config	
Draw Left Line	✓
Draw Right Line	☐
Draw Top Line	☐
Draw Bottom Line	✓
Lines Config	
Thickness	2
Color	
Dashed	☐
Dash Lenght	0
Dash Spacing	0

5. Charts

Current version of Awesome Charts supports LineChart, BarChart and PieChart. This chapter will focus on common features. Chart specific configuration is described in the following chapters.

Important note

When chart data or config is changed from the source code, SetDirty() method has to be called on chart instance, in order to make chart redraw.

All charts

Public properties

- LegendView - allow you to connect LegendView script, so chart will be able to fill it automatically based on data sets.

Axis based charts (LineChart, BarChart)

Public properties

- GridConfig(GridConfig config) - defines how grid is rendered, described detailed in Grid and Frame chapter
- FrameConfig(GridFrameConfig config) - defines how chart frame is rendered, described detailed in Grid and Frame chapter
- AxisConfig(LineChartAxisConfig/BarChartAxisConfig config) - defines how axis behaves and render, described detailed in Axis chapter

6. 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.

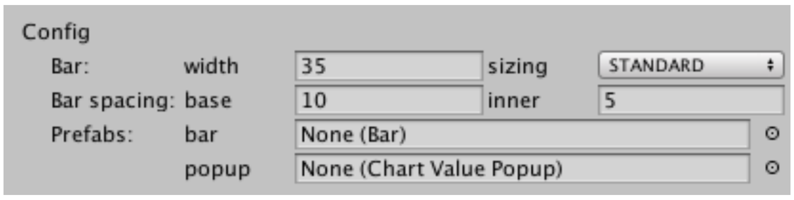
Public properties

- 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

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.

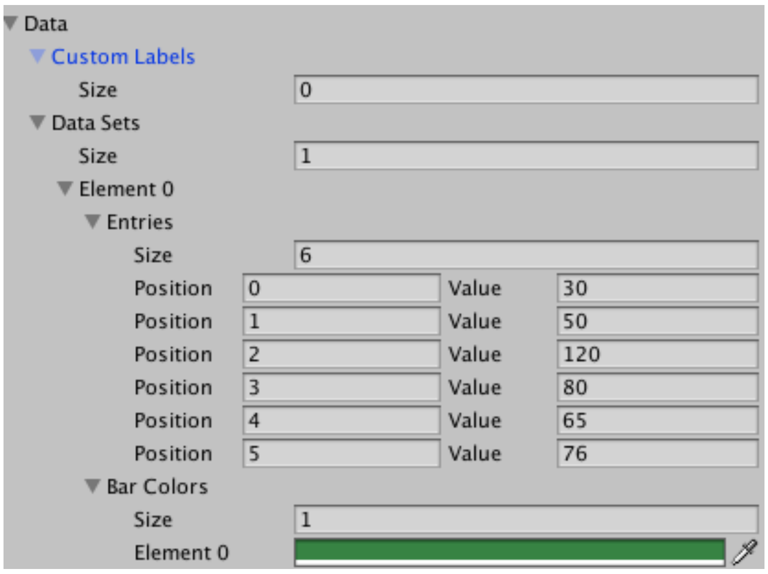
Properties

- 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 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.

7. 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.

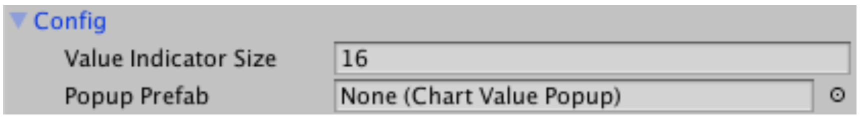
Public properties

- 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

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.

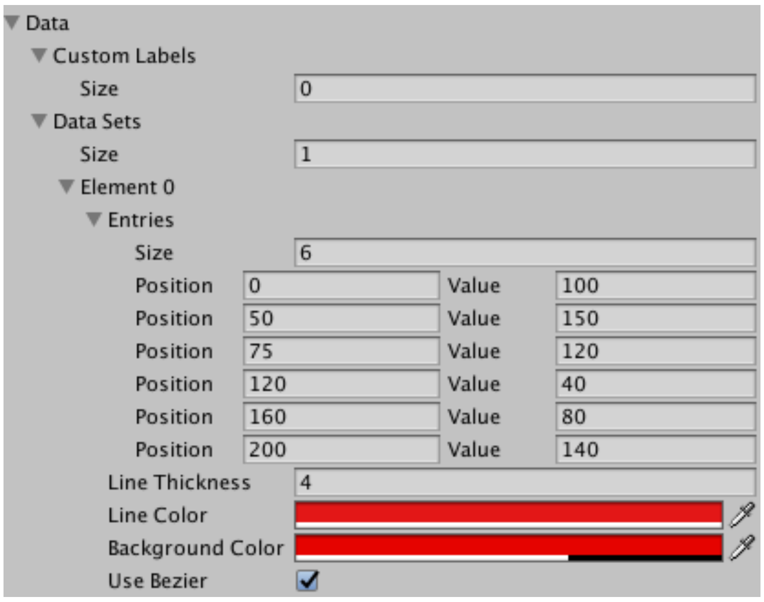
Properties

- 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));
// 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.

8. 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.

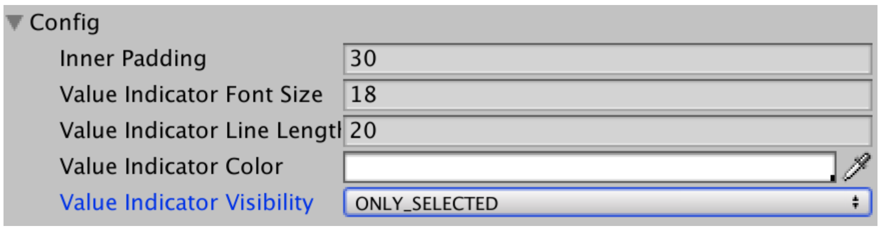
Public properties

- 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.

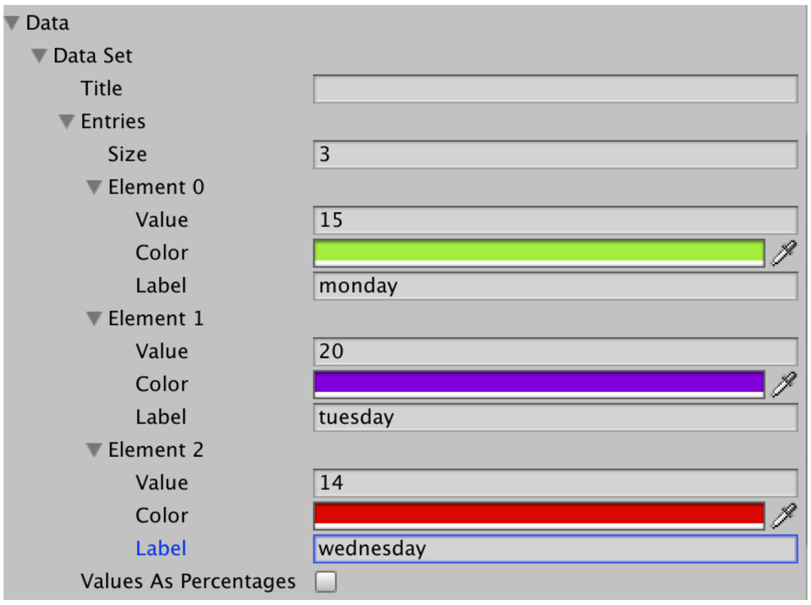
Properties

- 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:

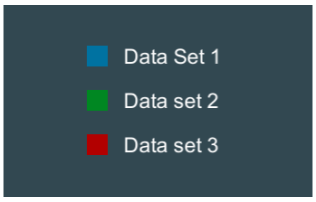
- SelectEntryAtPosition (int position) - makes entry at position selected, that entry slice will be larger and split from other slices
- DeselectEntryAtPosition (int position) - remove selection from entry at position
- IsEntryAtPositionSelected (int position) - returns true, if entry at position is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

9. Legend

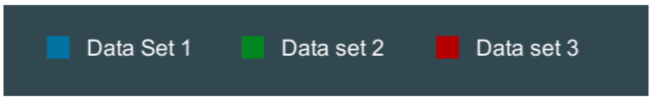
Legend is able to present information about data sets used by chart. Each legend entry is represented by title and icon.

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

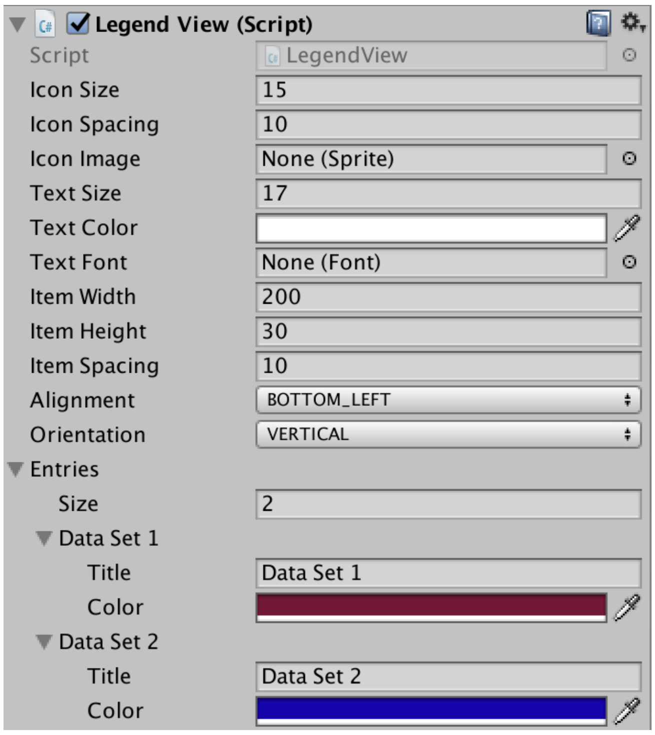


Horizontal legend

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	🟦

10. Migrations

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

1.1.5

Important!

Axis config has been split into Grid, Frame and Axis config with new options. This change gives more flexibility with config for those elements. Because this change touches core components, after library update from previous versions, those elements will have to be configured from the scratch. Source code might also require changes. Be careful with update and always backup your code.

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 setup of this properties according to your needs.

1.0.0 - 1.1.3

No actions needed

11. Example scenes

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

Examples

- /BarChart - scenes with BarChart examples
- /LineChart - scenes with LineChart examples
- /PieChart - scenes with PieChart examples
- /Mixed - scenes with different chart types