ProChart Manual for Unity3D

Introduction

ProChart has been designed to create different type of graphs and charts within a Unity3D project. The basic idea is to split the visualization and data set, so you can have an interactive multiple chart system with any number of data set. The data set can be manipulated from anywhere in the code without the needs of taking care of the invalidation of visualization.

Currently, ProChart support Bar, Line, Curves and Pie charts and 1D or 2D data sets. It is capable of rendering the charts to Canvas system (Unity3D built-in UI) or to 2D/3D meshes.

Installation

Installation is simple as importing the ProChart plugin into your Unity3D project. Open the project in Unity3D Asset Store and click download and import.

Content of the package

The package contains severals files, each located under Assets/CP folder. The code uses namespace **CP.ProChart** to avoid conflicts with other projects.

ProChart: this includes the core of ProChart

- Docs: Documentation
- Scripts: source codes
- Shaders: shader codes for Mesh rendering

ProChartDemo: A demonstration about how to use ProChart within Unity3D project.

- Resources: materials and sprites used in the demo
- Scenes: The demo contains several scenes to showcase it's features. Each scenes located here. Use the Menu.unity scene as a starting point.
- Scripts: source code for the demo, showcase the use of each feature of ProChart

How is it works

ProChart supports four types of charts to render, each has it's own script. The package divided into 3 main codes:

- Interactive canvas for Unity3D built-in UI system.
- None-interactive 2D/3D mesh for any 2D/3D object using mesh rendering (support older version of Unity3D)
- Data set for describe the data to be visualise.

Use the Canvas version, when you want to create charts with using Unity3D built-in ui system (above version 4.6), this allows you to mix charts with any other canvas items and effects (like shadow, outline or custom made effects).

Use the mesh rendering version, when you want to bypass the Unity3D built-in ui or if you want to use ProChart in an older unity version (older then 4.6).

Both charts Canvas and Mesh based are only responsible for visualise the data. None of them contains the actual data of the values. To set the data values, you have to set the data in the data set using the ChartData class. This allows you to connect multiple charts view to the same data set and to modify the data without knowing who and how to render it's data.

The source code for each parts are located under the Source folder:

Interactive canvas versions:

- BarChart.cs
- LineChart.cs
- PieChart.cs

None-Interactive 2D/3D Mesh versions:

- BarChartMesh.cs
- LineChartMesh.cs
- PieChartMesh.cs

Each Chart uses an abstract base class Chart.cs and ChartMesh.cs, which can be used to customise the features of ProChart. Don't use these in your project.

Data set for Charts:

- ChartData.cs implements ChartData1D and ChartData2D classes.

Use the 1D data set for Pie chart and 2D data set for Bar or Line charts.

Capturing the user action on a chart, like over state or click on a bar/line. ProChart uses events via delegates. These can be easily bind to your own functions.

- OnSelectDelegate: notification when user select a data on chart, if Interactable flag is on
- OnOverDelegate: notification when user mouse is over a data, if Interactable flag is on
- OnEnabledDelegate: notification when chart's parent gameobject activated

Use the **Interactable** flag to turn these events on or off.

How to use

First, create an empty gameobject for your chart. Add it under Canvas if you wish to use the canvas version of chart.

Then, create a chart with adding the appropriate chart script to the gameobject. (drag'n'drop into your gameobject within your hierarchy)

Create a script to manage the chart, add this script into your own location within your project.

Import the ProChart namespace in your script:

```
using CP.ProChart;
```

This script needs to have access or reference to your gameobject. You can use GameObject.Find() and GetComponent() to find it or simply add a public property and drag'n'drop your chart object in the editor:

```
public BarChart barChart;
```

Create a Data set. In this example we are creating Bar Charts which use 2D data:

```
ChartData2D dataSet = new ChartData2D();
```

Bind the data set with the chart(s). You can bind the same data set to any number of charts at the same time! This will allows you to see the changes on the same data set on each charts automatically.

```
barChart.SetValues(ref dataSet);
```

If you want to get notification on user activities, bind the corresponding function to your own handler function. Make sure you make these function in this script first.

```
public void OnSelectDelegate(int row, int column)
{...}

public void OnOverDelegate(int row, int column)
{...}
```

It is a good practice to bind the in your `OnEnable()` function and remove them in the `OnDisable()` function. This will make sure your script will only get notification if it's active.

```
void OnEnable()
{
    barChart.onSelectDelegate += OnSelectDelegate;
    barChart.onOverDelegate += OnOverDelegate;
}

void OnDisable()
{
    barChart.onSelectDelegate -= OnSelectDelegate;
    barChart.onOverDelegate -= OnOverDelegate;
}
```

Fill in the data[row, column], you can do this later, since it's real time visualised by the chart.

```
dataSet[0, 0] = 50;
dataSet[0, 1] = 30;
dataSet[0, 2] = 70;
dataSet[0, 3] = 10;
dataSet[0, 4] = 90;
```

Use this dataSet to update the value of the chart anytime you want. The data set will notify each charts to refresh and show actual data.

Properties

ProChart uses a few properties to set the visual looks of each charts. These can be manipulated either in the Inspector window or directly from the code.

PieChart properties

- InnerRadius: size of inner hole in percentage of diameter of chart, float in range 0f to 0.9f
- StartAngle: degree of zero position in clock-wise direction, float in range of 0f to 360f, where 0f = top (12 hr).
- ChartSize: degree of size, float in range of -360f to +360f

LineChart properties

- ChartType: type of the chart, enum values *LINE* or *CURVE*
- PointType: type of points at data, enum values *NONE*, *CIRCLE*, *RECTANGLE* or *TRIANGLE*
- Thickness: thickness of the lines, percentage of the chart size, float in range of 0f to 0.05f
- PointSize: size of points, percentage of the chart size, float in range of 0f to 0.05f

BarChart properties

- Spacing: Distance between data groups, float in range of 0.1f to 1f, where 0.1f is minimum space.
- Thickness: Thickness of bars, float in range of 0.1f to 1f, where 1f is the thickest bar.

For further information on how to use the ProChart features, please check the project and source code of any of our demos located in Assets/CP/ProChartDemo.

Have Fun!
Creative Pudding