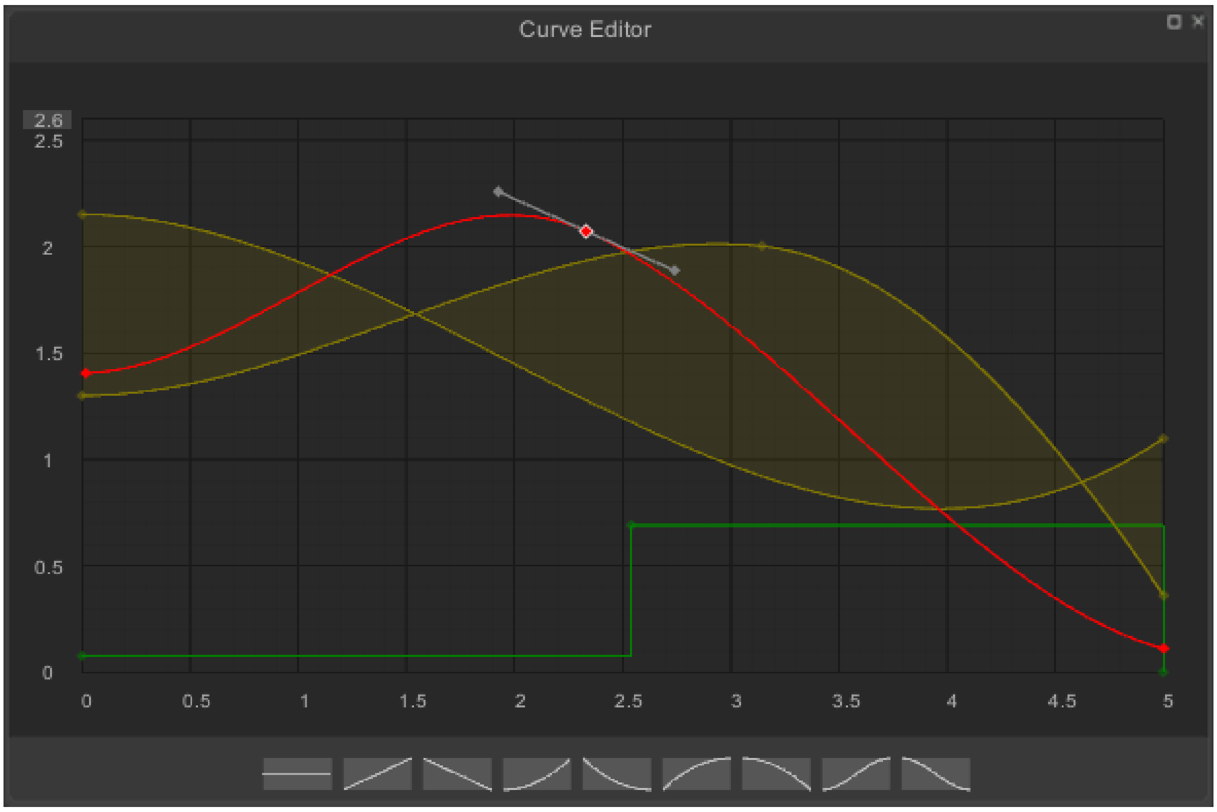

Runtime Curve Editor

The runtime curve editor can be used as part of an Unity application to **visually** edit ,at runtime, an AnimationCurve object.It has similar functionality (almost identical) to the Unity's animation curve editor.The curve equation used by this package is 100% the same with the equation used in Unity engine.

This package has been tested on PC/Mac,Web,iOS and Android.It should work flawlessly on any other platform too.The package doesn't need Unity Pro, it makes no use of any library, and requires no other asset from the store. All the code is C# ,available to the user(for reading/understanding/modyfing) .

Remember this is the only solution available in the store,for visually editing a curve at runtime, the way you can do it in the Unity Editor itself!

This package is not an Editor extension; inside the editor,it can be seen in action only in the play mode, also it makes you able to edit simultaneously more than one curve, so even more, giving it an advantage over the builtin curve editor from Unity Editor,where you can edit one single curve at a time.

Please watch also the youtube demmo video .

The package is structured on 3 layers:

- **core functionality** ,all the scripts under:
  RTAnimationCurve/Scripts/CurveEditor
- *RTAnimationCurve.cs* , which makes the **interface layer ;**
- *Demo.cs* ,which makes the **application layer** ;

This document describes ,mainly ,the 2nd layer(RTAnimationCurve.cs file).

mailto:rus_artur@yahoo.com

RTAnimationCurve component acts as an interface for the runtime curve editor module.

It provides the methods:

public void ShowCurveEditor();//pops up the window
public void CloseCurveEditor();//hides the window
public bool IsCurveEditorClosed();//true if the window is closed

public bool Add(ref AnimationCurve animCurve);//adds the curve to
//the window, returns false **only** if the curve window is not yet initialized.

public bool Add(ref AnimationCurve animCurve1, ref AnimationCurve animCurve2);
//similar with the above version of Add but it adds a path of two curves

public void Remove(AnimationCurve curve);//removes the curve from
//the window

public void SetGradYRange(float yMin,float yMax);//sets y axis's gradations range
public void SetGradXRange(float xMin,float xMax);//sets x axis's gradations range

public void SetLayers(int curveEditorLayer,int colliderLayer);//sets the layers for the curve editor and the collider(by default ,'Default' is selected for both) ; the camera's culling mask will match the curve editor's layer,so use this method if you want the collider's layer to match a different camera's culling mask.

public void SaveData(string name,Object obj);//saves the AnimationCurve fields related data,from 'obj', into the configuration with the given 'name' argument.

public void LoadData(string name,Object obj);//loads the AnimationCurve fields related data,into 'obj', out of the configuration with the given 'name' argument.

public void NewWindow();//removes all curves from editor and positions/resizes the window to initial.

public bool CurveVisible(AnimationCurve curve1);//is the curve added to the editor.

public bool CurveVisible(AnimationCurve curve1,AnimationCurve curve2);//is the path of the two curves added to the editor.

public bool DataAltered();//true , if any change in the editor has been made.

public List<string> GetNamesList();//get the list of configurations saved so far.

public void DeleteFile(string name);//deletes the named configuration .

public string GetLastFile();//get the name of the last loaded configuration or null, if no configuration has ever been saved.

The properties of the class are :

public Rect gradRect;//gets/sets gradation ranges for the grid
public AnimationCurve activeCurve;//read only
public Color activeCurveColor;//read only

The code and comments can be found inside **RTAnimationCurve.cs**.
For understanding how to use this component, open up ***DemoCurveUnity.scene*** and play it,a basic demmo application that enable you to edit two curves and a pair of curves as a path is available.
The code that makes use of RTAnimationCurve is in Demmo.cs.

It's also possible to save/load a configuration , a configuration means the current data of all curves, that are fields of Demmo component,and the coordinates of the editor's window. The curve's data, means both the AnimationCurve data,from Unity Engine, and curve's related data from the editor (e.g. each curve once added to runtime editor, has attached a rect of gradation ranges).

The grid size(the **gradation ranges)** being tested are :

for **x** axis,default only(0 to 5),

for **y** axis, 0 to any positive value(e.g. 0 to 15) or any simetrical range (e.g. -17 to 17).

! Attention

Pay some attention to the game objects hierarchy of the curve editor,(it's visible when you first show the curve editor), you'll notice a **Collider** game object ,it's just keeping a Collider component and it handles mouse inputs separately.

In the provided example, all the game objects are culled onto the same layer,but when using the curve editor in a system which already has a camera and it's doing mouse input handling, the curve editor's Collider game object should be assigned the same layer as those existing colliders from the respective system.

So in that case ,the Collider game object ,should have a different layer from the rest of the CurveEditor's game objects.

mailto:rus_artur@yahoo.com