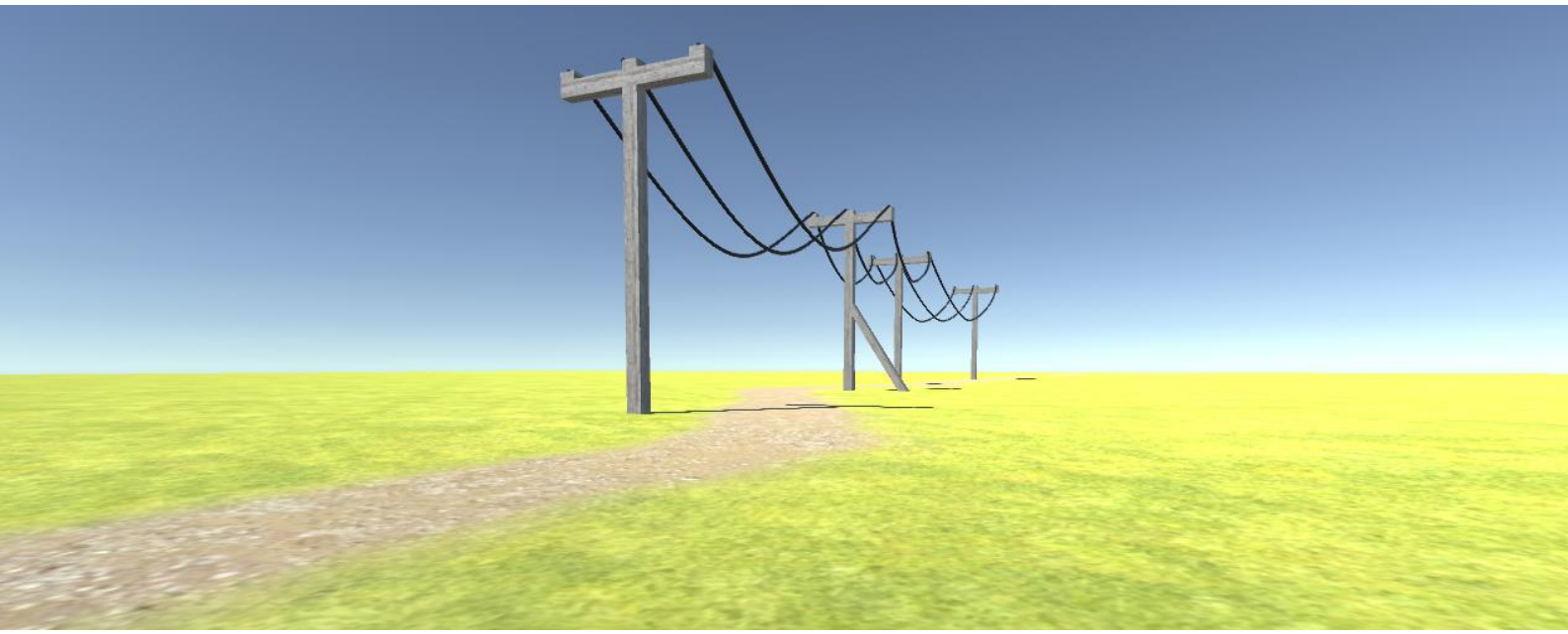

WIRES

Ver.2 07.03.2016

Introduction

Wires editor extension will help you easily create and maintain wires in your project.

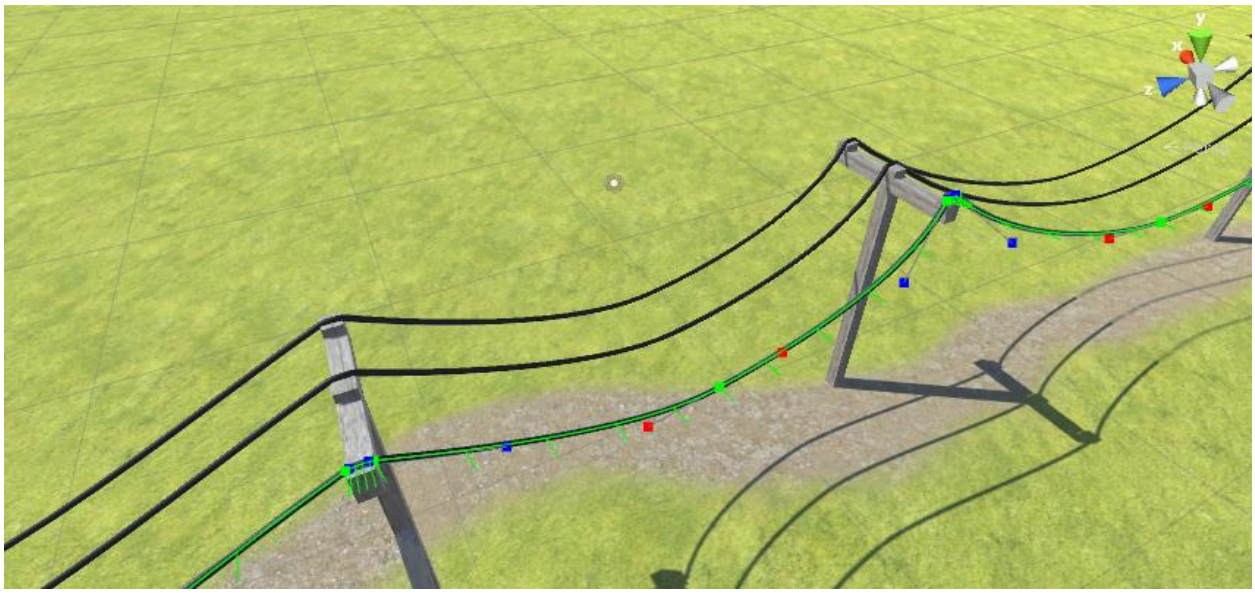

Fig.1 Wire Curve

Features

- Spline loops;
- Spline smoothing;
- Spline control point modes: Free, Aligned, Mirrored;
- Spline control points editing (support unity snap mode);
- Non-deforming (rigid mode);
- Configurable curve count per mesh;
- Configurable LOD;
- Configurable Thickness;
- LineRenderer Support
- Full editor Undo/Redo support;

Package Structure

Wires Package is located in **Assets/Battlehub/Wires2**

Wires Package was rewritten from scratch, and incompatible with previous version 1.0. **Version 1.0 is no longer supported**. If you still want to use 1.0 version it is located in WiresLegacy.unitypackage.

Script file located in **Assets/Battlehub/Wires2/Scripts**

Editor scripts and menu in **Assets/Battlehub/Wires2/Scripts/Editor**

Demo scene in **Assets/Battlehub/Wires2/DemoPackage.unitypackage**

Saved meshes in Assets/Battlehub/SavedMeshes

Menu

There are two submenus and five menu items:

1. **Create** menu item allow you to create wire
2. **Set Mode** submenu allow you to set spline or control point mode. There are three control point modes (**Free**, **Aligned** and **Mirrored**) and special spline **Rigid** mode, which forces Free control point mode. All these modes explained in **Control Point Modes** section.
3. **Postprocessing** contains menu items which typically useful when you finish editing your wire:
    - **Create LineRenderer –** Create LineRenderer from wire
    - **Remove Deformer** (wire component removed but all wire segments remains as is)
    - **Rollback** return wire to initial state, remove deformer and all wire segments;
    - **Combine And Save** (combine segments and save combined wire to **Assets/Battlehub/SavedMeshes folder**)
4. Use **Append/Prepend** to extend wire and curve.
5. **Remove curve**.
6. **Duplicate** duplicates mesh deformer and clone all meshes.

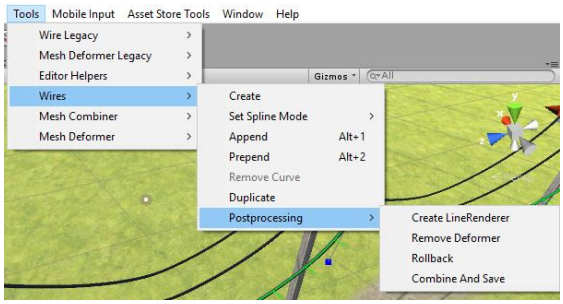

Fig.2 Menu

Control Point Modes

Wires Extension use spline similar to described in this tutorial. Everything described in tutorial will work with Wires. Each control point can be Free, Aligned or Mirrored. All of these modes shown in fig.3.

1) Free – move Control points independently
2) Aligned – variable distance between control points
3) Mirrored – same distance between control points

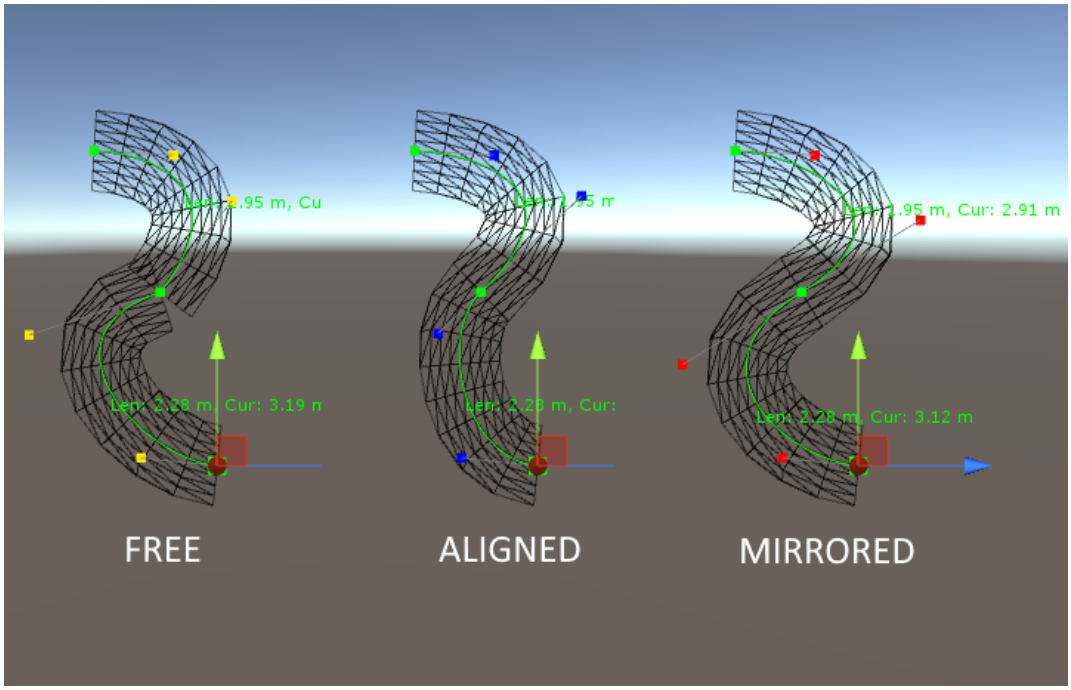

Fig.3 Control Point modes

Rigid mode forces all Control points of the curve to be aligned to a line. You can't bend and twist wire in this mode.

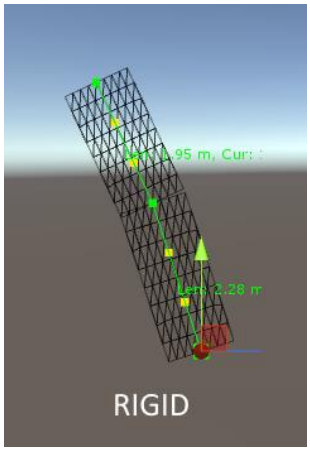

Fig.4 Rigid Mode

Wire Editor

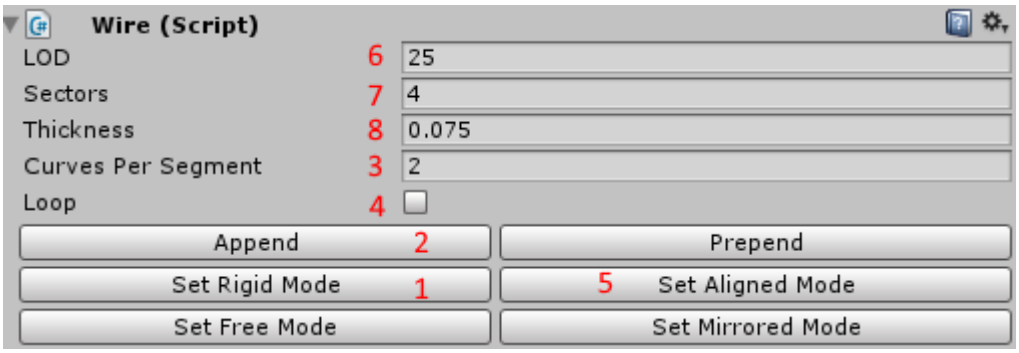

Fig.5 Wire Editor

1. Switch all control points to rigid mode
2. Append mesh duplicates
3. Change curves per segment count (more curves gives more control over the shape of the wire)
4. Create spline loop
5. Set all control points to aligned mode
6. LOD controls how precise wire will repeat trajectory of the spline. Greater value gives better visual effect. However this value should be kept as low as possible.
7. Sectors (4, 6, 8, 10…. wire arc sectors)
8. Wire thickness

Control Point Editor

This editor allow you to change individual control point mode and set twist angle and twist offset.

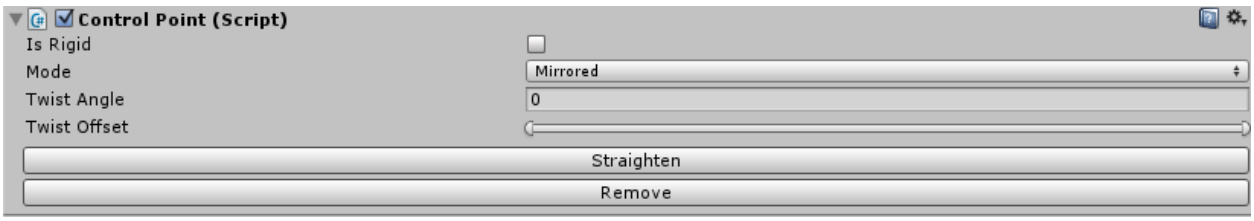

Fig.6 Control Point Editor

Wire Script

Description

class in Battlehub.Wires2

Wire class contains main functionality of package

Properties

Wire Settings
```
public override int Approximation //a.k.a LOD
public int SectorsCount
public virtual int CurvesPerMesh
public virtual float WireRadius
public override bool Loop
```

Spline Settings
```
public int CurveCount
public int ControlPointCount
```

Methods

Reset deformer to initial state
```
public void ResetDeformer()
```

Rebuild deformer
```
public void WrapAndDefromAll()
```

Remove curve with curveIndex. Returns gameObject associated with curve
```
public GameObject Remove(int curveIndex)
```

Extends Deformer by adding new curve and mesh duplicate.
Returns object associated with curve.
```
public GameObject Extend(bool prepend = false)
```

Straighten mesh and curve containing control point defined by pointIndex parameter
```
public void Straighten(int pointIndex)
```

Turn On/Off Rigid mode on curve containing control point with pointIndex
```
public void SetIsRigid(int pointIndex, bool isRigid)
```

Spline measurement methods
```
public float EvalLength(int curveIndex)
public float EvalCurveLength(int curveIndex, int steps)
```

Spline data access methods (ControlPointMode: Free, Aligned, Mirrored)
```
public ControlPointMode GetControlPointMode(int index)
```

Spline data access methods (World Space)
```
public Vector3 GetPoint(float t, int curveIndex)
public Vector3 GetPoint(float t)
public Vector3 GetControlPoint(int index)
public Vector3 GetVelocity(float t, int curveIndex)
public Vector3 GetVelocity(float t)
public Vector3 GetDirection(float t, int curveIndex)
public Vector3 GetDirection(float t)
```

Spline data access methods (Object Space)

```
public Vector3 GetPointLocal(float t, int curveIndex)
public Vector3 GetPointLocal(float t)
public Vector3 GetControlPointLocal(int index)
```

Spline data access methods (Twist Angles measured in degrees)

```
public float GetTwist(float t, int curveIndex)
public float GetTwist(float t)
public Twist GetTwist(int index)
```

Spline data modification methods

```
public void SetControlPointMode(int index, ControlPointMode mode)
public void SetControlPoint(int index, Vector3 point)
public void SetControlPointLocal(int index, Vector3 point)
public void SetTwist(int index, Twist twist)
```

Spline smooth method

```
public void Smooth()
```

Limitations and Issues

- Runtime editing may cause performance issues (will be fixed in future releases)
- Twist angles are not editable using mouse. Editing possible using editor or script.

Support

If you have any questions, suggestions, you want to talk or you have some issues please send mail to Vadim.Andriyanov@outlook.com or Battlehub@outlook.com.