Wrapping Rope v2.2

Information

Wrapping rope is a tool for creating thin rope stretched between two points. Wrapping rope does not provide realistic physics, but very useful if you want to show very long rope, that could bend, if it collides with other objects. For example, you can use this tool for creating super hero's grappling hook, wire for cable railway, lashing effect, bungee cord etc.

Key features:

- Works in editor
- Profile customizing
- Fore texturing modes
- The body of the rope could be rendered as finite segments or procedural mesh.
- Elastic and swinging physics

A short video instruction you can see at https://youtu.be/dmvq58QOM5g

Table of Contents

Migration from 1.X versions

Prefabs and game objects in the scene

If you have a game object with rope component, created in the previous version of package, then fix properties in inspector:

- **Body**. By default, this property is set to **Finite Segments**. If procedural generation of the mesh was used in the previous version then set this property to **`Continuous`**.
- **Material**. Assign to this property the same material that was used in Mesh Renderer in the previous version.
- **Extend Axis**. Assign to this property the value that was used in the previous version.

Scripting

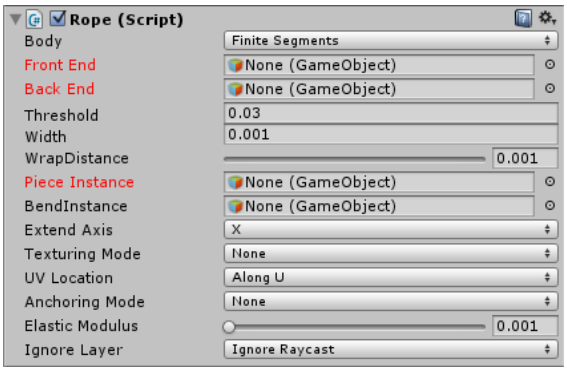
If you have scripts that process any objects of the older version of this package, please

- fix references to namespaces (see API section);
- change the type of sender parameter in **`ObjectWrap`** `event handler from` **`Rope`** `to` **`RopeBase`** `(see Events subsection)`

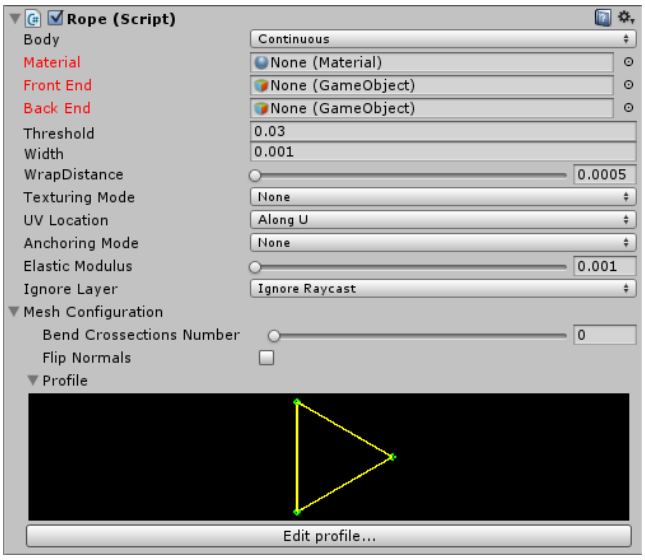
How to use

Creating of Rope

To create a rope use menu **GameObject/Wrapping Rope**. Depending on the **Body** property, the created rope can have two different sets of properties. Red color means required properties. Note, that created rope have Mesh Renderer and Mesh Filter components that not shown in figure.



Body set to Finite Segments



Body set to Continuous

Fig. 1 Inspector view

Name	Description
Body	The body type of the rope. This property provides two values: **Finite Segments** and Continuous. More information about body type see in Rope Body section. This property defines a list of other properties in inspector.
Material	The material which used when **Body** property set to **Continuous**.
Front End	Assign any game object to set one end of rope.
Back End	Assign any other game object to set another end of rope.
Threshold	Minimal size of objects, that reliably will be processed in collisions with rope.
Wrap Distance	Distance between object surface and rope in wrap zone.
Piece Instance	Assign game object with mesh filter and mesh renderer to this property for rope rendering.
Bend Instance	Assign game object with mesh filter and mesh renderer to this property to place instances of this object in joints of linear pieces of rope. This property can be used to smooth transition between linear pieces of rope. Note, that instances of assigned object are scaled automatically according to rope width and so initial object scale must be 1x1x1.
Width	The width of rope.
Extend Axis	The direction of piece instance expansion (in local coordinate system).
Texturing Mode	Specifies how texture of material should be expand (see in Texturing section).
UV Location	Specifies how texture of material should be placed (see in Texturing section).
Anchoring Mode	Specifies the anchored end of rope for swinging physics (see in Anchoring section).
Elastic Modulus	This property specifies elastic physics (see in Interaction with Game Objects section).
Ignore Layer	The layer assigned to objects that shouldn't be processed in collisions with rope.
Bend Crossections Number	The number of cross sections between linear pieces. The more value of this property, the smoother transition between linear pieces (see Rope Body section).
Flip Normals	The direction of normals in procedural generated mesh is dependent on initial position of rope. Sometimes normals are not defined correct and this property is used to fix normals.

Collisions

For properly processing of collisions follow a few rules.

1) Static objects could not be processed in collisions.
2) Object should have Collider.
3) Moving objects should have **Rigidbody** with unchecked **Is Kinematic** flag or have a script inherited from **IRopeInteraction** interface.
4) **Scale** property of game objects should have positive values.
5) Avoid clamping of rope between objects, which should be processed in collisions with rope.

Texturing

Texturing Mode

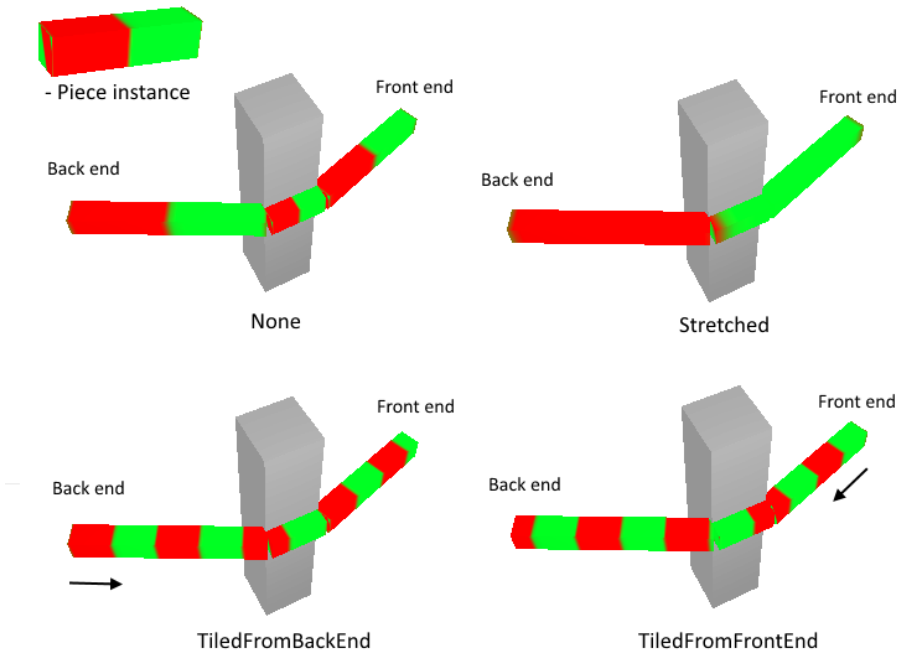


Fig. 2 Texturing Mode options

- **None** - texture not changed
- **Stretched** - texture stretches between back end (specified by **Back End** property) and front end (specified by **Front End** property)
- **TiledFromBackEnd** - texture anchored to back end and tiled along the rope
- **TiledFromFrontEnd** - texture anchored to front end and tiled along the rope

UVLocation

Texturing algorithm can't define how texture mapped along extend axis (specified by **Extend Axis** property), so this property used for solving this problem. Using texturing, remember, that extend axis always directed from back end to front end.

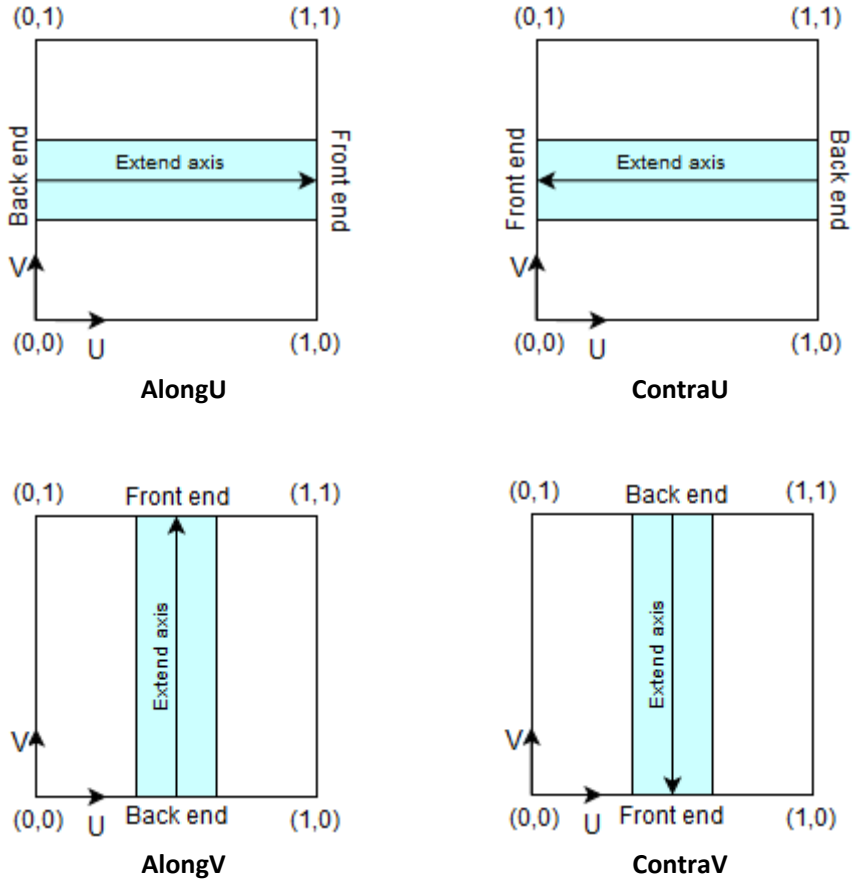


Fig. 3 UVLocation options

Texturing when Body set to Continuous

When **Body** set to **Continuous** (see Rope Body section) **UVLocation** controls how the texture of **Material** will be projected to the rope mesh. In this case **AlongU** and **ContraU** means that U axis of texture is projected along rope length, while **AlongV** and **ContraV** means that V axis of texture is projected along rope length.

Anchoring

This feature is useful for imitation of swinging physics. Swinging object should have Rigidbody with unchecked **Is Kinematic** flag and should be assign to **Front End** or **Back End** property. To control swinging physics use **Anchoring Mode** property.

- **None** - anchoring feature is off
- **By Front End** - The rope end, specified by **Front End** property, will be fixed in space, while other end of the rope will be suspended. In this case, **CutRope** method call will result in movement of the Back End to Front End.
- **By Back End** - The rope end, specified by Back End property, will be fixed in space, while other end of the rope will be suspended. In this case, **CutRope** method call will result in movement of the Front End to Back End.

Interaction with Game Objects

Interaction with game objects is possible when collision is properly processed (see Collisions section). There are three cases of collision of the rope with game objects.

1) The rope is moving and the game object is not moving.
2) The rope is not moving and the game object is moving.
3) The rope is moving and the game object is moving.

The first case requires that the game object has a collider. The second and the third cases require that the game object has **Rigidbody** with unchecked **Is Kinematic** flag or has a script inherited from **IRopeInteraction**

interface (see IRopeInteraction interface section). When game objects are moved by physics simulation in a game the **Rigidbody** is sufficient for the second and the third cases. When game objects are animated by script, character animation or dragged by mouse in editor the **Rigidbody** is useless while **IRopeInteraction** works fine. The asset has two examples of classes that inherited from **IRopeInteraction** interface. The first class **DefaultRopeInteraction** could be used as a script component or as a base class for other classes with some custom logic. The second class **CharacterRopeInteraction** is used for interaction with a biped animated character in a demonstration scene. One important condition to use these classes is that a rope script should be executed before these classes.

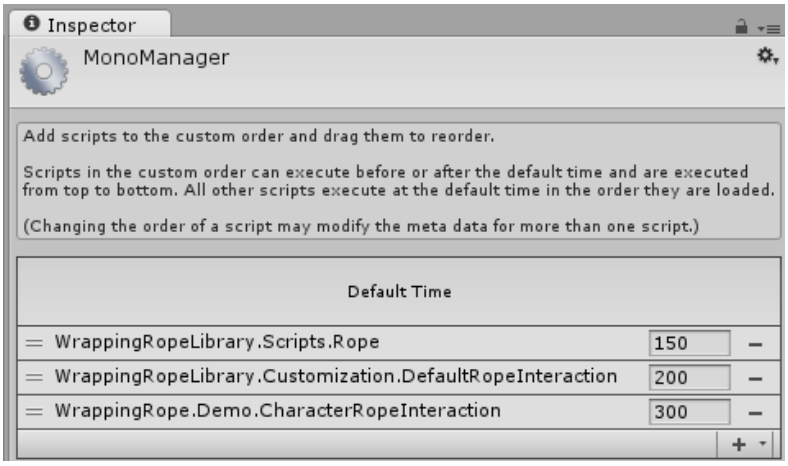
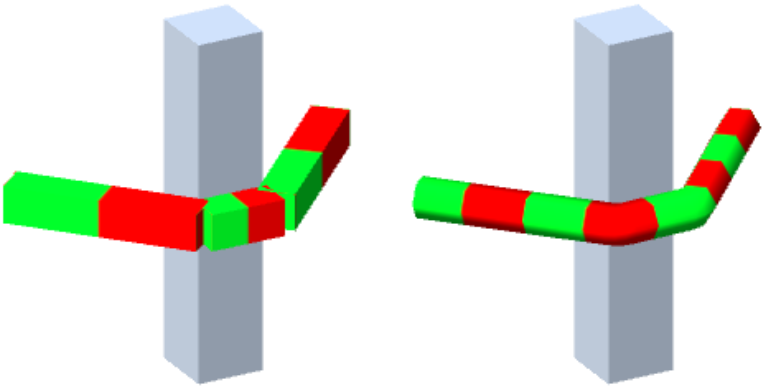


Fig. 4 Script execution order

When collision takes place, the rope could interact with object by applying a force. This feature is available in the same conditions as for collision: a game object should have **Rigidbody** with unchecked **Is Kinematic** flag or a script inherited from **IRopeInteraction**. The degree of interaction is specified by **Elastic Modulus** property. The less value of this property, the less force for wrapped object.

Rope Body



Finite Segments Continuous

Fig. 5 Body types

To select the body type use **Body** property, which provides two values:

- **Finite Segments** – connections of linear instances of object, assigned to **Piece Instance** property
- **Continuous** – procedural mesh

To configure procedural mesh use **Mesh Configuration** section (see Creating of Rope section).

Polygon Editor

To edit profile of rope when **Body** property set to **Continuous** use Polygon Editor. To open Polygon Editor use the button below profile preview in inspector or menu **Window/Polygon Editor**.

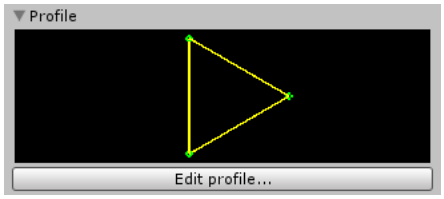


Fig. 6 Profile preview

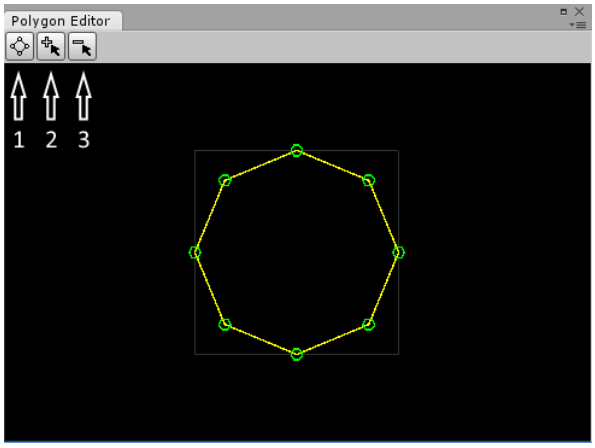


Fig. 7 Polygon Editor

- 1 – make regular polygon
- 2 – insert points
- 3 – delete points

The grey square limits an area with size of 1x1 Unity's units. To zoom use mouse wheel. Note, that minimal count of points is three. The polygon couldn't be self-intersecting.

API

Rope script

Namespace: `WrappingRopeLibrary.Scripts`

Public Fields

Minimal size of objects, that reliably will be processed in collisions with rope.
```
public float Threshold
```

Distance between object surface and rope in wrap zone.
```
public float WrapDistance
```

The direction of piece instance expansion (in local coordinate system).
```
public Axis ExtendAxis
```

Specifies how texture of material should be expand.
```
public TexturingMode TexturingMode
```

Specifies how texture of material should be placed.
```
public UVLocation UVLocation
```

Specifies the degree of elastic physics.
```
public float ElasticModulus
```

Public Properties

Gets a body type of the rope.
```
public BodyType Body { get; }
```

Gets a material used when Body property set to Continuous.
```
public Material Material { get; }
```

Gets a game object used for position of the front end of the rope.
```
public GameObject FrontEnd { get; }
```

Gets a game object used for position of the back end of the rope.
```
public GameObject BackEnd { get; }
```

Gets or sets a width of the rope.
```
public float Width { get; set; }
```

Gets or sets an anchored end of the rope for swinging physics.
```
public AnchoringMode AnchoringMode { get; set; }
```

Gets a first piece in the beginning of the rope.
```
public Piece FrontPiece { get; }
```

Gets a last piece of the rope.
```
public Piece BackPiece { get; }
```

Gets a scale of PieceInstance to the width of the rope by two dimensions and 1 unit by third dimension depending on ExtendAxis.
```
public Vector3 PieceInstanceRatio { get; }
```

Gets a uniform scale of BendInstance to the width of the rope. Scale factor is defined by x axis of bounds of BendInstance.
```
public float BendInstanceRatio { get; }
```

Gets a number of cross sections between linear pieces.
```
public int BendCrossectionsNumber { get; }
```

Public Methods

Changes the length of the rope. If value of AnchoringMode property is AnchoringMode.None, the length of the rope will be only decreased, otherwise the length of the rope can be whether decreased or increased depending on the sign of the length parameter.

```
public void CutRope(float length, Direction dir)
```

Shrinks the length of the rope.

```
public void CutRopeNotAnchoring(float length, Direction dir = Direction.BackToFront)
```

Events

Triggered when the rope is about to wrap a game object

```
public event ObjectWrapEventHandler ObjectWrap
```

Enums

Namespace: `WrappingRopeLibrary.Enums`

```
public enum Direction
{
    FrontToBack,
    BackToFront
}

public enum BodyType
{
    FiniteSegments,
    Continuous
}

public enum Axis
{
    X, Y, Z
}

public enum TexturingMode
{
    None,
    Stretched,
    TiledFromBackEnd,
    TiledFromFrontEnd
}

public enum UVLocation
{
    AlongU,
    ContraU,
    AlongV,
    ContraV
}

public enum AnchoringMode
{
    None,
    ByFrontEnd,
    ByBackEnd
}
```

Events

Namespace: `WrappingRopeLibrary.Events`

```
public delegate void ObjectWrapEventHandler(RopeBase sender, ObjectWrapEventArgs args)
```

Public Properties of ObjectWrapEventArgs Class

Gets a game object that the rope is about to wrap.
```
public Vector3[] Target { get; }
```

Gets an array of points in space that specifies the wrap path.
```
public Vector3[] WrapPoints { get; }
```

Gets or sets a value indicating whether the event should be canceled.
```
public bool Cancel { get; set; }
```

IRopeInteraction interface

Namespace: `WrappingRopeLibrary.Customization`

Public Methods

Gets the velocity of the game object at the point worldPoint in global space.
```
Vector3 GetPointVelocity(Vector3 worldPoint);
```

Applies force at position.
```
void AddForceAtPosition(Vector3 force, Vector3 position, ForceMode mode)
```

Support

If you have any questions or there are any problems, please email me at: dkirillovd@mail.ru – Denis