Thanks for purchasing this nice asset. If you have any questions or there are any problems, please email me at: dkirillovd@mail.ru – Denis

List of changes

Version	Section	Changes
1.1	Collisions	Removed condition of Convex flag in rule 1.
1.1	Texturing	New section with description of rope texturing.
1.2	Creating of Rope	Bend Instance description
1.2	Shrinking of Rope	CutRope method addition
1.2	Anchoring	New section.
1.2	Interaction with Game Objects	New section.
1.3	Procedural mesh	New section
1.3	Creating of rope	Extension of description of Piece Instance, Bend Instance and Extend Axis.
1.3	Texturing	The description of work of UVLocation in case of procedural mesh generation.
1.4	Procedural mesh	The description of profile editing and Flip Normals property.
1.4	Polygon Editor	New section
1.4	Events	New section

Information

Wrapping rope is a tool for creating thin rope stretched between two points in Unity 3d scene. 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, or lashing effect.

A short video instruction you can see at https://youtu.be/THtFphWoE2Q

How to use

Creating of Rope

Create empty game object and add **Rope** script component. You also can use rope prefab from Prefabs folder of package. Then define some script properties:

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. In case of procedural mesh generation (see Procedural mesh section) this property has no sense and could be null.
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. In case of procedural mesh generation (see Procedural mesh section) this property has no sense and could be null.
Width	Width of rope.
Extend Axis	The direction of piece instance expansion (in local coordinate system). In case of procedural mesh generation (see Procedural mesh section) this property has no sense.
Ignore Layer	The layer assigned to objects that shouldn't be processed in collisions with rope.

Note: for procedural mesh generation (see Procedural mesh section) add Mesh renderer and Mesh filter components to the rope game object and set material to Mesh renderer.

Shrinking of Rope

You can reduce rope length using function of **Rope** class:

public void **CutRope**(float **length**, Direction **dir**);

Parameters

length	A difference of rope length before and after shrinking.
dir	Specifies fixed and movable ends of rope. This parameter uses **Direction** enumerator that provides two values.

Direction values

FrontToBack	The rope end, specified by **Front End** property, moves to another end of the rope, that fixed in space.
BackToFront	The rope end, specified by **Back End** property, moves to another end of the rope, that fixed in space.

Note, that if **Anchoring Mode** value is not equal to **AnchoringMode.None**, the **dir** value will not have any effect, since the shrink direction will be determined by the value of **Anchoring Mode**.
(see Anchoring section).

Collisions

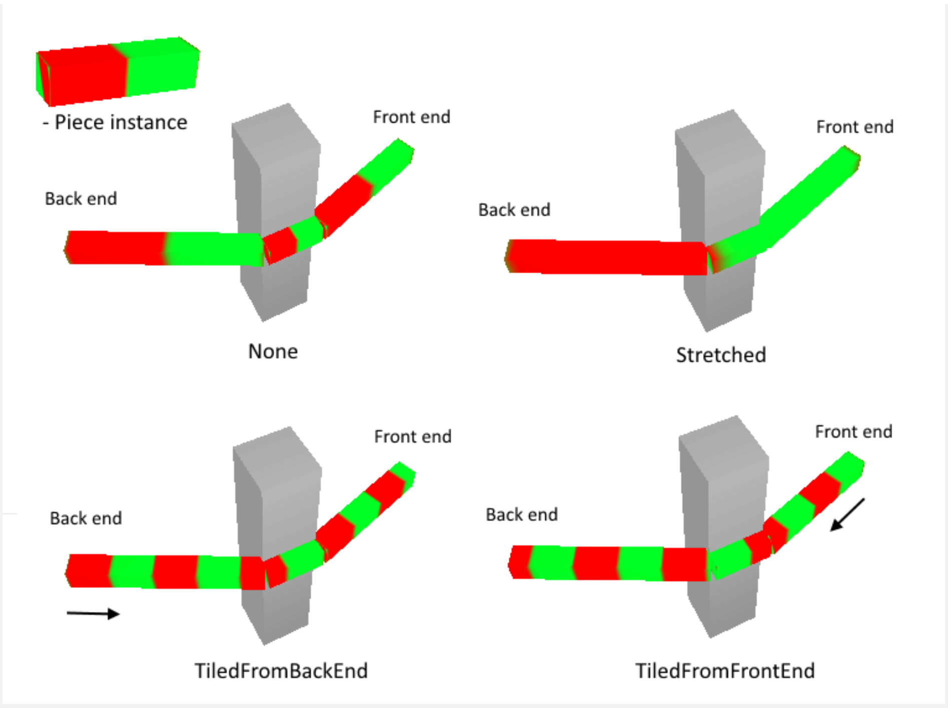
For properly processing of collisions follow a few rules.

1) Static and moving objects should have mesh collider.
2) Moving objects should have Rigidbody with unchecked **Is Kinematic** flag.
3) **Scale** property of game objects should have positive values.
4) Avoid clamping of rope between objects, which should be processed in collisions with rope.

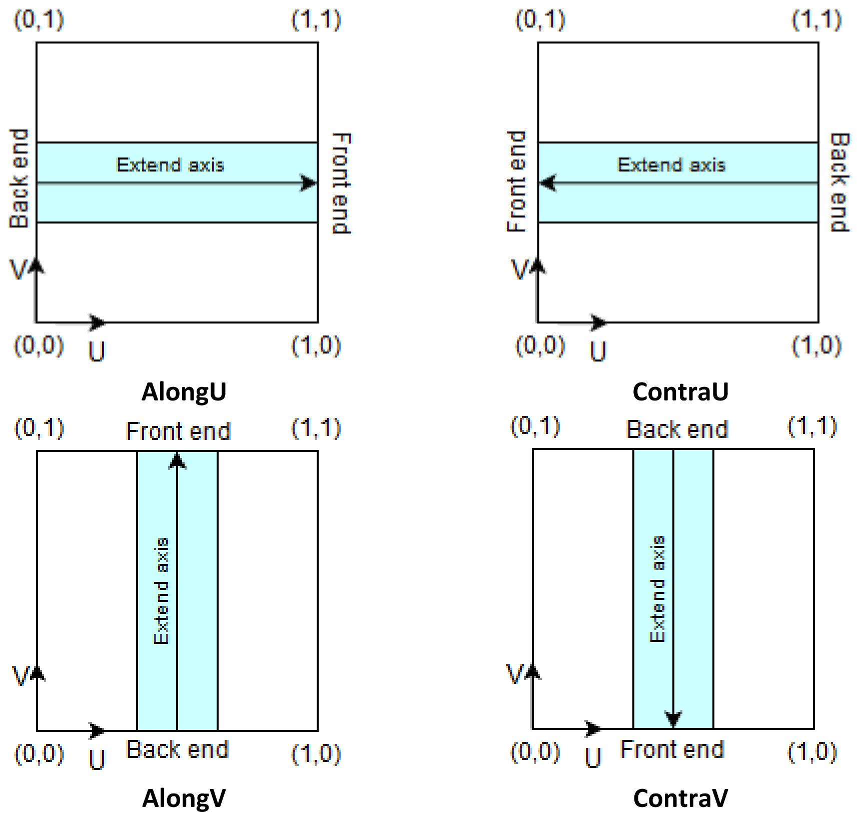
Texturing

Rope component provides several properties to control the way texture of game object, specified by **Piece Instance** property, should be used.

Texturing Mode	This property provides values: **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. Picture below shows available mapping schemes for texturing algorithm. Each scheme corresponds one of four property values: **AlongU**, **ContraU**, **AlongV**, **ContraV**. Using texturing, remember, that extend axis always directed from back end to front end.



Note: In case of procedural mesh generation (see Procedural mesh section) **UVLocation** controls how the texture of mesh renderer component of the rope 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 pendulum or grapple objects. For this feature use **Anchoring Mode** property. This property uses **AnchoringMode** enumerator, that provides three values.

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** independently from **dir** parameter value.
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** independently from **dir** parameter value.

Suspended end of the rope should have a Rigidbody with unchecked **Is Kinematic** flag.

Interaction with Game Objects

The degree of interaction of rope with Rigidbodies is specified by **Elastic Modulus** property. The less value of this property, the less effect to Rigidbodies.

Procedural mesh

Wrapping rope provides two methods of creating rope's body. The first one based on linear pieces connections and the second one based on procedural mesh generation. First method uses object, assigned to **Piece Instance**, as instance for linear pieces, and joints between pieces looks not good in this case. Second method provides smooth transition between linear pieces, so rope's body looks more realistic.

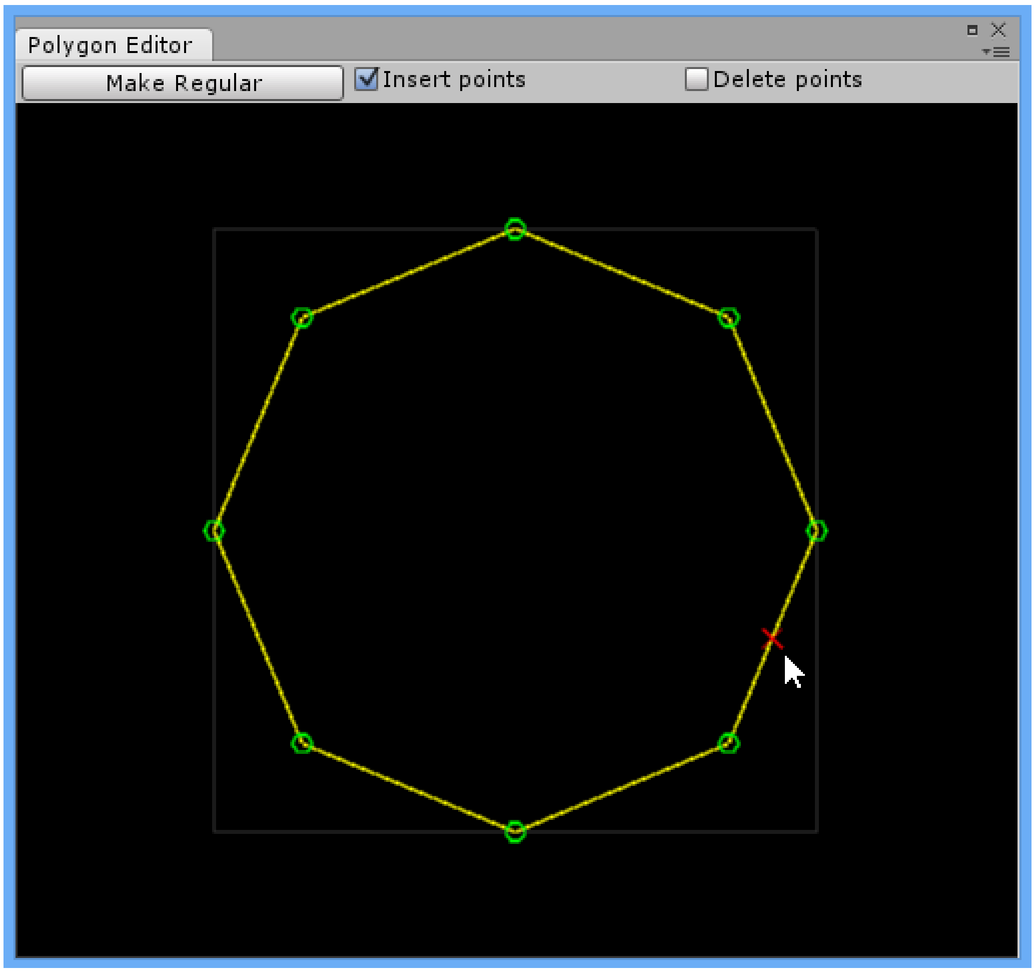
Note: for procedural mesh generation rope's game object should have Mesh renderer with material and Mesh filter components.

To configure procedural mesh generation use Mesh Configuration property, that contains children:

Bend Crossections Number	The number of cross sections between linear pieces. The more value of this property, the more smooth transition.
Flip Normals	The direction of normals is dependent on initial position of rope. Sometimes normals are not defined correct and in this case set this property on.
Profile	The shape of rope profile. For editing the shape of profile use Polygon Editor. Polygon Editor is an editor window, that could be opened from menu Window / Polygon Editor. For more details see Polygon Editor section.

Polygon Editor

Polygon Editor is a simple vector editor for customization of rope's profile. To open Polygon Editor use menu **Window** / **Polygon Editor**. Picture below shows the interface of Polygon Editor.



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.

Events

ObjectWrap

The **ObjectWrap** event is raised when the rope is about to wrap a game object. The event handler receives arguments:

Rope **sender**	The source of event.
ObjectWrapEventArgs **args**	An object that contains a data of event.

The following **ObjectWrapEventArgs** properties provide information specific to this event.

bool **Cancel**	If this property set to **true** a game object would not be wrapped. If this property set to **false**, the wrap will be happening.
GameObject **Target**	A game object that rope is about to wrap.
Vector3[] **WrapPoints**	Array of points in space that specify a path of wrap.