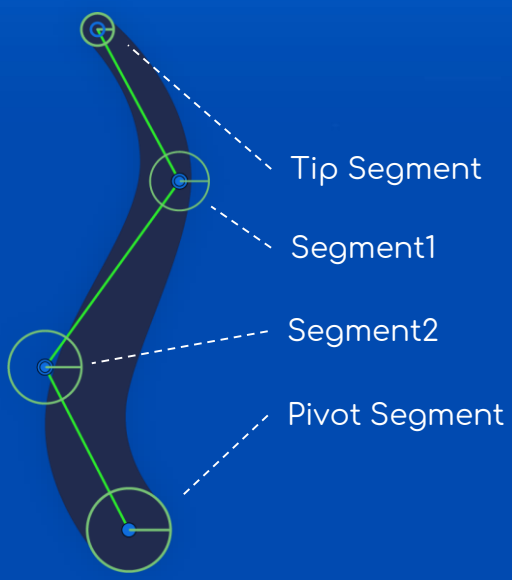
Tentacles 2D Documentation

About

Tentacles2D is a Unity asset that will provide you with the 2d procedural tentacles, ready to use in your project. The behavior of the tentacles is based on the Unity's 2D Physics, so they are fully compatible with all its elements (gravity, effectors, etc.). The visual part combines the properties from MeshRenderer and SpriteRenderer.

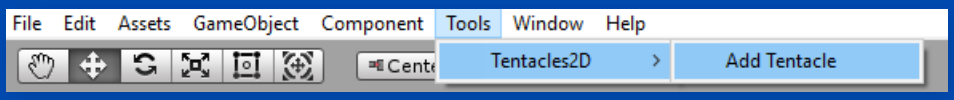


For performance reasons each tentacle represented by four segments (see pic). Each segment uses rigidbody connected by hinge joint with the rigidbody of the previous segment, starting from the pivot. For each segment you can enable the circle collider, and it will have the size of the tentacle's width at this place.

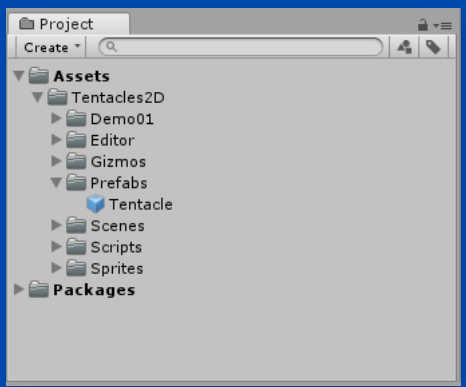
Getting Started

There is two ways to add the tentacle to your scene:

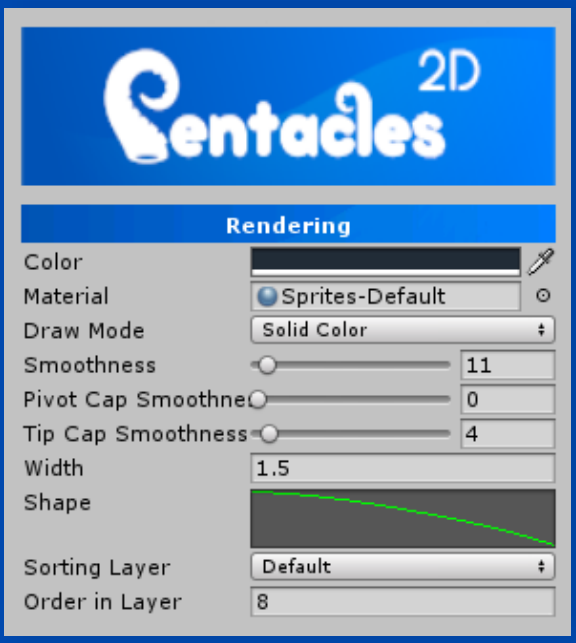
1. From the Main menu: Tools – Tentacles2D – Add Tentacle;



2. Drag the Prefab itself from the Project folder: Assets – Tentacles2D - Prefabs – Tentacle.prefab.



The inspector supports multiply editing and prefab modifications.



Rendering

Color - tint color for the material;

Material - material to be used by tentacle;

Draw Mode - specify the draw mode for the texture;

- Solid Color - the solid color will be used as the texture;
- Stretchy - the texture will be stretching and shrinking with tentacle;
- Tiled - the texture will be repeated and it's proportions will remain the same;

Smoothness - the smoothness of the mesh (this isn't a count of Rigidbodies);

Pivot Cap Smoothness - smoothness of the mesh at the pivot of the tentacle;

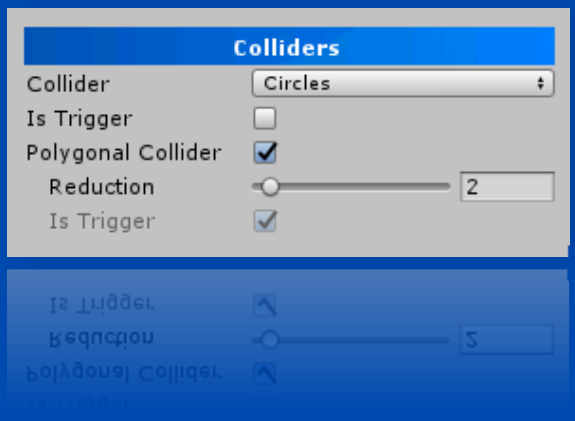
Tip Cap Smoothness - smoothness of the mesh at the tip of the tentacle;

Width - width of the tentacle;

Shape - shape of the tentacle;

Sorting Layer - name of the Renderer's sorting layer;

Order in Layer - renderer's order within a sorting layer.



Colliders

Collider Type - type of the collider the tentacle will use;

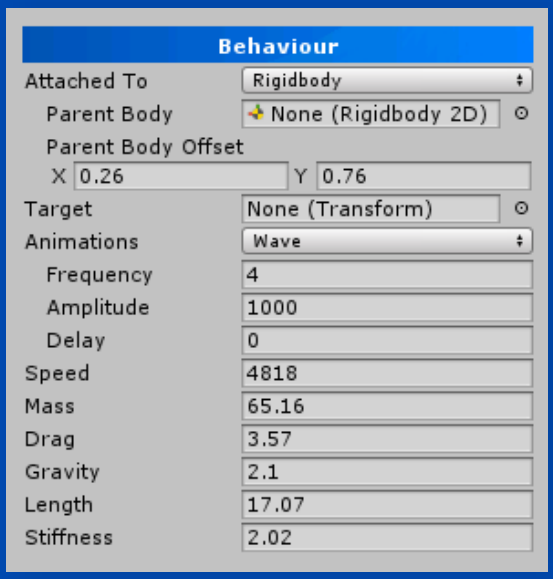
- Circles - circle colliders on 4 segments;
- Circle On Tip - circle collider on the tip segment only;
- None - the colliders are disabled;

Is Trigger - whether the enabled colliders behaves as a triggers or not;

Polygonal - whether polygonal collider enabled or not;

Reduction - simplification of the polygonal collider to boost performance;

Is Trigger - polygonal collider is building at the realtime so it can only behave as trigger to avoid glitches.



Behaviour

Attached To - parent of the tentacle;

- World - tentacle's pivot will be snapped to the world point at current position;
- Rigidbody - tentacle will be connected to another rigidbody2d by the pivot;
- Detached - tentacle's pivot will be detached;

    Parent Body - the rigidbody2d of the parent if Rigidbody type was chosen;

    Parent Body Offset - the local-space anchor from the rigidbody2d of the parent if Rigidbody type was chosen;

Target - tentacle will be trying to reach this target;

Animation - additional physics-based animation of the tentacle;

- None - no animation;
- Wave - wave-like animation;
- Swing - tentacle will swing from side to side;

    Frequency - frequency of the tentacle's animations;

    Amplitude - strength of the tentacle's animations;

    Delay - delay to start the animation (for randomization purposes);

Speed - tentacle will be trying to reach the target with this speed (strength);

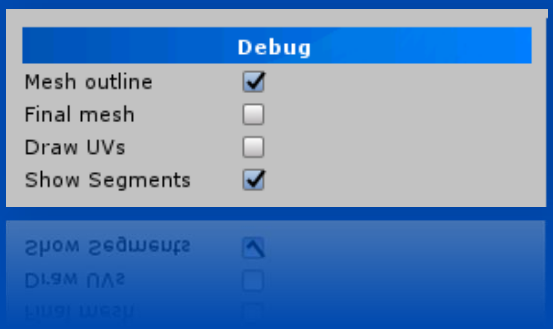
Mass - the mass of the tentacle;

Drag - the drag of an each segment of this tentacle;

Gravity - how much gravity affects each segment the tentacle;

Length - the length of the tentacle;

Stiffness - the stiffness of an each segment of this tentacle.



Debug

Mesh Outline - will highlight the mesh (magenta color);

Final Mesh - will outline the final triangulated mesh with triangles (magenta color);

Draw UVs - show UVs created for the mesh, at the center of the scene (cyan color);

Show Segments – hide / show all child gameobjects of this tentacle in the hierarchy window.

Scripting API

To get access to the Tentacle class from your script create respective field and drag the gameobject with Tentacle component (script) on this field in the Inspector:

```
[SerializeField] private Tentacle tentacle;
```

Or find the Tentacle component directly from your script:

```
private Tentacle tentacle;
private void Awake() => tentacle = GetComponent<Tentacle>();
```

class Tentacle	
Public fields, properties and enums:	
public float Width	The width multiplier of the tentacle's mesh.
public AnimationCurve Shape	The AnimationCurve that represents the shape of the tentacle.
public Color Color	The tint color of the tentacle.
public float Mass	The mass of the whole tentacle
public float Drag	The drag of an each segment of this tentacle
public float Gravity	How much gravity affects each segment of this tentacle
public float Length	The length of the whole tentacle
public float Stiffness	The stiffness of an each segment of this tentacle
public Rigidbody2D Tip	Returns the Rigidbody2D component of the tentacle's tip.
public Rigidbody2D Pivot	Returns the Rigidbody2D component of the tentacle's pivot.
public Rigidbody2D[] Segments	Returns an array of middle segments of the tentacle.
public SpringJoint2D[] Joints	Returns an array of SpringJoint2D components, with which segments are connected by.
public Rigidbody2D ParentRigidbody	Parent rigidbody2d the tentacle is attached to
public Vector2 ParentBodyOffset	Local-space anchor from the parent rigidbody2d

public Transform TargetTransform	Transform of the attached target.
public Rigidbody2D TargetRigidbody	Rigidbody2D of the attached target. Will reset TargetTransform property with the transform of this rigidbody when set.
public enum Animations	Enumeration that represents the additional animations of the tentacle
public Animations Animation	The type of the additional animation applied to this tentacle
public bool IsAttached	Whether tentacle's pivot attached to another rigidbody
public bool IsTargetSet	Whether the field TargetTransform and / or TargetRigidbody not equals null
public bool IsHoldingTarget	Whether tentacle is connected to the target's rigidbody
Public methods:	
public void Catch()	Catch (connect tentacle's tip to the) target if TargetRigidbody is set.
public void Release()	Release target.
public void Attach()	Attach tentacle's pivot segment to the world point at the current position.
public void Attach(Rigidbody2D rigidbody)	Attach tentacle's pivot segment to another rigidbody.
public void Detach()	Detach tentacle.

Example script that makes tentacles reach the target when you press Space key and deactivates them with the next press:

```
using Tentacles2D;
public class TentaclesController : MonoBehaviour
{
    [SerializeField] private Tentacle[] tentacles;
    [SerializeField] private Rigidbody2D target;
    private bool isActive;

    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            if (isActive)
            {
                for (int i = 0; i < tentacles.Length; i++)
                    tentacles[i].TargetRigidbody = null;
                isActive = false;
            }
```

```
            else
            {
              for (int i = 0; i < tentacles.Length; i++)
                  tentacles[i].TargetRigidbody = target;
              isActive = true;
            }
    }
}
```

The see some live implementations see the demo scene at your assets folder: Tentacles2D/Demo2/Demo2.scene.