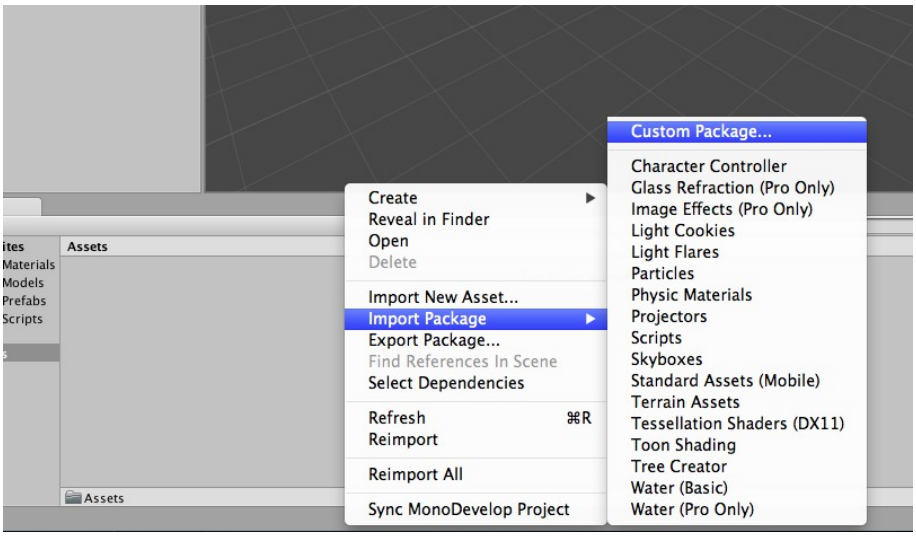
Quickstart Guide

Include the Filo package in your project. Right click in the project window, "Import Package" "Custom Package", then select FiloCables.unitypackage.



Support / Contact

If you have any suggestions, questions or issues, drop us a line:

filo@virtualmethodstudio.com

User manual

How it works

Filo is a physics add-on specialized in cable-driven system simulation. It approaches the problem in a very different way to many of the existing simulators that share the same goal.

It does not model the cable as a chain of small rigidbodies, and does not influence wheels, pulleys and other objects trough frictional contacts. Instead, it connects objects using custom joints that store the length of cable between them as a single scalar value.

Compared to the traditional *rigidbody chain* approach, Filo's approach has several important advantages:

- Performance: detecting and simulating frictional contacts against arbitrary 3D shapes is a very performance demanding task. So is solving a long chain of constraints in an iterative way. By avoiding both, Filo achieves lightning fast simulation.

- Scalability: Long chains of constraints tend to be excessively stretchy, and the longer they are, the higher the performance hit. Filo considers a single constraint between each pair of connected objects regardless of the distance between them, so cables can be as long as needed, and affect a large number of objects without becoming excessively stretchy.

- Stability: Due to the absence of collision detection/resolution, the low amount of joints used (even by extremely complex setups) and the loose timestep requirements, Filo offers unprecedented stability.

It also has several drawbacks, that can be alleviated by combining it with more traditional solvers:

- It cannot simulate free cables, and its ability to represent loose cables is very limited. It can only accurately model completely tensioned cables.

- Cables cannot collide with geometry other than the specified list of bodies linked by the cable.

- Cables cannot self-collide, become tangled or form knots.

- Cables do not model torsion or twisting effects.

- It cannot simulate frictional contacts or elastic cable: all cables are assumed to have perfect friction, and be perfectly stiff.

Understanding Filo's streghts and weaknesses compared to other approaches are essential when it comes to choosing the appropiate simulation method. Using the wrong tool for the task will sign you up for a frustrating experience.

Basic concepts

Filo is made of 4 basic components:

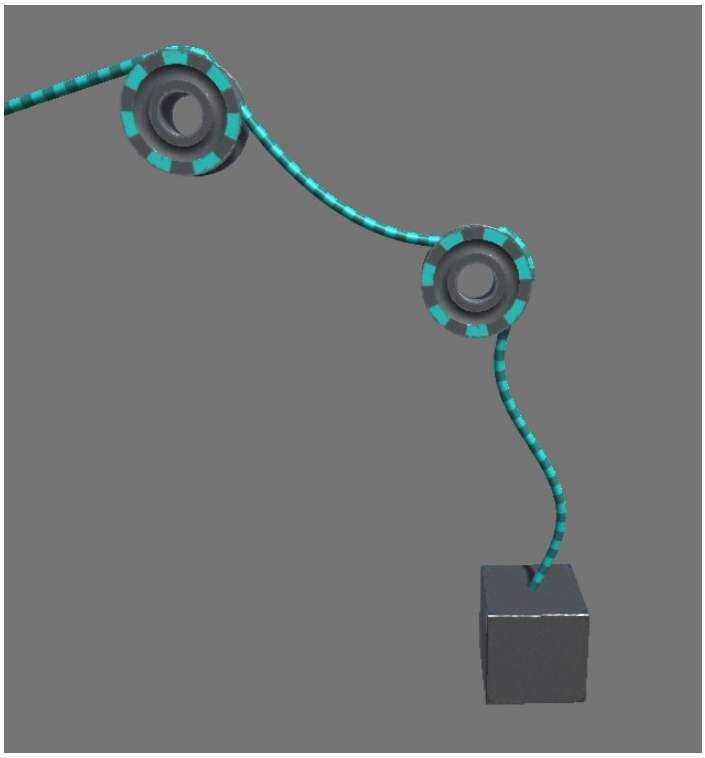
- Cable
- Cable Renderer
- Cable Solver
- Cable Body

Cables link several Cable Bodies together.

Each Cable must have a Cable Renderer component in order to be visible in the scene.

All cables belonging to the same system (that is, cables sharing at least one Cable Body) must be added to the same Cable Solver, who will be in charge of simulating the system.

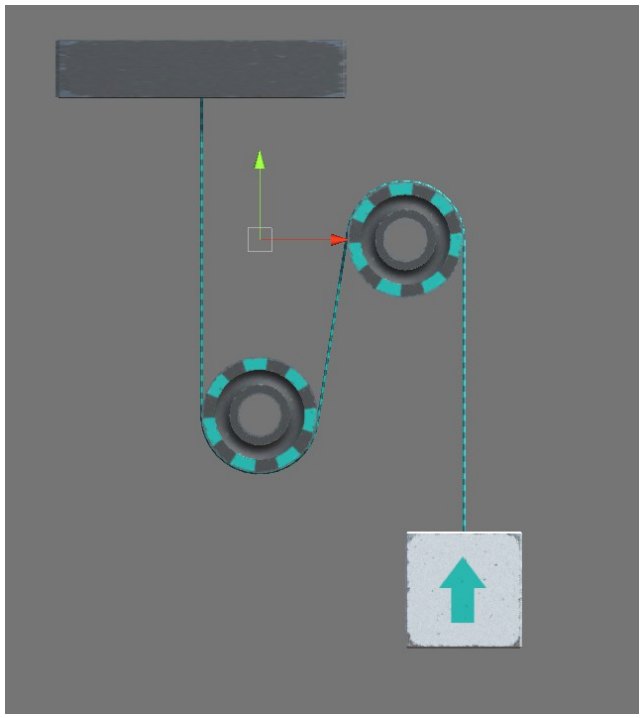
Filo considers cables to be massless, so cables will only apply forces to the objects they're linked to when completely tense.



When a cable section between two bodies has some slack, it is visually represented by drawing either a catenary curve or a sinusoid between the common tangent points to the 2 bodies linked by the cable section. It is possible to control the intensity of the catenary effect.

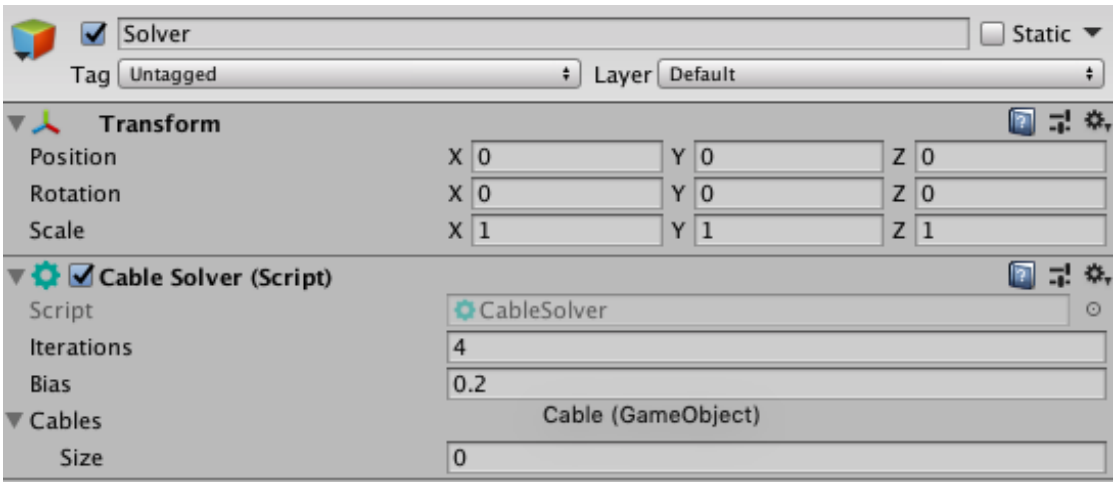
Setting up your first cable

Let's do a very simple compound pulley setup:

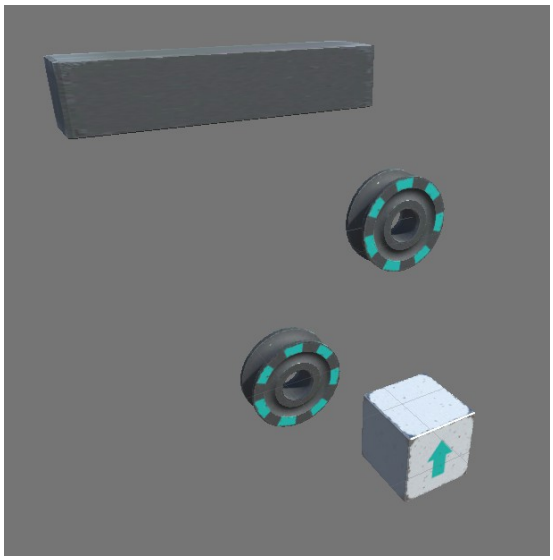


This setup consists of a single cable. No matter how many cables our setup has, we will need at least 1 Cable Solver in our scene. It does not matter which GameObject you add this component to, as long as it is present in the scene.

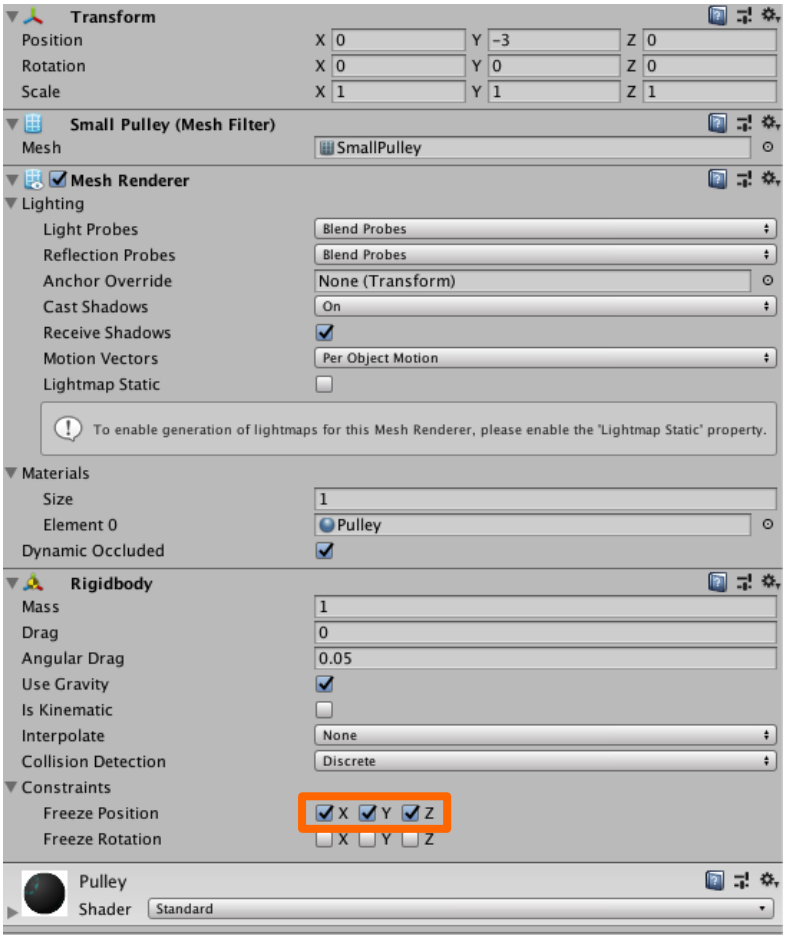
Go to "GameObject-->Filo Cables-->Cable Solver" to create a solver:



In addition to the cable itself (that we will create later) we'll need 4 Cable Bodies: 1 attachment point, 2 pulleys, and 1 weight.

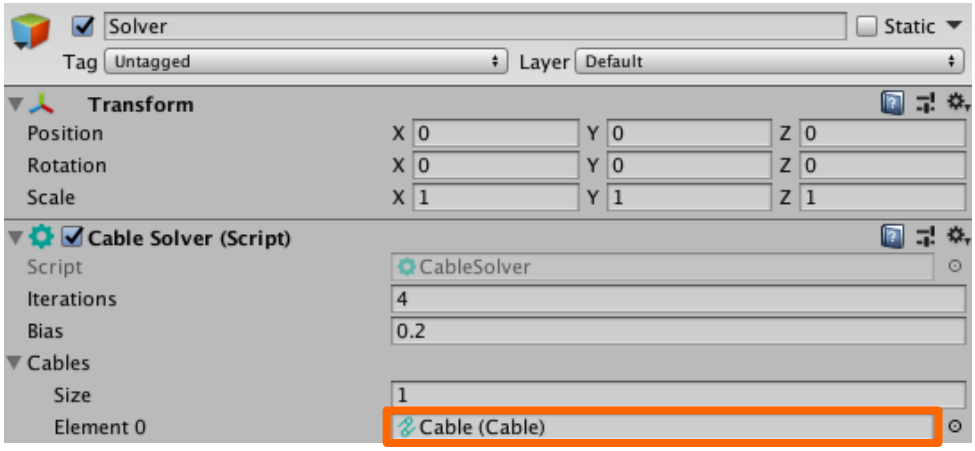


The pulleys and the weight will have physics applied, so we need them to have Rigidbody components. The attachment point does not, at is just a static ceiling. Create and position/rotate them correctl. Now enable freeze position in both pulley's constraints foldout, so that they don't fall into the void when we simulate them:



We will add a Cable Point component to the ceiling and the weight, and a Cable Disc to each pulley. The cable will be attached to both points, and roll over the discs.

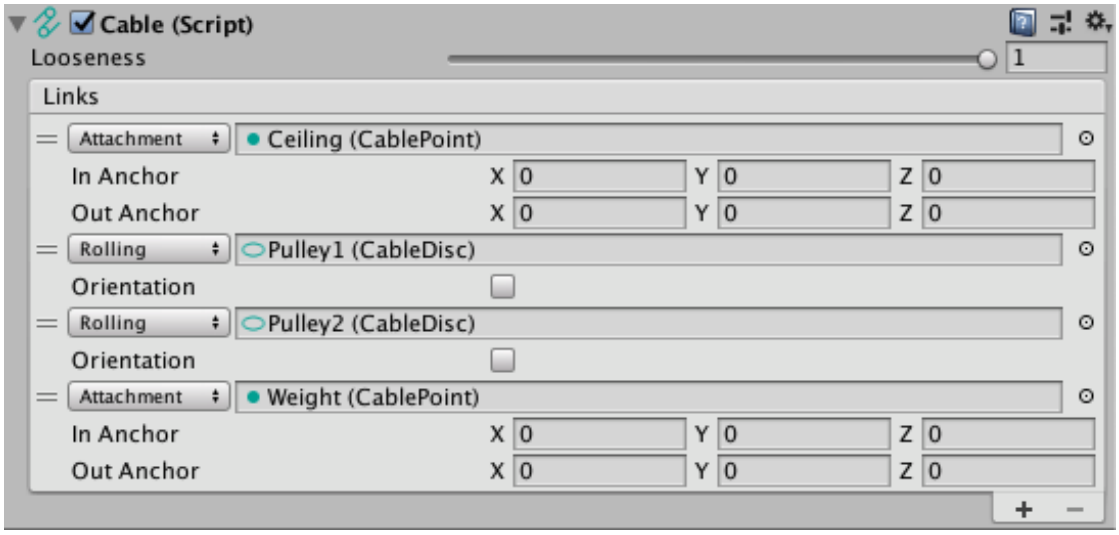
On to the cable itself. Go to "GameObject-->FiloCables-->Cable", this will create a new GameObject with the Cable and CableRenderer components. Then, drag the Cable to the Cable Solver's "cables" list. This is an important step, if you skip it the cable will not be simulated:



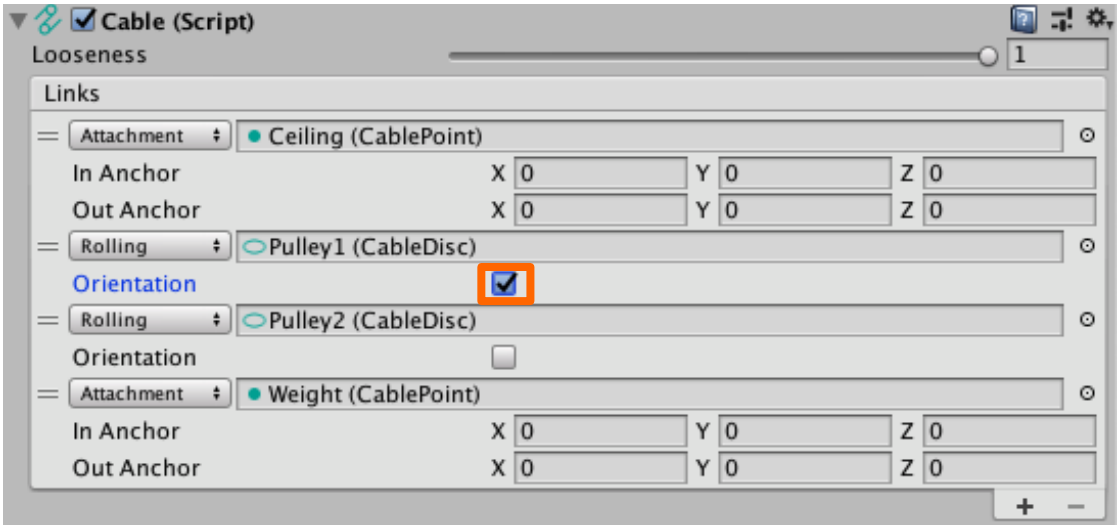
All that's left to do is tell the cable which objects it should connect and how it relates to them. For this we use links. Each link defines how the cable passes trough or attaches to a certain body.

Add 4 links to the cable by clicking on the small "+" button at the bottom of the link list. Then drag the 4 bodies in this order: ceiling, first pulley, second pulley, weight. The order in which bodies appear in the list defines the order in which the cable will pass trough them.

Now, use the link dropdowns to set the link type: the first and last links (ceiling and weight respectively) will be of type "Attachment". Both discs need to be of type "Rolling":



If the cable wraps around any of the pulleys in the wrong way (the first pulley probably will), toggle the "orientation" checkbox of the corresponding link. This controls which way (clock or counter-clockwise) should the cable wrap around a rolling link.



We are ready to go. Hit play, then move the pulleys or the ceiling around and see how the weight reacts to them.

Let's sum up all we did:

1. Create a Cable Solver.
2. Create the objects that the cable will interact with (add Rigidbodies where necessary)
3. Add Cable Bodies (Cable Discs, Cable Points or Cable Shapes) to the objects.
4. Create a Cable/CableRenderer combo, and add it to the solver's cable list.
5. Configure cable links.

Making more complex setups is usually just a matter of adding more links to the cable. You can also use multiple cables as long as you add them to the solver.

Component reference

Cable Solver

Iterations:

Number of constraint solving iterations per physics step. The default is 4. Higher values will yield more accurate simulation.

Bias:

Baumgarte stabilization factor. The default is 0.2. Higher values will yield faster positional correction response (less rubbery simulation). Too high values (approximately > 0.5) can degrade simulation stability.

Cables:

List of cables that should be solved together by this solver.

Cable Renderer

Section:

Cable section asset used to render the cable. For more info scroll down to Cable Section.

UV Scale:

UV scaling applied to the cable mesh. By default, U wraps once around the cable section, and V spans 1 meter of cable length.

Thickness:

Pretty self explanatory: cable thickness. Note that increasing the renderer thickness will not keep the cable from clipping trough Cable Bodies (discs, points, etc). You should adjust the disc radius, point anchor points, etc manually.

Edge Loop Spacing:

Distance in meters between edge loops when building the cable mesh. Smaller values will yield a denser, higher-quality mesh at a performance cost. Larger values will yield a coarser mesh.

Max Edge Loops Per Loose Joint:

Maximum amount of edge loops used to draw loose joints (both horizontal and vertical). This caps the amount of edge loops determined by *Edge Loop Spacing*.

Max Edge Loops Per Taut Joint:

Maximum amount of edge loops used to draw taut, straight joints. This caps the amount of edge loops determined by *Edge Loop Spacing*.

Looseness Scale:

Percentage of actual cable length visualized when the cable is not completely tense. A value of 0 will ensure cables look straight at all times. Larger values will use the catenary equation to represent cable slack.

Max Loose Cable:

Maximum length of loose cable visualized per segment when the cable is not completely tense. The actual cable length is first clamped so that it does not exceed MaxLooseCable, then multiplied by LoosenessScale.

Vertical Curlyness:

Number of waves used to draw loose vertical cables.

Vertical Threshold:

Maxiumum distance between the two links at the ends of a cable segment to consider the cable is vertical. This distance is measured as the distance between the projection of both tangent/attachment points on the XZ plane.

Cable

Dynamic Split/Merge:

Enabling this allows the cable to dynamically add/remove links when needed. New links will be generated when the cable intersects a body (this intersection test uses a Raycast, so make sure the body has a Collider). Links with negative stored cable (these that should no longer be in contact with the cable) will be removed.

Links:

List of objects linked by the cable. The order in which they appear in the list is the order in which the cable will pass trough them. You can add, remove or drag links to a new position in the list.

Every link type has an "orientation" attribute that determines how the cable is wrapped around the link (clock or counter-clockwise), and a "slack" attribute that adds extra cable length between that link and the next one.

There's four different link types:

- **Attachment: The cable will be firmly attached to two points defined in the body's local space: "In Anchor" and "Out Anchor". Its "Spawn Speed" determines the length of cable generated (or consumed, if negative) per second.**
- **Hybrid: A certain length of cable will be rolled around the body, simulating a spool. Once the cable is completely unrolled, the link will behave as an attachment.**
- **Rolling: The cable will be rolled around the body, but not attached to it.**
- **Pinhole: The cable will be able to slide trough two points defined in the body's local space: "In Anchor" and "Out Anchor".**

Cable Bodies

All cable bodies (Cable Disc, Cable Point and Cable Shape) have two properties in common:

Plane:

Plane on which the cable should wrap around the object. The plane is defined in the body's local space.

Freeze Rotation:

Freeze rigidbody rotation on all axis except the cable plane's normal. For instance, if the body's plane is set to "XY" and freeze rotation enabled, the body would only be allowed to rotate around the local Z axis. This is very convenient most of the time.

Note: If you're constraining the body's orientation using *joints,* then you must disable freeze rotation to maintain stability.

Cable Disc

Radius:

Radius of the disc in local space (takes uniform scaling into account).

Edge Loop Spacing Mulitplier:

Multiplies the spacing between edge loops for cable passing trough this specific disc. Allows you to fine-tune the quality of the mesh generated when rendering the cable.

Setting it to zero will disable cable rendering for this disc, this is specially useful when you have hybrid links (spools) with a large amount of stored cable and don't want to waste performance on generating and rendering such a large amount of cable mesh.

Max Edge Loops:

Maximum amount of edge loops used to draw cable passing trough this disc.

Cable Point

No properties of its own.

Cable Shape

Convex Hull:

A reference to the ConvexHull asset used by this shape. For more info on this, see the following section.

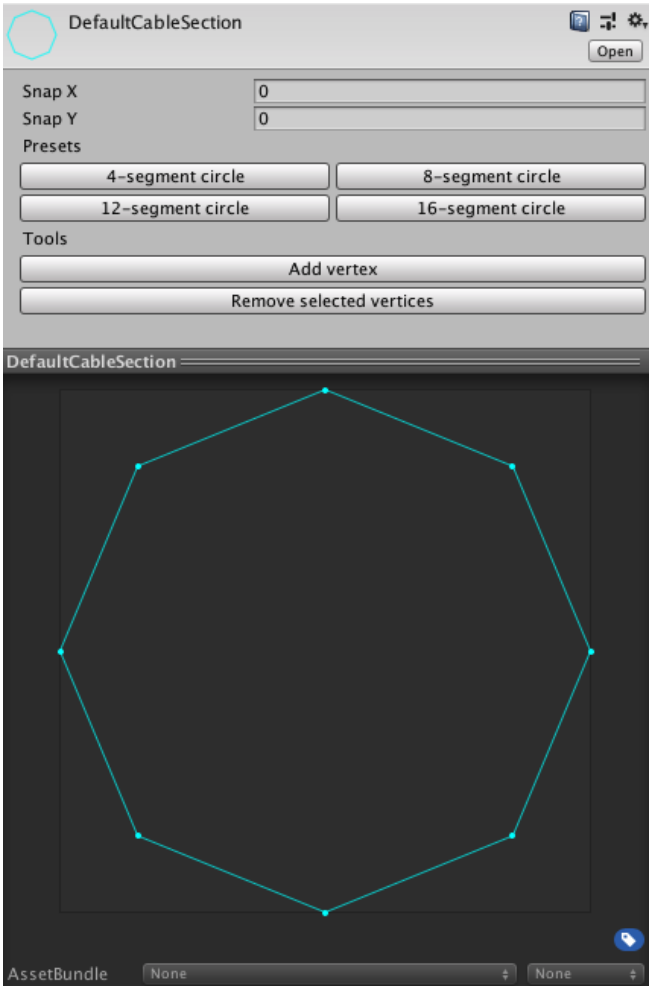
Asset reference

Cable Section

You can create a custom cable section asset by right clicking on the project window-->Create-->Filo Cables-->Cable Section.

Cable sections are used by Cable Renderers. They extrude the cable section along the cable's path to create a mesh that is later rendered by Unity.

This is what the Cable Section inspector looks like. You can click and drag section vertices around to create custom shapes, or generate one of the 4 pre-defined circular sections.

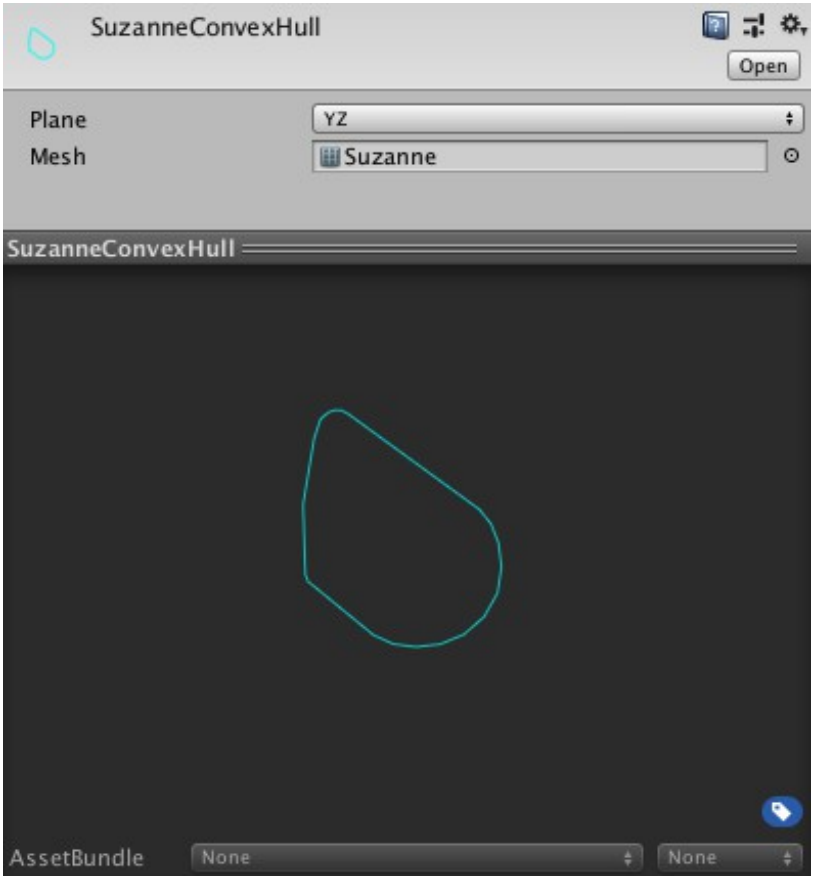


Convex Hull

In addition to the usual, disc-shaped pulley, Filo cables have the ability to wrap around any shape, both convex and concave.

A completely tense cable wrapped around any geometric shape always lies in the shape's convex hull, and Filo takes advantage of this. To calculate and store the convex hull of a shape, Filo provides the ConvexHull asset.

You can create one by right clicking on the project window-->Create-->Filo Cables-->Convex Hull. The asset will calculate the convex hull of the 2D intersection of the provided mesh with any of the 3 planes: XY,XZ,YZ.



Once generated, you can feed this asset to a Cable Shape component.