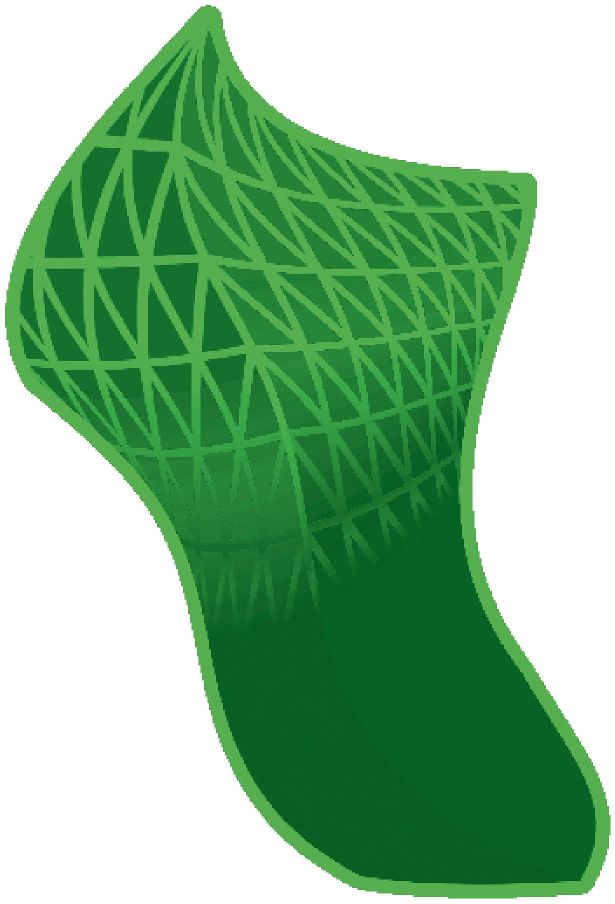
MushyMesh Documentation

version 1.0.0

Jack Herd, Wyatt Joyner, Ryan Kann, Julian McClinton
Practical Code LLC

June 6, 2019



Contents

1 Overview

MushyMesh is an intuitive tool that makes makes it easy to add simple Softbody physics for any Sprite. It works for any size and shape, and effectively replaces the Rigidbody component for game objects with sprites, with added on 'mushy' settings. Currently, MushyMesh only works for 2D sprites, however, future iterations will have support for 3D models.

Note: works with Unity 2019.1.2+

2 Getting Started

2.1 Downloading and Importing the Asset

Te get started using the MushyMesh tool, first download and install the package from the Unity Asset Store. The package includes both the core of the tool located under the /Scripts folder, and a series of demo scenes under the /Demos folder that may be useful to see what MushyMesh is capable of but not necessary to use the tool.

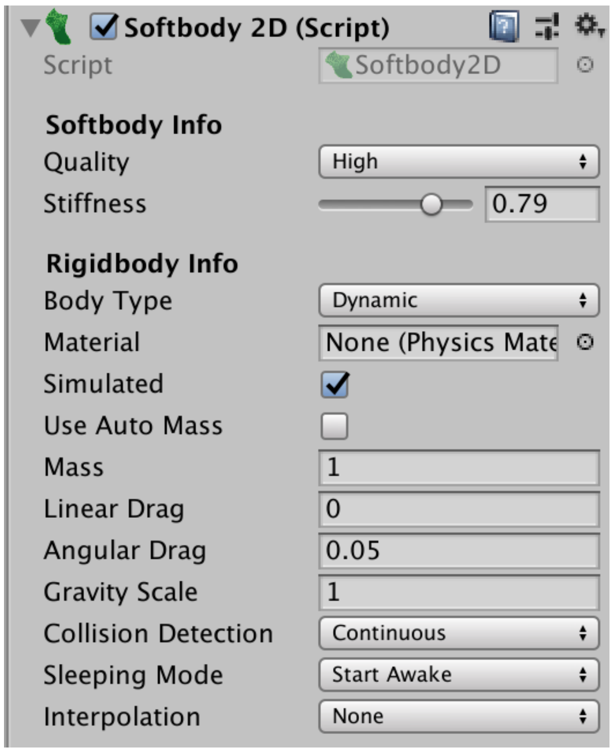
2.2 2D Softbody Test

Once you have the package installed, select the Sprite you wish to '*mushify*', and add the Softbody2D component. From there, you will see two Softbody settings, and the standard Rigidbody2D settings. At this point, the Softbody will automatically generate when pressing Play.

3 Navigating the Asset

3.1 Softbody2D

Used for creating a 2-dimensional Softbody. Requires a SpriteRenderer component.



Softbody 2D (Script)

Script	Softbody2D
Softbody Info	
Quality	High
Stiffness	0.79
Rigidbody Info	
Body Type	Dynamic
Material	None (Physics Mate
Simulated	✓
Use Auto Mass	☐
Mass	1
Linear Drag	0
Angular Drag	0.05
Gravity Scale	1
Collision Detection	Continuous
Sleeping Mode	Start Awake
Interpolation	None

3.1.1 Quality

Quality mandates the number of samples used in constructing the Softbody. Sampling is grid-based, with typical sample counts ranging from 25-250 points depending on quality. The only reason not to use the highest setting is if you're getting performance issues, which varies from computer to computer. In general, use *Ultra* when you have 1-4 Softbodies in the scene, *High* when you have 5-10, *Medium* when you have 11-30, and *Low* when you have 30+. Also note that you can mix and match these settings if you have some Softbodies that are more important than others.

3.1.2 Stiffness

Stiffness is a direct scale between floppiness and bounciness. The closer it is to 0, the floppier it becomes, and the closer it is to 1, the stiffer and bouncier it becomes. Typically speaking, you will get more consistent and realistic Softbody effects in the range of 0.7-1.0. Any lower than that, and you risk creating an overlapping visual bug.

3.1.3 Rigidbody Info

This section contains the exact same info present in Rigidbody2D. The reasoning for this is that Softbody2D is designed to circumvent the need for a Rigidbody2D. In that sense, Softbody2D is effectively a subclass. One thing that is missing, however, is the Constraints section, as it is not the purpose of a Softbody to have these Constraints.

3.2 Softbody2DEditor

This script currently doesn't do anything except make sprites read/write enabled.

4 Extended Functionality

4.1 Collision Events

Collisions are non-trivial with MushyMesh, as a singular object is comprised of multiple Rigidbodies and Colliders. As such, a collision on one piece does not inherently notify the parent. This limitation has been circumvented by directly sending the pieces' collision messages to the parent. To treat the object as a singular Collider, you must use the Collision prototype methods present in Softbody2D. If you need to use collisions in another script, it is recommended set up an Event to fire when those functions are called, and subscribe another function to that Event.

5 Limitations

While MushyMesh is functional, and can be used very broadly, there are some limitations to its capabilities. The current known limitations are as follows:

- MushyMesh doesn't handle curves too well
  - Sampling is based on a grid structure. As such, each sample is always equidistant in a lattice, and can't conform to curves. The resulting effect is that curved edges will often appear to have square chunks removed from them.
- Too many items can easily hurt the performance of your project
  - MushyMesh works by segmenting a single object into many discrete Rigidbodies connected with spring joints. Especially with higher qualities, having many objects can get incredibly taxing with Physics calls.
- At high velocities, Softbodies can permanently deform
  - As mentioned above, MushyMesh is based on a grid-like sample of points. At high velocities, these points can momentarily exceed the spring forces that keep them in place, causing them to get stuck in between 2 or more other points.

6 Features in the Works

- Allow the generation of the Softbody mesh to execute in the editor
  - We would like a special Editor interface to allow the user to generate *Mushies* without having to play the scene.
- Store the generated Softbody in a '(temp)' folder as a prefab
  - This is a standard feature of many different 'generator' assets. Instead of making the user drag in the *Mushy* GameObject from the scene to create a prefab, we'd like MushyMesh to handle that for the user.
- Port the mesh realignment to a Compute Shader

  - MushyMesh works by continuously realigning the vertices of a 2D mesh to a grid of physics-enabled CircleColliders. Shifting this process to a compute shader would greatly optimize the performance!

- Allow dynamic resolution changing to optimize performance
  - *Mushies* don't have to always perform at maximum mushiness. Sometimes, dynamically reducing the number of CircleColliders, and thus the intensity of the physics operations, could improve performance.

- Fix bounciness at high stiffness
  - At high stiffness, especially with higher Quality, the spring joints have very high spring forces that compound into a resulting bounciness. Can be fixed by tweaking with the drag on said joints.

- Include floppy mode, not just mushy
  - Sometimes you don't need a full softbody model. Instead, you might need something floppy, like an eel. Using roughly the same techniques we use to generate the *Mushy* bodies, we can allow this possibility. Our first implementation actually did exactly that, but the code wasn't very portable. We'll put it back in soon!

- Create optional advanced mode that allows lower level control
  - Functionality, such as control over the exact number of layers to have has been abstracted to a much higher level. The purpose of this is to keep it simple, and not overwhelm the user with a bunch of variables with many degrees of freedom. The presented choices provide a decent range of capabilities and behaviors, but we would like to allow the user to change those lower level variables if they know what they are doing.

- Make Softbody3D
  - While there are numerous solutions for 3D jellies, we think it would be a nice addition to the asset to incorporate 3D functionality. Convenient!