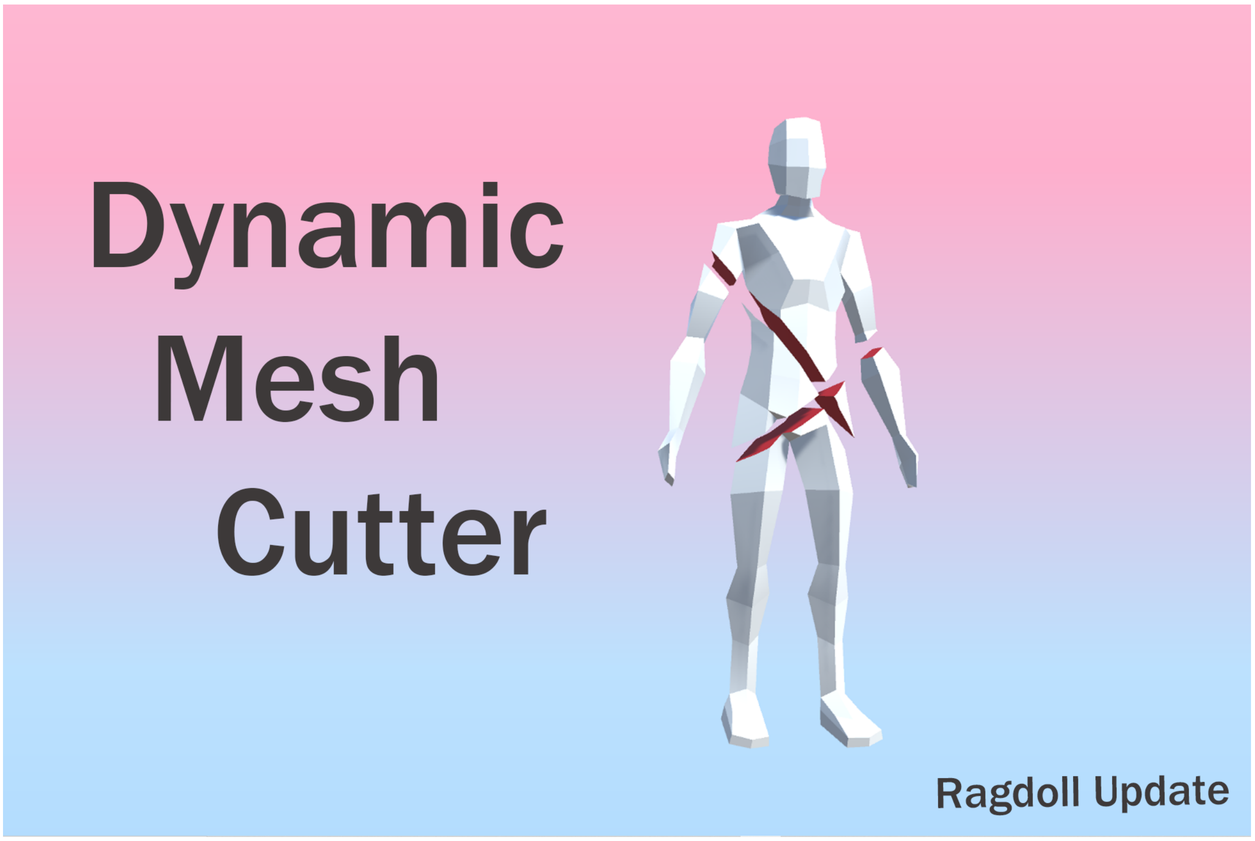
Dynamic Mesh Cutter Manual



Contact: philipbeaucamp.inquiries@gmail.com
Website: sites.google.com/view/philipbeaucamp
Unity Asset Store:http://u3d.as/2Hwc

Table of contents:

1. Quick overview

Dynamic Mesh Cutter can cut any mesh into **multiple** smaller meshes by passing a plane (i.e. a position and a normal vector) into the algorithm. Some the highlights that this package offers include:

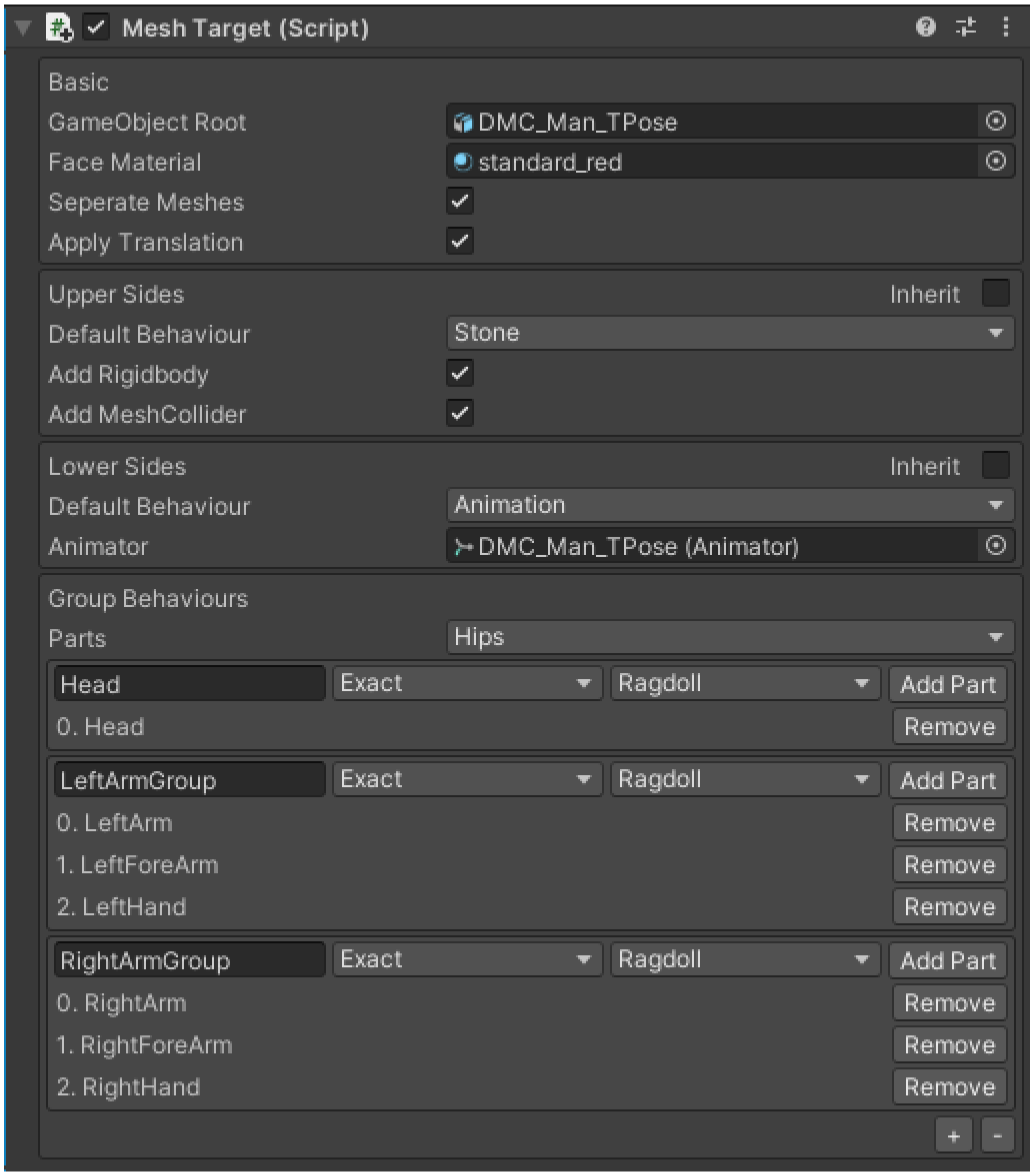
- Using multithreading to cut the meshes **asynchronously** while not blocking the main thread.
- Separating the newly created meshes/objects into **multiple different objects** (not just a “left” and “right” side) depending on the topology of the meshes.
- Skinned mesh cutting including **animated meshes** while respecting boneweights.
- “**Ragdoll**” cutting option to cut physics driven meshes.

This tool is optimized for performance to work even on **mobile or VR devices** and the user can manually change settings (see below) to further increase or decrease performance based on the desired end result of the cut.

2. Explanation of components

The scripts are separated into two folders, **Core** and **Utility**. Interested readers are welcome to browse the former to get insights into how the cutting, separation and creation of meshes is done, but it is not necessary to do so. The latter includes the MonoBehaviours which will initiate the cut and which hold the settings that require some attention.

2.1 MeshTarget.cs



Mesh Target (Script)

Basic	
GameObject Root	DMC_Man_TPose
Face Material	standard_red
Seperate Meshes	✓
Apply Translation	✓
Upper Sides	Inherit ☐
Default Behaviour	Stone
Add Rigidbody	✓
Add MeshCollider	✓
Lower Sides	Inherit ☐
Default Behaviour	Animation
Animator	DMC_Man_TPose (Animator)
Group Behaviours	
Parts	Hips

Head	Exact	Ragdoll	Add Part
0. Head			Remove
LeftArmGroup	Exact	Ragdoll	Add Part
0. LeftArm			Remove
1. LeftForeArm			Remove
2. LeftHand			Remove
RightArmGroup	Exact	Ragdoll	Add Part
0. RightArm			Remove
1. RightForeArm			Remove
2. RightHand			Remove

This script needs to sit on the object that holds the mesh renderer we wish to cut. Without it the algorithm doesn't know which mesh to cut. There are several options on the mesh target on how to treat the mesh when it is being cut.

Basic options:

> **GameObjectRoot**
> This reference needs to be assigned to what the user considers the root for the object. It can be the transform of the mesh itself, but should be set to the transform of the Animator or DynamicRagdoll components if present in the hierarchy. This root will also

be used to destroy the original object after the cut. If left blank the transform of MeshTarget.cs will be used instead.

FaceMaterial
Material that will be used for the new vertex faces created during the cut.
If left empty, the first instantiated material assigned to the renderer will be used instead.

SeparateMeshes
If true, the mesh on the left and right side of the cut will automatically be separated further into disjoint meshes. Since this comes with a performance cost it is recommended disabling this option if the mesh is already convex (for example if it has the shape of a ball or a cube).

ApplyTranslation
If true, translates the created meshes by a distance specified in the CutterBehaviour along the normal of the cutting plane, in the positive or negative direction depending on which side of the plane the mesh is.

Upper/Lower Sides:

For more control over the cut, a behaviour of type “stone”, “ragdoll” or “animation” can be set for the upper or lower side of the cut separately. The “Upper” side is quite literally the side that is higher up than the other one and can be both the left or right side of the cut. (If there are multiple “upper” or multiple “lower” sides because of mesh separation, then the settings apply to all of them).

Stone is the default case which will simply create and bake a new mesh when cut. (If a skinned mesh gets cut with the behaviour set to stone, the resulting mesh will be a non-skinned one). Optionally, a non-kinematic Rigidbody and/or convex MeshCollider can be automatically created after the cut is done. The latter option can occasionally throw errors if the cut creates an unusually small (or large) set of vertices, i.e. a mesh with less than 4 vertices etc.

Ragdoll requires a correctly set up DynamicRagdoll component to sit on a parent transform. If it is missing or if the ragdoll is set up incorrectly(for example missing colliders, etc) , the cut will automatically default to the “stone” behaviour. Else one can choose the result of the created ragdoll, setting it kinematic, non kinematic or leave the rigidbody the same as it was before the cut.

Animation requires an Animator component on a parent transform. If so, the resulting mesh (be it skinned or not) will keep playing its animation. If the component is missing, the cut will instead be that of the “stone” behaviour.

Remark When working with skinned meshes it is important that the root bone is correctly assigned in the inspector of the skinned mesh renderer.

If **Inherit** is true, the new MeshTarget will have the same Upper/Lower sides options as the original MeshTarget. If false, the created MeshTargets Upper/Lower options will **both** be the same behaviour as the one that was used to create it, i.e. if it is a "stone" mesh, both options for upper and lower will be stone too.

Group Behaviours:

Lastly, **if a DynamicRagdoll component is being used**, one can specify and overwrite the behaviour of the cut (Stone, Ragdoll or Animation) based on the collider groups of the ragdoll.
For example, one could choose the default behaviour to be "Animation" but want the "Ragdoll" behaviour if and only if a certain group of colliders has been separated from the rest of the original object, i.e. an arm from the rest of the body.
Group Behaviours can be added by pressing the "+" button and adding the individual parts (which are configured in the DynamicRagdoll component) to it. The behaviour can be set to execute when the newly created mesh

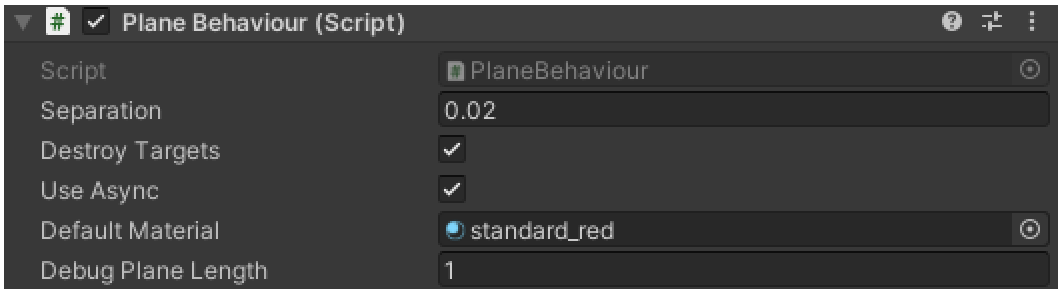
- contains **exactly** the specified collider groups,
- contains **any** of the specified collider groups, or
- contains **all** of the specified collider groups, but not exclusively.

If none of the above applies, the GroupBehaviour will be ignored.

2.2 CutterBehaviour.cs

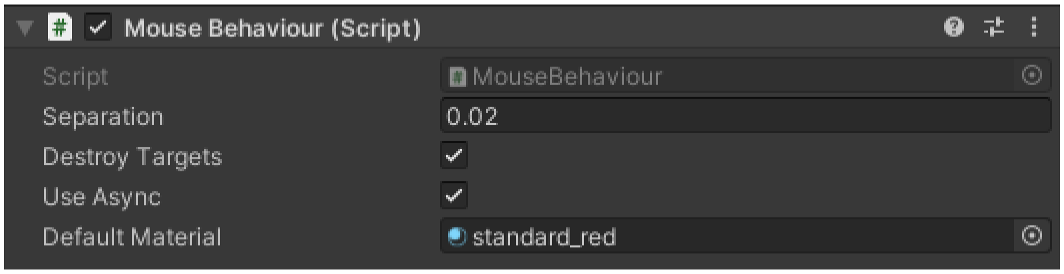
A MonoBehaviour which starts the cut and receives its callbacks. Currently, in the ExampleScene there are two different kinds of derived "CutterBehaviours", **PlaneBehaviour** and **MouseBehaviour**. The cutters can be adjusted as one sees fit and merely exist as a link between the input and the scripts that do the actual work.

PlaneBehaviour defines a plane using its position and forward direction as point and normal of the plane. Pressing the button labeled "Cut" during runtime will cut any mesh intersecting with the plane that has a MeshTarget.cs component added.


Plane Behaviour (Script)

Script	PlaneBehaviour
Separation	0.02
Destroy Targets	✓
Use Async	✓
Default Material	standard_red
Debug Plane Length	1

MouseBehaviour uses the user's mouse input to create the plane. When the behaviour is enabled, press the left mouse button down and drag (in the game view) to initiate the start point of the cut and release to create a plane (using the starting point, release point and the MainCamera position) which cuts any mesh intersecting with it that has a MeshTarget.cs component added.



Mouse Behaviour (Script)

Script	MouseBehaviour
Separation	0.02
Destroy Targets	✓
Use Async	✓
Default Material	standard_red

Both scripts allow for the following settings to be adjusted.

Separation
The distance by which the positive (negative) meshes will be moved in the (opposite) direction of the plane normal.

DestroyTargets
If enabled, the original object will automatically be destroyed after it has been cut.

UseAsync
If enabled, the cutting work will be done in a separate thread asynchronously, freeing the main thread without interrupting or freezing the game. Can be disabled, but should, except perhaps for performance debugging or if the target platform can't use multithreading, never be done.

DefaultMaterial
If the target does not have any materials specified in the mesh renderer nor the MeshTargetc component, this material will be used instead for the created faces of the cut.

Remark:
Two very useful callbacks of this class are:

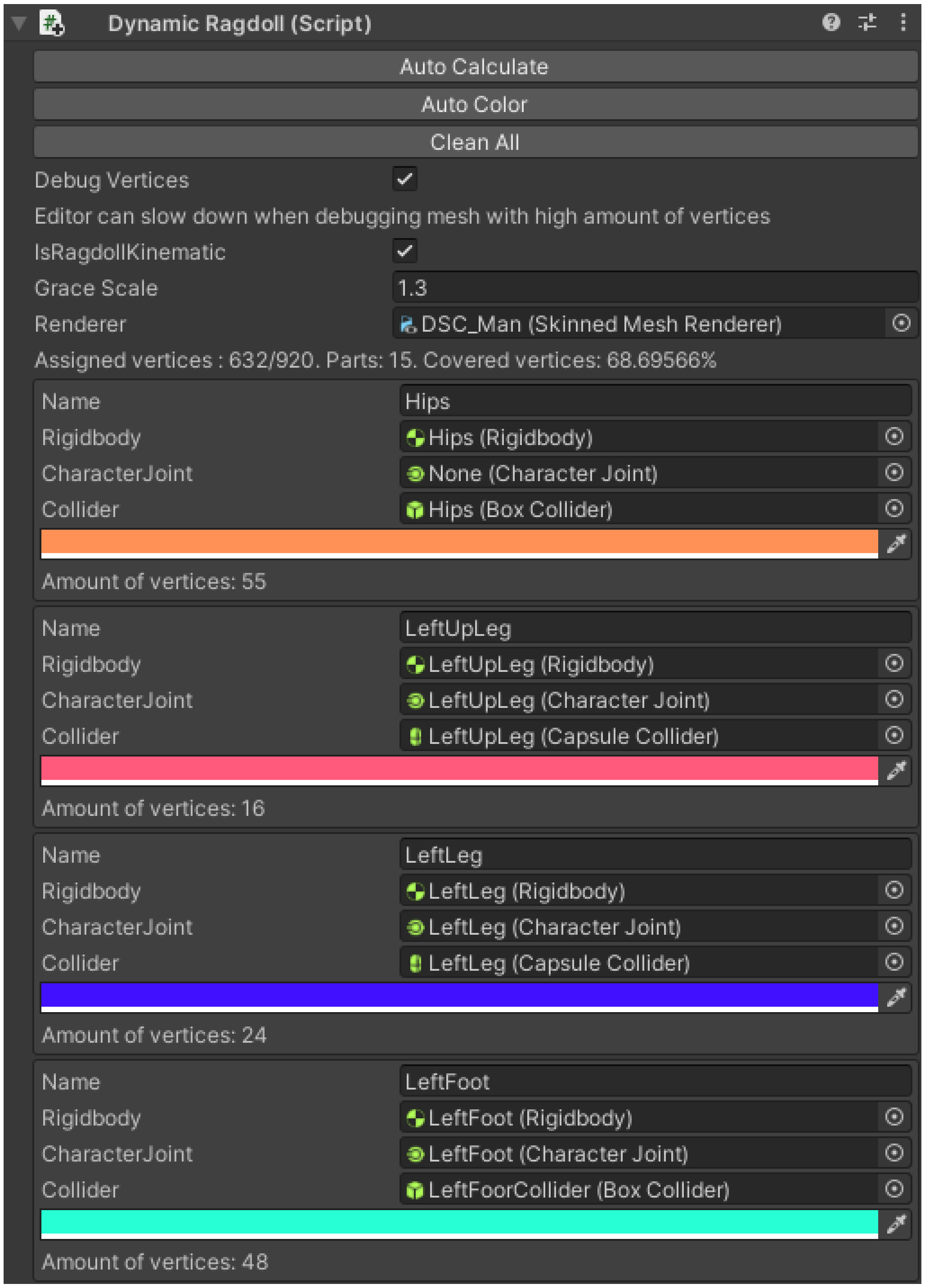
OnCut(bool success, Info info)
Called immediately after the cut algorithm has finished but **before** any objects are created or destroyed. The bool success will be false for any mesh that didn't intersect with the plane.

OnCreated(Info info, MeshCreationData creationData);

Called after a mesh has successfully been cut and passes the newly created objects as parameters.

2.3 DynamicRagdoll

The last core component is the optional DynamicRagdoll which can be added to dynamically separate joints, rigidbodies and colliders.



Debug Vertices ✓

Editor can slow down when debugging mesh with high amount of vertices

IsRagdollKinematic ✓

Grace Scale 1.3

Renderer DSC_Man (Skinned Mesh Renderer)

Assigned vertices : 632/920. Parts: 15. Covered vertices: 68.69566%

Name	Hips
Rigidbody	Hips (Rigidbody)
CharacterJoint	None (Character Joint)
Collider	Hips (Box Collider)

Amount of vertices: 55

Name	LeftUpLeg
Rigidbody	LeftUpLeg (Rigidbody)
CharacterJoint	LeftUpLeg (Character Joint)
Collider	LeftUpLeg (Capsule Collider)

Amount of vertices: 16

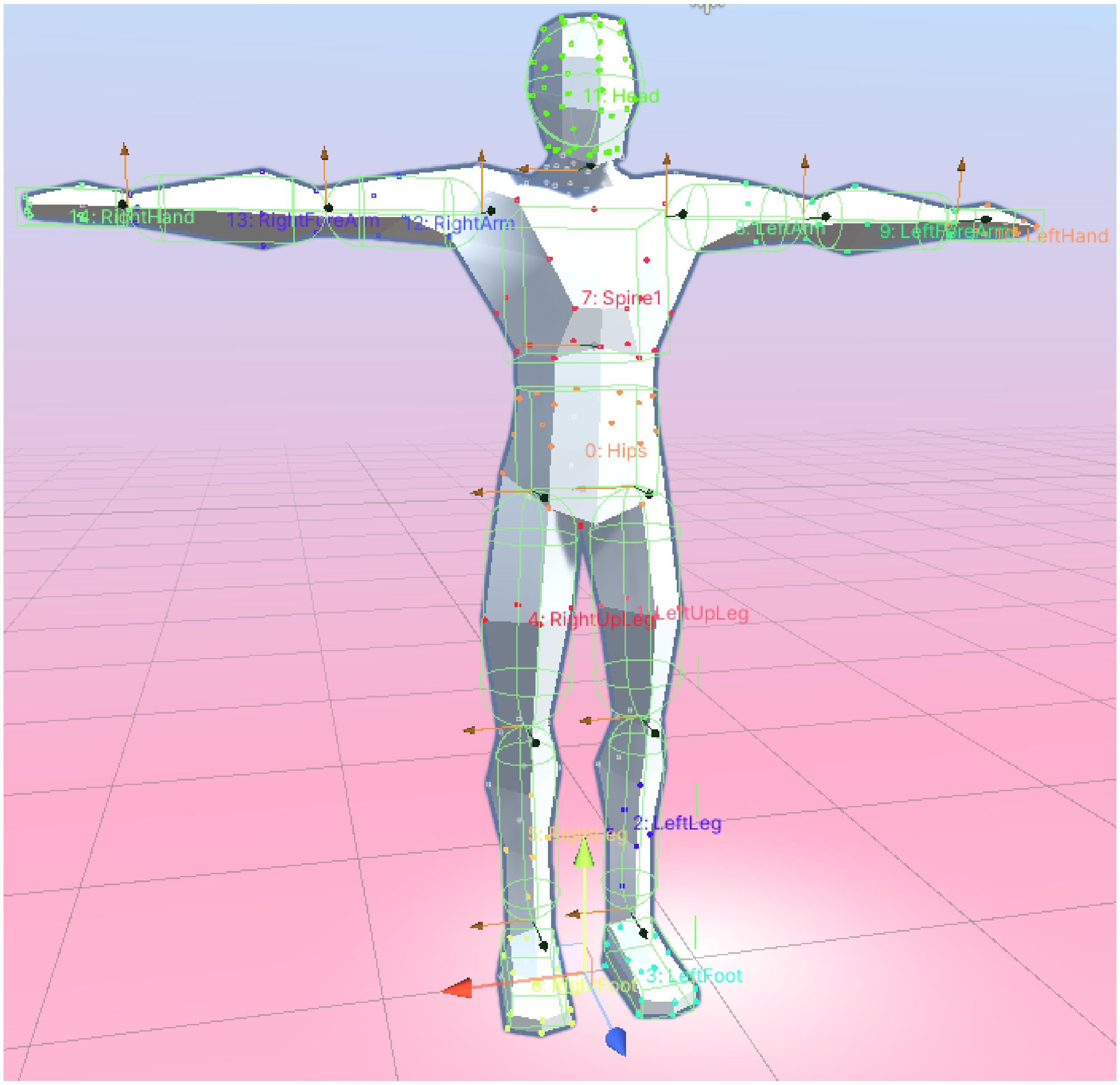
Name	LeftLeg
Rigidbody	LeftLeg (Rigidbody)
CharacterJoint	LeftLeg (Character Joint)
Collider	LeftLeg (Capsule Collider)

Amount of vertices: 24

Name	LeftFoot
Rigidbody	LeftFoot (Rigidbody)
CharacterJoint	LeftFoot (Character Joint)
Collider	LeftFoorCollider (Box Collider)

Amount of vertices: 48

To make use of the component, the object firstly needs to be set up with colliders, joints and rigidbodies. (This part of the process has to be done manually). Once the physics have been set up to one's likings, a click on the button "**Auto Calculate**" will automatically find and group all the colliders, rigidbodies and joints, as well as **assign and flag** all the vertices that fall within a collider group (a group can have multiple colliders belonging to one rigidbody).



When the mesh is cut, it will be separated into multiple smaller meshes potentially containing multiple collider groups, based on the flagged vertices present in the smaller, cut mesh. If enough flagged vertices of the collider group are present, the original collider groups components (rigidbody,joint and colliders) will be copied over to the new mesh.

Enabling **Debug Vertices** will show the color of the assigned vertices as well as the collider group label.

The **Grace Scale** can be thought of as a multiplier of the collider size which will be applied to calculate which vertices are inside (or close) to a collider when pressing “Auto Calculate”. For example, a box collider of size (1,1,1) with Grace Scale set to 1.3 will find all vertices of the mesh within a box collider of size (1.3,1.3,1.3) at its original center, and add them to the corresponding group. This setting exists, because oftentimes the colliders of a ragdoll do not encompass the actual vertices of the mesh, so adding a small padding might be useful.

A name and color can be given to each group. Pressing “**Auto Color**” quickly randomizes the colors for each group.

Remark:
Not all vertices have to be assigned to a collider group and the “covered vertices” percent does not indicate whether the dynamic ragdoll is set up good or bad. It is generally better to have a grace time close to 1 and accurate colliders than the opposite. The following example might help illustrate the point.

Example:
Take the above picture as a reference as assume there is only one sphere collider on the head and no other colliders. If the collider has a reasonable size and the GraceScale is for the purpose of this example set to a very, very high number, say 1000, then almost surely every vertice will fall into the collider group that we will call “head”. This means that even when the character is cut around waist height, the algorithm will assume the “head” collider group has been cut since all the vertices, even the ones below the waist, have been assigned to the “head” collider group. Thus, both the upper and lower newly created meshes will contain the “head” collider, because both meshes show vertices that belong to the group “head”.

3. Current limitations

- For the sake of performance, the current algorithm does **not** work well with holes inside meshes and might create unwanted triangles in cases where the face of a cut surface is not convex.

- Changing the animator's scale or transform hierarchy can easily result in unwanted behaviour which is the result of Unitys strict animation constraints. It is recommended to work with unit scales whenever possible.

Remarks on performance:
- The greatest contributor to a worsening of performance is the **amount of vertices** of the mesh, followed by the “separate meshes” option. Working with low poly meshes is therefore always recommended for performance.
- The algorithm is fastest for **convex meshes** that don’t have bone weights in which case **SeparateMeshes** can be turned off to increase performance drastically.
- If physics, ragdolls or Group Behaviours are not needed, there is no need for the optional DynamicRagdoll component. Adding one comes with a slight performance loss, although not as great as the “separate meshes” performance loss.

- Stone < Animation < Ragdoll in terms of performance, with Stone being the fastest. One should keep in mind that “Ragdoll” will use Unity Physics which can greatly impact the performance after the cut, especially when many rigidbodies and colliders are being used.