Useful Links:

Mesh Slicer Free on GitHub: https://github.com/hanzemeng/MeshSlicerFree
Mesh Slicer Free on Unity Asset Store: https://assetstore.unity.com/packages/slug/283149
Mesh Slicer Free's online documentation:
https://docs.google.com/document/d/1Muqt7BsIIzq-GR4BRaghu5qdzuZcEfcTd27qJ-Gr0GU/edit?usp=sharing

Purpose:

Mesh Slicer Free slices meshes into exactly two meshes. Mesh Slicer Free can handle some pretty complex meshes, such as those with holes. In general, as long as the mesh forms a closed volume and every triangle intersects other triangles only at the vertices, Mesh Slicer Free will slice correctly and, relatively, efficiently.

Slicer Usage:

You want to use this namespace:

```
using Hanzzz.MeshSlicerFree;
```

You want to create a Slicer object by:

```
Slicer slicer = new Slicer();
```

You want to slice the object by:

```
slicer.Slice(objectToSlice, slicePlane, intersectionMaterial);
```

objectToSlice is of type GameObject and has a Mesh Filter and Mesh Renderer component attached. **Make sure the mesh in the Mesh Filter component has the read/write option enabled.**

slicePlane is of type Plane. **Make sure this plane is defined using world space coordinates.**

intersectionMaterial is of type Material. The slicer will fill the intersections with this material.

If the slicer operates correctly, it returns:

```
Slicer.SliceReturnValue sliceReturnValue
```

The definition of SlicerReturnValue is:

```
public class SliceReturnValue
{
    public GameObject topGameObject;
    public GameObject bottomGameObject;
}
```

topGameObject is part of the original object that was on top of the slice plane. topGameObject has exactly 4 components attached: GameObject, Transform, Mesh Filter, Mesh Renderer. Notably, the Transform is at the root of the hierarchy and has the same position, rotation, scale as the original game object;
the Mesh Filter uses the sliced mesh;
the Mesh Renderer uses the original material(s) in addition to the intersection material.

Same story for bottomGameObject.

Note that the slicer does not modify any of the input parameters. If you need to destroy the original game object after the slice, you need to do it yourself (you got this).

The slice operation can be performed asynchronously:

```
Slicer.SliceReturnValue sliceReturnValue = await slicer.SliceAsync(originalGameObject, plane, intersectionMaterial);
```

Skinned Slicer Usage:

Mesh Slicer Free offers the SkinnedSlicer class to slice skinned meshes. Its syntax is similar to that of the Slicer class.

You want to create a SkinnedSlicer object by:

```
SkinnedSlicer slicer = new SkinnedSlicer();
```

You want to slice the object by:

```
slicer.Slice(originalGameObject, skinnedMeshRendererIndex, rootIndex, plane, intersectionMaterial);
```

originalGameObject is of type GameObject and is the parent of the object to be sliced.
skinnedMeshRendererIndex is of type int and should be the index of the child game object that has the mesh renderer component.
rootIndex is of type int and should be the index of the child game object that has the root bone.

SkinnedSlicer's SlicerReturnValue is identical to that of the Slicer.

Things That Should Work, But I Have Not Tested Them:

Should be able to slice meshes with multiple sub meshes.
Should be able to slice meshes with vertices that have all 8 UV channels.
Should be able to slice meshes with vertices that have color.

Things That Should Not Work:

If the mesh is weird (like having triangles clipping into each other, or having triangles that only form a plane), then do not expect Mesh Slicer Free to slice correctly.

Possible Extensions:

I would like to add the following extension to this product, but need help and incentive:
Need a better way to give uv coordinates to intersection vertices (right now all intersection vertices have uv coordinates of (0,0)).
Multithreading?

Contact Author:

hanzemeng2001518@gmail.com