Mesh Slicer 3

Content

About the product

You can cut objects by plane. It could be as a single static mesh or the a complex object consisting of several meshes, even skinned meshes.

Here you can find some folders: ObjectSlicer, CharacterSlicer, ObjectSlicerSamples and CharacterSlicerSamples. In your project, you can remove xxxSamples folders, because there are only examples of usage.

You can try a free version of this package. Check the reference to a free version on the product page.

If you have any problems, you can contact me. See contacts on my publisher page.

I would very much appreciate it if you would leave a review on the product page.)

Performance

Test on quite a complex model (vertex count - 17004) takes ~ 55 milliseconds of time on my core i7 2.2GHz in a release. With the option “Object Grouping” it will take ~ 4 times more time.

JIT compiler adds a significant 100 milliseconds to a first call, but you can avoid it by using IL2CPP.

How does it work?

To cut a simple non-skinned mesh you have to attach a BzSliceableObject component to your object and call it’s SliceAsync method with the await operator.

Example:

1) Get the “mesh slicer” component by the interface: var sliceable = GetComponent<IBzMeshSlicer>();
2) Create a plane by which you are going to cut: var plane = new Plane(...);
3) Cut: var result = await sliceable.SliceAsync(plane);

Main slicer component

There are two types of such a “slicer” components:

- BzSliceableObject - for a static mesh renderer;
- BzSliceableCharacter - for a skinned mesh renderer.

Both of this components are derive from the BzSliceableBase class and implements the ‘IBzMeshSlicer’ interface:

```
/// <summary>
/// Interface that implements mesh separation logic
/// </summary>
public interface IBzMeshSlicer
{
             /// <summary>
            /// Start slicing the object
            /// </summary>
            /// <param name="plane">Plane by which you are going to
slice</param>
            Task<BzSliceTryResult> SliceAsync(Plane plane);
            /// <summary>
            /// Wait all internal tasks to be completed
            /// </summary>
            Task WaitTasksAsync();
            /// <summary>
            /// Add a task to a slicer waiting queue
            /// </summary>
            /// <param name="task">Task to wait</param>
            void AddTask(Task task);
}
```

Base class BzSliceableBase

This is a base class, so its properties will be available for each implementation of the IBzMeshSlicer.

The base class provides you some methods that you can override:

- GetComponentManager - you can override the default component manager.
- GetNewObjects - control cloning of cutted parts.

And some base properties that you can use in the inspector:

- DefaultSliceMaterial - material that will be used on sliced faces;
- Asynchronously - set this flag if you want the object to be sliced asynchronously.
- WaitIfBusy - by default if a previous slice is in process (asynchronous case), the new one will be rejected. If this flag is set, it will wait previous to be processed and then process it again.
- ObjectGrouping - By default objects are sliced into two parts, negative and positive. For example, if you cut a table, all legs will be one object. Enabling this option will separate legs of the table into several objects. **Important! This option hits performans a lot!**

BzSliceableObject component

Add this component to the object that you want to cut. This component allows you to cut only a static mesh with the “MeshRenderer” component.

BzSliceableCharacter component

Add this component to the character that you want to cut. This component allows you to cut a skinned mesh.

Slicing of a skinned mesh renderer is very interesting, but it has some downsides:

- Performance is not very good. So, as maximum, you can use it for a low poly models.
- If the character was successfully sliced, no way it will continue its animation.

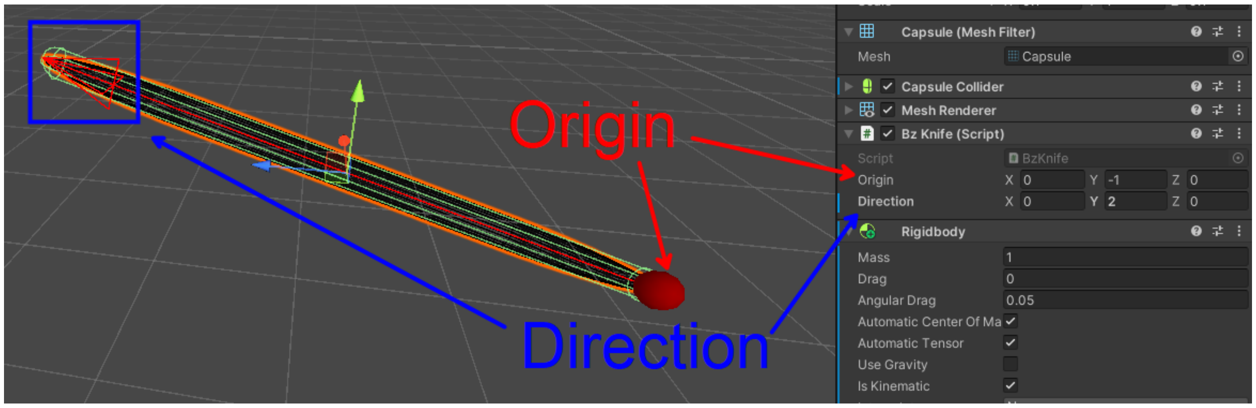
To be able to cut a character you have to:

- Add your character to the scene;
- Setup a ragdoll;
- Add a BzSliceableCharacter component;
- Then you can call a SliceAsync method of this component.

Slice objects with a knife

In the sample folder you and can find the BzKnife file.

1) Attach a collider and BzKnife component to you knife.
2) Setup knife origin and direction.
3) Move the knife around the scene to cut other objects.



```
Mesh                      Capsule
Script                    BzKnife
Origin                    X 0    Y -1   Z 0
Direction                 X 0    Y 2    Z 0
Mass                      1
Drag                      0
Angular Drag              0.05
Automatic Center Of Ma    ✓
Automatic Tensor          ✓
Use Gravity
Is Kinematic              ✓
```

The example scene of the knife component: testSceneKnife.

BzSliceConfiguration

If your sliceable object contains multiple renderers, you can configure each one differently by applying BzSliceConfiguration script to it.

Here you can configure some properties:

- Slice type - select one of available options:
    - Slice - object will be sliced
    - Duplicate - the object will be duplicated after the slice
    - KeepOne - object will not be sliced, and it stays only on one side of the plane
- Slice material - override a default cut material.
- Create cap - close openned part after the cut done
- Skip if not closed - do not create a cap if sliced part does not have a full closed loop.

Garbage Collector for infinitely falling objects

If you cut an object into very small parts, it sometimes happens that some very small parts fall through the ground and fall infinitely. These objects are useless and consume CPU time. To prevent it, use the component FallingObjGC to destroy them.

This component should be only one per scene. So, attach it to some static empty game object.

Set Min Pos Y to determine the minimum Y threshold. And if an object with a BzSliceableBase component will be below minimum position, it will be destroyed.

Extensions/Events

If you attach an object that implements an IBzObjectSlicedEvent interface, it will be invoked at certain points of time:

```
/// <summary>
/// Object Slicer event handler interface
/// </summary>
public interface IBzObjectSlicedEvent
{
		/// <summary>
		/// Called when a "Slice" is called
		/// </summary>
		/// <returns>If false, slice will be stopped</returns>
		bool OnSlice(IBzMeshSlicer meshSlicer, Plane plane);
		/// <summary>
		/// called when the object was successfully sliced
		/// </summary>
		void ObjectSliced(GameObject original, GameObject[] resultObjects, BzSliceTryResult result);
}
```

The method OnSlice is called befor the slice.

The method ObjectSliced is called after successful slicing.

Handler BzDrawDebugInfoHandler

This handler allows you to see a debug information.

Features:

- Show last slicer time.
- Draw center of the mass.
- Draw last slice attempts (draw a plane). It draws up to 5 last attempts.

Handler BzFixMass

BzFixMass - is an example implementation of the IBzObjectSlicedEvent interface where it fixes the weight and center of the mass of a sliced object.

Usually the center shoud be fixed automatilly by the PhysX, but in some cases it doesn't. This handler will help you if it doesn't.

Handler BzAvoidOversliceHandler

For example, you want to slice your model with a knife each time it will collide with it. You started to move your knife, as it close to the object you want to slice, it will collide with it. Then slice will be processed. But you knife is still close to it and you are still moving it though it. It could start new slice with an already sliced object. And the slower you move your knife the more times it will be sliced.

This handler allows you setup a minimum delay between slices and maximum number of the same object could be sliced.

Handler BzReaplyForce

If you slice an object, it creates several new objects that do not inherit velocity and angularVelocity. This handler fixes this issue.

Handler BzDeleteSecondJoint

Suppose you slice an object that was attached to another object via Joint. But new duplicated copies will all have their own Joint. The script deletes the Joint from the farthest objects from the anchor.

Handler BzSmoothDepenetrationHandler

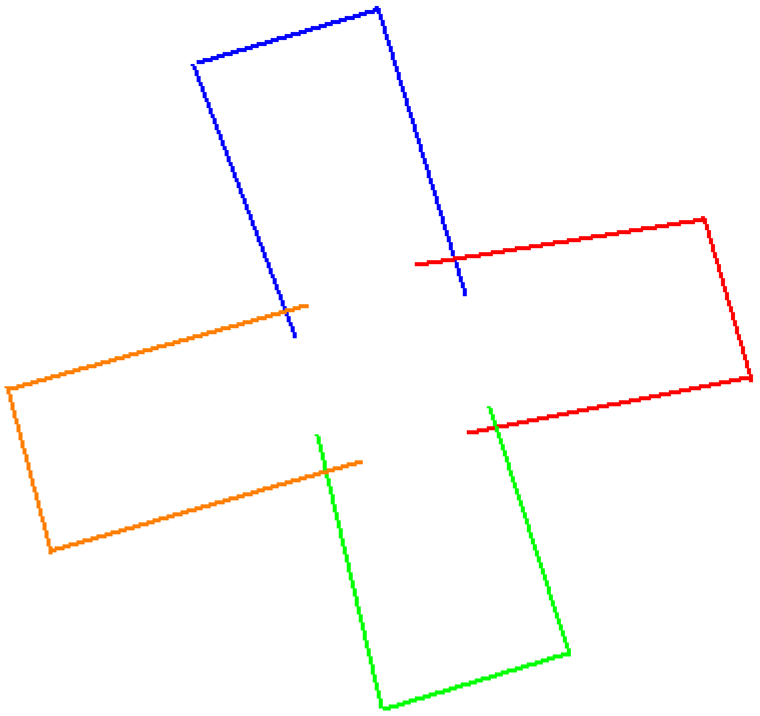
After the object sliced it creates two new objects and its colliders could bounce away from each other. This handler reduce depenetration velocity for a short period of time to avoid "bounce" effect.

Problem zones (issues, limitations)

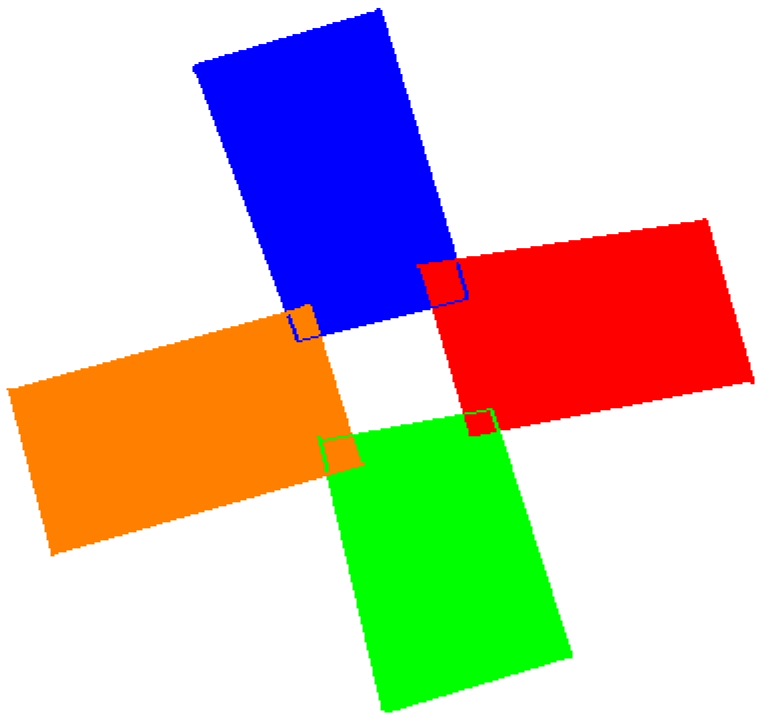
Of course, I'm working to get rid of problem zones. But they still exist. And there are some that I noticed:

Different sequences

If section view looks like this:

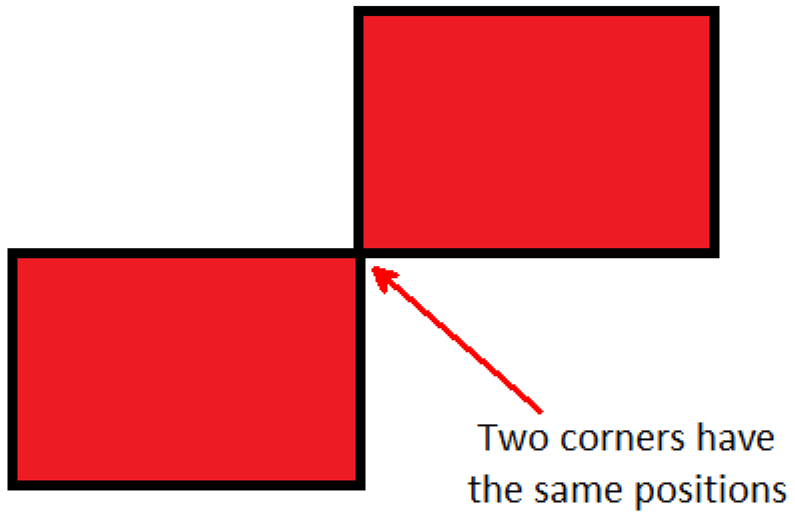


Each connected sequence of vertices will be joined to its own polygon. The result is 4 polygons, but the center will stay empty:



Chessboard

In this case, the model could be closed incorrectly



Flipped normals

If your model has back-flipped normals, it will close the sliced part incorrectly.

Non-1 scale

The scale of each game object should be 1,1,1.

First slice lag

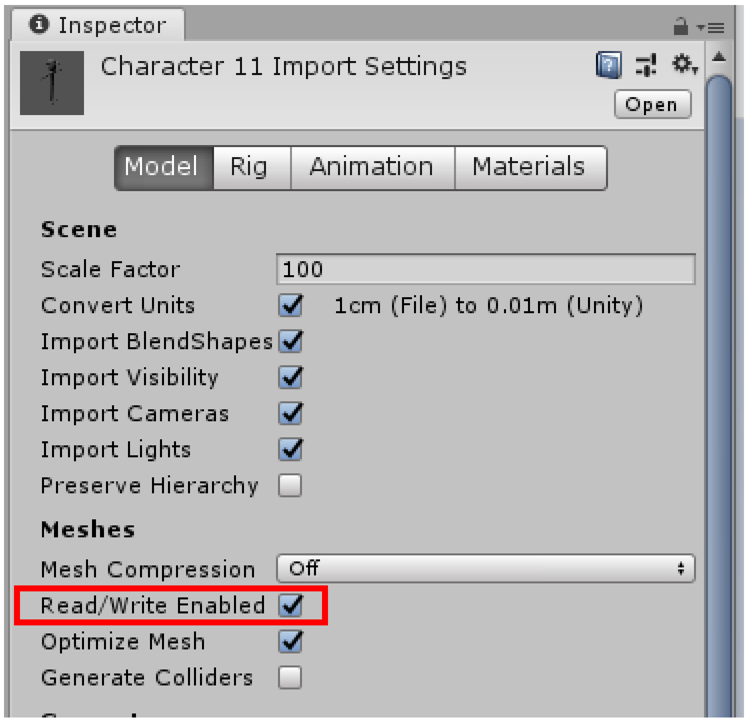
This issue is related to JIT implementation. On the first slice, it has to compile the whole IL code to native machine code and it takes extra time.

To solve this use IL2CPP.

FAQ

Error: Not allowed to access vertices on mesh '...' (isReadable is false; Read/Write must be enabled in import settings)

Read/Write operations are disabled for your model. You can enable it in the model properties window:



Scene

Scale Factor 100
Convert Units ✓ 1cm (File) to 0.01m (Unity)
Import BlendShapes ✓
Import Visibility ✓
Import Cameras ✓
Import Lights ✓
Preserve Hierarchy ☐

Meshes

Mesh Compression Off
Read/Write Enabled ✓
Optimize Mesh ✓
Generate Colliders ☐