Mesh Slicer

Content

About package

You can slice objects by plane synchronously/asynchronously. You call the Slice method and after a while, a callback method will be called.

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

You can try free version of this package: on GoogleDrive

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

If you're interested, you can send me references to your game with this plugin and I'll add them to this documentation.

I would very much appreciate 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 release.

How does it work?

Each sliceable object in game have to implement 'IBzSliceableAsync' interface:

```
/// <summary>
/// Asynchronously sliceable object
/// </summary>
public interface IBzSliceableAsync
{
      /// <summary>
      /// Start slicing the object
      /// </summary>
      /// <param name="plane">Plane by which you are going to slice</param>
      /// <param name="sliceId">To prevent multiple slice requests you should
use sliceId</param>
      /// <param name="callBack">Method that will be called when the slice
will be done</param>
      void Slice(Plane plane, int slicdId, Action<BzSliceTryResult> callBack);
}
```

The IBzSliceableAsync interface is implemented in abstract BzSliceableObjectBase class which you have to inherit. Example of inherited class is ObjectSlicerSample.

How to slice:

1) Get sliceable component: var sliceable = GetComponent<IBzSliceableAsync>();

2) Create plane by which you are going to slice: var plane = new Plane(...);
3) Slice: sliceable.Slice(plane, sliceId, null);

To Slice method you need to pass sliceId to avoid slicing the same object several times within one swing. So it is better to create a slice_id manager to get a new unique id each time. Or you can pass zero to rely on delay between slices.

Base class BzSliceableBase

This is a base class. But you should not inherit from it directly. You need to use one of its derived classes BzSliceableObjectBase or BzSliceableCharacterBase.
Anyway, base class provides you some methods that you could override:

- StartWorker - method that creates a worker thread. You may need it, for example, if you want to use your own thread pool.
- OnSliceFinishedWorkerThread - this method will be called in worker thread after the work done.
- GetNewObjects - control cloning of slicing objects.

And some base properties that you can use in inspector:

- DefaultSliceMaterial - material that will be used on sliced faces
- Asynchronously - set this flag if you want that the object will be sliced asynchronously.
- UseLazyRunner - split processing to several frames. Increase performance.

Implementations of BzSliceableObjectBase

In your class you have to make your implementation of abstract BzSliceableObjectBase class. It requires you to implement this methods:

- protected abstract BzSliceTryData PrepareData(Plane plane) - Prepare data before slice
- protected abstract void OnSliceFinished(BzSliceTryResult result) - Will be called when the slice will be finished

ObjectSlicerSample

For example, you can see sample implementation of ObjectSlicerSample class:

```
public class ObjectSlicerSample : BzSliceableObjectBase
{
	protected override BzSliceTryData PrepareData(Plane plane)
	{
		// remember some data. Later we could use it after the slice is done.
		// here I add Stopwatch object to see how much time it takes
		// and vertex count to display.
```

```
                ResultData addData = new ResultData();

                // count vertices
                var filters = GetComponentsInChildren<MeshFilter>();
                for (int i = 0; i < filters.Length; i++)
                {
                        addData.vertexCount += filters[i].sharedMesh.vertexCount;
                }

                // remember start time
                addData.stopwatch = Stopwatch.StartNew();

                // colliders that will be participating in slicing
                var colliders = gameObject.GetComponentsInChildren<Collider>();

                // return data
                return new BzSliceTryData()
                {
                        // componentManager: this class will manage components on sliced objects
                        componentManager =
                                new StaticComponentManager(gameObject, plane, colliders),
                        plane = plane,
                        addData = addData,
                };
        }

        protected override void OnSliceFinished(BzSliceTryResult result)
        {
                // on sliced, get data that we saved in 'PrepareData' method
                var addData = (ResultData)result.addData;
                addData.stopwatch.Stop();
                drawText += gameObject.name +
                        ". VertCount: " + addData.vertexCount.ToString() + ". ms: " +
                        addData.stopwatch.ElapsedMilliseconds.ToString() + Environment.NewLine;
        }

        static string drawText = string.Empty;

        void OnGUI()
        {
                GUI.Label(new Rect(10, 10, 2000, 2000), drawText);
        }

        // DTO that we pass to slicer and then receive back
        class ResultData
        {
                public int vertexCount;
                public Stopwatch stopwatch;
        }
}
```

CharacterSlicerSampleFast (beta)

Simple implementation for skinned mesh slicer. Why I'm saying that it is beta, is because it has some downsides:

- Performance is not very good. So, as max you can use it for low poly models.
- Number of slicing of the same model have to be limited to, as max, 2.
- If the character was successfully sliced, no way it will continue its animation.
- slicing will not break connections on each side even if visually nothing is connecting it.

So do not blame me for that, I'm trying to make it better)).

BzSliceableCharacterBase

Slice objects with a knife

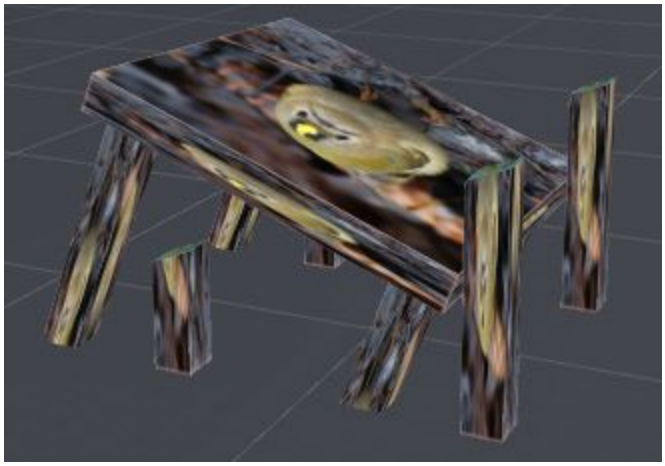
In sample folder you and find two files "KnifeSliceableAsync" and "BzKnife":

- KnifeSliceableAsync.cs - for objects that you want to slice via knife (e.g. an apple);
- BzKnife.cs - for object that you want to slice with (e.g. knife).

KnifeSliceableAsync must be attached to the objects with rigidbody on it.

BzKnifeSliceable

Simple implementation that slices mesh.
Test scene: "BzKovSoft/ObjectSlicerSamples/testSceneAsync". If you click on the Table in the Game window, it will be sliced to two objects. And every time you click on it, it will be sliced again:



Important to note, the table sliced only into two objects, so legs at the bottom will be connected.

BzSliceConfiguration

If your sliceable object contains multiple renderers, you can configure each differently by applying BzSliceConfiguration script to it.
Here you can configure slice material and slice type.
Slice types:

- Slice - object will be sliced
- Duplicate - object will be duplicated after slice
- KeepOne - object will not be sliced, and it stays only on one side of the plane

Garbage Collector for falling objects

If you cut an object to very small parts, it sometimes happens that some very small parts fall through the ground and fall infinitely. These objects are useless and consumes CPU time. To prevent this, use component FallingObjGC.
This component should be only one per scene. So attach it to some static empty game object. Set Min Pos Y to determine minimum Y threshold. And if some objects with BzSliceableBase component will be below this value, it will be destroyed.

Extensions/Events

If you attach an object that implements an IBzObjectSlicedEvent interface, will be invoked after slicing:

```
public interface IBzObjectSlicedEvent
{
        void ObjectSliced(GameObject original, GameObject resultNeg, GameObject resultPos);
}
```

The method ObjectSliced will be invoked after successful slicing where it passes old object and new instantiated.

BzFixMass

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

BzFixMassSmart

Correctly fixes weight and center of the mass of sliced objects. It works correctly only for closed objects.

BzReaplyForce

If you slice an object, it creates two new objects that do not inherit velocity and angularVelocity. This class fixes this problem.

BzDeleteSecondJoint

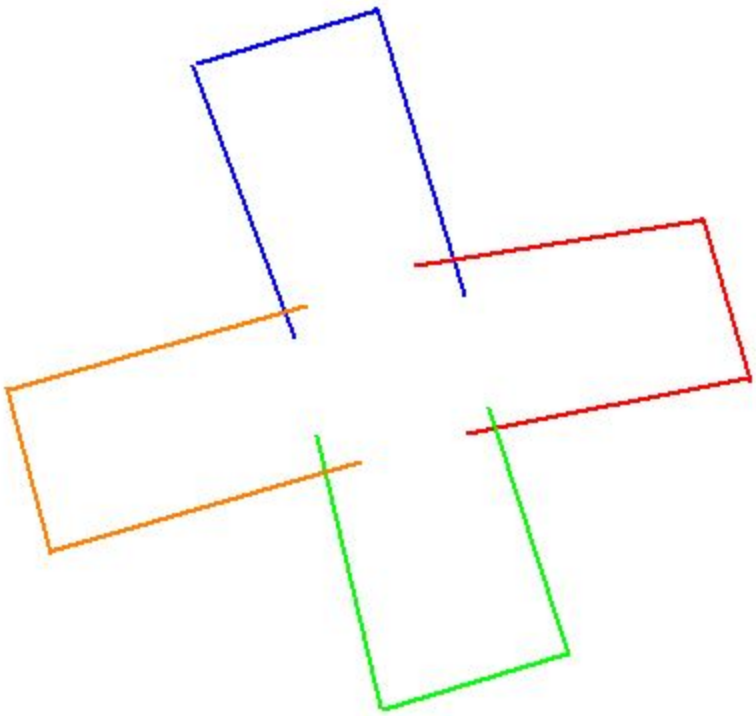
Suppose you slice an object that is attached to another object via Joint. But the two duplicated copies will both have their own Joint. The script deletes the Joint from the farthest object from the anchor.

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 vertex will be joined to polygon. The result is 4 polygons, but the center stays empty: