This documentation is just an extension for light version package (http://u3d.as/qSN) that describes only specific to this package aspects.

About package

This package allows you to slice your character.
It totally includes mesh slicer package that you can find here: http://u3d.as/qSN
For more latest documentation, always use online version of this document.

Limitations

- One character can be sliced only once.
- Slicing is taking some time. For example, on my 2.2 GHz it takes ~ 30-80 milliseconds for my character model with 12974/3 triangles.
- slicing do not break connections on each side even if visually nothing is connecting it.

How to use it

1) Create your own implementation of BzSliceableCharacterBase
2) Add this component (your implementation) to your player.
3) Find IBzSliceableAsync on object you want to slice and call its Slice method.
4) As BzSliceableCharacterBase implements IBzSliceableAsync interface, so it will be sliced if plane intersects your object.

BzSliceableCharacterBase implementation

In your class you need to implement two methods:
- BzSliceTryData PrepareData(Plane plane)
- void OnSliceFinishedChar(BzSliceTryResult result)

In PrepareData you need to prepare data important for slicer and return it in BzSliceTryData class:
- componentManager - see IComponentManager implementation
- plane
- addData - you can pass any data. You will get it in OnSliceFinishedChar method.

In OnSliceFinishedChar you need to convert your player to ragdoll. Also you can add any effect you want: spraying blood, sparks etc.
This method will be called after slice will be done.
The income parameter BzSliceTryResult is a summary data, that, for example, you can use to find points where it intersects with plane.

```
public class BzSliceTryResult
{
```

```
	public BzSliceTryResult(bool sliced, object addData)
	{
		this.sliced = sliced;
		this.addData = addData;
	}

	public BzMeshSliceResult[] meshItems;
	public readonly bool sliced;
	public readonly object addData;
	public GameObject outObjectNeg;
	public GameObject outObjectPos;
}

public class BzMeshSliceResult
{
	public BzSliceEdgeResult[] sliceEdgesNeg;
	public Renderer rendererNeg;

	public BzSliceEdgeResult[] sliceEdgesPos;
	public Renderer rendererPos;
}

public class BzSliceEdgeResult
{
	public PolyMeshData capsData;
}

public class PolyMeshData
{
	public Vector3[] vertices;
	public Vector3[] normals;
	public Vector2[] uv;
	public int[] triangles;
	public BoneWeight[] boneWeights;
}
```

Also there are some optional methods that you can override:

- StartWorker - for example, if you want to use your thread pool.
- OnSliceFinishedWorkerThread - this method will be calld in worker thread after the work done.

IComponentManager implementation

IComponentManager is needed to manage GameObjects and its components like joints, so that after slice it would be connected properly.
Now there is only one implementation for character slicer:
CharacterComponentManagerFast.
To create one, you need to pass 3 parameters to constructor:

- Root gameObject of you player
- Plane

- Colliders that will participate in slicing.