_projDawn

Version 1.0.0

Project Dawn | Dots Mesh Slicer | Tool

Overview

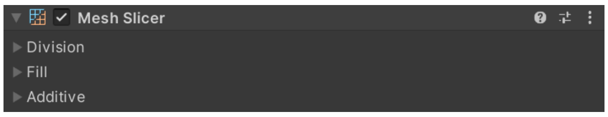
This package contains high performance mesh slicer tool. It is built to be used as an extension for existing Unity builtin packages (etc. SIMD mathematics, Collections, Jobs and Burst compiler).

Library is developed to fit well with Unity DOTS standards. This includes performance by default, simplicity and most importantly, readability of the code. Driven to take the best HPC# can offer.

Mesh Slicer

Using a mesh slicer is quite simple and the workflow follows a very similar path to most of Unity already existing components. For any game object that has `MeshFilter` and `MeshRenderer` you need to add a new component called `MeshSlicer`. This component will mark game object as slicable object. The component also includes useful settings to modify slicing logic that will be covered later on.

Component can be added following menu item `Component/Mesh/Mesh Slicer`.

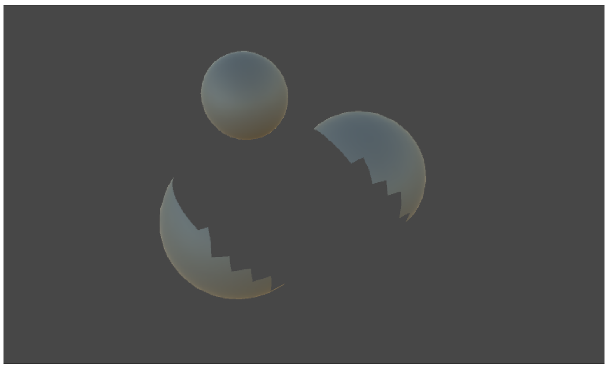
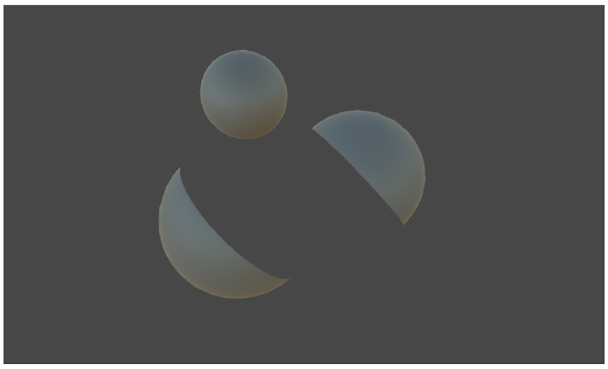


Mesh Slicer component inspector

Division Settings

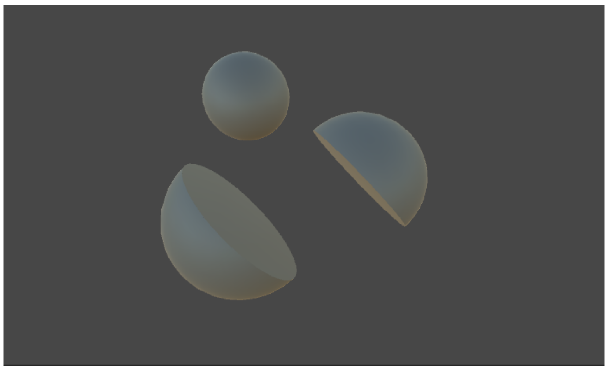
Settings controls how mesh will be divided.

Discrete vs Linear:

Fill Settings

Settings controls the filling generation after the slice. Also known as cap.



> **Note:** Currently only convex is supported. Concave support will be added in the future.

Addive Settings

Enables scheduling multiple slices for a single `MeshSlicer`. Internally, all jobs are scheduled into a binary tree structure that gets executed.

Mesh Slicer Manager

As mentioned in section above, the `MeshSlicer` component is there to mark game object slicable and provide settings. However to schedule slice jobs you need at least one game object with `MeshSlicerManager` component as manager. It is recommended to use `MeshSlicerManager.GetOrCreateManager` to get default manager.

The main methods:

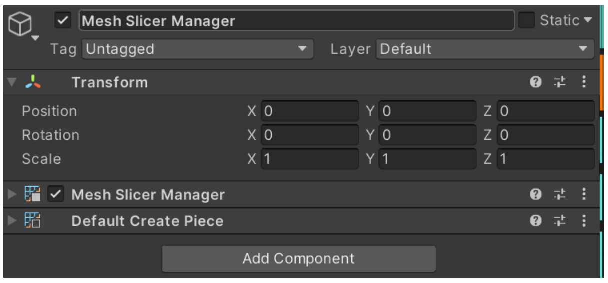
- `MeshSlicerManager.GetOrCreateManager` - returns or creates a new manager. Typically, `MeshSlicerManager` is a singlaton. For better performance, it is recommended to cache the returned manager.
- `MeshSlicerManager.Slice` - schedules slice job.
- `MeshSlicerManager.IsCompleted` - returns true if slice job is completed.
- `MeshSlicerManager.Complete` - waits on thread until slice job is completed.

Modifiers

To extend `MeshSlicerManager` slice logic there is special concept called modifiers. Modifier is simply a `MonoBehaviour` component that implements specific mesh slicer interfaces.

Add Modifier

Adding a modifier is quite trivial. Firstly, you need to create game object with `MeshSlicerManager` component. Finally, you add any modifier component (etc. `DefaultCreatePiece`).


Mesh Slicer Manager
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Mesh Slicer Manager
Default Create Piece

Mesh Slicer Manager game object with modifier `DefaultCreatePiece`

Custom Modifier

As mentioned in the section above, to create a custom modifier, you need to create a new `MonoBehaviour` component that implements one of the following interfaces (Also, the component must be added to the same `GameObject` that contains `MeshSlicerManager`):

- `IGetMeshSurfaces.GetMeshSurfaces` - callback to collect mesh surfaces from the mesh slicer.
- `ICreatePiece.CreatePiece` - callback to handle when the new piece is created after the slice. Can return null to skip this piece.
- `IScheduleSliceJob.ScheduleSliceJob` - can be used to schedule additional jobs to modify `MeshSurface`. One of possible use cases is to add texture coordinates, tangent.

> **Note:** If you need different modifiers, you can always request them (Check support section).

As example:

```
class MyCustomCreatePiece : MonoBehaviour, ICreatePiece
{
    public MeshSlicer CreatePiece(MeshSlicer source, Mesh.MeshDataArray
meshDataArray, int index)
    {
        // TODO: Code here
    }
}
```

Low Level

For more extensive modifications, it is recommended to directly use the low level slicing API `Slicer.Slice`. This is the low level function that schedules slice job and outputs a **single** piece of `MeshSurface`.

Tests and Samples

The package contains samples and tests, so it is quite recommended to check them to get an overall sense of API usage.

Limitations

- Fill mesh (Also known as cap or internal mesh) generation is currently only convex.
- No mesh separation. As example, slicing letter C vertically will not produce three pieces.
- No skinned mesh slicing support.

> **Note:** These limitations will be removed in future releases.

Dependencies

- Tested with Unity 2020.3
- Package com.unity.mathematics@1.2
- Package com.unity.collections@0.9
- Package com.unity.burst@1.4
- Package com.projectdawn.dotsplus@1.5

Support

If you have questions, bugs or feature requests use Discord.