Mesh Melter v1.0

by Cloverbit (pixeldust@cloverbit.com)

Introduction

An easy way to apply a meltdown effect to any mesh.

Mesh Melter aims to provide a drag and drop noninvasive solution to integrate meltdown effects in your game.

See below an example of Mesh Melter in action.

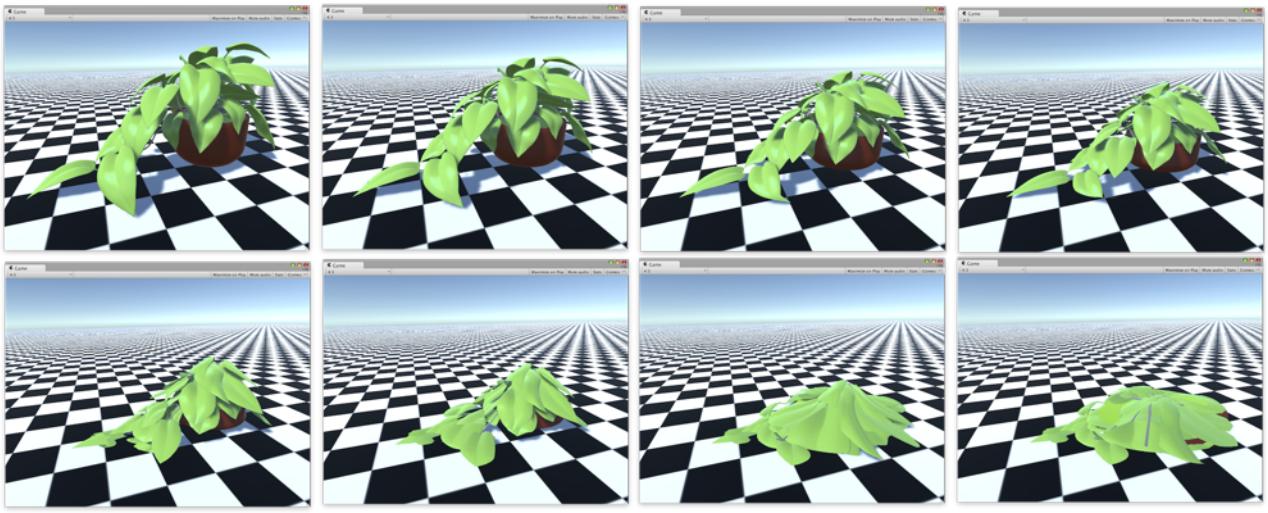


Figure 1: Mesh Melter applied to a plant model.

Mesh Melter can melt composite objects with many children; each child with its own local rotation and scale. See an example in the following picture.

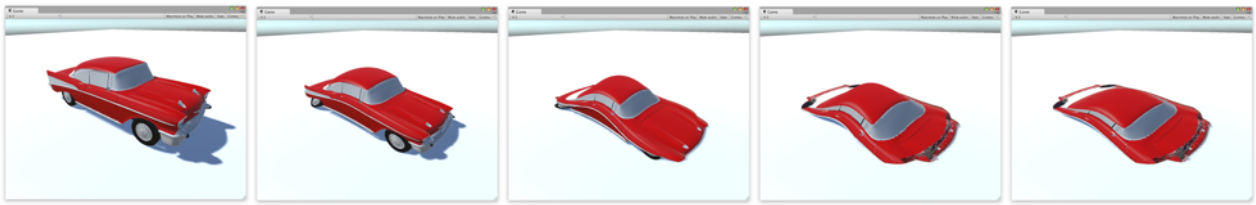


Figure 2: Mesh Melter applied to a car model with 63 meshes using 19 different materials.

Setup

Locate the Melter script inside the MeshMelter folder and add it to the target gameobject. The gameobject will not be modified in any other way.

When melting composite objects, you need to add the component only to the root object.

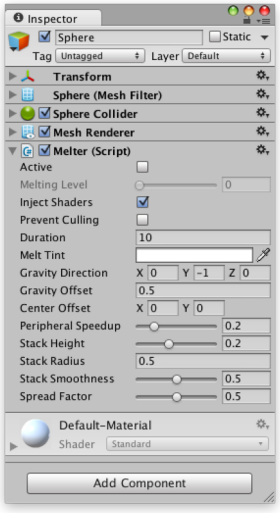


Figure 3: *Melter inspector panel*

Component Parameters

The default values should be fine to start tweaking.

Active

When this flag is set, the melting is ongoing. You can pause at will by changing its value. Once the meltdown is complete, the value will be automatically set to false.

If this flag is set in the editor, the effect will start as soon as you start the game.

Melting Level

The current melting level applied to the mesh.

NOTE: This parameter becomes available only after the melter has been activated for the first time. This is because all the geometry and interface parameters must be push to the shader first.

Inject Shaders

The first time the effect is started, all materials used by the object will be cloned (this way we are not going to change your assets). If this flag is set, all shaders will also be set to the melting shader.

If you unset this flag, the component will assume the melting shader has already been set and will run using the current materials. This can be useful if you need to melt only a portion of your gameobject.

NOTE: If this flag is not set and you forget to set the shader manually in at least one material, there will be no melting at all.

Prevent Culling

The actual name for this flag should be "Prevent Frustum Culling".

To optimize performance, unity is not pushing to the graphic pipeline any mesh outside of the camera frustum. As a result, a displaced mesh will disappear when the original vertices are all out of the viewing area. This will happen if you zoom very close to the melted mesh, even in the editor window.

NOTE: This flag will prevent frustum culling but might introduce some artifacts in the projected shadows.

Duration

The total effect duration, in seconds.

Melt Tint

While melting, this color will be gradually applied to the mesh. Set it to white for no changes.

Gravity Direction

This will be the melting movement direction. This vector is in **world coordinates** and is shown as a makeshift green arrow in gizmos when the gameobject is selected.

Gravity Offset

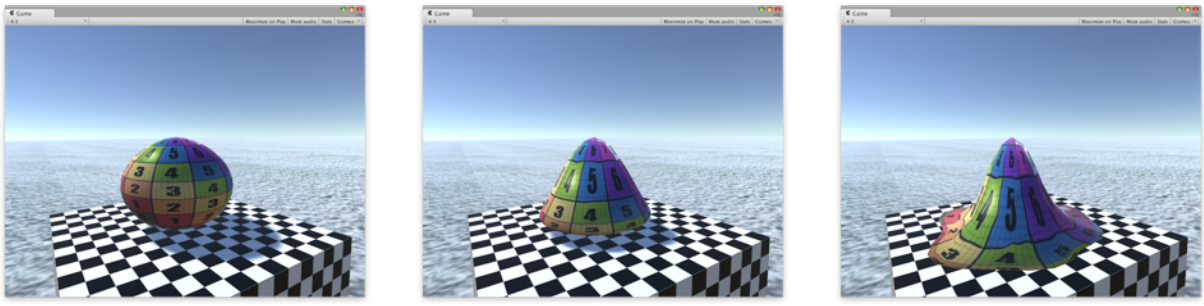
This is the distance in **world space** the mesh will fall during the meltdown. By default, it is calculated using the bounding box along the gravity direction. Changing this offset may be useful when, e.g., melting an object which is suspended in midair. The collapsing plane is shown as a red gizmo rectangle.

Center Offset

This 2-dimensiona offset in **world space** will be applied to the geometric center location (the blue ball in gizmos) shifting it along the collapsing plane. The top of the melted stack will be relocated accordingly. Changing this offset may be useful when the mesh is asymmetrical and the main body is not aligned with the center of the bounding box.

Peripheral Speedup

External vertices should be dropping faster than central ones. This parameter is an amplification factor for that. Putting this flag to 0 will result in all vertices falling at the same (unrealistic) speed while setting it to 1 will make external vertices hitting the ground much earlier. See the following pictures for an example.



Peripheral speed 0 | Peripheral speed 0.5 | Peripheral speed 1

Figure 4: Spheres at melting level 25% with different peripheral speedup values.

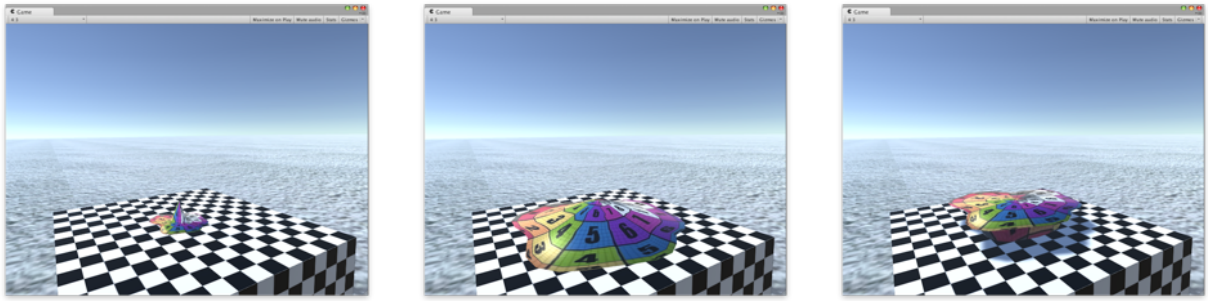
Stack Height

The melted mesh is not going to be flat but will accumulate in a hill-shaped stack. This parameter set the height of the hill as a **fraction of the original mesh bounding box** calculated along the gravity axis. Setting this value to 0.5 will result in a melted stack which is high as half of the original mesh.

Stack Radius

This is the radius of the final melted stack as a distance in **world space**.

NOTE: the melted mesh will adjust the stack shape but will not fill it. Setting this flag to large values will result in holes below the melted stack or having a melted mesh stay away from the ground. See the following pictures for an example.

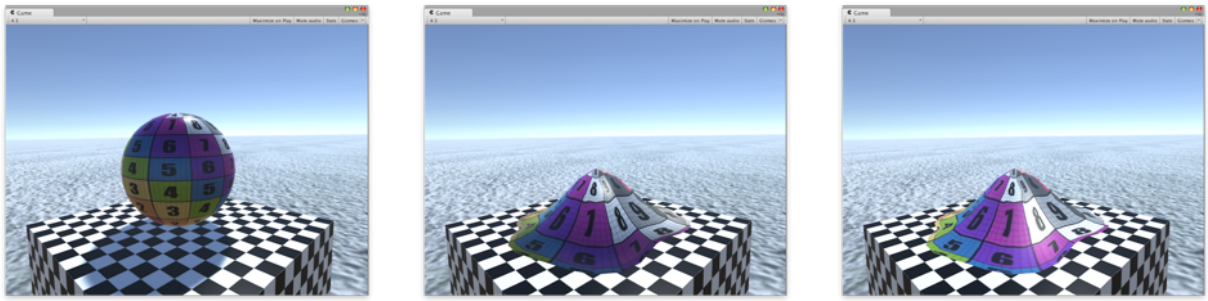


Stack radius 0 | Stack radius 0.5 | Stack radius 2

Figure 5: Different valued for the stack radius for a melted sphere.

Stack Smoothness

This parameter controls the normals during meltdown. If it is set to 0 the original normals will be kept, giving the impression of a rough surface. If it is set to 1 normals will converge to the melted stack normals, giving a very smooth surface. Keeping it in the middle will produce a melted stack with some hints of the original mesh. See the following pictures for an example.

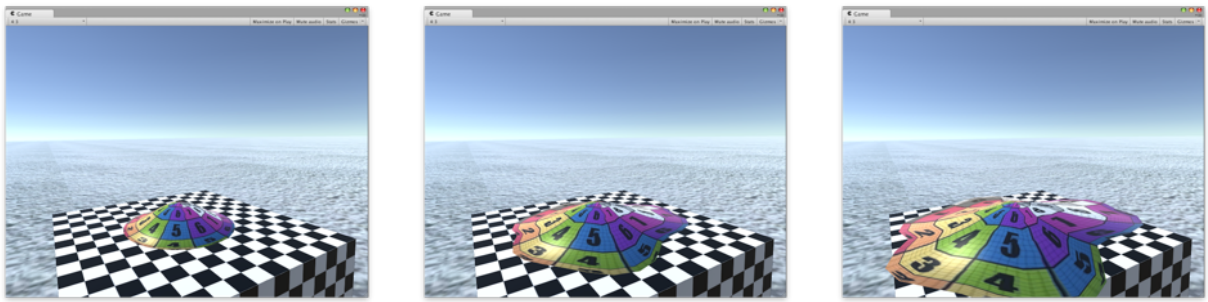


A sphere | The sphere melted with stack smoothness 0 | The sphere melted with stack smoothness 1

Figure 6: Comparison between melting a sphere with different values of stack smoothness.
Note that if smoothness is 0 shadows stick to the same position as in the original model.

Spread Factor

When reaching the ground, the mesh will spread away to simulate a puddle. This parameter sets how far the puddle will extend from the melted stack border. Setting the spread factor to 0 will make a round stack with the requested radius while setting it to 1 will spread the borders up to the double of the stack radius. On top of this, a radial noise is applied to get irregular borders. See the following pictures for an example.



Spread factor 0 | Spread factor 0.5 | Spread factor 1

Figure 7: Different spread factors applied to a stack radius of 0.5.

About the Shader

Please, read this section!

The melter shader is a surface shader based on the standard one.

If your model is using the standard shader for rendering, it is unlikely you will see any major artifact during meltdown. If, on the other hand, you are using a legacy or custom shader, it may be possible to have a noticeable visual difference. In such case, your best option is to modify the surface part of the shader to suite your requirements.

VERY IMPORTANT: **Leave the shader in a Resource folder when you build**.

The shader is loaded at runtime. So, the build will not contain it if there is not a static reference in at least one material. If the melting effect is working when playing inside the editor but not in a standalone build, it is possible you have this problem.

Another option is to include the shader in the "Always Included Shaders" list. To access this panel from the menu use Edit → Project Setting → Graphics.

Putting Mesh Melter Into Your Projects

As already mentioned, adding Mesh Melter into your project scripts is quite easy.

When you want to start a melt, just set the active field to true;

```
myGameObject.GetComponent<Melter>().active = true;
```

Synchronizing Melts with the Rest of your Game

To synchronize an effect with other scripts in your game, the InOut component is exposing two delegates:

```
public delegate void MeltCallback (GameObject go);

public MeltCallback StartCallback;
public MeltCallback EndCallback;
```

When set, StartCallback will be executed when the effect starts whereas EndCallback will be executed after the effect is completed. In both cases, the melting gameobject will be used as parameter.

As a quick example, let's say we want to melt an object and add some nice particle effects to it.

```
// when setting up the scene
GameObject myGameObject;

myGameObject.GetComponent<Melter>().StartCallback = TurnOnParticles;
myGameObject.GetComponent<Melter>().EndCallback = TurnOffParticles;

// somewhere else, where you need it
myGameObject.GetComponent<Melter>().active = true;

// somewhere else, where convenient in your code
private void TurnOnParticles(GameObejct go) {
    // code to turn on particles
}

private void TurnOffParticles(GameObject go) {
    //code to turn off particles
}
```

Customizing The Melting

Yes, if you know how to script you can do that.

In the package everything is provided as source code. You will also find comments to help you out.

VERY IMPORTANT: **WE ARE NOT GIVING SUPPORT TO YOUR CODE.**

We are sorry to say this, but our support is limited to bugs in the original product.

If your project is not working because you changed the component or the shader code, please consider reinstalling the package and start over.

Demo Scenes

Bundled in the package you will find two demo scenes: “Melting sphere” and “Melting robot”.

In both scenes, there will be a checkered cube with a model on top (a sphere or the a robot). When the game is started, the camera will orbit around the cube while the model will melt. The rotation script is applied to a pivot object, which is a child of the cube.

Getting Support

If you believe you found a bug, please send an email to *pixeldust@cloverbit.com* stating:

1. A short description of the problem
2. The exact version of unity you are using (including operating system details)
3. A way to replicate the bug

If we cannot replicate the bug, we will ask you for additional information.

Acknowledgements

Models for the demo scenes and in the pictures have been taken from the following sources:

The Robot

Robot By mirov777

Model available online at: http://www.blendswap.com/blends/view/47587

License: CC0 1.0 (http://creativecommons.org/publicdomain/zero/1.0/)

The Plant

Indoor Plant In A Pot by MZiemys

Model available online at: http://www.blendswap.com/blends/view/84195

License: CC0 1.0 (http://creativecommons.org/publicdomain/zero/1.0/)

The Car Model

1957 Chevrolet Bel Air by *spacetug*

Model available online at: http://www.blendswap.com/blends/view/4045

License: CC0 1.0 (http://creativecommons.org/publicdomain/zero/1.0/)

Disclaimer