Dynamic melting effect

Kjartan Valur Kjartansson
Sveinbjörn Berent Birgisson

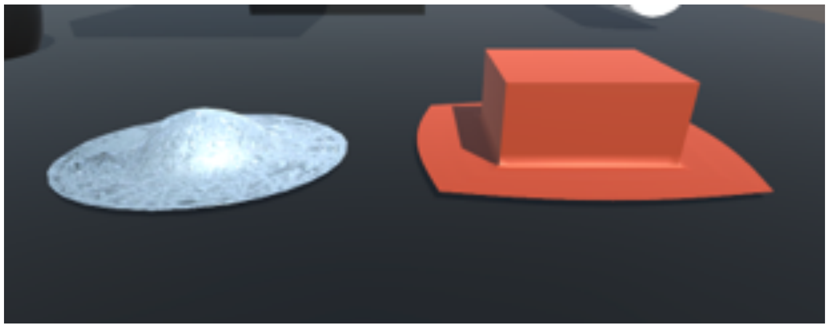

Figure 1: *Melting a sphere and a cube*

Introduction

In this paper we explain our process as we try to find a way to dynamically simulate a melting effect via vertex manipulation using meshes in the game engine 'Unity´. This effect is usually done with an animation and when meshes are involved the simple solutions are unrealistic and cheap and on the other hand the realistic ones are very complex and processor heavy. We present a solution that can make meshes melt by sinking down, spreading out, covering objects and flowing over edges.

1 Motivation

The idea itself came from Kjartan when he was reminiscing about an old game (Metal Gear Solid 2) where you could tip an ice bucket over and the ice had the cool effect of melting over time. Instead of simulating liquid, we wanted to melt down more solid objects like plastic or metals, something that would look a bit more thick and viscous when melted. We looked around the Unity Asset store and, while we did find some mesh deformation packages which could potentially be manipulated for melting, there weren't any that we could find which did exactly what we were aiming for.

2 Similar work

Whether it be due to lack of material online, or just uneffective searching, we were not able to find much to base our work on. Only a few things which were similar. Cloth effects, which are now built into Unity, are somewhat related to our solution in that they involve deforming a mesh around a hard surface.
A package we found on the Unity Asset Store, called Megafiers, could potentially be used for creating a melting effect, but that's not what it's made for. It's more of a general purpose mesh deformation tool, which seems more focused on rubbery deformation.
We were able to find one post on the Unity forums in which as user (username: Nanako) had asked about melting meshes. One user (username: mgear) posted a link to their solution: http://pasteboard.co/1QARh81p.gif
This solution presents an object which reacts to an invisible heat source. The melting effect in this solution is quite different from what we had in mind. The object, rather than sinking and flowing out, just shrinks away from the heat source and changes color to look like molten metal or lava.

3 Method

The melting effect is entirely implemented through vertex manipulation. Firstly, we apply a collider to our object, and the melting starts once collision is detected. At that point, a couple of things happen. The melting object is made kinematic and its colliders are removed. Because the mesh is melted down, but not the collider, we end up with an empty collider sitting in the object's initial position, so it's removal is to prevent any other objects which might be in the scene from bumping into something invisible, causing our melted puddle to start bouncing around. Another thing which happens is that we find the lowest y-position of the lowest vertex in the melting object, to define a "floor" value. Then the melting can begin.
The code iterates through every vertex in the melting object, checking it is within a miniscule distance from the "floor" of the object. If the current vertex is not close to the floor, a down vector is applied to it. This makes the object sink down. If the vertex is close to the floor, a normal vector (with its y-value set to 0) is applied, thus making the object spread itself out.
We were quite happy with the way this looked when simply melting the object on a flat surface. This was where we had to start thinking about making the object melt over the edges of other surfaces. We started by trying to implement this by gathering the coordinates on the edges of the surface the object is melting on, using them to construct bounds, thereby being able to set a condition that if a vertex crosses those bounds, a down vector should be applied to it again. Our first pass at this implementation produced some strange results wherein the mesh only partly dripped over the sides, while it also spread out forming a plus shaped puddle. (https://www.youtube.com/watch?v=YzQfPd_sw1k)
Even if we could've fixed that, the result wouldn't have been that great, since it would be pretty cumbersome to manage, would not

work on curved surfaces, and if there were a larger surface below the first one, the mesh would not react to it.

We scrapped that implementation, and instead realized that using a raycast could be a good solution. We simply have every vertex that has reached the mesh's floor shoot a raycast straight down, a very short distance. If the raycast hits something within that short distance, the mesh spreads out more. If the raycast hits nothing, a down vector is applied again. This produced terrific results, making the mesh not only flow over the edges of the platform it's on, but also making it react correctly to whatever it may hit once it's gone over the edges. (https://www.youtube.com/watch?v=BK3iWXSmyLY) However, now the problem was that the material, once it started going over the edges, clipped through them quite nastily and didn't look as thick as we wanted. We didn't manage to entirely solve the clipping, however we managed to lessen it considerably. Firstly, we had the material elevate itself slightly over the surface it's on. This makes the material seem thicker, as well as making sure that it's not as close to the surface when it goes over the edge, thus slightly reducing clipping. Another thing we did was use raycasts in the reverse normal direction (with y set to 0 again) on each vertex which is flowing down. This is to check if there's a surface close by, if there is one, move the vertex out slightly. This serves to produce a pretty nice ripple effect when the material is flowing, as well as slightly reducing the clipping problem.

The next problem we set out to fix was that of how much the material spreads. Because there's only so many vertices we have in an object, once they reach a certain amount of spread, they inevitably start getting far too stretched from the source and it ends up looking rather ugly. (see **Figure 3**) So we set a public value, which is easy to change, which sets how much the material should be allowed to spread out before it stops. Once the bounds of the material are that far from the center, it will stop spreading. We also used this value to produce an effect which makes the material slow down the more it spreads out, by scaling the spread of the material by the difference between the max spread and the current spread, divided by the max spread. This makes it look considerably more realistic.

Finally, we implemented a small bonus feature. The ice cubes in Metal Gear Solid 2, where this idea came from, melted slower if they were close together. So as a small homage to this project's inspiration, we added a similar feature. We set a proximity modifier which the melting speed is scaled by. The default value for it is 1. But if there are many melting objects in the scene, and they're tagged as "melter", the proximity modifer will be equal to 1 minus the distance to every other melter in the scene divided by 100. This produces a very slight difference in the melt speed.

4 Future work

Our method is a step in the right direction, but could do with some improvements. Some fixes to our current project would be the starting point. Adding some features like more of a random puddle since it melts almost in the same shape as the object. Starting the melting process when near an object marked as the heat source would be an excellent feature. Instead of just flowing over objects, letting the liquid fill up a closed up area like a mould would be a desired function. To get a better performance it might be desirable to remove vertices when the surface is flat but increase them as they gets close to an uneven surface to remove unwanted clipping through edges.

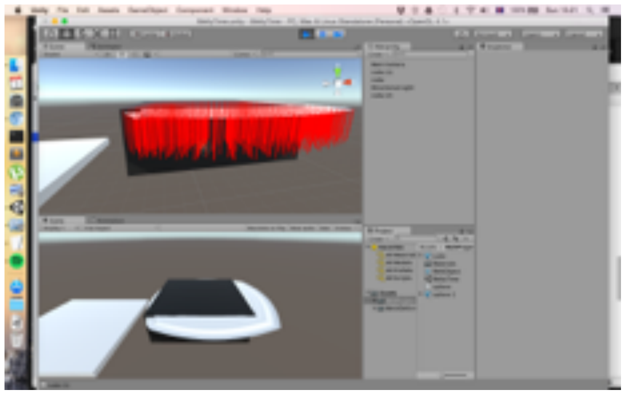

Figure 2: *Shooting raycasts directly down to detect a surface*

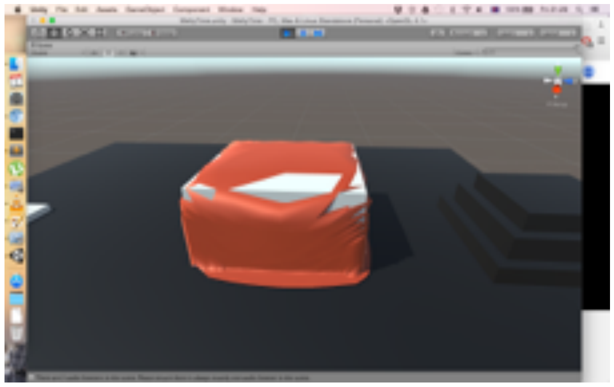

Figure 3: *Material clipping after spreading out too much*

5 Videos

Here are links to a few videos of the finished project:
https://www.youtube.com/watch?v=-qAF60CK5R0&feature=youtu.be

https://www.youtube.com/watch?v=n1uXliTT2pY&feature=youtu.be

https://www.youtube.com/watch?v=gLN_lucLFHw&feature=youtu.be

https://www.youtube.com/watch?v=jcYbRXZtFvA&feature=youtu.be