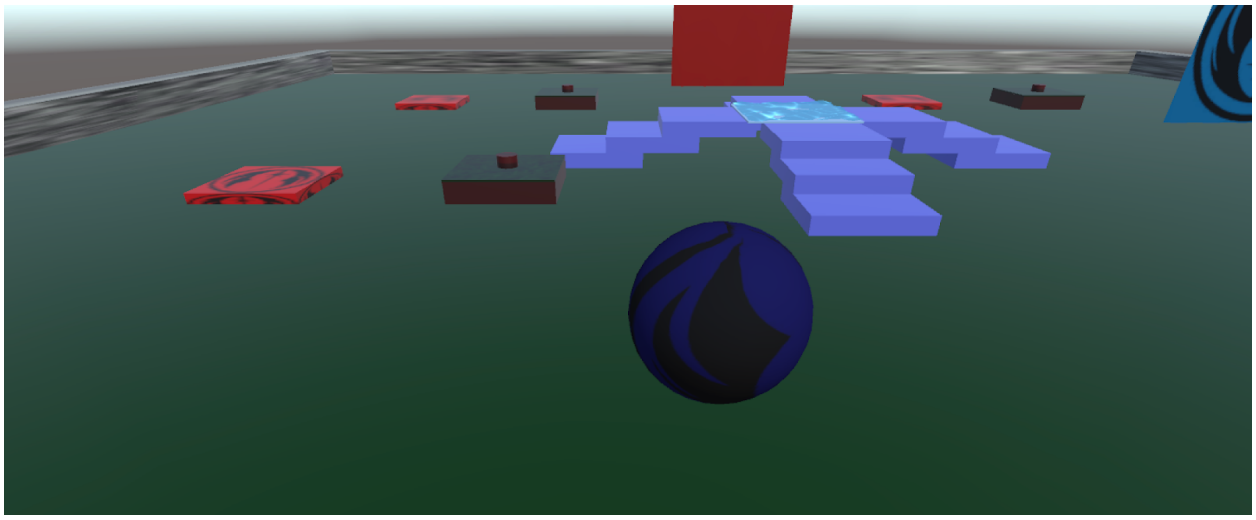
CS175 Final Project: Rolling Ball

Nina Chen and Moni Radev

Introduction

For our final project, we decided to explore Unity. We first worked through a couple of tutorials to get the basics down, and then went off on our own.

When the user first enters our world, they will see the following:



Roughly speaking, our project focused on implementing several different interesting features, while unifying them in a coherent manner. The entire project is implemented on a single plane, with the movement accomplished by a rolling ball. There are several orange platforms on the ground. When the ball rolls over one of them, the corresponding piece of shrubbery and tree stump will light on fire. The platform itself will also slightly sink into the ground when the ball is rolled onto it, so as to simulate a "clicking" functionality. Additionally, if the ball rolls over the fire, then it itself will also ignite with its own flame. However, it won't cause surrounding objects to catch on fire. Towards the center of the plane there is a small piece of water that will "douse" the ball, so that when the ball rolls on top of the water the fire will go out (if it was lit in the first place). Lastly, we also added some cloth to our plane. We provide two pieces of cloth hanging from above, with which the ball can fully interact. In our world, cloth is fire-resistant, because it is guarded by the symbol of the Jedi Order (the Force allows the symbol's protection to transfer onto all pieces of cloth), hence why the material doesn't catch on fire.

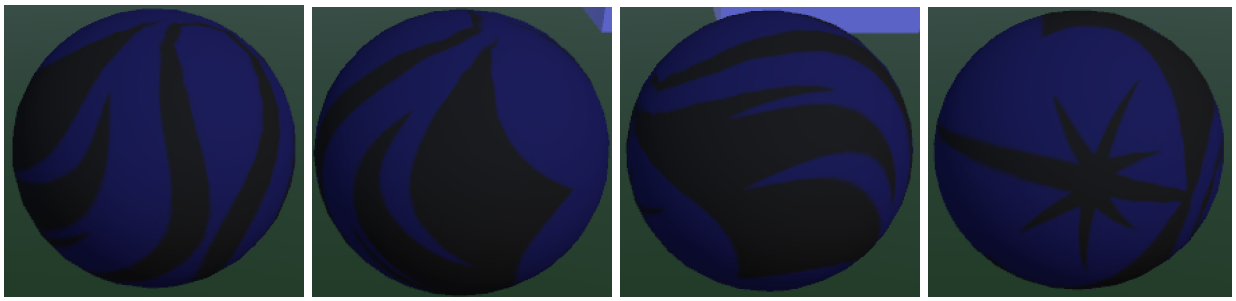
Overall, our main goal with this project was to learn the basics of Unity, since neither of us have used Unity before, and to see more broadly the relevance of what we studied this semester. In this project, we had to consider the impact of different resistance forces, like drag and friction, along with vector-component manipulation. We also had the chance to work with Shaders (for our water implementation, and several less successful spin-offs not displayed). Additionally, we considered the physics behind particle simulation for our implementation of fire, and worked with collision forces to implement fire ignition, amongst other features.. Finally, we also worked with meshes to implement cloth.

Ball Rolling

To implement a rolling ball, we attached a RigidBody component to a simple sphere. We assigned it a mass of 1 and a drag coefficient of 2. We did this because we made our velocity force quite high, and so we wanted some resistance. Following this point, we also assigned our object some friction. Specifically, the coefficient of dynamic friction is 0.8 and for static friction is 1. We selected these high values because we wanted the ball to very quickly stop moving as soon as the user stops pressing a key that contributes some force to the ball.Additionally the higher friction coefficients is what generally gives a better rolling motion, otherwise it will be as if the ball was moving on ice, in which case we would be merely translating our ball across an axis rather than rolling it on a surface.

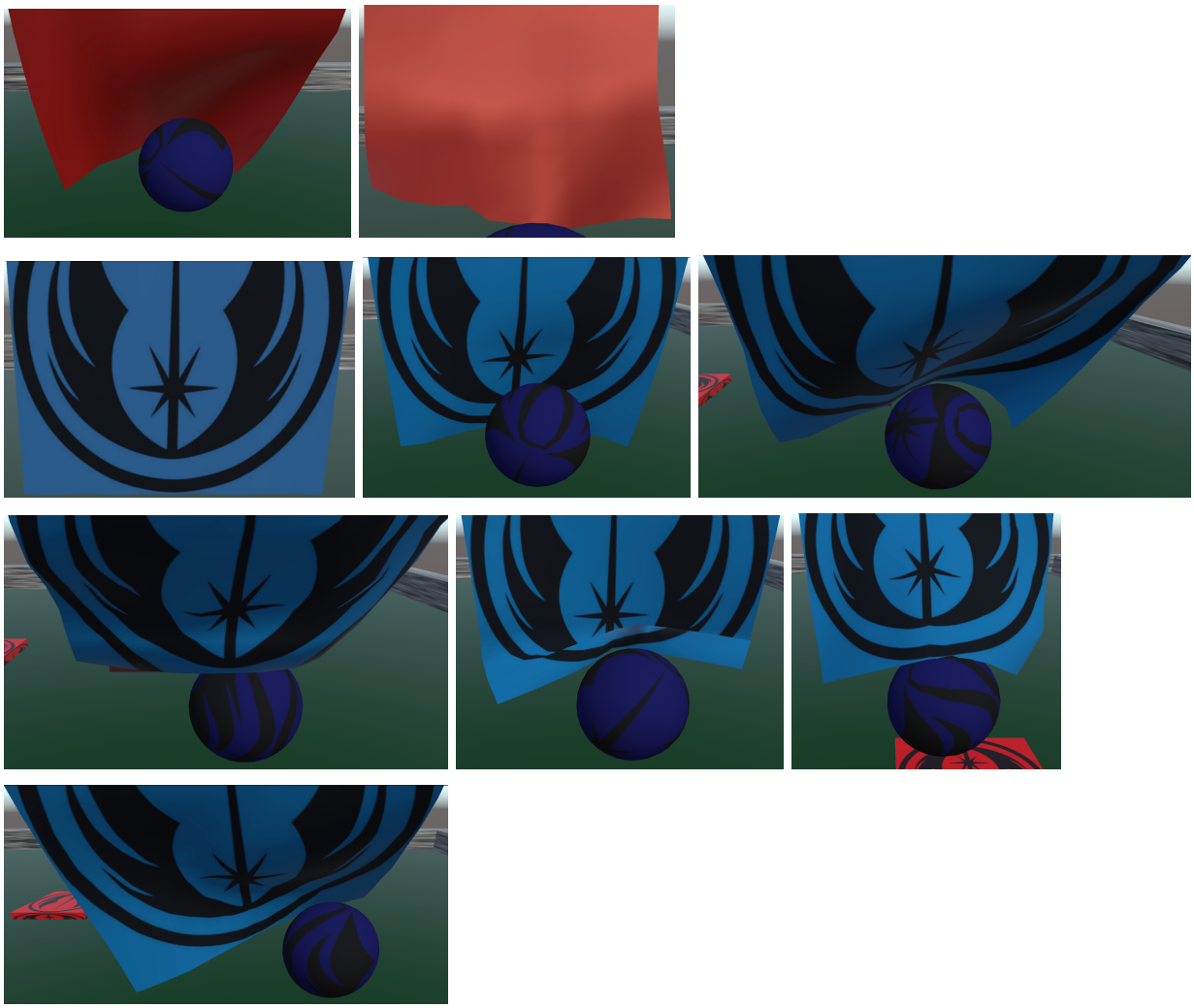
In order to make the camera follow the ball, we wrote a script to manually change the position of the camera depending on the ball's location. We hard-code some predetermined good camera offset and then once per frame re-position the main camera according to this offset.

Finally, to actually implement the movement, we add a force component. On every Physic update, we get the new horizontal (left-and-right arrows) and vertical momentum (up-and-down arrow), and make a new Vector3 with these values. We then add this Vector3 as a force to the rigidbody component of the ball, thus creating movement.



Cloth

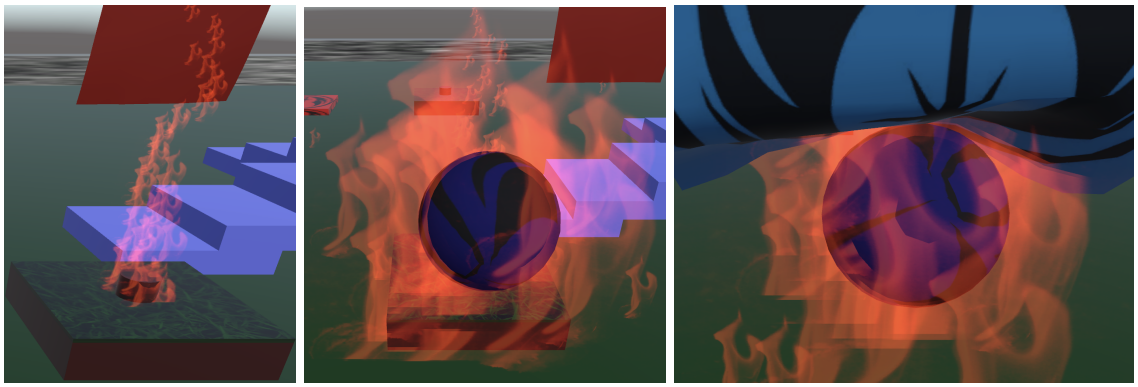
Our cloth is implemented on a Skinned Mesh Renderer. We first had to add constraints to the top of the cloth to have it “suspended”. We did not implement falling cloth. What gave it actual cloth-like functionality, however, was by assigning our ball as a sphere collider to our cloth. This was a more trivial implementation in terms of sheer number of steps, but was very interesting to learn and implement nonetheless.



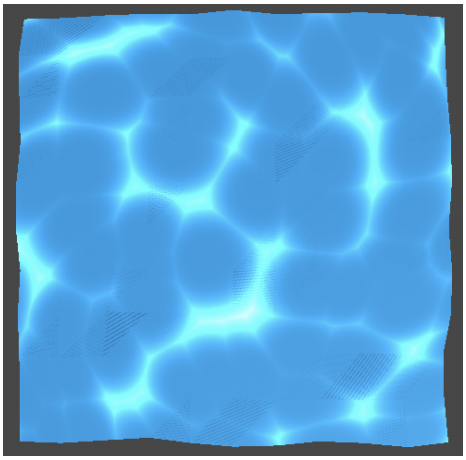
Fire

To make fire, we first started with a particle system. We assigned some shape to this particle system (a sphere for the ball, and cone for the others). We assigned the system a start lifetime of 10 and start speed of 2. The interesting part is perhaps the particle system curve. This is what

allows the top of the flame to taper off and make the start of the flame look denser. As mentioned, we made a separate particle system design for the sphere, which is why the flame emission looks different.



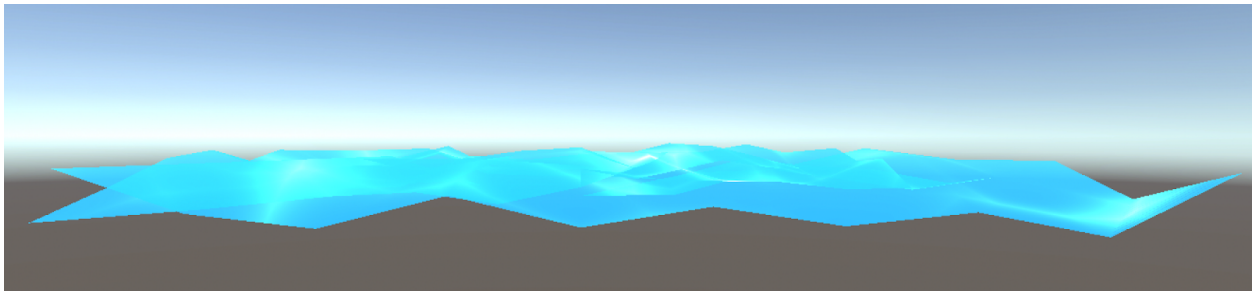
Water



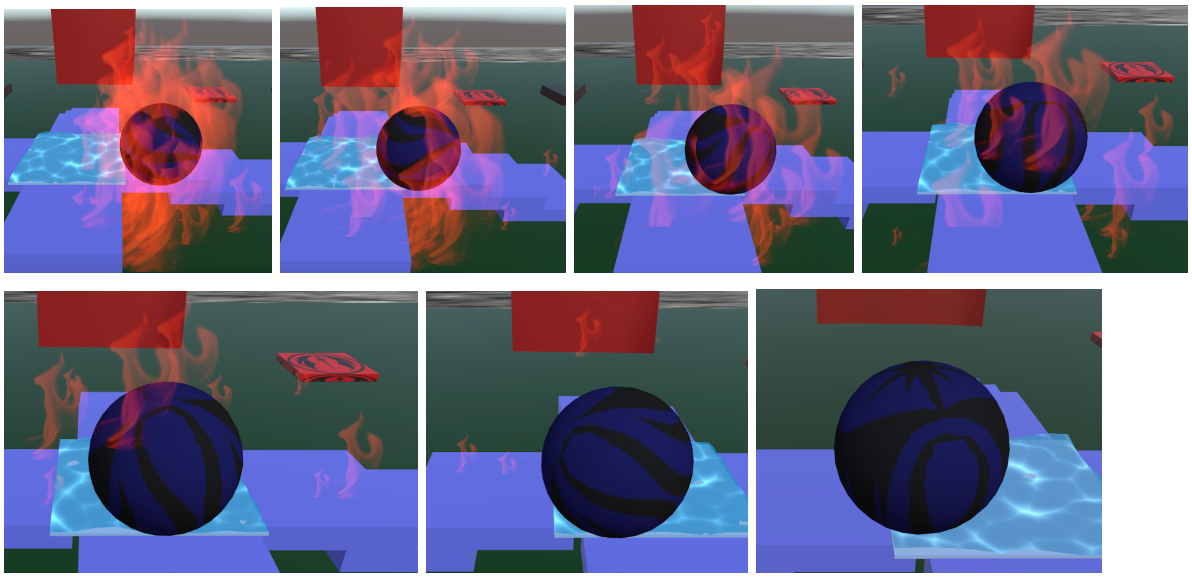
Next, we also created water in Unity. To create water, we first created a new shader graph named WaterShader for the water material. In our WaterShader graph, we first gave our water a bright blue base color. To add the ripples, we added a voronoi node. The voronoi node creates cells that, when distorted, will look like ripples in the water. By changing the angle offset in the voronoi node over time, we are able to create nice ripple effects by making the cells move around. Next, we attached the output of the voronoi node to a power node to make the motion of the cells more smooth. We also added a more intense, brighter blue ripple color to contrast with our base color. We also decreased our alpha to 0.8 to make our water a bit see through. We then added a radial shear node to further distort the water effect to make the movement of the ripples less uniform. We add the radial shear node as an input to our voronoi node to distort our voronoi by rotating the cells around a center point.

To make the plane less flat and more wavy, we added vertex displacement to our WaterShader. We first added a gradient noise node. We want the noise to move over time instead of being static. To do so, we need a timing and offset noise to change the gradient noise. The timing and offset noise changes its offset over time through the attachment of a time node. Next, we took the gradient noise and multiplied it with the normal vector of our object (the plane) to distort the

plane on the y-axis. We add this noise on top of our current position, which is also in object space. Finally, we update our position with the result of this addition to create the wave-like motion of water.



In our scene, we used water as a mechanism to put out the fire.



Conclusion

This project was an experiment in what we can do with the basic features of Unity. We learned quite a bit about the framework as a whole, even about features we did not end up implementing. We now feel comfortable with implementing some basic games in Unity, and with working with different 3D objects and connecting them with scripts. We became relatively adept at interacting with all of our objects through scripts, and learned quite a bit about detecting collisions with functions such as OnCollisionEnter() and OnTriggerEnter(). We also learned some C#. We learned about making prefabs and about assigning objects to layers, so that we can distinguish

between collisions with different objects. Lastly, we also gained experience with Shaders, in order to make the water.

The features that took less time to implement were cloth, fire, and the rolling ball. Granted, there was a lot of fine-tuning to be done, but we were surprised by how quick it was to get some of the basic functionality in a rolling ball, for example, up and running. Making simple fire using a particle system was also fairly straightforward once we tried different methods the first few times. We made cloth later on in the project, so it is also quite possible that we just had more experience with Unity by the time it took to make some of the features. Oddly, some of the hardest things we did were designing the actual 3D objects. It was surprisingly difficult to get the sizes we wanted, and the exact shapes. Since child objects are sized relative to their parent, and since we started out without any parent-child relationships, there was a lot of re-configuring when we decided it would be wiser to make our fire block prefabs, for example. The script also took quite a bit of time to develop, but that was a bit more of a programming problem that we knew how to solve, rather than something completely new.

One thing that could have been further improved was the water. Though the basic functionality is implemented, it can be made to look much more realistic, and it would be nice to have the ball float in the water, not just roll on top of it. Now, the physics of it make it feel like solid ground. In general, however, we found working with shaders somewhat difficult. In addition to the water featured in our project, we tried extensively to implement more sophisticated water by doing a lot more work in the shaders. We tried changing the shades of the waves, and making the water look just more realistic in general by adding a Normal mesh. We also tried to give our water some foam. We were not able to get the shades quite right though, and ultimately decided to stick with the simpler version. However our trials and tribulations with water taught us a lot about working with shaders.

One other thing we tried which did not work out was using Blender. We initially wanted to add some objects into our Unity world by creating them with Blender, but could not get the shapes we wanted even after several tutorials. We were able to make some basic things like a pyramid, etc., but did not have time to learn enough in order to make some more sophisticated objects. However we remain quite intrigued about using Blender to design some really neat objects, like a Star Destroyer (or two).

Future Works

Some features which could nicely augment our project would include making some sort of elevator functionality. We at first wanted to make our plane at a higher level than ground, where the ground layer would have been an entire ocean. Then, we would use some sort of elevator to

move and up down to carry water to douse the fire. Other features to augment our project would include improving our water. We mentioned this above, but we tried making some more realistic water by doing more complex shader work. While we did not have time to see that to fruition, it would be a nice addition. Additionally, instead of a rolling ball it would be nice to have an avatar. Otherwise we could have made the ball have more realistic rolling functionality. Right now it does not always behave very intuitively, so that could be improved upon. Lastly, we could have added some nice custom-made objects using Blender, as mentioned above.

Resources

- https://youtu.be/pwZpJzpE2lQ
- https://www.youtube.com/watch?v=Vg0L9aCRWPE&ab_channel=Brackeys
- https://youtu.be/hZ8DgasVT_M
- https://youtu.be/CRX05E14XPg
- https://youtu.be/qShjsxopbfQ
- https://youtu.be/uuzB93F39P8