Michaela Kane & Jason Stein
Computer Science 175: Computer Graphics
Professor Steven Gortler

Final Project Write-Up

For this project, we decided we wanted to learn more about working and designing in Unity, given its popularity and its wide use in the computer graphics world. We began by watching and learning from both the Roll-a-Ball and Space Shooter tutorials, before we turned to our own project: fire simulation and ragdoll character.

Using the ground and walls we learned to make in the tutorials, we started with a simple scene the user could explore without falling :

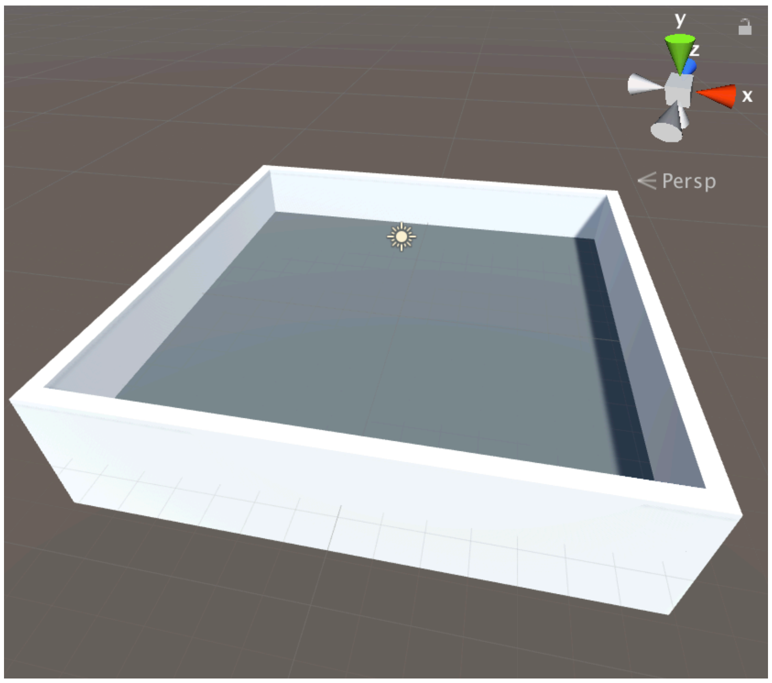
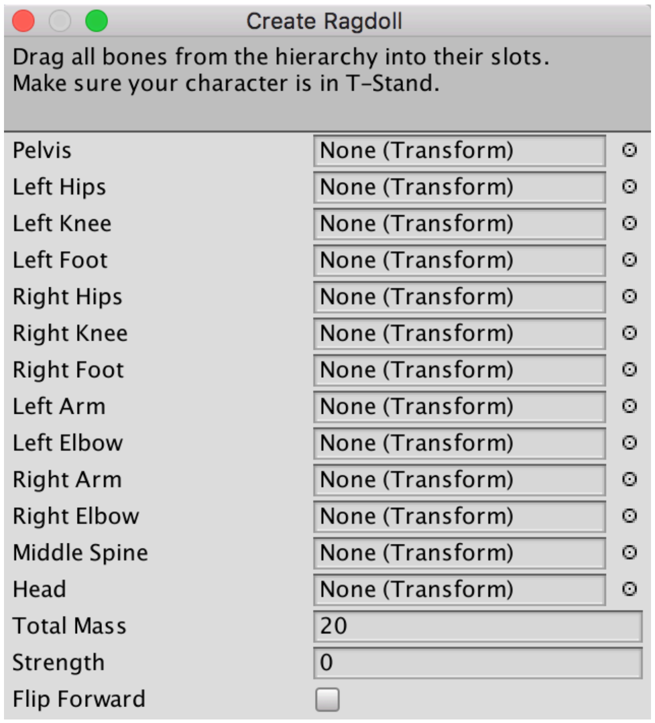


Figure 1 Empty Arena for the Scene

We then used the free, built-in assets from the Supercyan Character Pack Free Sample to create a moveable player character. Although the pack had the underlying "robot" parts set up, we configured each joint using the unity ragdoll GameObject.


Create Ragdoll

Drag all bones from the hierarchy into their slots.
Make sure your character is in T-Stand.

Pelvis	None (Transform)
Left Hips	None (Transform)
Left Knee	None (Transform)
Left Foot	None (Transform)
Right Hips	None (Transform)
Right Knee	None (Transform)
Right Foot	None (Transform)
Left Arm	None (Transform)
Left Elbow	None (Transform)
Right Arm	None (Transform)
Right Elbow	None (Transform)
Middle Spine	None (Transform)
Head	None (Transform)
Total Mass	20
Strength	0
Flip Forward	☐

Figure 2 Unity's Ragdoll Creation Template

After adding the appropriate joint transforms to these fields, the provided textures and character animation script, we had a fully fleshed out moveable unit. We removed some of the excess functions that gave the character extra motions (such as picking up objects), but kept the built-in animation and movement script such that the player GameObject can move around with either the arrow keys or the WASD combination, and can jump when the spacebar is hit. Much like what we studied in class, the character is made up of joints and child joints that help dictate its movement. Finally, we made the main camera a child of the player, such that it follows the character around as it moves.

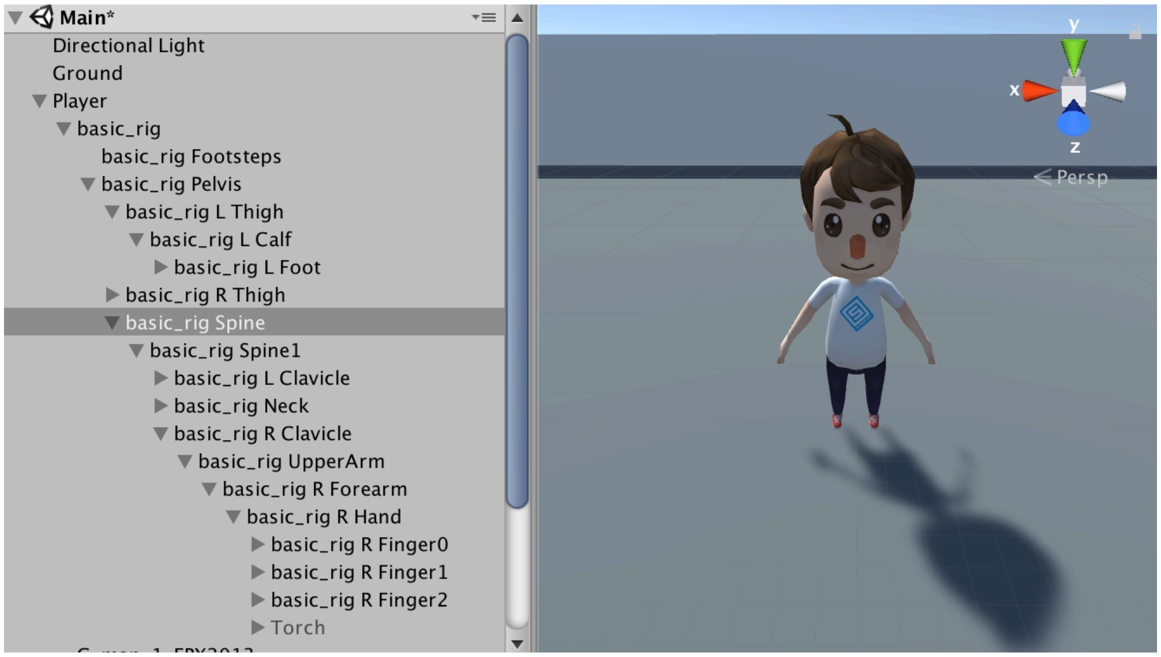


Figure 3 Completed Player GameObject with Child Joints

Once we created our moveable character, we moved on to flame simulation. We used the Unity Particle Pack as a source of inspiration, and decided to create a series of flame simulations with child light components. We began by creating a torch for our character, made up of a cylinder stick GameObject and built in as a child of the right hand, such that it appears that our character is holding it. We then added a small particle system GameObject to the end of the stick to create the flame itself. The particle system included a great deal of built-in functionality, so all that was left to do was to edit a few of its fields. First, we prewarmed the system such that it would already be actively emitting particles once the game started. We also cut Start Speed down to 0 to keep the particles in a contained area, and increased particle size – each particle can be thought of as a wisp of fire in the simulation. We altered the Emission parameter to decrease the rate of particle emission, and changed the emission shape from a cone to a circle, thus keeping the flame particles confined to a specific area. We changed Velocity over

Lifetime to increase gradually on the Y-axis, while the Size over Lifetime decreases exponentially. This gives the image of the flames flickering out as they ascend further from the source. Next, we colored the flames with a gradient such that they start at an orange color, before turning red and then fading away. We also activated the Noise module in order to give the flames a greater sense of randomness in their generation. Finally, in order to change the particle shape from circular motes, we used the built in billboard of FlameRoundParticleSheet provided in the asset package, such that each particle would take on the shape of one of those flame images.

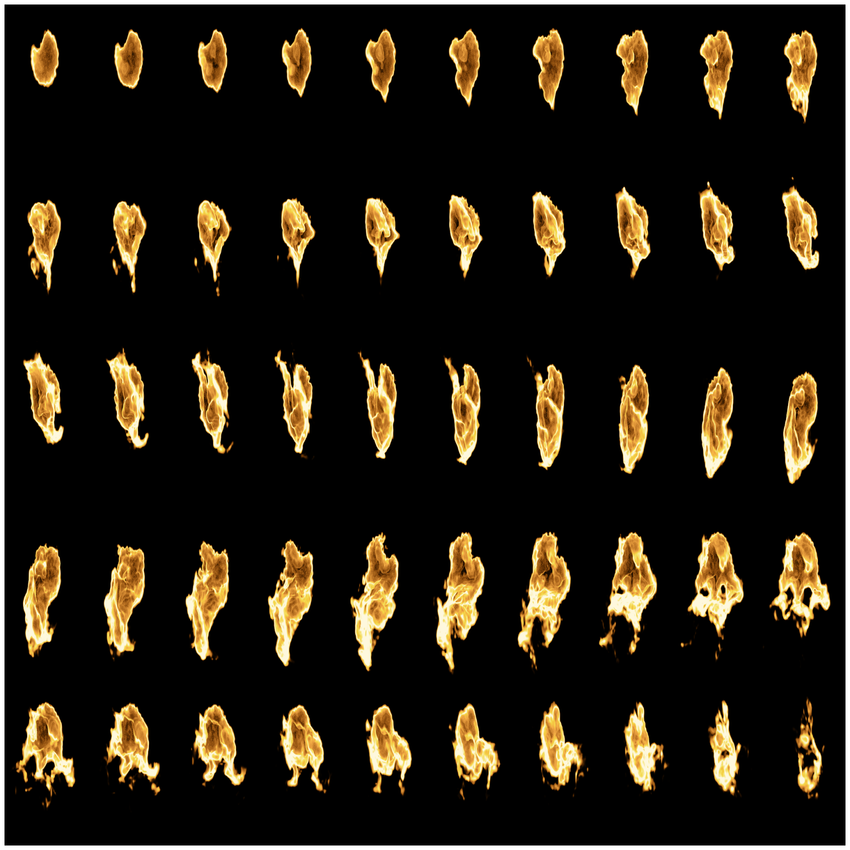
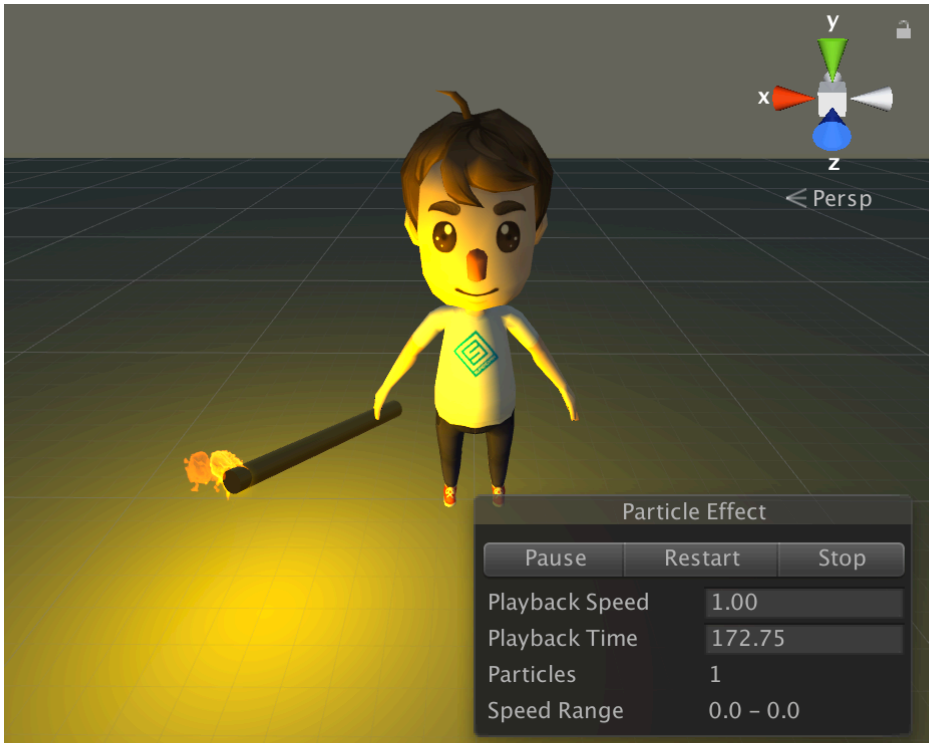


Figure 4 FlameRoundParticleSheet Provided by the Unity Particle Pack

Finally, we created another Particle System as a child of the Flames system to generate a pulsating light. We originally tried to simply activate and use the Lights module of the flames particle system, but the volatility caused by the rapidly generating particles created an unnatural flickering effect, so we instead used a single, static particle that stayed in the middle of the flame, and used the ParticlesLight prefab, again provided by the Unity Package, to create a more steady, pulsing light that would illuminate the scene.



Playback Speed 1.00
Playback Time 172.75
Particles 1
Speed Range 0.0 – 0.0

Figure 5 Player Character Holding Torch

Once we created a player with a fire stick, the natural next step was to give him something to light on fire. So we created a set of four fire pit objects, which consisted of stacked cylinder “logs,” and a paused particle system. We created a sphere collider trigger for the flames on our torch, as well as a sphere collider trigger for each of these fire pits, and, using C#’s OnTriggerEnter function, we made it such that once the fire of the torch comes into contact with the collider on the fire pits, the “wood” ignites and the fire pit flame particle system starts.

As a final touch to explore more of the components of the particle systems, we made each of these fire pits with different colored logs generate flames and lights of different colors. This only involved duplicating the original fire pit and altering the Color components of both the light and flame particle systems. The result is a scene where the player can light any combination of fire pits, and observe the light patterns it creates both on the player character, and on the highly reflective walls. Finally, in order to highlight the colored lights and effects more, we deactivated the directional light GameObject, such that most of the illumination comes from the flames.

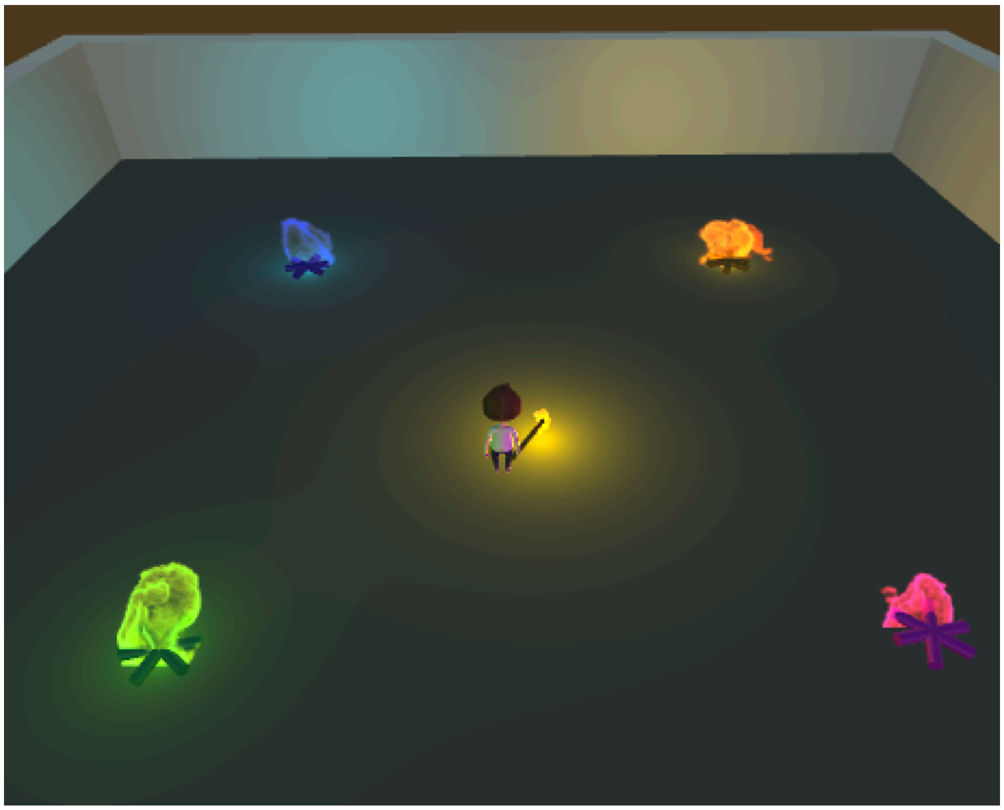


Figure 6 Final Scene with All Fire Pits Lit

From this project, we learned the basics of programming in unity; specifically, C# scripting, ragdoll functionality, and particle systems. We also learned about the huge number of resources available on the unity site to both further our knowledge of programming in unity, as well as how to best use and alter packages available in the asset store.

Resources

Unity Tutorials: https://unity3d.com/learn/tutorials

Textured Character Pack: https://www.assetstore.unity3d.com/en/#!/content/79870 (for character animations and textures)

Unity Particle Pack: https://www.assetstore.unity3d.com/en/#!/content/73777 (for particle textures and reference)