Mystical Grove

HLSL Shader scene using Unity HDRP

Aaron Kanehl
Computation Media
University of California, Santa Cruz
Santa Cruz, California
akanehl@ucsc.edu

Kaio Barbosa
Computation Media
University of California, Santa Cruz
Santa Cruz, California
ksbarbos@ucsc.edu

Wai Chun Leung
Computation Media
University of California, Santa Cruz
Santa Cruz, California
wleung11@ucsc.edu

Evita Lobo
Computation Media
University of California, Santa Cruz
Santa Cruz, California
elobo@ucsc.edu

Wesley Smith
Computation Media
University of California, Santa Cruz
Santa Cruz, California
wetsmith@ucsc.edu

ABSTRACT

This paper describes the process of creating the project **Mystical Grove** as part of the final project for course Game Graphics and Real-Time Rendering offered at the University of California, Santa Cruz. This document will showcase the shader and rendering techniques behind creating a realistic and believable 3D scene in Unity using the high-definition render pipeline (HDRP).

INTRODUCTION

In this shader project, we wanted to create a surreal forest scene complete with moving trees, swaying foliage, clouds, ponds, and various elemental crystals. The shader techniques that are applied to the scene include vertex displacement, volumetric lighting, particle systems, bloom lighting, flowing water/liquid systems, cloud simulation, and other lighting models. We use these techniques to create a mystical grove setting conceptualized in *Figure 1*. We will describe the process of making four different elemental crystals, clouds, trees and foliage, water, and miscellaneous particles such as fireflies and fallings leaves found within the scene.

Figure 1: **Visual concept of the Mystical Grove setting**

1 Elemental Crystals

The Mystical Grove scene includes crystals with four different types of elements: shadow, electric, fire and water. Each of the four crystals applies one or more shader techniques ranging from glowing effects to animated particle systems.

1.1 Shadow Crystal - Wesley

The objective of designing the shadow crystal was to design an ominous shader set. A combination of particles and shaders are applied to the crystal asset to create the effect.

The primary shader used is a dissolving shader, which creates a “melting” visual on the surface of the crystal. The prototype (not shown in project) adjusted the alpha values based on the blackness of a scrolling noise texture. The final rendition is slightly more stylized for the context. It employs a more regional dissolve - for all values beneath an oscillating threshold, the transparency is full. A “straddle” border is also calculated, and then used to shade the off-gray outline that follows the dissolving regions.

The inky miasma that swirls around the crystal is a combination of a particle system and a wind zone. Particles that flare and vanish as their lifetime progresses are blown slightly by a wind zone. Their orbital trajectory is also modified by noise functions within the particle system. Finally, a similar dissolve shader is applied to the trails, making the swirl and the crystal vanish and emerge in synchronicity.

1.2 Electrical Crystal - Evita Lobo & Aaron Kanehl

The Electrical Crystal showcases a shimmering crystal that oscillates between two colors- red and pink using the same shader as the fire crystal that oscillates between red being the albedo and the pinkish blue color being the final color. I used another shader that uses the same cracked texture as the fire crystal and dissolves near the edges of the crystal to create an effect that looks like it is wrapping around the crystal. In addition to that, I added a lightning particle system that uses noise to displace the trail rendered onto the particle system. the size of the particle gets smaller with time to reflect that of realistic lightning and has varying lengths of its trail.

1.3 Fire Crystal - Aaron Kanehl

1.1.1 A Crystal set ablaze. For the Fire Crystal, I wanted to create a shader that would emulate lava and cracked obsidian. I used a shader that started with a deep red albedo, modified through the editor, and then added a cracked texture over the top. Using nodes that control the time, I added a sine time graph to the texture, making it fade in and out. I limited the outputs of sine time in order to avoid it phasing out completely.

1.4 Water Crystal

1.1.1.1 A Crystal of waning sea. For the water crystal, I took a simple caustics texture, and overlayed it over a crystal that has a deep sea blue albedo. I changed the color of this texture, and then used a tiling and offest node (I learned how to do this during homework 3), and used it to make the texture move across the surface as an element of time. This made the crystal look alive within itself, rather than using the global camera to make its elements appear.

2 Clouds - Evita Lobo

The clouds used a transparent shader that used Perlin noise and displaced the vertices based on a time function. The vertices shift over time to create a globular 3D object that moves similar to a gaseous mass. In addition to applying the Perlin noise sphere towards the top of the scene, there are also particle systems that are layered to give the clouds more depth. The particle system renders each particle as a sphere mesh and uses the same cloud material that the noise function uses. The shape is all concentrated as a hemisphere and the of each sphere also grows smaller with time. The color of the cloud is brightened or darkened according to the directional lights *y* direction. This is done so that when it is nighttime (directional light’s *y* direction equals one) the cloud becomes dark and is no longer visible.

3 Trees, Foliage, and Wind - Wai Chun

3.1 Foliage and Grass. We wanted to create a forest scene that feels refreshing. In order to achieve that. foliage was an important aspect in conveying a believable setting. The first thing I did was create a grass shader that would be animated as if a breeze of wind is passing over the grove. To produce this effect, I created a vertex displacement on a grass mesh based on a linear interpolation of a simple scrolling noise function multiplied by Unity’s time. I applied this result to the position of the mesh and made sure that the base of the grass stays put while the top of the grass moves to the wind by using a gradient. Adjustable properties are then added that allow changes in wind density and strength.

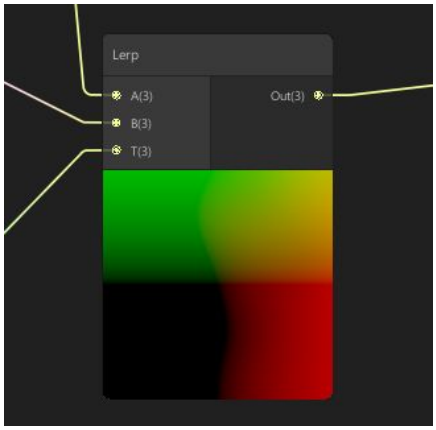

Figure 2: **Linear Interpolation between the position (A), the time(B) and the UV of the mesh (T)**

3.2 Trees and Leaves. For the trees in the grove, we wanted to use low-polygonal trees to express the mystical and fantasy side of the scene. Similar to the grass foliage sway, I applied a vertex displacement with time and a scrolling noise function to simulate windy trees. The gradient was adjusted in order to stop the swaying of the tree trunk, and the wind strength property was lowered. Since the leaves of the tree are polygons, I applied the shader on each of the triangles to display individual leaf sway, which adds to the unique movement in the scene.

4 Water and Terrain

4.1 Water - Kaio Barbosa

Water was created using Unity's HDRP stack. The render type of the material is Transparent, which makes refraction material properties available. The alpha value of the main material color was tuned to the point where the bottom of the river could be seen under the surface of the water. A planar reflection probe component was added to a plane to create the realtime reflections. The reflection probe creates a cubemap from the location of the the plane it is applied to, and captures reflections of all objects in the scene plus the skybox. This is done every frame, and this is why we were capable of seeing the reflection of fireflies as the flit over the water surface and other dynamic light changes.

To create water ondulations, a noise texture was converted into a normal map, and that normal map was applied at a low intensity to show calm but variable water. To make the ondulations move, the normal texture was offset by time and a speed factor in a C# script that accesses the HDRPs Lit shader "_BaseColorMap" texture properties. This texture name is unique to shader, so always make sure to look at the documentation of the shader you use.

4.2 Terrain - Kaio

The terrain was originally made in Blender and was a solid 3D model. A script that converted meshes into Unity Terrain made the terrain easier to work with. Free environment textures were used to hand paint the areas with grass, dirt, and cobblestone on the map.

5 Miscellaneous Particles

Miscellaneous particles were added to the scene in order to add movement and heighten the mystical feeling of the grove.

5.1 Fireflies - Wesley

This component is relatively straightforward. Small local wind zones "cluster" the fireflies, which fly along organic, noisy trajectories. They primarily showcase the beautiful lighting of the high definition render pipeline. The soft, magical glow is the result of the HDRP's bloom and lighting mechanics.

5.2 Falling Leaves - Wai Chun

Using Unity's built in particle system, I created simple falling leaves by adjusting the shape to take on a down-facing cone. The arc and radius of emission is widened to simulate a large area of falling leaves around a tree. A gradual noise function is then applied to the particles to give off the illusion that the leaves are lightweight and affected by wind. To complete the full effect, the leaves are set to disappear on collision with the terrain.

Figure 3: **Final Result, nighttime**