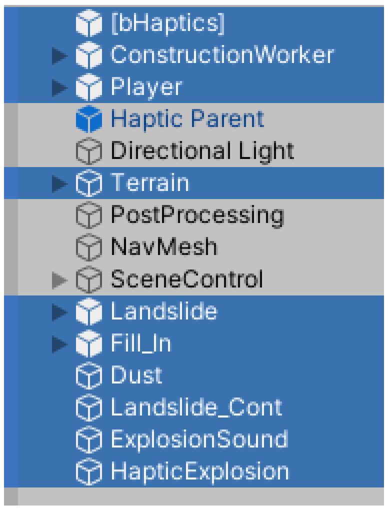
Landslide Example Scene

Description:

The example landslide scene is meant to demonstrate several key facets of the Unity development environment and to give an example of how a landslide event could be simulated. This document will depict those facets and give guidance on how to setup similar events.

Scene Hierarchy:

The scene was built using the remnants of the construction safety simulator asset demonstration scene. It incorporates terrain assets used for the ditch, a player character for viewing the scene in VR, a construction worker that used as the target of the event, and objects used to simulate the landslide such as 3D assets, animations, sound sources, and haptic feedback sources. Those elements are highlighted below:

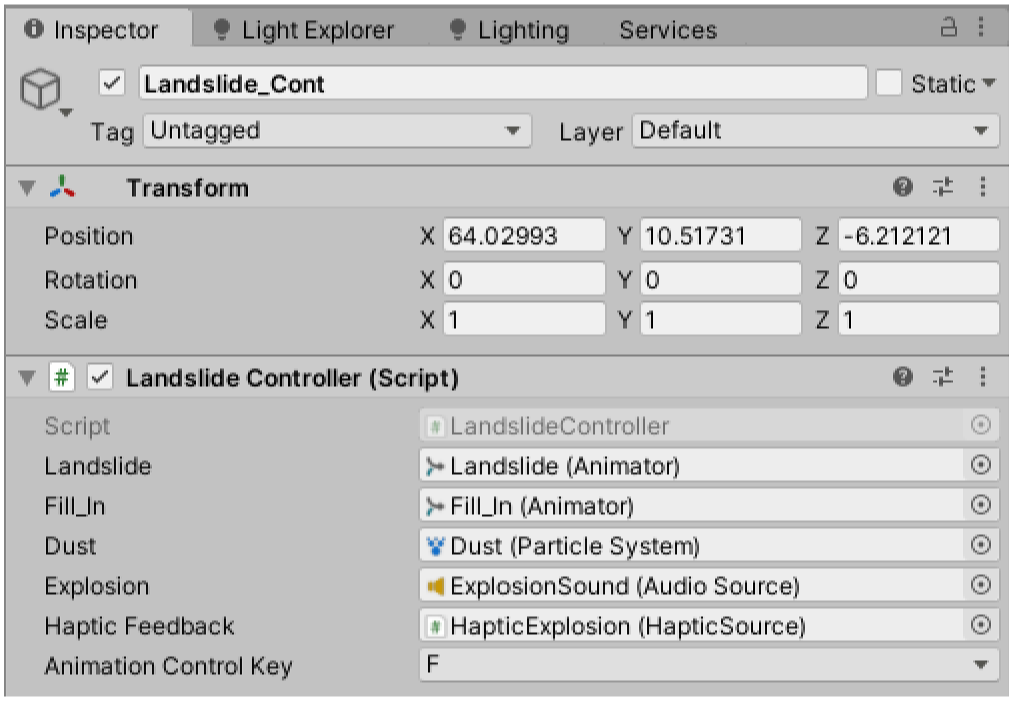


```
[bHaptics]
ConstructionWorker
Player
Haptic Parent
Directional Light
Terrain
PostProcessing
NavMesh
SceneControl
Landslide
Fill_In
Dust
Landslide_Cont
ExplosionSound
HapticExplosion
```

We will walk through the functionality of each of these assets in this document. We will start with the landslide control script.

Landslide_Cont and Script:

This is a single asset in the scene that is meant as the main control mechanism for triggering the events of the landslide. It has a single component attached to it which is the control script. The control script makes public its references to the other assets in the scene that it was designed to control: the side-wall landslide objects, the fill-in objects used for the bottom of the ditch, the dust particle system, the explosion sound, and the haptic explosion feedback. Each of these elements contains some form of animation, audio, or haptic feedback controller. The script also makes available the selection of a keyboard key to initiate the landslide event. Below is a picture of the inspector view for the Landslide_Cont object:



Landslide_Cont

Tag Untagged

Layer Default

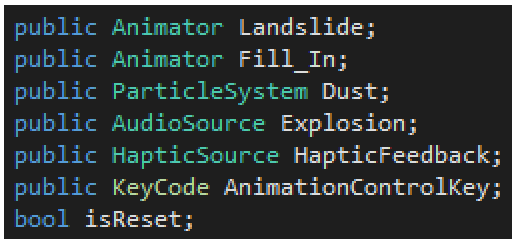
Transform

Position	X 64.02993	Y 10.51731	Z -6.212121
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Landslide Controller (Script)

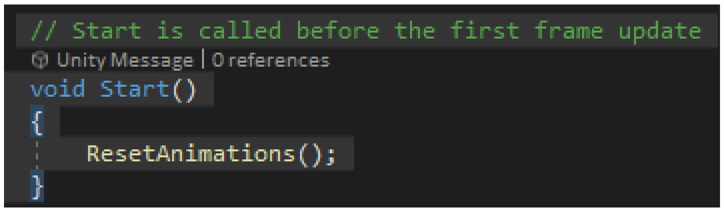
Script	LandslideController
Landslide	Landslide (Animator)
Fill_In	Fill_In (Animator)
Dust	Dust (Particle System)
Explosion	ExplosionSound (Audio Source)
Haptic Feedback	HapticExplosion (HapticSource)
Animation Control Key	F

The script itself is simple. Here are its internal parameters. They correspond with the above references, with the exception of the isReset variable used to track if the landslide event is in progress



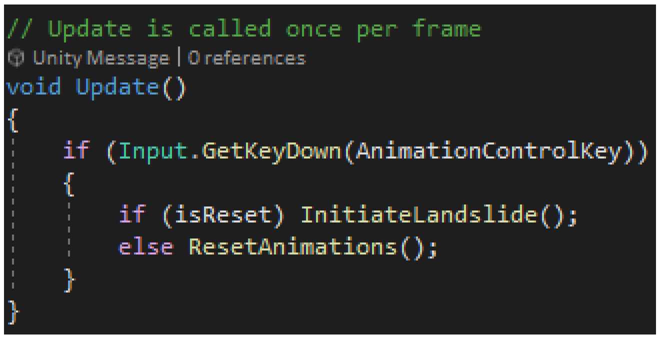
```
public Animator Landslide;
public Animator Fill_In;
public ParticleSystem Dust;
public AudioSource Explosion;
public HapticSource HapticFeedback;
public KeyCode AnimationControlKey;
bool isReset;
```

On the start of the simulation, the script runs the ResetAnimations method that initializes all the animations systems to their beginning state.



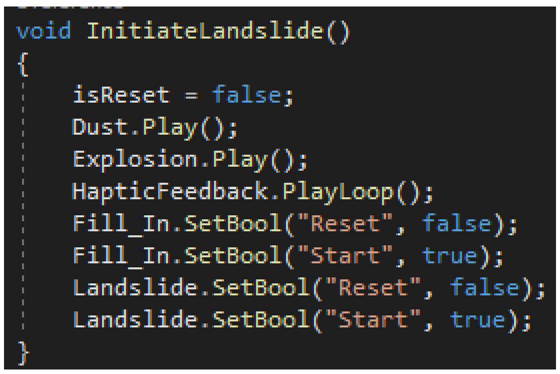
```
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    ResetAnimations();
}
```

In the Update function that is called each frame, the code checks to see if the animation control key has been pressed. If it has it can take two different paths: either it will trigger the landslide event if it is not in progress, or it will stop it if it is in progress. This is only triggered on the key down event:



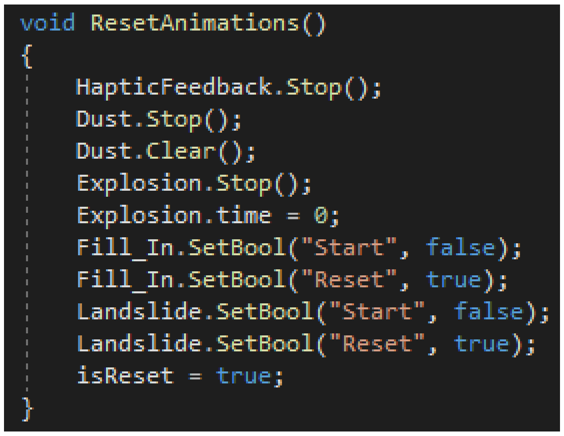
```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    if (Input.GetKeyDown(AnimationControlKey))
    {
        if (isReset) InitiateLandslide();
        else ResetAnimations();
    }
}
```

The InitiateLandslide method will set the Boolean value isReset to false to indicate that the landslide event is in progress. It will also start all of the animations sequences for the landslide.



```
void InitiateLandslide()
{
    isReset = false;
    Dust.Play();
    Explosion.Play();
    HapticFeedback.PlayLoop();
    Fill_In.SetBool("Reset", false);
    Fill_In.SetBool("Start", true);
    Landslide.SetBool("Reset", false);
    Landslide.SetBool("Start", true);
}
```

The opposite of this method is the ResetAnimations method. It sets isReset to true, and stops and resets all of the animation sequences that are in progress.

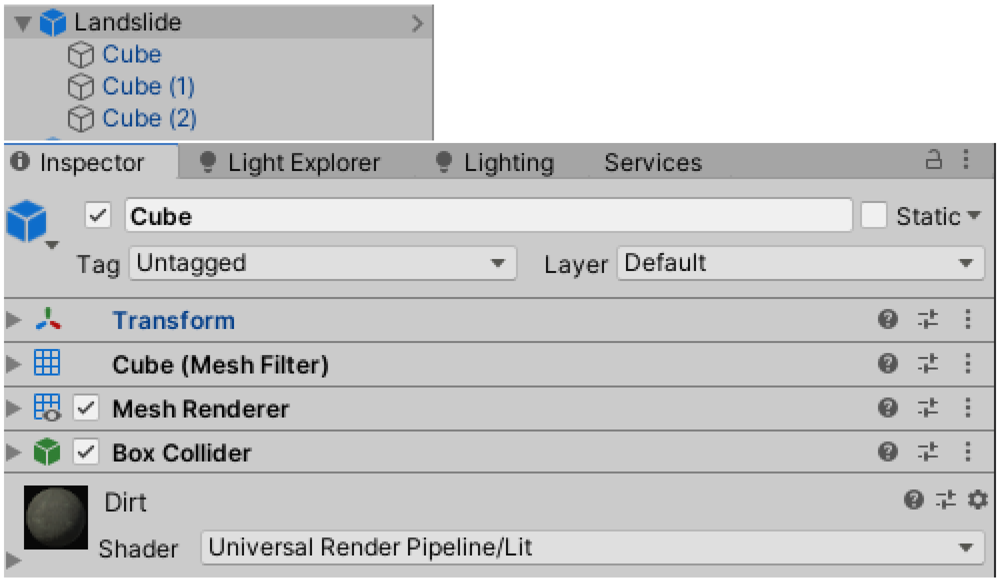


```
void ResetAnimations()
{
    HapticFeedback.Stop();
    Dust.Stop();
    Dust.Clear();
    Explosion.Stop();
    Explosion.time = 0;
    Fill_In.SetBool("Start", false);
    Fill_In.SetBool("Reset", true);
    Landslide.SetBool("Start", false);
    Landslide.SetBool("Reset", true);
    isReset = true;
}
```

Animations:

There are two main animations associated with actual objects in the simulation: Landslide and Fill_In. They will be discussed independently.

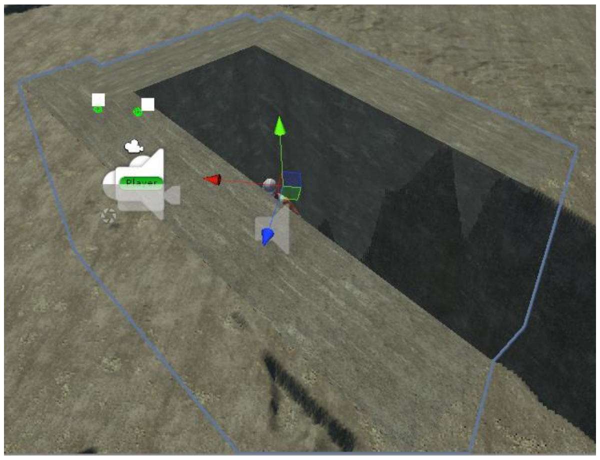
1. **Landslide:**
    a. This asset in the game has three sub-objects that are cubes. A material representing dirt was applied to each of these assets.

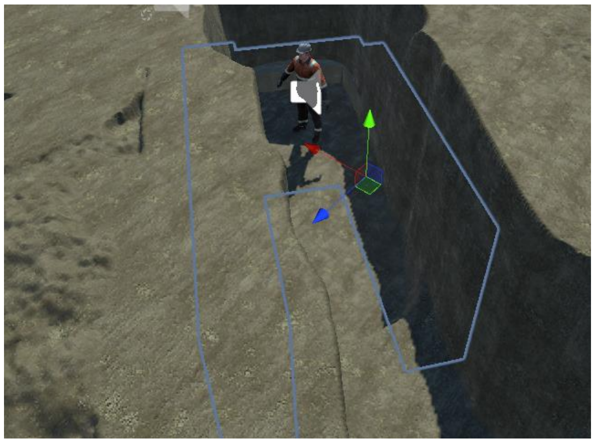


Landslide
Cube
Cube (1)
Cube (2)

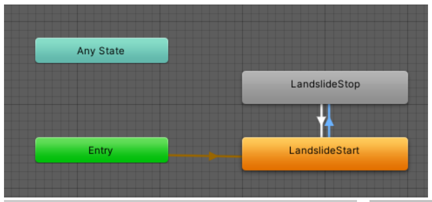
Cube
Static
Tag Untagged
Layer Default
Transform
Cube (Mesh Filter)
Mesh Renderer
Box Collider
Dirt
Shader Universal Render Pipeline/Lit

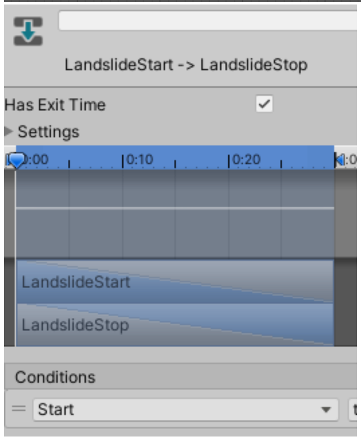
b. These objects were positioned and scaled such that they would properly fit around the edges of the ditch. The ditch sides themselves were eroded away to a position that reflected what the scene looked like after the land slide. When positioned all the way at the top, the Landslide objects made it appear as if the ditch was perfectly in tact.
c. An animation was applied to the parent object. Two positions were defined, both an initial position, and an end position.


Upper Position


Lower Position

d. Using Unity's animator system, a transition was defined from the initial state to the end position of the parent object. When the keyboard trigger is pressed, this transition initiates. Coincidentally, another transition was put in place defining movement back to the initial position. This maintains the state machine dynamic that Unity uses for it's animation system.




LandslideStart -> LandslideStop

Has Exit Time ✓

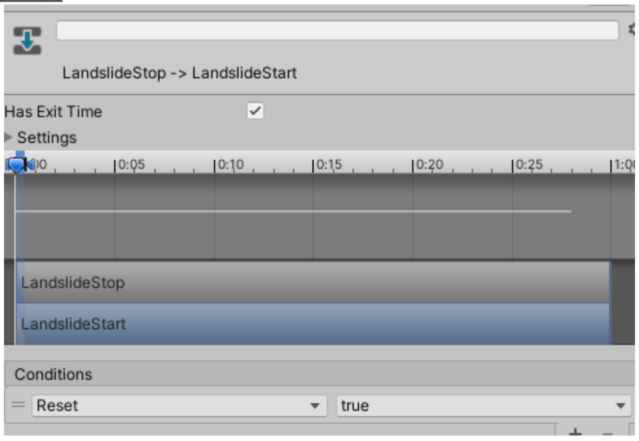
Settings

LandslideStart

LandslideStop

Conditions

Start


LandslideStop -> LandslideStart

Has Exit Time ✓

Settings

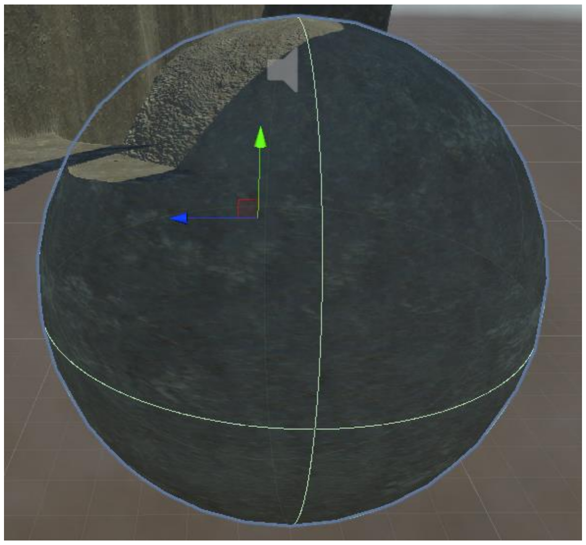
LandslideStop

LandslideStart

Conditions

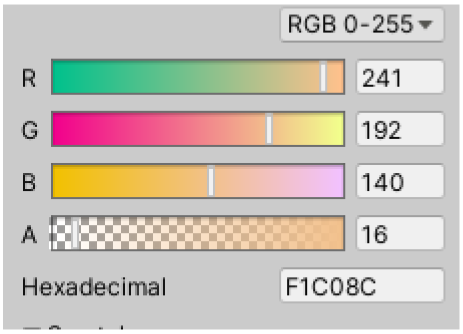
Reset true

2. **Fill_In:**
    a. This asset works the same as the Landslide asset, however in this case the animation scales the objcects. The Fill_In object itself is just a sphere with the same dirt material used on the terrain and the Landslide objects. The sphere is made to scale upward from an initial size to a larger size on its triggered state. This larger size engulfs the construction worker character making it look like he is buried in the event.



Dust Particle System:

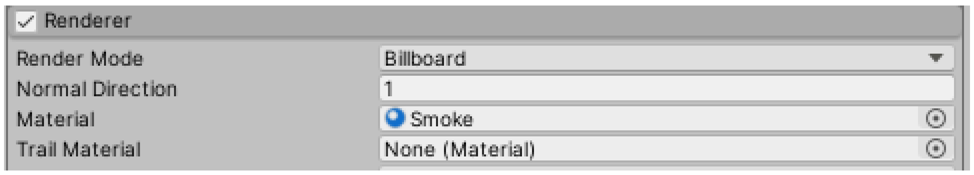
For the dust explosion I created a new empty object and added a particle system component to it. I found a transparent cloud image on the internet and imported this into Unity. I changed its generic material to indicate that the material was a alpha blend. This allowed tinting the color to make it look like dust as well:



RGB 0-255

R	241
G	192
B	140
A	16
Hexadecimal	F1C08C

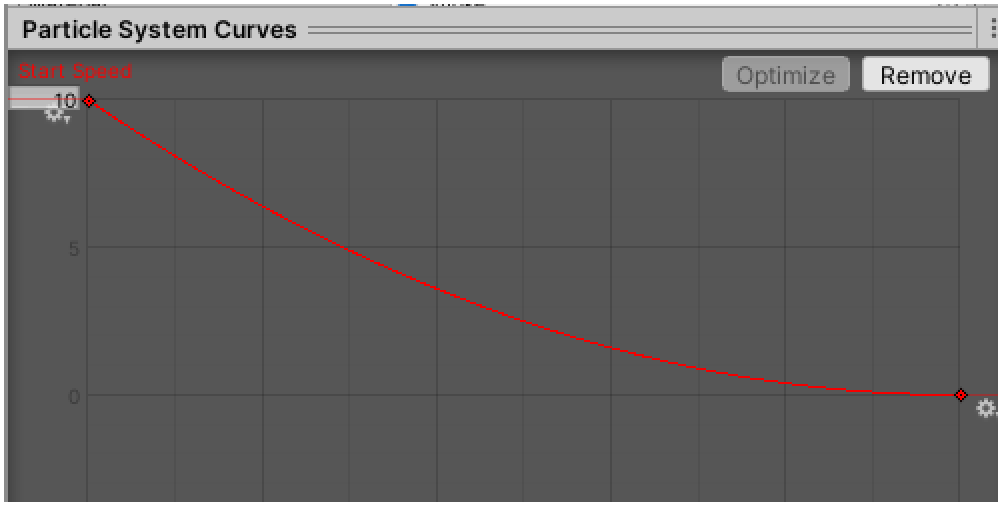
This was applied under the Renderer setting for the particle system.



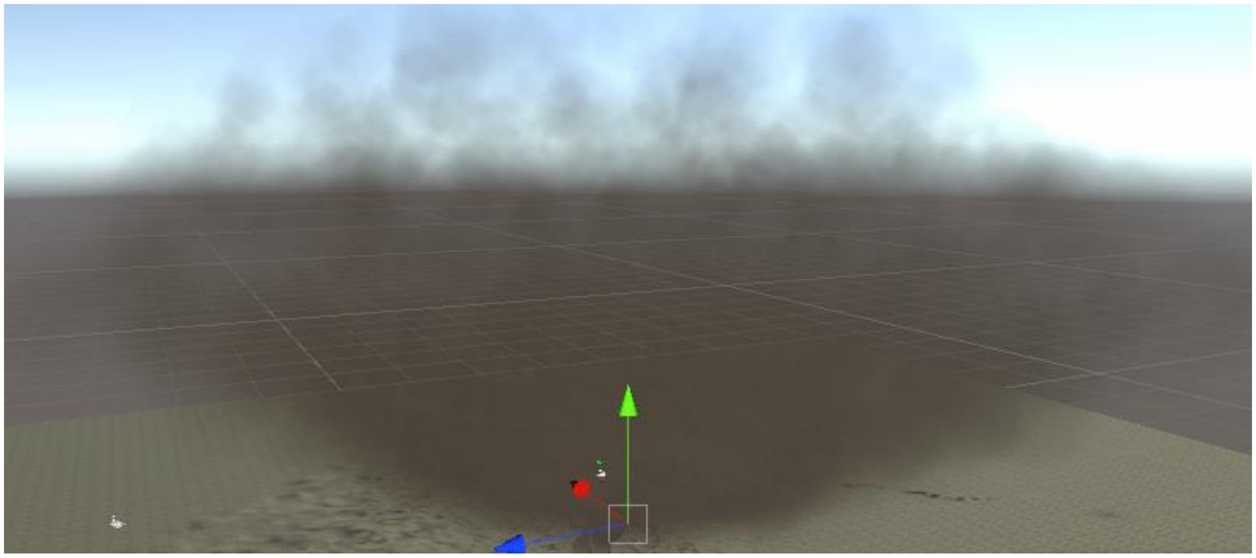
✓ Renderer

Render Mode	Billboard
Normal Direction	1
Material	Smoke
Trail Material	None (Material)

From there I changed various other parameters of the particle system in order to get the desired effect of a dust explosion. I changed the start lifetime to be 20 seconds, and the start speed to be a sloped transition from a high speed to a lower one. This gave the appearance that the dust was exploding outward at a rapid rate in the beginning, and then slowed down over time.



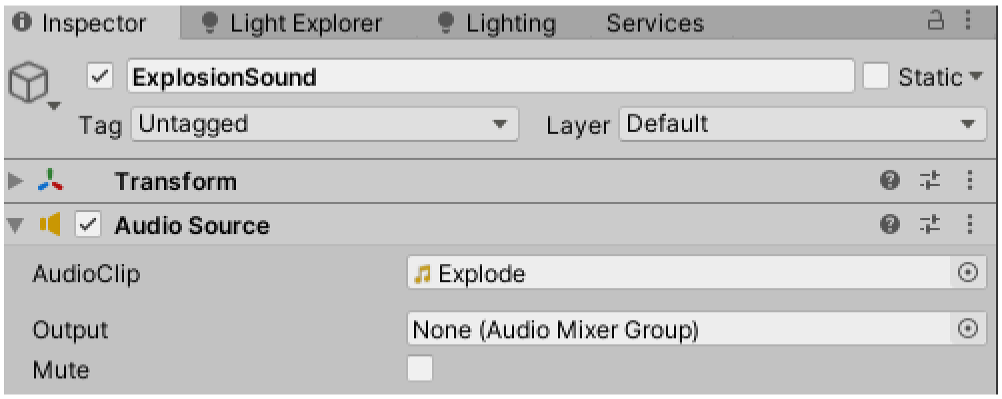
A gravity modifier was applied to the particles so that they would eventually fall when their speed was low enough. Additionally, the size and rotation of the particles were made to increase and spin, simulating the expansion of individual dust clouds. This created an explosion that slowly soared into the atmosphere, and the drifted downward as the animation progressed. This is an example of the plume produced:



When the player is standing next to the scene it produces a very realistic event as the dust cloud fully envelopes the player. This also makes it difficult to see in the ditch, adding to the overall realism.

Explosion Sound:

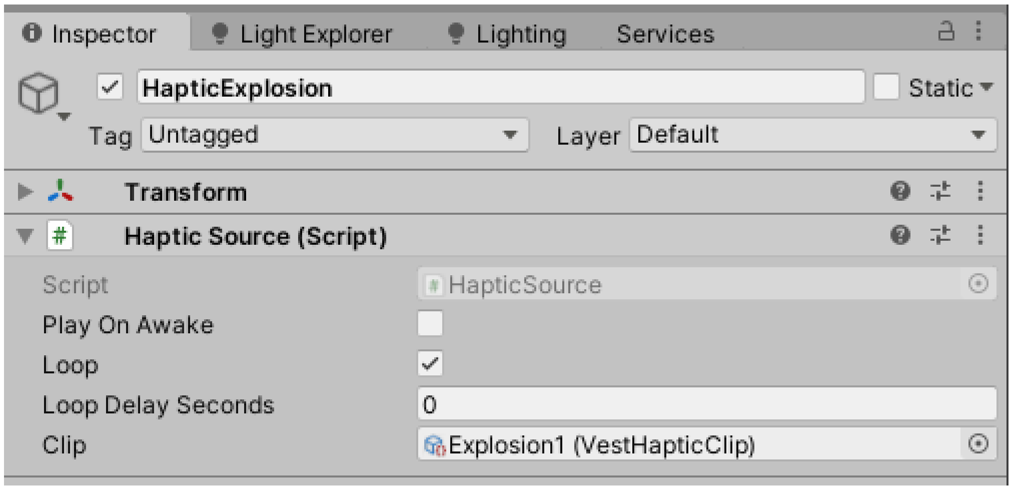
For the explosion sound a new object was added to the scene. An Audio Source component was then added to it. The audio was then set to an explosion sound that was imported from an internet source into Unity.



ExplosionSound
Tag Untagged
Layer Default
AudioClip Explode
Output None (Audio Mixer Group)
Mute

Haptic Explosion:

For the haptic explosion, a new object was added to the scene, and a Haptic Source component was added to it. From there, the clip for the haptic feedback was set to a pre-recorded haptic clip that came with the bHaptics SDK. This completed the haptic feedback object.



HapticExplosion
Tag Untagged
Layer Default
Script HapticSource
Play On Awake
Loop ✓
Loop Delay Seconds 0
Clip Explosion1 (VestHapticClip)

Conclusion:

This implementation of a construction safety event has several important aspects that can be taken from it in order to build more scene in the same vein of content. A control script is utilized to trigger events (animations) within other objects. These are made public for easy interface by developers. The script need only initiate the animations and reset them, as necessary. The script also provides the ability to select a desired keyboard button to initiate the entire event.

The objects themselves are varied but give a good indication of how to manipulate the environment to achieve the desired effects. 3D assets can be made to move and reset to their initial positions by way of animating them. Particle systems can be devised that replicate real life situations in detail. This could be expanded to steam, smoke, physical objects such as leaves or snow falling, sparks, flames, or electrical bolts. The particle systems of Unity have a vast number of parameters that can be changed to suit the needs of the event. Finally, sound and haptic feedback are also easy to develop, simply by creating empty objects within the scene, and the attaching audio or haptic source components to them. All these assets are referenced by the original script, which controls the overall playing of the event. As a result, this scene can be used as a good reference for creating more scenes like it.