

Game Audio: Final Project

Dave Maynard

Medieval Encounters: The Quest for Nerk

“1950’s Aliens Invade the Kingdom”



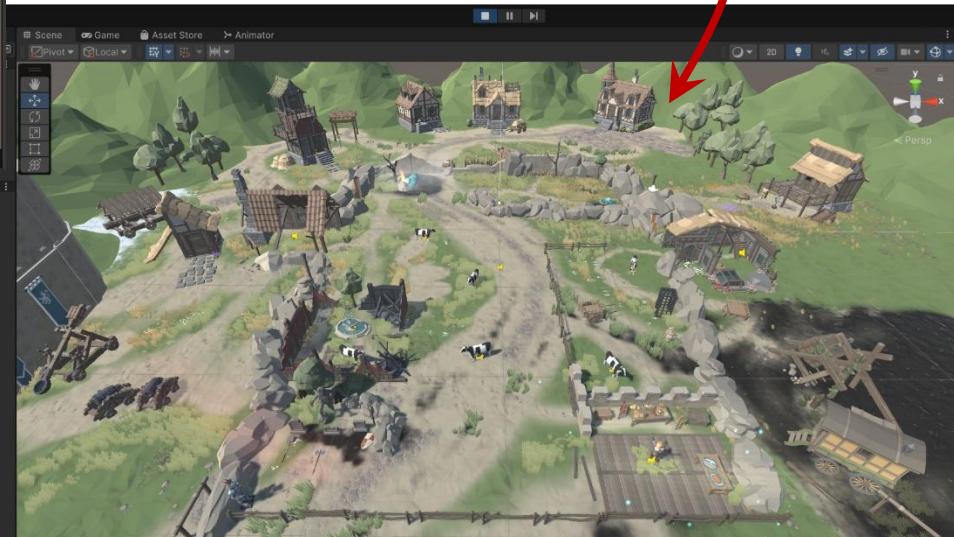
Custom Midjourney AI Generated Artwork

Game Audio: Final Project

Dave Maynard



Turned this into that ☺
in 2 Weeks



Game Audio: Final Project

Dave Maynard

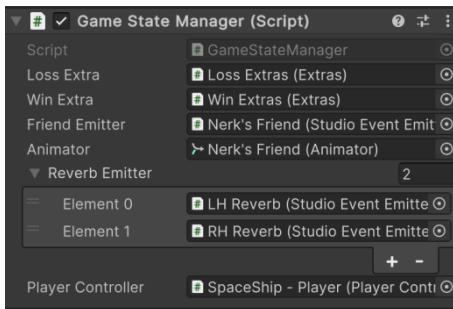
NOTE

You can hit the [ESCAPE] key to exit this assignment program.
You can also hit the [1] key to reset the level without exiting.

These are merely code snippets; the complete code encompasses much more than what is presented here, simplified for the sake of clarity.

```
Unity Message | 0 references
void Update()
{
    if (Input.GetKeyUp(KeyCode.Escape))
    {
        Application.Quit();
    }

    if (Input.GetKeyUp(KeyCode.Alpha1))
    {
        ResetLevel();
    }
}
```



I've incorporated a Singleton Game State Manager class into the project, which is now part of the game's hierarchy. It's designed to detect when the [Escape] key is pressed and ensures the program exits gracefully. It's also designed to detect when the [1] key is pressed to reset the level without exiting the program.

```
public void ResetLevel()
{
    foreach (var emitter in reverbEmitter)
    {
        emitter.Stop();
    }

    playerController.StopFMOD();
    SceneManager.LoadScene(SceneManager.GetActiveScene().name);
}

public void StopFMOD()
{
    musicEv.stop(FMOD.Studio.STOP_MODE.IMMEDIATE);
    flyingEv.stop(FMOD.Studio.STOP_MODE.IMMEDIATE);
}
```

For those who might be curious, these methods are how I ensure that all FMOD audio, including the music and the reverb zones, are forced to stop and the level reloads.

Game Audio: Final Project

Story of “1950’s Aliens Invade the Kingdom”



Our player lands on a planet with a rich Kingdom. He's looking for his buddy "Nerk" who came here last Wednesday.



He's sad to find his buddy crashed and his ship is on fire! But maybe he's around somewhere.



But YAY there are cows to abduct on this planet. He's happy again!



And there are lots of places to explore!



There are also places throughout that are special nooks that give everyday sounds a playful twist!

Game Audio: Final Project

Story of “1950’s Aliens Invade the Kingdom”

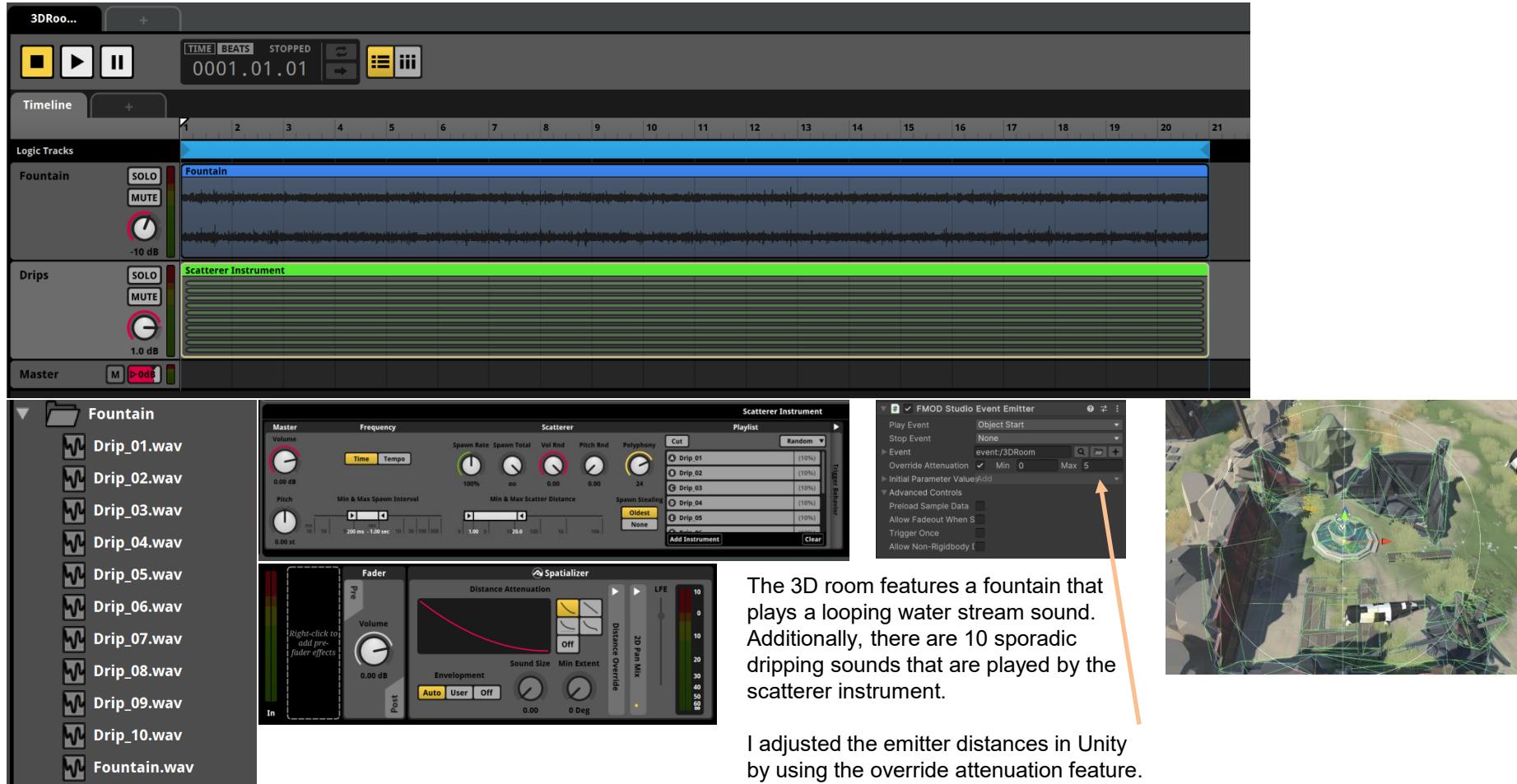


If our player finds Nerk's friend with all the snacks he WINS !



If our player finds the graveyard he is locked in and LOSES!

Game Audio: Final Project: FMOD Creation & Integration



The image shows the FMOD Studio interface and a 3D game environment in Unity. The FMOD interface displays Logic Tracks for 'Fountain' and 'Drips'. The 'Fountain' track uses a Scatterer instrument with 10 instances of 'Drip_01.wav' to create a looping water stream sound. The 'Drips' track uses a Scatterer instrument with 5 instances of 'Drip_01.wav' to create 10 sporadic dripping sounds. The Unity scene shows a 3D room with a fountain and green network lines representing audio emitter distances.

FMOD Studio Interface:

- Timeline:** Shows a timeline from 1 to 21 seconds.
- Logic Tracks:**
 - Fountain:** Solo, Mute, -10 dB. Scatterer instrument, 10 instances of 'Drip_01.wav'.
 - Drips:** Solo, Mute, 1.0 dB. Scatterer instrument, 5 instances of 'Drip_01.wav'.
- Master:** Mute, 0 dB.

Asset Browser: Shows a folder named 'Fountain' containing 11 audio files: Drip_01.wav through Drip_10.wav, and Fountain.wav.

Scatterer Instrument Settings:

- Master:** Volume (0.00 dB), Pitch (0.00 st).
- Frequency:** Time (Tempo), Min & Max Spawn Interval (1200 ms - 1.00 sec), Min & Max Scatter Distance (1.00 - 20.00).
- Scatterer:** Spawn Rate (100%), Spawn Total (oo), VolRnd (0.00), PitchRnd (0.00), Polyphony (24), Spawn Stealing (Oldest, None).
- Playlist:** Cut (Random), Playlist (Drip_01, Drip_02, Drip_03, Drip_04, Drip_05).
- Spatializer:** Fader (0.00 dB), Volume (0.00 dB), Distance Attenuation (curve), Envelopment (Auto, User, Off), Sound Size (0.00), Min Extent (0.00), Distance Override (20.00), Pan Mix (0.00), LFE (10).

FMOD Studio Event Emitter:

- Play Event: Object Start
- Stop Event: None
- Event: event/3DRoom
- Override Attenuation: Min 0, Max 5
- Initial Parameter Value: Add
- Advanced Controls:
 - Preload Sample Data
 - Allow Fadecut When S
 - Trigger Once
 - Allow Non-Rigidbody

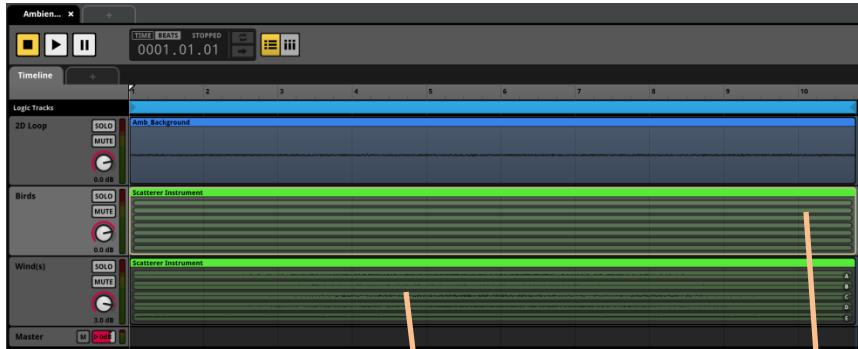
Unity 3D Room: A 3D environment with a fountain and green network lines representing audio emitter distances.

Text:

The 3D room features a fountain that plays a looping water stream sound. Additionally, there are 10 sporadic dripping sounds that are played by the scatterer instrument.

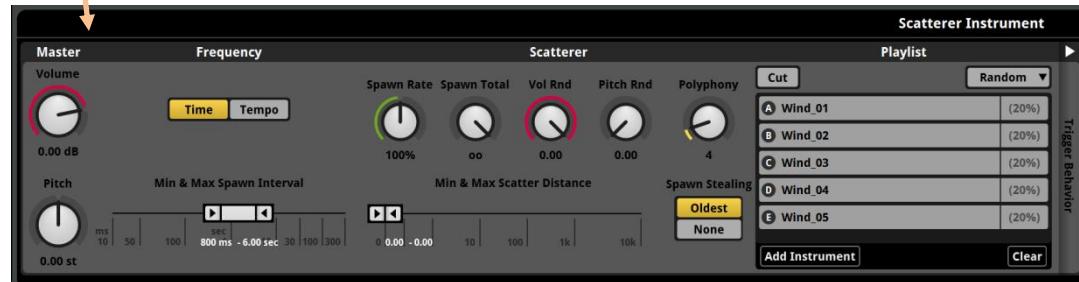
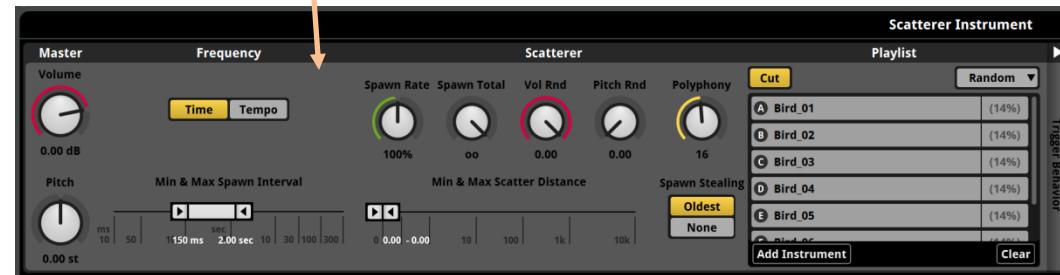
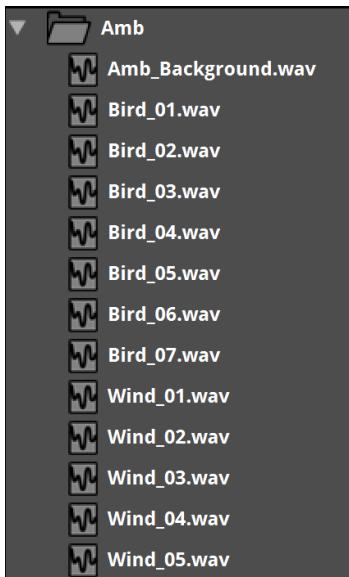
I adjusted the emitter distances in Unity by using the override attenuation feature.

Game Audio: Final Project: FMOD Creation & Integration

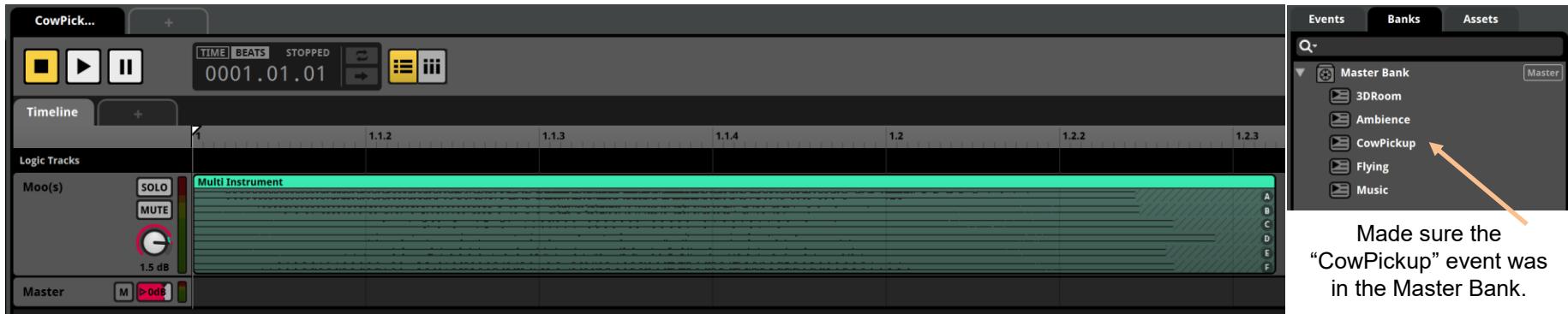


The ambient timeline includes a continuous loop of park sounds with no human voices. Additionally, it features two scatterer instruments

One randomly plays seven distinct bird calls, and another that randomly emits five varieties of wind sounds. I have fine-tuned the parameters of the scatterer instruments, such as polyphony, minimum and maximum spawn intervals, and minimum and maximum scatter distances, to achieve a blend that I thought best fit the scene.



Game Audio: Final Project: FMOD Creation & Integration



The screenshot shows the FMOD Studio interface. The top bar displays the event name 'CowPick...', time '0001.01.01', and status 'STOPPED'. The timeline shows a single logic track named 'Multi Instrument' with a single event labeled 'Moo(s)'. The event has 'SOLO' and 'MUTE' buttons, and a volume fader set to '1.5 dB'. The 'Master' track is selected. The right panel, titled 'Banks', shows the 'Master' bank with sub-banks: '3DRoom', 'Ambience', 'CowPickup' (which is highlighted with an orange arrow), 'Flying', and 'Music'.

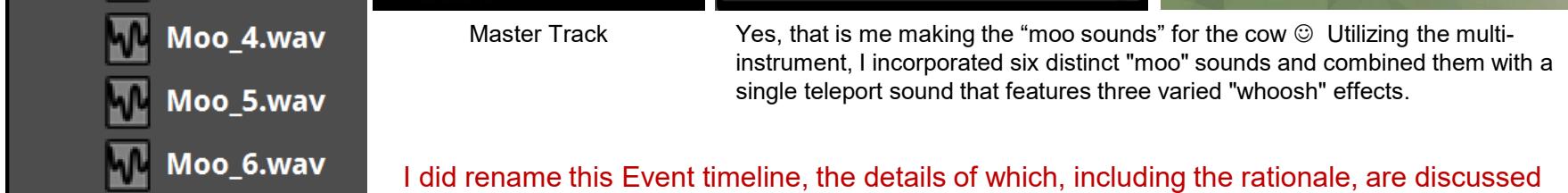
Made sure the "CowPickup" event was in the Master Bank.



The screenshot shows the 'Moo' event structure. It contains six wav files: 'Moo_1.wav', 'Moo_2.wav', 'Moo_3.wav', 'Moo_4.wav', 'Moo_5.wav', and 'Moo_6.wav'. The 'Master' track settings are shown, including a 'Fader' with 'Volume' at 0.00 dB and 'Reverb' at -20.0 dB, and a 'Playlist' with five entries: 'A Moo_1' (17%), 'B Moo_2' (17%), 'C Moo_3' (17%), 'D Moo_4' (17%), and 'E Moo_5' (17%).

Master Track

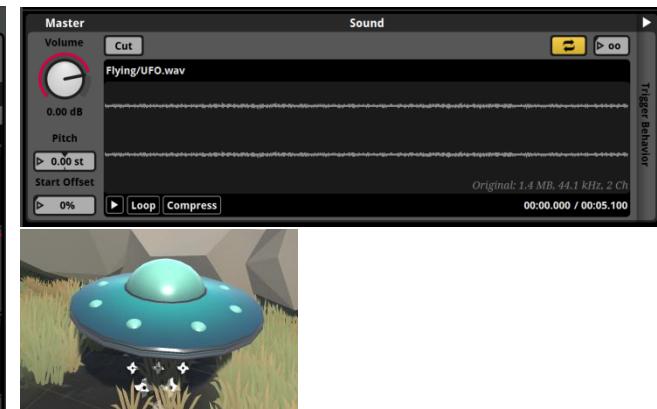
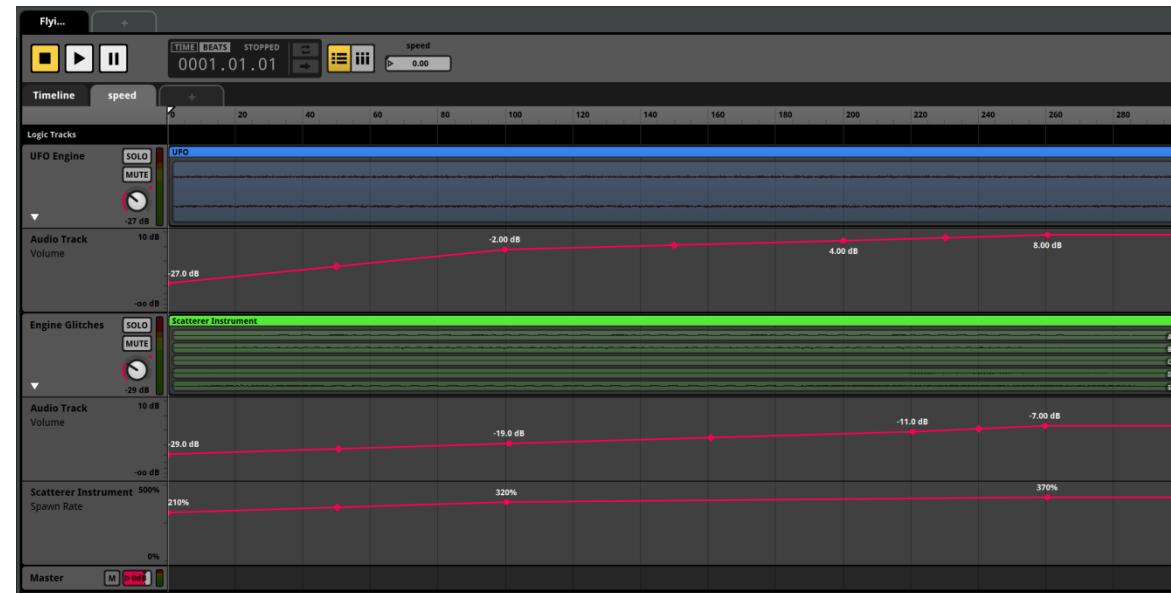
Yes, that is me making the "moo sounds" for the cow 😊 Utilizing the multi-instrument, I incorporated six distinct "moo" sounds and combined them with a single teleport sound that features three varied "whoosh" effects.



The screenshot shows the 'CowPick...' event timeline. The timeline is labeled 'Multi Instrument' and contains a single event named 'Moo(s)'. The event has 'SOLO' and 'MUTE' buttons, and a volume fader set to '1.5 dB'. The 'Master' track is selected. The right panel, titled 'Banks', shows the 'Master' bank with sub-banks: '3DRoom', 'Ambience', 'CowPickup' (which is highlighted with an orange arrow), 'Flying', and 'Music'. The bottom right shows a 3D cow model in a game engine environment.

I did rename this Event timeline, the details of which, including the rationale, are discussed in the "Extra" section of this write-up.

Game Audio: Final Project: FMOD Creation & Integration



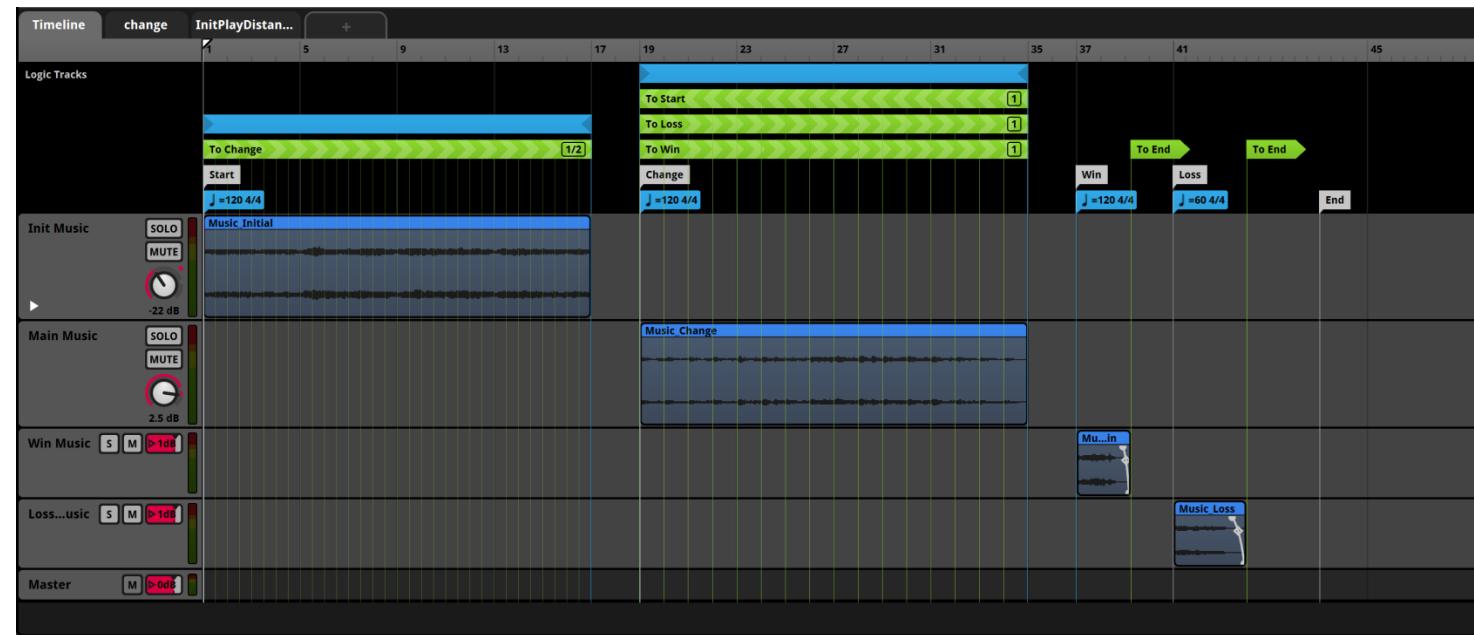
Yes, that is also me making the main engine sounds using my mouth 😊 along with a few other sounds baked in. I employed the scatterer instrument to integrate five unique engine glitching sounds.

By using the ship's speed data from the game engine, I automated the adjustment of the overall volume(s) and the spawn rate of the engine glitches.



Made sure the "Flying" event was in the Master Bank.

Game Audio: Final Project: FMOD Creation & Integration



Music

- Music_Change.wav
- Music_Initial.wav
- Music_Loss.wav
- Music_Win.wav

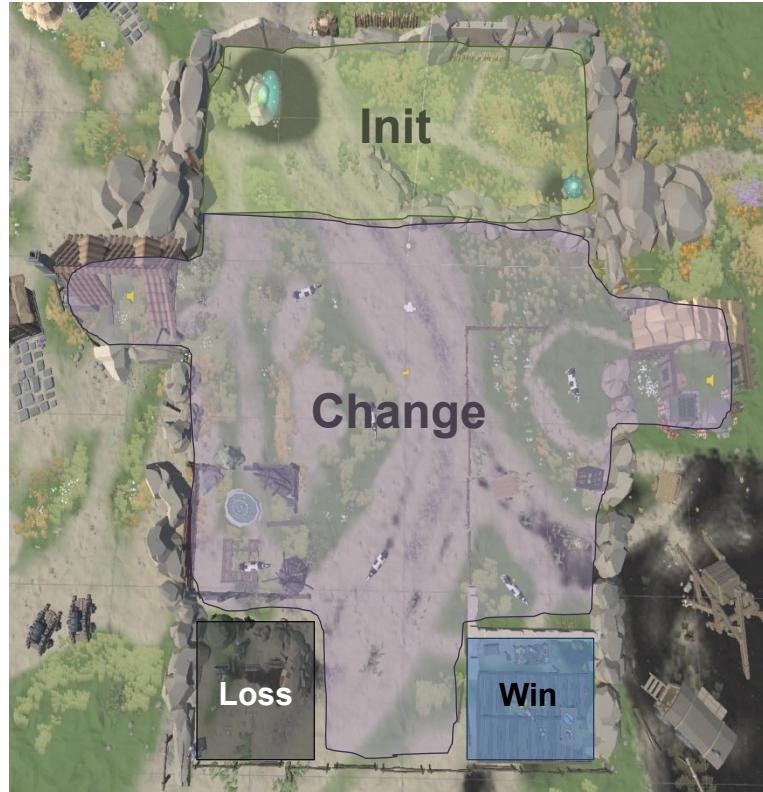
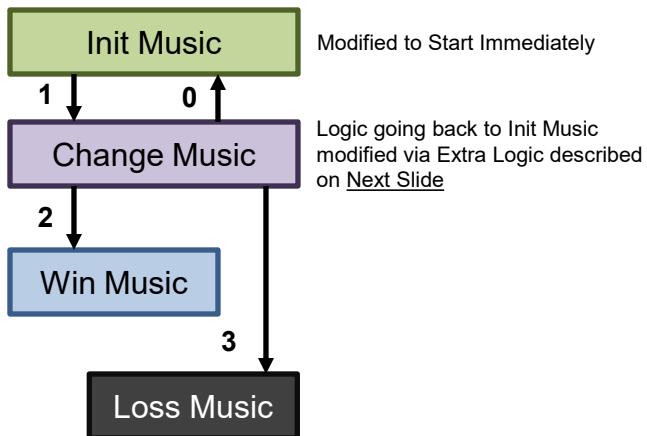
Following the specifications of the assignment, I created an Initial Music piece, Change Music, and both Win and Loss Stingers. My approach to the compositions was influenced by the style of 1950s science fiction, with a comedic twist.

I deviated slightly from the original transition markers and logic requirements. This was primarily because the layout of the game area made it impossible to transition directly from the "initial" music to the win or loss zones; one must pass through the "change" music area to enter the win or loss areas.

Game Audio: Final Project: FMOD Creation & Integration

Logic Map for Music Transitions

= 'change' parameter value

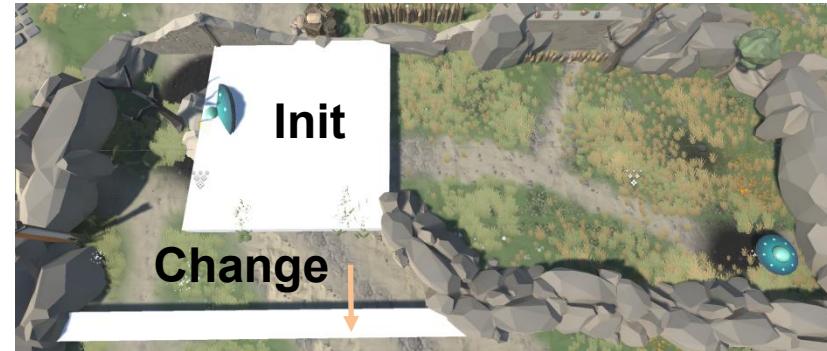
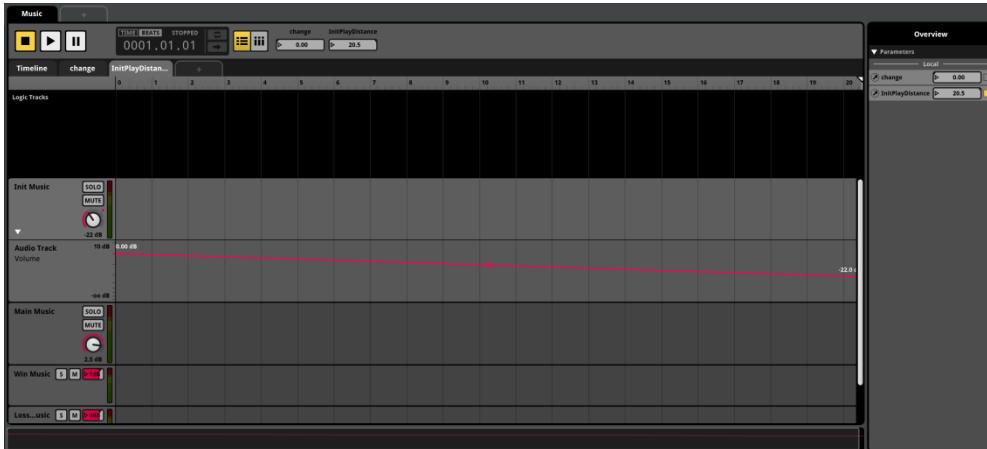


Due to this logical mapping, the 'Win' or 'Loss' stingers cannot be triggered from the 'Init' area.

Consequently, the logic presented in this and the previous FMOD slide adheres to the established flow of the game's physical layout.

Very Rough Overlay

Game Audio: Final Project: FMOD Creation & Integration

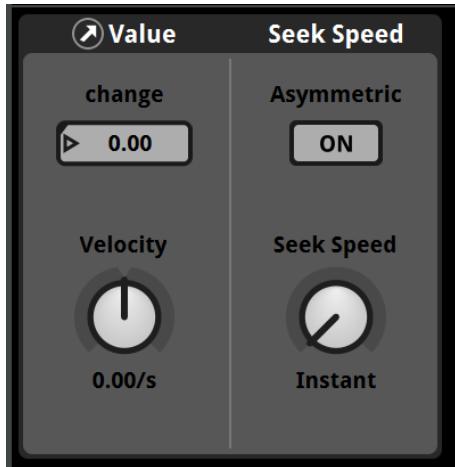


I intentionally added a new floating-point distance parameter in FMOD to craft a dramatic crescendo for the 1950s-style crash site discovery. Although I'm aware that this effect could potentially be achieved using a 2D/3D timeline and emitter setup, I chose to understand this challenge through coding within this assignment.

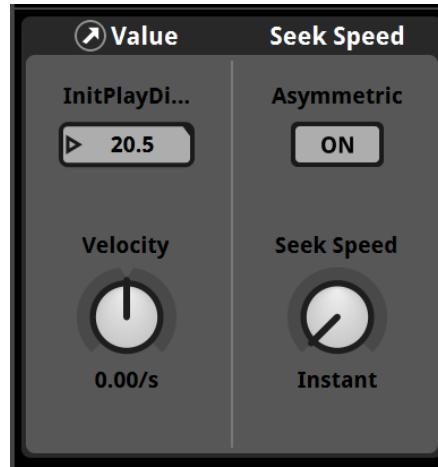
The resulting audio effect is that as the player's ship approaches the crash site, the volume of the dramatic "initial" music increases, and it decreases as the distance grows. Furthermore, the code I developed allows for a seamless transition back to the dramatic "initial" music from the "change" music, contingent on the player's interaction with a collider cube and their proximity to the crash site.

The code is discussed in the "Extra" section of this write-up.

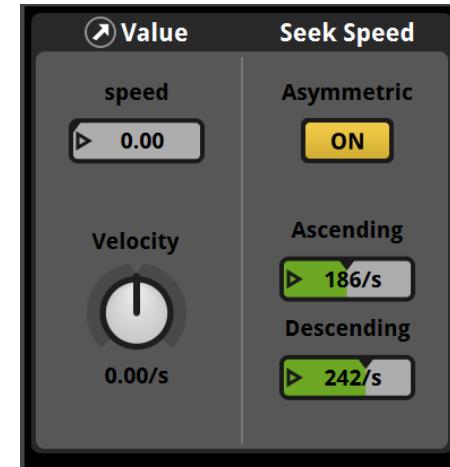
Game Audio: Final Project: FMOD Creation & Integration



"change" Parameter



"InitPlayDistance" Parameter



"speed" Parameter

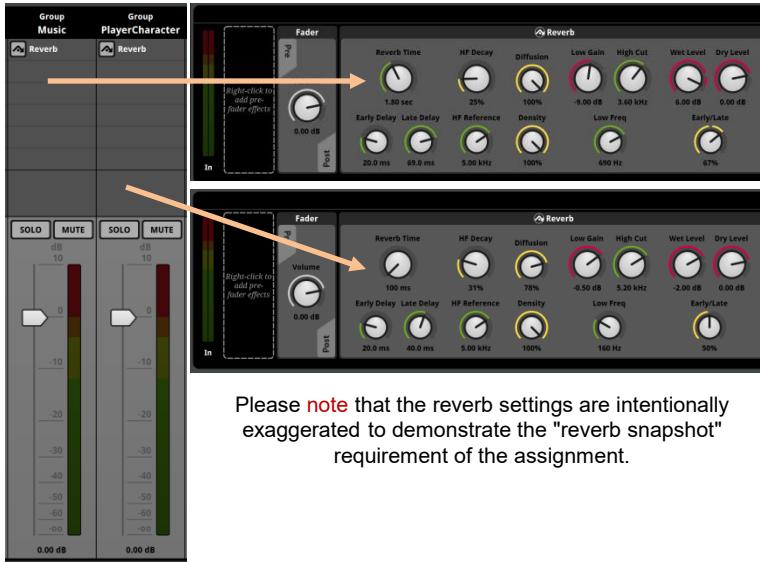
The seek speeds for the "change" and my custom "InitPlayDistance" parameter must be instantaneous. For the "change" parameter, this is because it represents a discrete event, shifting from one numeric value to another, which in turn alters the music state. As for the "InitPlayDistance" parameter, it's a real-time value critical for adjusting the volume of the "initial" music associated with the crash site.

Conversely, the "speed" parameter required a bit more flexibility in its values to provide a sense of acceleration and deceleration in the ship's engine sounds, allowing for a fast but gradual increase or decrease in volume.

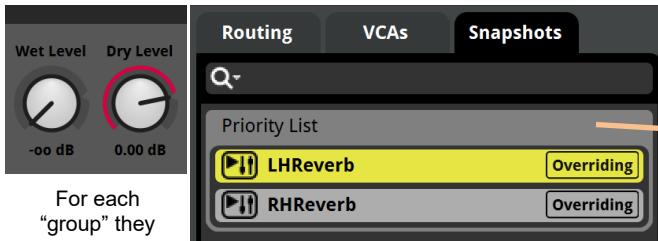
Game Audio: Final Project: FMOD Creation & Integration

Reverb Snapshot (Left House)

Please note that both the ship's flying audio and the music are affected by the reverb zones. Since moving the ship is required to hear the effect, adding the music was thought to enhance the demonstration of the reverb zone's impact.



Please note that the reverb settings are intentionally exaggerated to demonstrate the "reverb snapshot" requirement of the assignment.



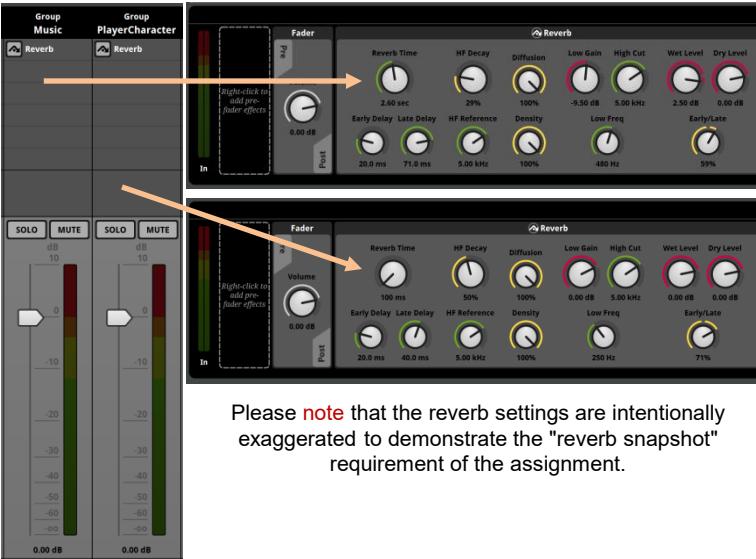
For each "group" they have the "Wet Level" set to 'off' when the snapshots are not active.



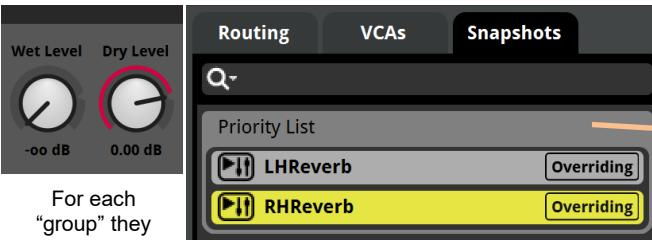
Game Audio: Final Project: FMOD Creation & Integration

Reverb Snapshot (Right House)

Please note that both the ship's flying audio and the music are affected by the reverb zones. Since moving the ship is required to hear the effect, adding the music was thought to enhance the demonstration of the reverb zone's impact.



Please note that the reverb settings are intentionally exaggerated to demonstrate the "reverb snapshot" requirement of the assignment.



For each "group" they have the "Wet Level" set to "off" when the snapshots are not active.

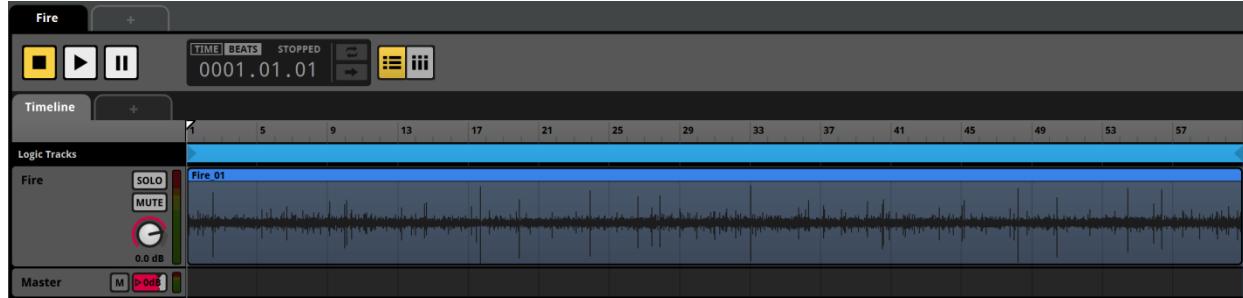


For the "Right House," the reverb zone is limited to the area under the roof, demonstrating how the ship can enter the open area to become reverb-free again.



Game Audio: Final Project: FMOD Creation & Integration

Fire at the Crash Site (FMOD 3D Timeline)



FMOD Studio Event Emitter

Play Event: Object Start

Stop Event: None

Event: event:/Fire

Override Attenuation: Min 1 Max 20

Initial Parameter Values: Add

Advanced Controls:

- Preload Sample Data:
- Allow Fadeout When S:
- Trigger Once:
- Allow Non-Rigidbody:

This was simply an extra 3D Audio Emitter placed in world with a looping fire sound.

Game Audio: Final Project: FMOD Creation & Integration

One-Shot SFX Played when Win Occurs



Events Banks Assets

Q- 3DRoom

Ambience

CowPickup

Fire

Flying

Friend

LossFX

Music

WinFX

Extras (Script)

Game Objects

- Element 0: cow 01
- Element 1: cow 02
- Element 2: cow 03
- Element 3: cow 04
- Element 4: cow 05
- Element 5: cow 06
- Element 6: HCFX_Explosion_03
- Element 7: Blocker

Box Collider Win Area (Box Collider)

Win...

TIME BEATS STOPPED 0001.01.01

Timeline

Logic Tracks

WinFX

SOLO

MUTE

0.0 dB

Master

0.0 dB

In

Right-click to add pre-fader effects

Fader

Volume 0.00 dB

Post

Send

Reverb -11.5 dB

```
Unity Script | 2 references
public class Extras : MonoBehaviour
{
    [SerializeField]
    private List<GameObject> gameObjects;
    [SerializeField]
    private BoxCollider boxCollider;

    2 references
    public void Engage()
    {
        boxCollider.enabled = false;

        foreach (GameObject go in gameObjects)
        {
            go.SetActive(true);
        }

        GameStateManager.instance.GoIdle();
    }
}
```

In summary, the updated code in the player controller triggers the one-shot SFX audio, activates the particle effect, disables a box collider to prevent multiple triggers, sets all GameObjects in the list to active (the cows and a collision blocker), and transitions "Nerk's Friend's" animation to an idle state, stopping her from waving and making her stand idle.

Game Audio: Final Project: FMOD Creation & Integration

One-Shot SFX Played when Loss Occurs



The image shows two screenshots of a game. The left screenshot shows a player character and a blue alien-like object in a graveyard setting. An orange arrow labeled "Loss" points from the player to the alien object. The right screenshot shows the same scene after the loss occurs, with the alien object now destroyed and lying on the ground.

FMOD Studio Project Overview:

- Events:** 3DRoom, Ambience, CowPickup, Fire, Flying, Friend, **LossFX** (highlighted in yellow), Music, WinFX.
- Banks:** None.
- Assets:** None.

LossFX Event Timeline:

- Timeline:** TIME BEATS STOPPED 0001.01.01
- Logic Tracks:** LossFX (SOLO, MUTE, 0.0 dB)
- Master:** M 0.0 dB

Script (Extras.cs):

```
if (other.gameObject.CompareTag ("losecube"))
{
    playbackState = 3;
    FMODUnity.RuntimeManager.PlayOneShot(lossSound);
    musicEv.setParameterByName("Change", playbackState);
    GameStateManager.instance.lossExtra.Engage();
}

public void Engage()
{
    boxCollider.enabled = false;
    foreach (GameObject go in gameObjects)
    {
        go.SetActive(true);
    }
    GameStateManager.instance.GoIdle();
}
```

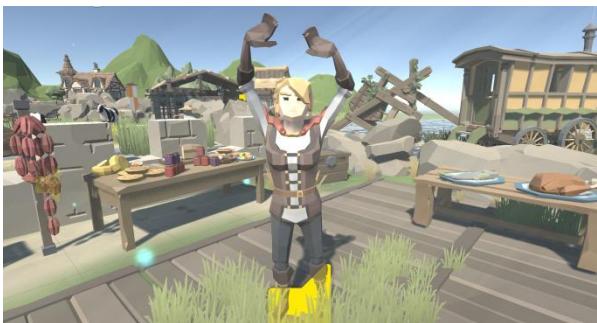
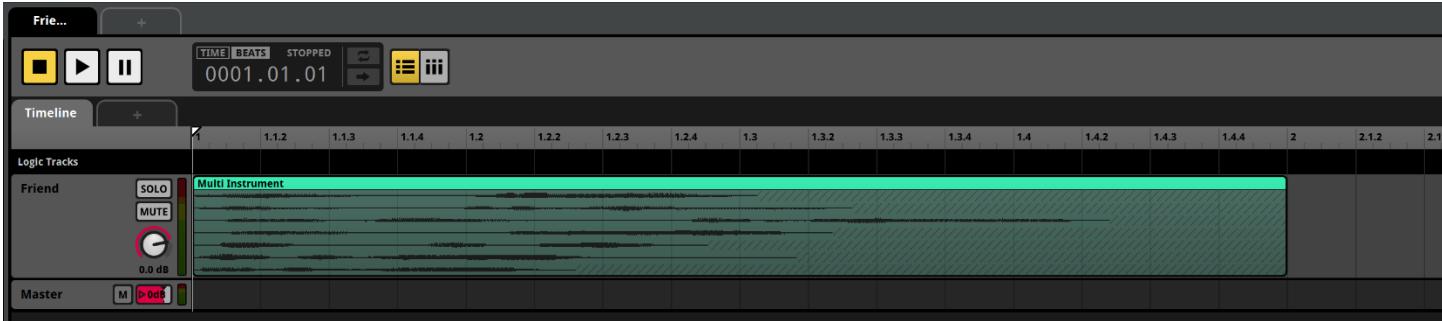
In summary, the updated code in the player controller triggers the one-shot SFX audio, activates the particle effect, disables a box collider to prevent multiple triggers, sets all GameObjects in the list to active (the gravestones, ballista and a collision blocker), and for completeness transitions "Nerk's Friend's" animation to an idle state, stopping her from waving and making her stand idle.

Game Audio: Final Project: FMOD Creation & Integration

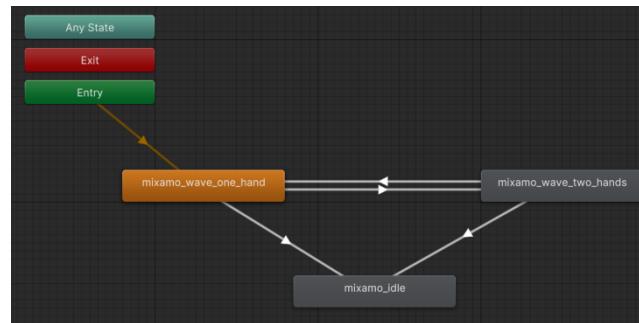
“Nerk’s Friend” (Code Triggered FMOD 3D Timeline)



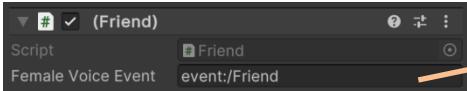
I navigated the Mixamo animation pipeline to integrate animations with my LowPoly Characters successfully.



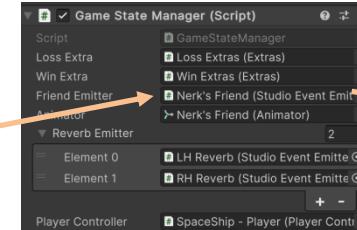
Admittedly, the process is somewhat complex.



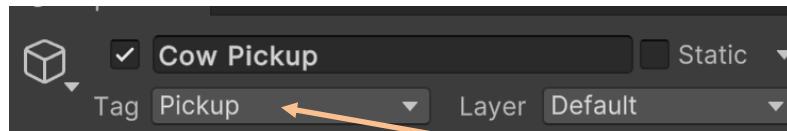
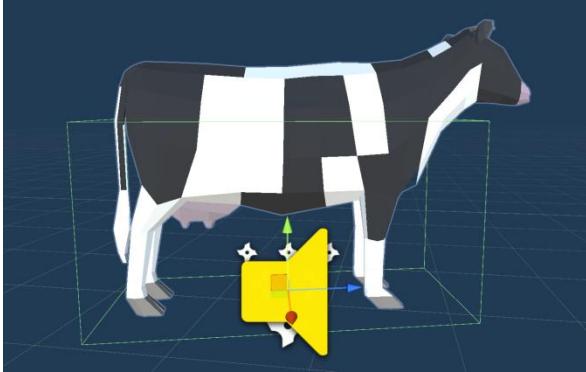
I utilized my license for ElevenLabs to create an AI-powered female voice for the lines above.



Essentially, whenever the animation switches to a two-handed wave, a call is made to the GameStateManager, triggering the FMOD Event Emitter on the "Nerk's Friend" GameObject.

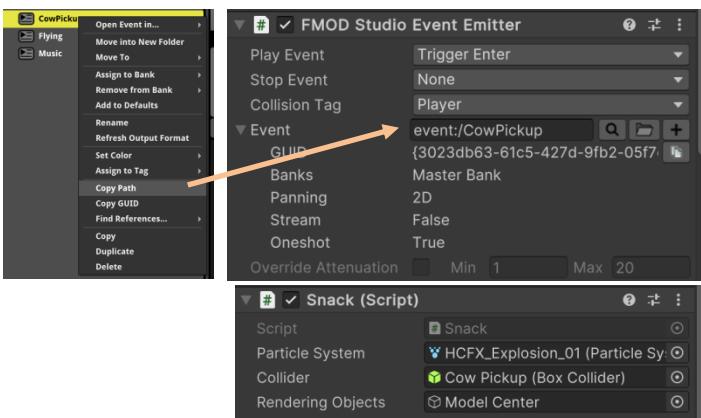


Game Audio: Final Project: Complete the Final Project: Extras: Cow Pickup



```
if (other.gameObject.CompareTag ("Pickup"))
{
    other.GetComponent<Snack>().RemoveSnack();
}
```

I created a new prefab of a cow that has the same tag as the project included code so that all I had to do was trigger my new code from the original code in the "PlayerController" class.



The path of the new event timeline needed to align with the specifications set in the "play event" 'trigger enter' within the FMOD Studio Event Emitter component.

Whenever there are changes of this nature, Unity provides a scanner for updates following a complete build in FMOD. This procedure is quite straightforward and is facilitated by a pop-up user interface within the project.

I won't delve into the specifics of the code below, but in essence, when the RemoveSnack() method is invoked, it disables the collider, turns off the renderer for the cow model, and then initiates a particle effect that coincides with the GameObject's destruction.

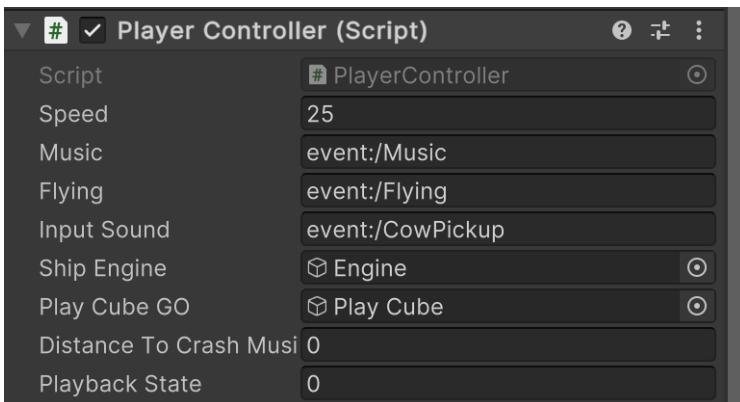
```
1  using System.Collections;
2  using UnityEngine;
3
4  #pragma warning disable CS0108
5  #pragma warning disable CS0108
6  public class Snack : MonoBehaviour
7  {
8      [SerializeField]
9      private ParticleSystem particleSystem;
10     [SerializeField]
11     private BoxCollider collider;
12     [SerializeField]
13     private GameObject renderingObjects;
14
15     public void RemoveSnack()
16     {
17         collider.enabled = false;
18         renderingObjects.SetActive(false);
19         particleSystem.Play();
20         StartCoroutine(WaitForParticleSystem());
21     }
22
23     private IEnumerator WaitForParticleSystem()
24     {
25         yield return new WaitForSeconds(particleSystem.main.duration + 0.1f);
26
27         if (!particleSystem.isPlaying)
28         {
29             Destroy(gameObject);
30         }
31     }
32 }
```

Game Audio: Final Project: Extras: “Initial” Music Work Product

```
// DSM
[SerializeField]
private GameObject shipEngine;
[SerializeField]
private GameObject playCubeGO;
private float tempVel = 0f;
public float distanceToCrashMusic = 0f;
private FMOD.Studio.PLAYBACK_STATE play_state;
public int playbackState = 0;
```

```
musicEv = FMODUnity.RuntimeManager.CreateInstance(music);
flyingEv = FMODUnity.RuntimeManager.CreateInstance(flying);
flyingEv.setParameterByName("speed", 0);
flyingEv.start();
```

Here I'm making sure the flyingEv is setup properly and the initial speed is set to 0 and started.



I added and modified the location of a few variables.

```
FMODUnity.EventReference Flying;
public string flying = "event:/Flying";
FMOD.Studio.EventInstance flyingEv;
```

Modified the name of the EventInstance from rolling to flying for clarity. FYI the EventReference isn't used nor needed.

```
// DSM
shipEngine.transform.localScale = new Vector3((tempVel / 100f), (tempVel / 100f), (tempVel / 100f));
distanceToCrashMusic = Vector3.Distance(this.transform.position, playCubeGO.transform.position);
musicEv.setParameterByName("InitPlayDistance", distanceToCrashMusic);
if (distanceToCrashMusic <= 9.37)
{
    if (playbackState != 0)
    {
        playbackState = 0;
        musicEv.setParameterByName("Change", playbackState);
        musicEv.start();
    }
}
```

Inside the FixedUpdate() method, I inserted code that essentially utilizes the speed, stored in the variable 'tempVel' for temporary velocity, to determine the size of the particle effect underneath the spaceship. This code also assigns the distance to a new variable 'distanceToCrashSite' that I pass to FMOD. I introduced another variable named 'playbackState' to store the value corresponding to the "change" parameter in FMOD. This setup allows me to revert to the "initial" music from the "change" music, contingent on the FMOD "change" logic, the 'playbackState' and the ship's distance from the crash site, which is defined by the location of the original play cube.

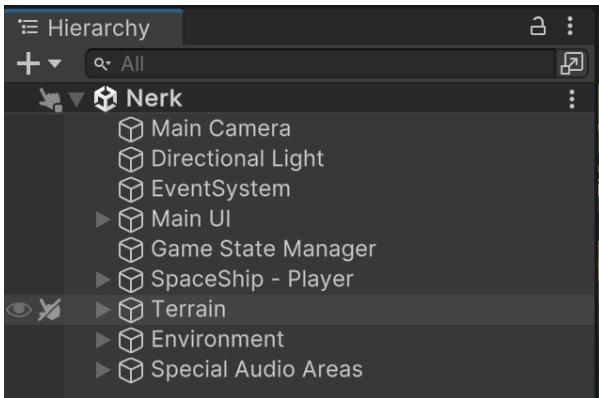
```
if (other.gameObject.CompareTag ("ChangeCube"))
{
    playbackState = 1;
    musicEv.setParameterByName("Change", playbackState);
}
```

Here is the slightly modified "ChangeCube" logic but with the playbackState being utilized.

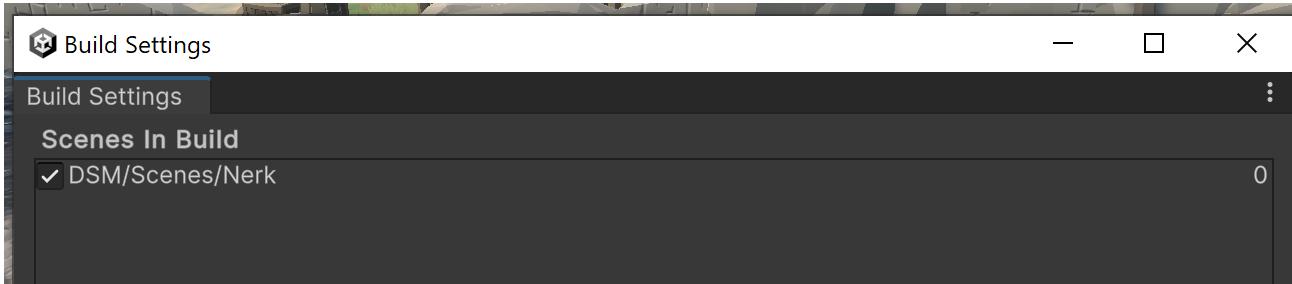
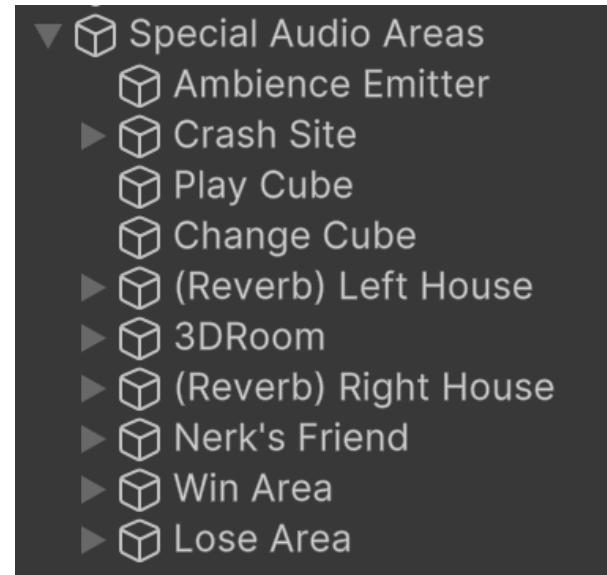
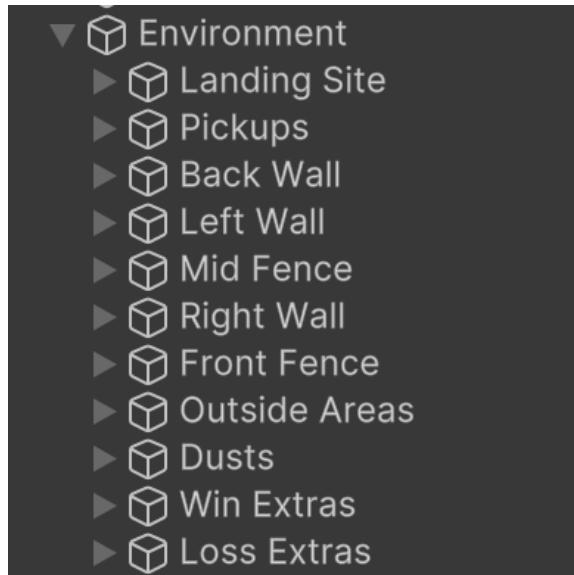
There were a few other items I changed both in code and the scene's hierarchy but nothing worth noting as ultra important.

Game Audio: Final Project: Extras: Game's Hierarchy

Cleaned up the Game's Hierarchy and Created a New Scene (from previous)



The game's hierarchy was restructured according to the collections of GameObjects and their overall purpose.



Made sure the correct scene was included in the build settings for appropriate build procedures.

Game Audio: Final Project: Extras: Very Simple UI



The image shows a game scene with a UI message at the top: "[ESC] To Exit Game :: [1] to Reset the Game". Below the game view is the Unity Editor interface, specifically the Canvas Scaler settings for a UI element.

Main UI (Selected)

- Panel
- UI Text (TMP)

TextMeshPro - Text (UI)

Text Input: [ESC] To Exit Game :: [1] to Reset the Game

Text Style: Normal

Main Settings

- Font Asset: LiberationSans SDF (TMP_Font)
- Material Preset: LiberationSans SDF Material
- Font Style: B I U S ab AB SC
- Font Size: 36
- Auto Size
- Vertex Color
- Color Gradient
- Override Tags

Canvas Scaler

UI Scale Mode: Scale With Screen Size

Reference Resolution: X 1920 Y 1080

Screen Match Mode: Match Width Or Height

Match: 0

Width: 0

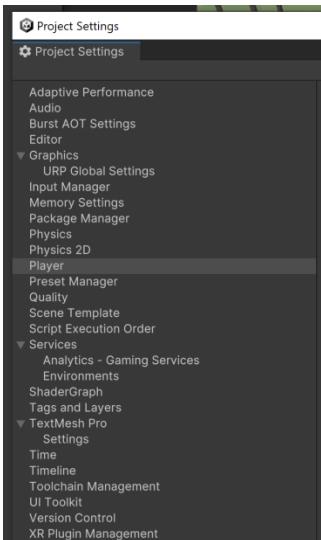
Height: 0

Reference Pixels Per U: 100

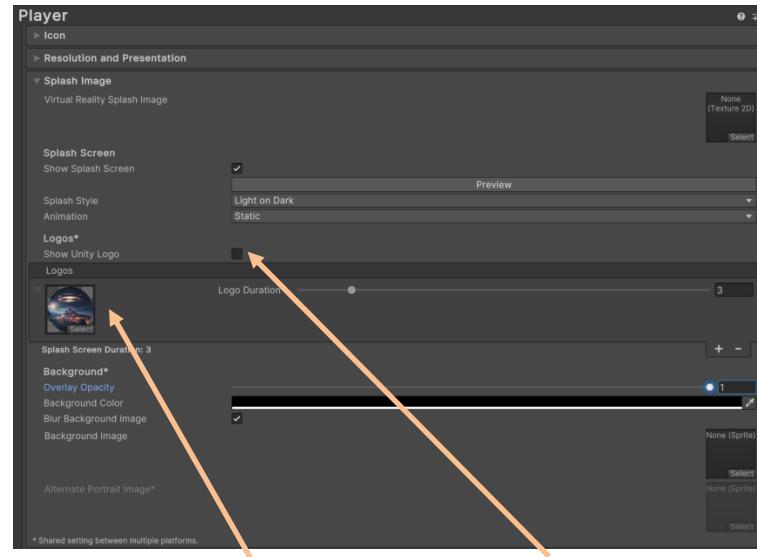
The UI scales based on the runtime resolution. The "Reference Resolution" serves precisely as its name suggests: it's what Unity uses as a benchmark when I constructed the very simple UI, ensuring it knows how and what to scale accurately.

Game Audio: Final Project: Extras: Splash Screen

Please **note** that a "Pro" license for Unity is required to perform some of the actions described here.

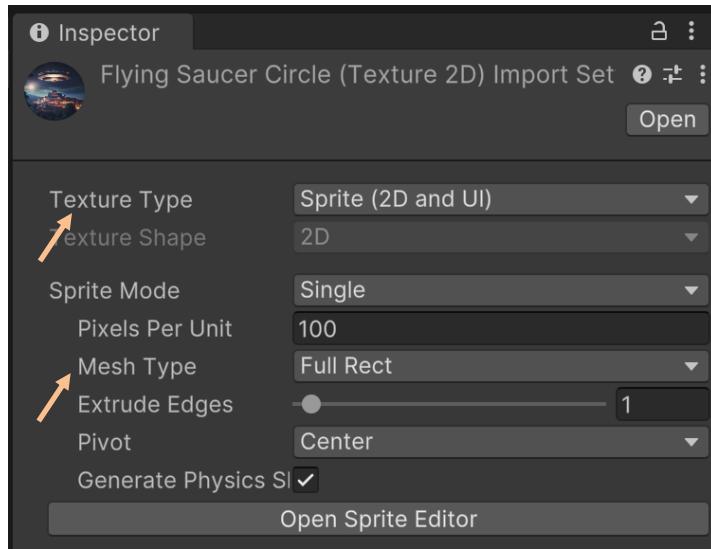


Edit to "Project Settings"
then "Player"



Replace this area with your
loading image.

Disabling the "Unity Logo"
option is likely only available
with "Pro" licenses.



Set the logo image's "Texture Type" to "Sprite (2D and UI)," and under "Sprite Mode," the "Mesh Type" should be adjusted to "Full Rect."