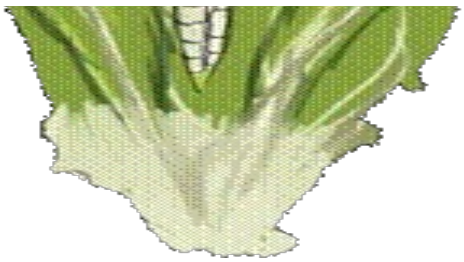
Viral Impact
Sound Design Document



By Bao Le

1. Overview

This Sound Design Document (SDD) serves as an implementation and informative paper about the Sound System that was established in Viral Impact. The document's main

purpose is to inform the reader about how to implement and what approach I took to create the sound effects.

You can find all the sounds I have made in FMOD by opening using **FMOD version 2.02.30. Please** continue using this version or the project is going to explode. Or u can update FMOD in Unity then Update FMOD studio.

2. Music Design

Worlds	Style	Instrumentation
Whatsapp	Medieval	Strings, Flute, Piano, Harp
Instagram	Dreamy, Lofi	Piano, lofi synth, pad
Scam	Glitchy, unsettling	Piano, square synth, strings

3. Sound Design

3.1. Worlds and Environment

Worlds	Elements	Notes

Whatsapp	Birds, Cricket, Pink Noise, Water waves	Whatsapp is a forest with multiple swamp. You should focus on the serenity as well as the danger lurking in the woods.
Instagram	Synth, dreamy pad, Pink noise	Instagram is like a dream world, it should be chill with some lo-fi elements.
Scam World	Glitchy, dark noise, dark drone	Scam World is unsettling and need some sound effect to contribute and give context to that feeling

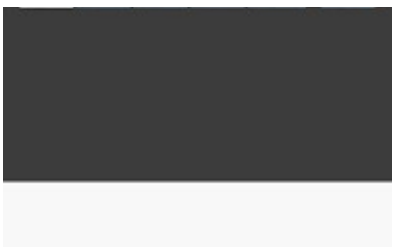
3.2. Gameplay Feedback

Categories	Sound Collections	Notes
Player	Slash Dash Hurt	All Implemented
UI	Text message notification Button Hovering Button Click Player death Player Spawn Wrong answer selection Right Answer Selection	Not Implemented Player Death Player Spawn Wrong Answer Selection Right Answer Selection

Enemies	Enemies getting damaged in general sound Flutter attack Billy Attack Chatram Attack Cappy Attack	All of these are implemented but you can change it if you want
Portal Ambience	Whatsapp Ambience Instagram Ambience Scam Ambience	Scam ambience hasn't been made but you can use the portal ambience as in-battle music. Although it wouldn't be very good for it.

4. Implementation

4.1. DAW



For this project, Ableton Live was used to create and edit sound effects and to build simple music / ambience layers. Serum 2 was used as the main synthesizer to design tonal sounds (UI tones, drones, pulses, sci-fi elements) and some instrument-like layers combine with using modified sound files from freesound.org and foley.

Workflow

1. **Sound creation**
    - Foley-style effects (impacts, footsteps, interactions) were created by layering recorded samples and processed audio.

- Synth-based effects (UI bleeps, power-up tones, ambience beds) were created in Serum 2 and bounced to audio.

2. **Editing and processing**
    - Common processing included EQ (remove rumble), compression (consistent level), saturation (extra character), and reverb/delay (space).
    - Sounds were exported “dry” when they needed to match the game space via FMOD/Unity (recommended), and exported with effects when the sound was meant to be self-contained.
3. **Export settings (recommended)**
    - Format: **WAV**
    - Sample rate: **48 kHz**
    - Bit depth: **24-bit** (or 16-bit if you prefer smaller files, but keep it consistent)
    - Mono vs Stereo:
        - **Mono** for 3D positional sounds (footsteps, hits, interactions)
        - **Stereo** for UI and music/ambience (usually non-positional)

File naming + folder structure
Use consistent names so FMOD events and Unity references stay clean.

Example:

- sfx_UI_ButtonClick_01.wav
- sfx_Player_Footstep_Grass_03.wav
- sfx_Env_DoorOpen_02.wav
- amb_ForestDay_Loop.wav
- music_Level1_Theme_Loop.wav

Looping notes
For ambience and music loops:

- Export loops as seamless as possible
- Test loops early in FMOD because loop clicks are easier to fix at the source than later.

4.2. FMOD

FMOD was used as the audio middleware to manage playback logic, parameters, mixing, and routing. Instead of triggering raw audio clips directly in Unity, Unity triggers **FMOD Events**, which handle variations, 3D settings, and volume control.

Why FMOD

- Centralised mixing (SFX/Music/Ambience busses)
- Easy randomisation (multiple takes, pitch/volume variation)
  Parameters for gameplay-driven sound changes (intensity, distance, surface type, etc.)
- Snapshot support for pausing, underwater, low health, menus, etc.

Project structure

- **Events**
  - event:/Music/Music/...
  - event:/Player/Attack/...
  - event:/UI/event:/...
- **VCA**
  - Master
    - Music
    - SFX
      - UI
      - Player
      - Environment

Typical event design

- **Single one-shot**: one audio file, usually UI clicks or simple interactions.
- **Multi-sound with randomisation**: multiple similar files (e.g., 8 footsteps) using a Multi Instrument with:
  - Random selection
  - Pitch randomisation (small range)
  - Volume randomisation (small range)
- **3D events**
  - Set event to 3D

- Use appropriate min/max distance
- Export in mono for best 3D localisation

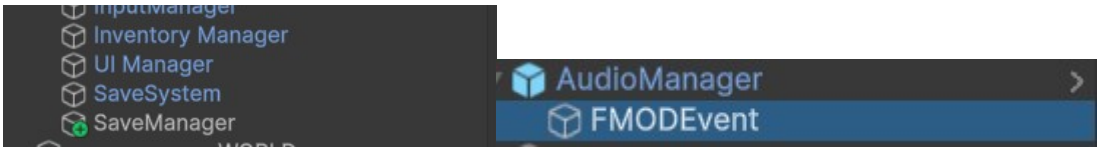
Parameters (examples)

- Value (int):Select different songs for different portals (Basic:0, Whatsapp: 1,etc.)

Build + bank workflow

1. Import exported WAVs into FMOD.
2. Assign them to events and configure randomisation/3D/looping.
3. Build banks and ensure the Unity project receives updated banks.
4. Keep event names stable once Unity scripts reference them. (If changes also change it in Unity)

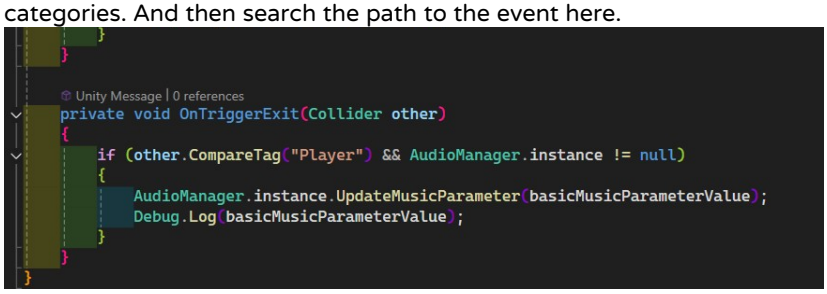
4.2. Audio Script Unity



We have the Audio Manager in Managers in every scene. This script focuses on containing all the volume of the sound effects as well as the function so we don't have to put Using UnityFMOD in every script. You just have to use the function here.

The FMODEvent Script contains every event inside of FMOD, you will have to add the event here if you want your sound effect to function

You have to add the same lines public EventReference sound {get; private set; } to its corresponding categories. And then search the path to the event here.

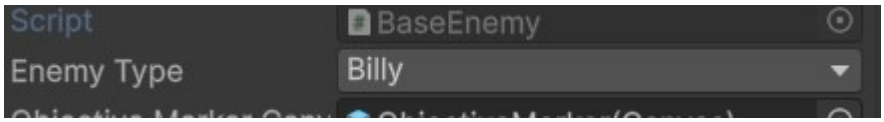


The FMODParameterUpdater Script is for changing the value of the parameter. This is used in the hub to change the portal music. You can use this script as a base and add more parameters in FMOD.

```
            case VolumeType.MASTER:
                volumeSlider.value = AudioManager.instance.masterVolume;
                break;
            case VolumeType.MUSIC:
                volumeSlider.value = AudioManager.instance.musicVolume;
                break;
            case VolumeType.SFX:
                volumeSlider.value = AudioManager.instance.SFXVolume;
                break;
        }
    }

    0 references
    public void OnSliderValueChanged()
    {
```

SliderVolume Script is for connecting the UI slider to the Audio Manager so the player can adjust the volume.



This BaseEnemy script is important to decide which sound that enemies make. There is an Enum here for each enemy and since every enemy has this script you have to add enum if you want to have new enemies, and also add the path to enemies sound effects.

5. Conclusion

I hope this document can help you have a deeper understanding of the sound design choice and how to implement the sound effects.