CMP407 Audio Programming Report

Callum Myers

For each of the **Foundational Requirements** in the [Assessment Brief](https://mylearningspace.abertay.ac.uk/d2l/le/content/38381/viewContent/938025/View), write a short paragraph explaining how you have implemented it within your project.

For your **Code Focus**, aim to write 3-4 paragraphs. These paragraphs should include:

* A broad overview of your implementation of that particular focus, concentrating on the individual elements involved and how they fit together. You may want to include a system diagram if that makes sense for your project.
* 2-4 examples of specific elements that you are particularly proud of, or that were challenging to implement, or that involved a complex implementation that requires a bit of explanation.

As this is a practical assignment the report should be fairly short. Aim for the main text of your report (i.e. everything up to the Credits section) to fill 3 pages A4, 4 pages at most. If you want to include images it’s fine if the report is a bit longer, but don’t go overboard.

# Using your project as an example of past work

Check this box if you’re happy for your project to be used as an example of past work on the module for future students of CMP407

# Video Evidence

<include a link to a (youtube) video of your work here>

# Foundational Requirements:

## Sound file playback

My project contains a variety of looping sounds and some oneshot sounds. The looping sounds include an outside ambient sound, a sound which plays when the player is nearby to, or in, the lake which is on the left of where they spawn in, each light in the house makes a faint buzzing sound when stood directly underneath and the radio plays a music track when turned on.

It also contains two oneshot sounds, which come from opening the door and activating the speaker. The door is just a simple track which plays once when opened and again when closed, and the speaker will randomly shuffle through a selection of music loops.

I have multiple sounds which may play concurrently at any point, primarily obvious when in the pond as the ambient garden sound and the sound of the water play together. There is also the door opening sound which can either play alongside the garden ambience sound when opened from the outside, or depending on where it is opened from when inside the building might be played with the sound of a light buzzing overhead.

## Spatial Localisation

For my spatial localisation I used Unreal Engine 5’s built-in localisation functions. I used binaural processing to ensure it would sound the best with headphones since this is usually what a player would be using if they wanted a full audio experience. By ensuring all my objects containing sound sources were placed appropriately in the world this made it easy to enable and use localisation. All sounds pan correctly with the player camera except for the audio of the garden ambience, which was a design choice because I didn’t feel the sound source of the wind should move with the player and it should be heard from all around.

I also enabled Unreal’s built in occlusion settings. I had originally planned to use a Steam Audio implementation however due to some issues I was having with the plugin I reverted to the built-in features. I set the low pass filter of my sound sources to a low value since this prevented sound from going through walls effectively, and I was able to set it up so that the sounds occluded correctly. The best example of this is with the ambient garden sound, where it is clearly audible from outside the house but once you enter the building and close the door you can no longer hear it.

## Attenuation

For attenuation I again used Unreal’s built in audio features. This was easy and quick to set up since all I had to do was ensure the inner radius, which is where the sound at full volume can be heard, and the falloff distance, the area outside of the inner radius where sound can be heard at a progressively lower volume as distance to the object increases.

One thing I did which I feel makes the project more realistic and that I am proud of was using a variety of attenuation shapes across my project. For example, inside of the building I changed my light sounds to use the cone shape instead of the sphere shape. This made it so that to hear the light buzzing you must stand directly underneath it, as opposed to how it was originally where it used a sphere shape and was less precise on where you could stand to hear it.

## Reverb

My project contains two reverb zones: inside of the house and when the player enters the small lake. To set these up I used Unreal’s “Audio Gameplay Volume” plugin. I then used the underwater reverb preset applied to the volume in the lake, and the large room preset for the house. I increased the gain and decay time in the underwater preset which helped to emphasise the effect, since increasing the decay time made it sound more muffled and increasing the gain made the effect affect the sound more vividly.

## Compressed File Formats

After looking into Unreal’s audio file formats I found that the six main type settings I could select from were: Bink Audio, ADPCM, PCM, Opus, Platform-Specific and RAD Audio. By default, Unreal will import sounds using Bink, since it is generally good enough for quick implementation with a good compression to sound ratio. However, after further research I decided to be more specific with my sound file choices:

* For my ambient looping sounds (the garden, underwater and light sounds), I used Bink because I was not too worried about the sounds since they are just background, and the user will not be too focused on them.
* For my music tracks (the speaker and radio) I used Opus because it has a very good compression to quality ratio, which allows the music to stay with good quality while also having slight compression. This would have been best for the ambient sounds too however due to its’ slightly higher CPU usage I didn’t want the project to be too resource intensive therefore I chose to prioritise the music tracks because they are more likely to engage the player.
* For my door I used PCM, because whilst it is the most resource intensive (large file sizes and high memory usage) it has the best sound quality. I was not too worried about the resource usage for this sound because it is a short sound and only plays very occasionally, so I would rather have the best fidelity possible.

## Sound Files and Mix Quality

Most of the files which I used were already well mixed however I checked each in Audacity just to make sure. I first normalized all of them to -6.0dB to ensure they were a mostly consistent volume to begin with. After importing them into the project, I adjusted the volume multiplier of each sound source component within Unreal Engine until all sounds seemed to be at a reasonable level in comparison to each other. I also made sure to trim off any extra silence which came about. This was particularly important when I was implementing the songs for the speaker, since I wanted a fade in and fade out effect, which lead to some silences at the beginning and end of files so I removed the silent sections to ensure it wouldn’t mess with the audio timing.

# Code Focus: **Complex soundscape**

The first, and in my opinion most impressive, feature in my project is my focus feature. This feature means that when the player plays the radio, all other sounds are muted, and the speaker will do the same. To implement this, I first added any ambient sounds, such as the wind and lights, to a sound class called AmbientSoundClass. Then when the player focuses on one of the objects, this sound class’s volume multiplier is set to 0, muting any sounds attached to it. When the player stops the focused sound from playing again, this volume multiplier it reset to 1. I also implemented a function in the Blueprint for both the radio and speaker which would allow it to mute its’ own volume multiplier from another Blueprint. I did this because I needed a way to mute them individually since they could not be added to the sound class, or they would end up muting themselves when activated as well. Instead, the radio will call the speaker’s volume adjustment function to match it to the volume of ambient, and the speaker will do the same with the radio.

The next feature I am proud of is my song shuffling feature on the speaker.

A screenshot of a video game

Description automatically generatedA screenshot of a computer

Description automatically generated

This is the function which is called when the song finishes playing on the speaker, which is called using the “On Audio Finished” node attached to the AudioComponent of the speaker Blueprint. In order to achieve this I added all the sound waves for my speaker songs to an array, and when the current song finishes the program will randomly select a number between 0 and the length of the array, then set the active sound property of the AudioComponent to the soundwave in that position of the array.

Overall I am happy with my project. I am happy that I managed to implement the focusing feature as it caused me a lot of trouble until I found out about the sound class functions available in Unreal Blueprints. If I had more time to work on this project I would try to add some procedural sounds to it. In particular, I would’ve liked to add procedural wind, water and music for the speaker as I feel this would have made the audio feel a lot more diverse and interesting.

# Credits

As well as the links listed below, I also used audio and models from the starter content available on Unreal Engine 5, and I collected some sounds using Soundly. The full names of the Soundly sounds are:

Ambience, Public Place, Botanic Garden, City Traffic Background, Birds 02 SND53689

Doors, Creak, Metal, Squeak, Large, Long SND0671

Water, Wave, Ocean, Beach Waves, Medium, Lapping SND0009

<https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-5-documentation>

<https://www.turbosquid.com/AssetManager/Index.cfm?stgAction=getFiles&subAction=Download&intID=1836802&intType=3&csrf=DED2595AC572F0718E42EF8DD8F8CED6382001AD&showDownload=1&s=1>

<https://dev.epicgames.com/community/learning/tutorials/vEP2/how-to-make-a-character-controller-in-unreal-engine-5-for-beginners>

<https://www.turbosquid.com/3d-models/vintage-soviet-radio-rodina-3d-model-1636852>

<https://www.turbosquid.com/3d-models/old-wooden-table1-lowpoy-pbr-3d-model-2278389>

<https://www.turbosquid.com/3d-models/old-wooden-chair-lowpoly-pbr3-3d-2279653>

<https://slooply.com>

# Generative AI Acknowledgement

If you have used any generative AI tools in the creation of your project and report, please list them here. Please include:

1. The name and version of the tool you used
2. The URL for the tool
3. Where and how you used it within your submission
4. Any prompts you used

# Rubric

Similar to what we did with the week 11 presentations, put an **X** in the rubric for each category, based on your own final assessment of your project. I’ll use this to get a sense for your understanding of the module content. If you’ve got a good handle on the module I expect your rubric will look very similar to mine; if you’ve maybe misunderstood some key concepts then our rubrics may look different, and this will help me focus on those areas in my feedback.

You may want to refer to the [full rubric](https://mylearningspace.abertay.ac.uk/d2l/le/content/38381/viewContent/948777/View) and [rubric FAQ](https://mylearningspace.abertay.ac.uk/d2l/le/content/38381/viewContent/948792/View) on MLS.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Application Functionality** | **Code Quality** | **Practical Audio Knowledge** | **Written Audio Knowledge** | **Reflection/Evaluation** |
| **A** |  |  |  |  | X |
| **B** | X | X | X | X |  |
| **C** |  |  |  |  |  |
| **D** |  |  |  |  |  |
| **F** |  |  |  |  |  |