CMP407 AUDIO PROGRAMMING REPORT

MICHAEL ENNIS 1902141

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

https://youtu.be/wPY-GnpSUS4

FOUNDATIONAL REQUIREMENTS:

SOUND FILE PLAYBACK

Looping Sounds – The dynamic music loops consistently throughout the level, inside of a music switch container. The music is updated using WWise states, when the 'SetChased' or 'SetHidden' events are posted. The heartbeat sound effect also loops when the player is low on health, responding dynamically to an RTPC value. The heartbeat stops looping when the player regains health.

One Shot Sounds – The player plays footstep sounds at specific moments on the running animation sequence. These footsteps utilize a WWise switch, depending on the actor tag assigned to the ground surface. The footsteps are segmented into three different random containers for water, grass and metal sounds. The turret AI plays a sound when firing and the projectile itself plays an impact sound, utilizing a WWise switch, depending on if it collides with an actor or a wall. The projectile also plays a pass-by sound effect if it misses the player. The chasing AI plays an explosion sound effect if it collides with a projectile or the player.

Concurrent Playback – All of the sounds play appropriately alongside the dynamic music. All the sounds and music tracks have been normalized to ensure that the audio plays at a similar volume to one another.

SPATIAL LOCALISATION

Spatial localisation is present in several key audio effects including the AI turret firing projectiles, the projectile itself passing by the player when dodged and when the projectile hits a wall. It is most noticeable in the pass-by effect as the sound will pan appropriately based on the direction the player dodged. Spatial localisation is also present on the explosion effect when an enemy is destroyed and is implemented by enabling 3D spatialization within the "Positioning" tab within WWise to "Position + Orientation".

A low pass filter is applied to the explosion and projectile firing sounds depending on the value of the 'CameraDirection' RTPC, which is updated based on the angle between the camera direction and the sound source. If the camera is facing away, a low pass filter is applied to muffle the sound.

ATTENUATION

Attenuation is present in several key sounds, including; when the AI turret fires projectiles, when the projectiles miss the player and on projectile impact. It is also utilized when an enemy plays an explosion sound. Attenuation was implemented using different 'AttenuationSets' within then 'ShareSets' section of the project explorer. This allows for an individual attenuation set to apply to multiple sound objects simultaneously, without having to copy the attenuation settings from one object to another.

REVERB

Reverb is present on the footstep sounds of the player in the starting area. This was implemented using a reverb zone that applies an auxiliary bus that the footstep sounds route through before being added to the master audio bus. The auxiliary bus applies a specific type of audio effect to the footstep sounds which changes how the sound is heard by the player. The auxiliary bus uses 'RoomMedium' preset to modify the sounds. The footsteps sound in WWise set the "Use game-defined aux sends" to allow the auxiliary bus to be allocated based on the reverb zone, rather than in WWise itself.

COMPRESSED FILE FORMATS

Originally the files were imported into WWise in the .wav format, completely uncompressed. The explosion sound and music tracks are compressed in WWise using the Vorbis, to lower the file size. The other sounds within the soundscape are uncompressed in the PCM format, as they are either highly repetitive or loop consistently, as such, repeatedly decompressing these sounds could require a large amount of processing power. These include the projectile firing, projectile pass-by, projectile impact, footsteps and heartbeat sounds.

The WEM Opus format could potentially be used in place of Vorbis, as it compresses files with better quality, however the processing power required to decompress this format is even higher, as such this format was not used for any of the sounds or music tracks in the soundscape.

SOUND FILES AND MIX QUALITY

All sound files were edited with Audacity first before being imported into WWise. This was done to remove any extraneous silence at the beginning or end, lowering the size of the sound file in the process. Also, any clicks were removed by applying "Fade In / Fade Out" features at the start and end of the sound files. The files were also normalized to -6 decibels to ensure a consistent volume across the soundscape.

The dynamic music loops were tested together as one music track within the 'SoundTrap' online music creation tool to ensure that they were cohesive when layered on top of each other.

CODE FOCUS: DYNAMIC MUSIC

The focus of this project was primarily on dynamic music, to showcase smooth transitioning and vertical layering based on AI gameplay mechanics. There are also a few overlaps with the sophisticated soundscape management focus, as the projectile pass-by effect and the other sounds within the level all contribute to a more well-rounded experience, allowing the gameplay mechanics to effectively showcase different audio techniques.

The key features implemented in the project include dynamic music transitioning, vertical layering of instrument tracks, projectile pass-by effect when the player dodges and the interpolating heartbeat mechanic using audio busses.

Dynamic Music Transitioning

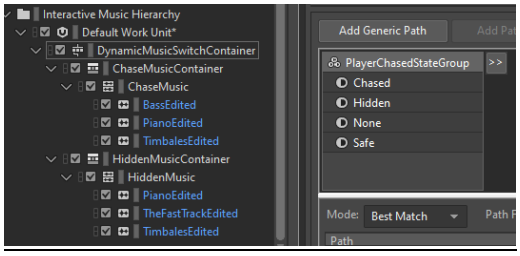
The dynamic music responds directly to AI, using WWise states. There are two types of AI that exist within the soundscape; a turret that fires projectiles and a hovercraft that chases the player. There are two different music types; 'Chase' and 'Hidden', that are stored within music containers that are children of the 'DynamicMusic' switch container. This container swaps the played music based on a WWise state, transitioning when there are any state changes.

The dynamic music state begins in the "Hidden" state, playing music that represents the relative safety of the player. When they get spotted by either of the AI types however, then the event 'SetChased' is fired, causing it to switch to 'Chased'. If the player manages to escape from the field of view of the AI, then the music transitions back to the "Hidden" state, after the 'SetHidden' event is fired. There is a short fade out/in that occurs when the music transitions, enabled within WWise under the 'transitions' tab of the music switch container. This allows for smooth transitioning, as the music track interpolates between the old and new music containers, causing the music to feel more fluid.

There is a short delay before posting a state transition event that ensures the transition feels more immersive and natural, as if the states transition too frequently, the transition could feel quite jarring to the player. Also, this ensures that if the player runs away from an enemy, becomes unspotted, but then runs straight into the line of sight of a new one, the music stays in the combat state, or similarly if an enemy spots the player for only a brief moment, the game will still play the hidden music, as the enemy AI barely had any time to react to the player.

The chasing gameplay mechanic is implemented using the "AudioEnemySubsystem.h" class. This class informs the player audio component when any enemy is chasing them and subsequently when they are safe. This utilizes an event-driven approach, informing the player and any other listeners when a new enemy is registered and when an old enemy is unregistered with the subsystem.

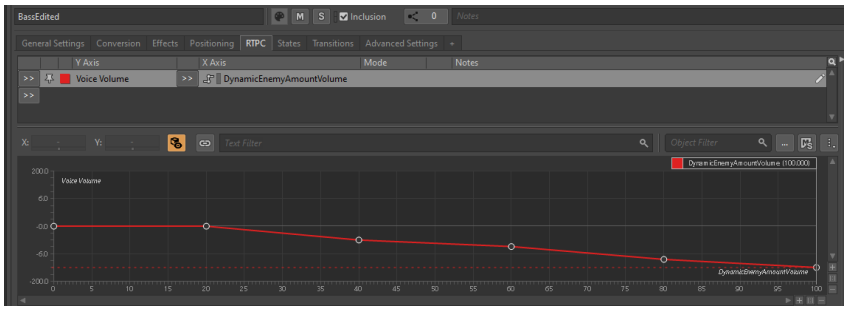
Project explorer showcasing 'DynamicMusicSwitchContainer' and states used within it



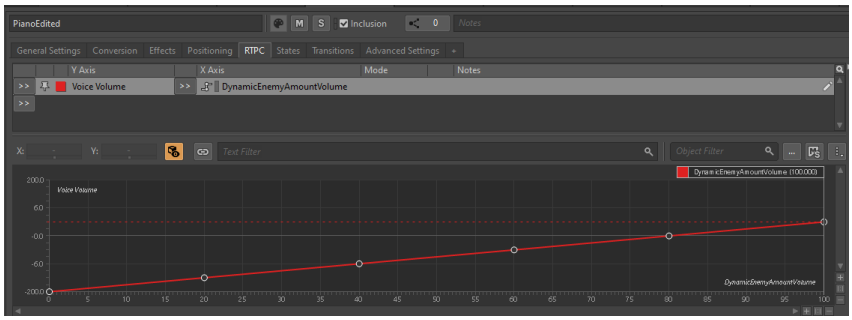
Dynamic Music Vertical Layering

Vertical layering is utilized when the player is being chased by enemies. As the number of enemies chasing the player increases, the 'PianoEdited' and 'TimbalesEdited' tracks increases in volume, while simultaneously the main track 'BassEdited' lowers. This is implemented using the RTPC "DynamicEnemyAmountVolume". This vertical layering occurs even when the player is in the 'Hidden' state, as this adds a certain instant reactive element to the dynamic music, without fully transitioning to the 'Chased' state. This allows for a more natural transition that adds to the player's immersion when traversing the soundscape.

'BassEdited' track lowering in volume as the 'DynamicEnemyAmountVolume' RTPC increases.



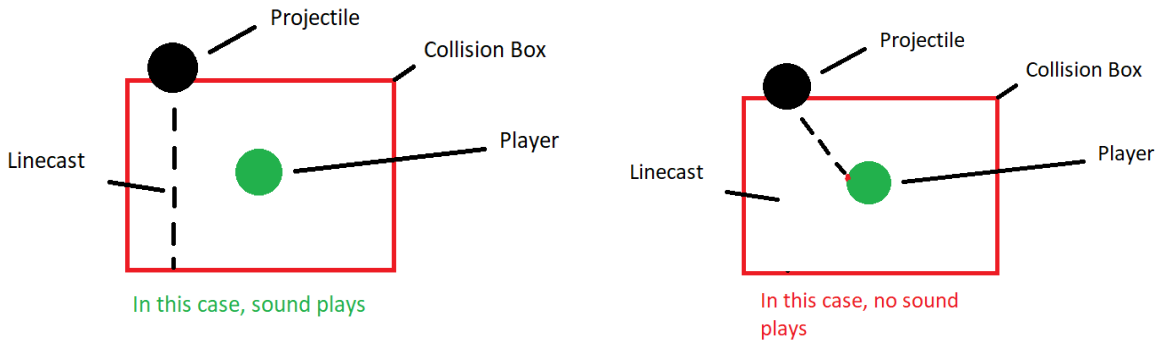
'PianoEdited' track increasing in volume as the 'DynamicEnemyAmountVolume' RTPC increases.



Projectile Pass-By Effect

When a projectile passes close by the player, but does not collide with them, the projectile plays a brief whizz sound effect. To ensure that the projectile does not collide with the player, it performs a line cast in the direction of the projectile, of the length of the 'BulletPassByBoxCollider' collision box, on collision with this box. If the line cast hits the player, then the whizz sound effect does not occur, otherwise it plays.

Diagram of projectile performing line cast to predict collision with player



Heartbeat Implementation

The heartbeat sound is played on a separate audio bus to everything else within the soundscape and begins completely silent. When the player loses more than half of their health, the RTPC 'HasBeenDamaged' becomes linked to the player's health, and interpolates alongside it. This RTPC increases the volume of the heartbeat audio bus, while simultaneously applying a low pass filter and lowering the volume of all other audio in the scene through the 'AffectedByHeartbeatAudioBus', as the player takes more damage.

Conclusion

Overall, the dynamic music system works relatively well when reacting to the AI that exist within the level. There were a few problems with implementing this however, as I noticed if the music switched during the fade in/fade out time (which is approximately 1 second) then the switch would fail to occur, causing the music to be stuck in the wrong state until another state change occurs. To fix this, I implemented the delay, which in turn also made the dynamic music transition sound more natural due to it not transitioning when brief interruptions occurred. To avoid this problem in the future, I should do the transition within the code itself instead of within WWise, using RTPC values and code-side interpolation.

The sounds within the level are relatively good, and I am especially proud of the projectile whizz sound. However, the actor projectile impact sound is far too quiet and could be easily missed with the volume of the other sounds. It was normalized to -6 dB alongside the other sounds, although it must have been a quiet sound to begin with. In the future, I should use a louder sound to compliment the soundscape more effectively. Also, the water footsteps seemed to last too long, causing them to sound very chaotic when running. I should source shorter water footsteps to avoid this issue.

Footsteps and bullet impacts were implemented using switches and checking the actor tag of the surface they hit. While this certainly works, errors can occur if the tag is spelled incorrectly when checking. To avoid this, ideally a 'UPhysicalMaterial' should be applied to the surfaces and checked with a linecast, as this would remove having to type the tag manually every time you want a surface to react a certain way, and instead you would just apply a new physical material.

CREDITS

Footstep Sounds

All footstep sounds were sourced from the Soundly Music Library, as well as Explosion sound. Full names referenced below-

Metal Footsteps: Footsteps, Human, Metal, Run, Isolated Steps 01
Water Footsteps: Footsteps, Human, Water, Run
Grass Footsteps: Footsteps, Human, Dirty Ground, Run
Explosion: Beeps, Lofi, Game, Retro, 8 Bit, Explosion 02

Music

Dynamic music tracks were sourced from loops found on SoundTrap. Full names of the loops below-

BassEdited: Memphis – Battery(Bass)
TimbalesEdited: World Percussion – Intense Timbales 04
TheFastTrackEdited: 23 – The Fast Trick(Instrumental)
PianoEdited: Piano – Octopus 2

Projectile Sounds

Projectile Firing, by Audionatics: https://freesound.org/s/133966/
Projectile Pass-by, by Audionatics: https://freesound.org/s/134024/
Projectile Impacts, by Mr Vibe SFX Youtube: https://www.youtube.com/watch?v=n49bNiFBvgU

Heartbeat Sound

Heartbeat, by user3331413297206, https://samplefocus.com/samples/heartbeat