Lecturer: Dr John McGowan & Dr Balandino Di Donato

Edinburgh Napier University CSI08104 – Intermediate Interactive Audio

Technical Notes

Max Torras Figuerola
31-12-2411

Words: 4442

Contents

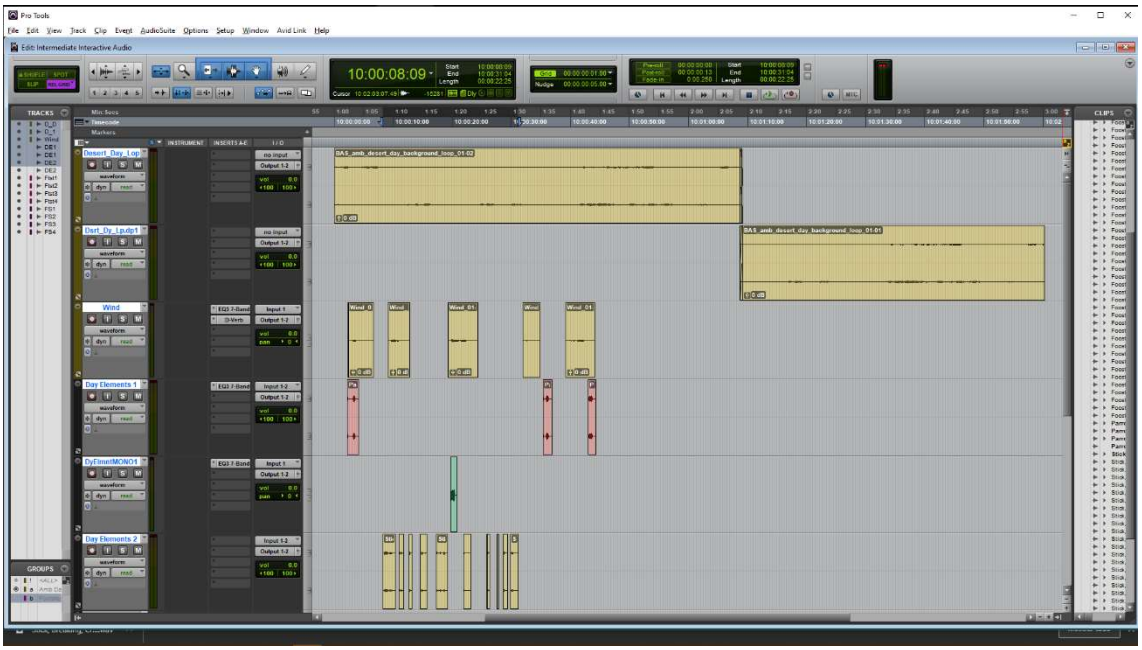
Introduction

This project is based on the integration of Wwise and Unity and designing interactive audio for a video game level. In this case, the level chosen is the Courtyard, from the free Unity assets. The final idea of this project was to create an immersive and realistic scene in a desert-based temple, where the character does not know where to go or what to do so that it can explore the whole world. Non-repetitive sounds are essential to make the game immersive and authentic, even if the player stays in the world long. The technical notes will cover everything done during the process of the project.

Library SFX

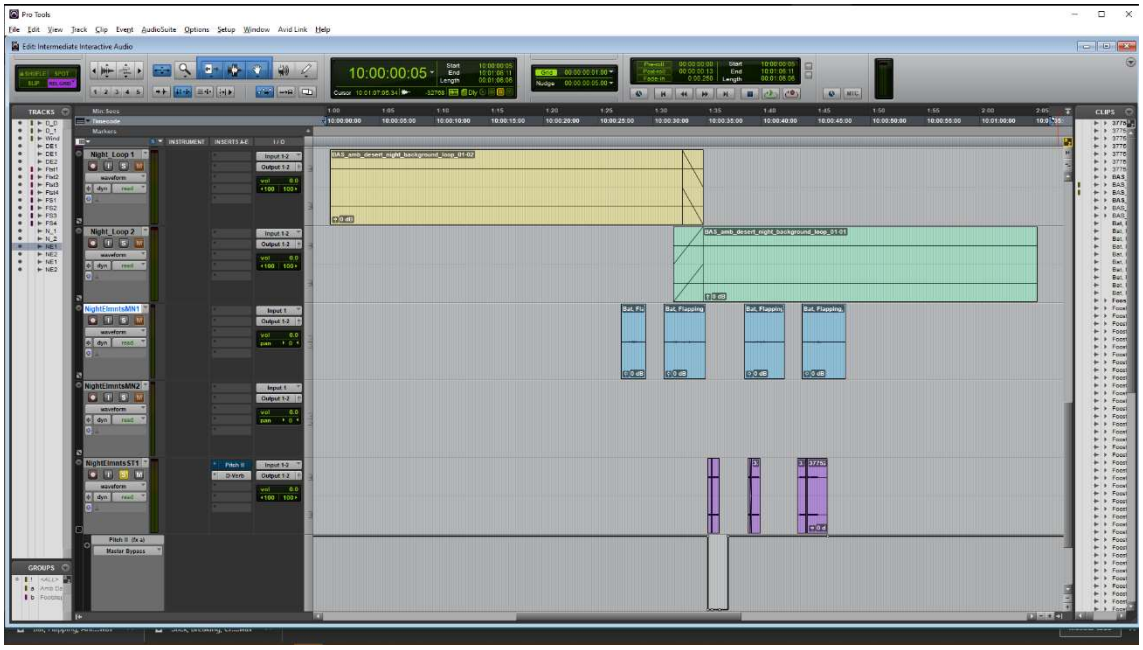
Most of the sounds that have been used are recorded by myself with an H5 Zoom Handy recorder. Others, though, have been extracted from freesounds.org or handed by module teachers. A list of all the sounds and the correspondent sources will be found at the end of the technical notes as an annex. With most of the sounds used, after sourcing them appropriately, they have been imported into a Pro Tools session to edit before importing them into the Wwise Project.

Day Ambience



For the day ambience it has been created a loop with an audio given by the module teacher. To do that, the clip has been split and swapped to match the loop. To complement the background loop, it has been created some wind noises with my own voice near to a Rode NT1A Rode Microphone. After that, for the day elements, it has been used some different African parrot sounds, leopard roaring sounds, and a crackling stick sound, which has been split to randomize them afterward. On the parrot sounds, an EQ has been used to remove low and high frequencies

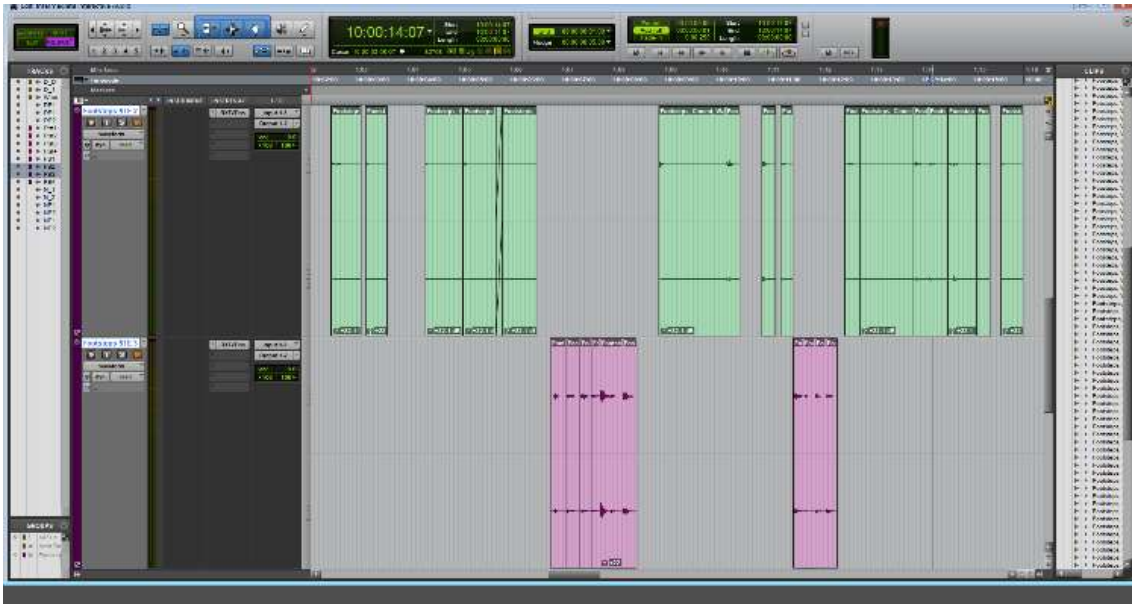
Night Ambience

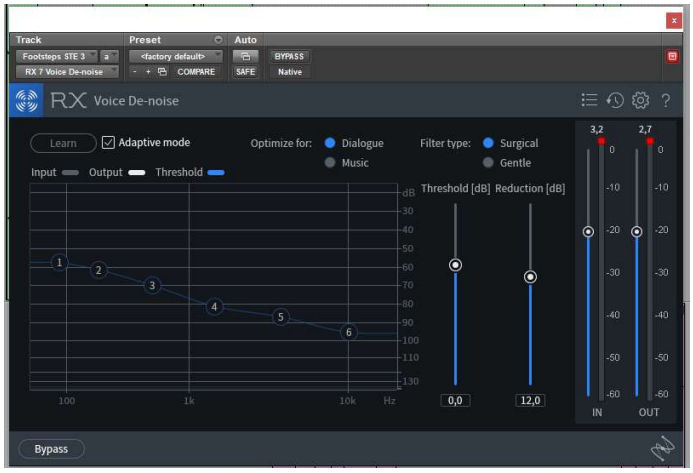


For the Night Ambience it has done approximately the same as for the day ambience, changing the looping clip. A part of that, two animal sounds have been used in the night ambience, which are a bat flapping and an owl. On the owl clips, it has been used a pitch shift add some variation in the ambience, and some reverb to make it more natural.

Pavement Footsteps

For the pavement footsteps, it has been recorded me at home, which there is a pavement floor that fits perfectly with the floor in the game. It has been recorded a long clip and then split in ProTools to create different sounds and make the footsteps more real inside the game with randomization. It has been added an iZotope plug-in named RX7 Voice De-Noise to remove unwanted noises. This plug-in has been used in all the footsteps.



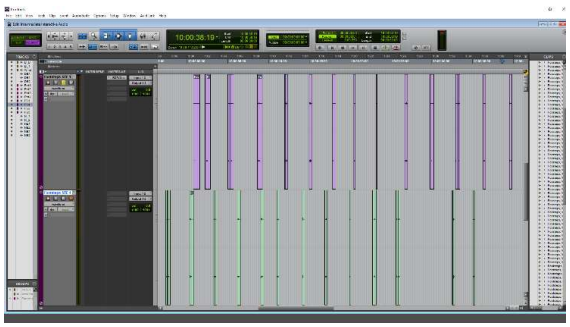


.

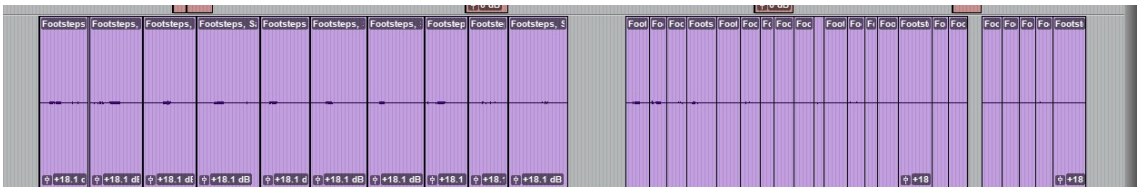
It has also been added some fades afterward to remove clipping and noises.

Water Footsteps

For the water footsteps, the procedure has been the same as with the pavement footsteps. The only thing that has changed is that these footsteps have been recorded inside a bath to simulate an artificial lake. It has also been added the iZotope plug-in with the same settings.



Sand Footsteps



The same method has been used for the footsteps on the sand. In this case, though, the audio is from freesound.org, and it is a long clip split and stretch for the running effect.

Artificial Lake Shore.

For the artificial lake waves, it has been tried to recreate the waves using a bath and creating some manual waves. No side effect has been implemented to this clip in ProTools.

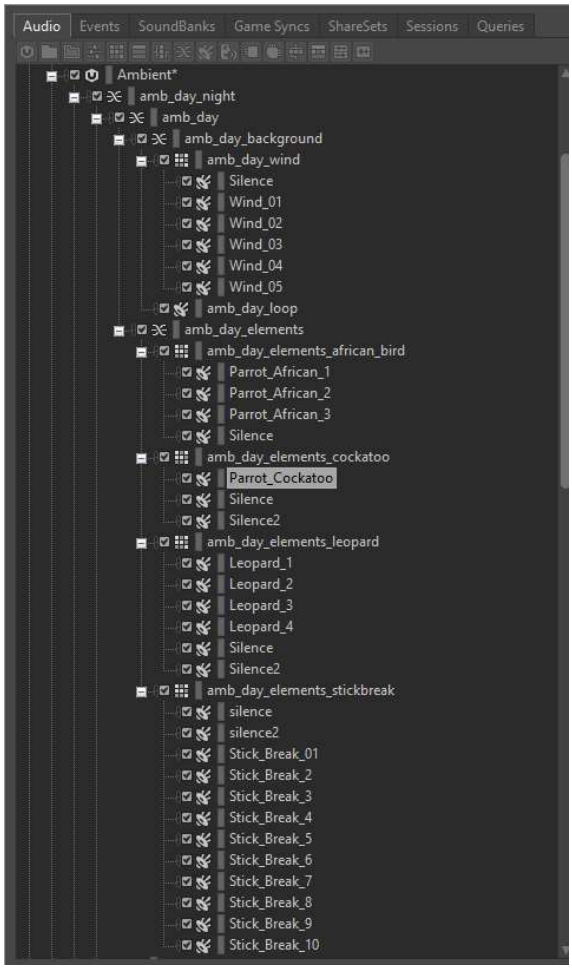
Dialogue

For the dialogue in this project, it has been decided not to record any dialogue properly, but instead, it has been added some goat sounds, simulating any chat. This is done because it has been thought that the game it is easy enough to figure it out what to do.

Sound FX

Finally, for the library SFX, it has been added the sound effects. In this project, there are a total of four sound effects for different purposes. Two of them are used for the collection system, one for tension using a heartbeat, and the last one is some fluorescent lights to create ambience in the game.

Environment Sound Design

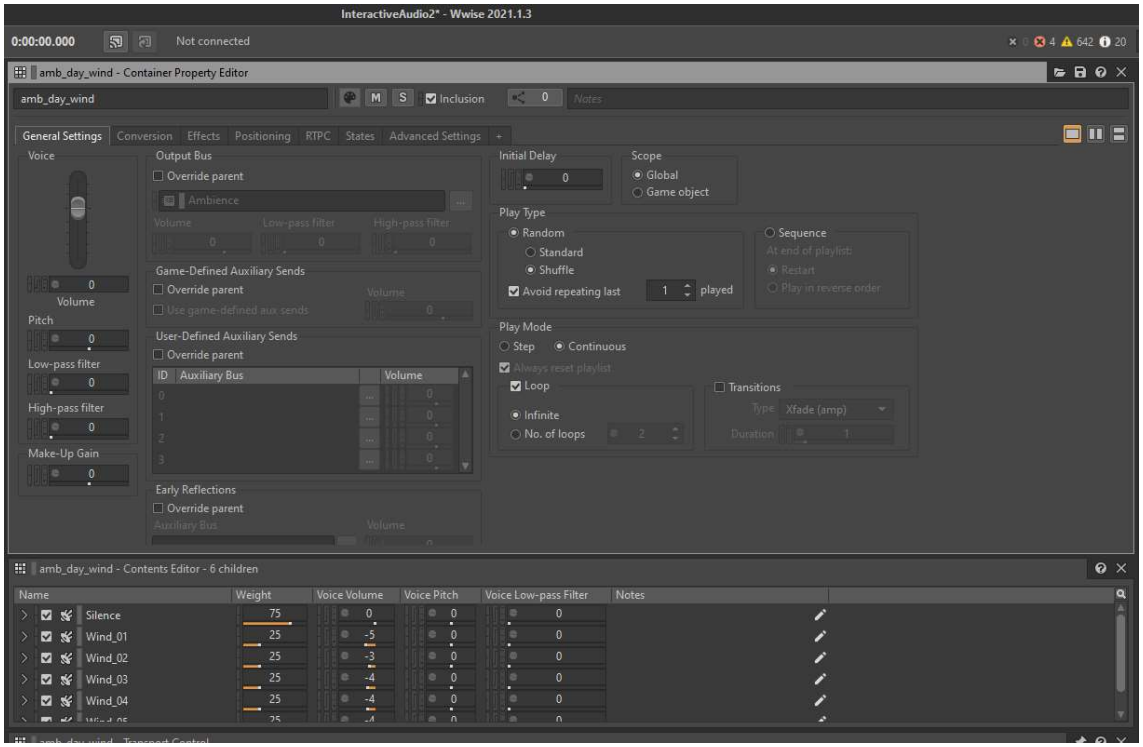


These are all the audio clips that have been used to create the day-night cycle in Wwise to create a perfect environment for the game. There are about 50 sounds, including animal sounds, background loops and other effects. Each animal and sound effect (wind and stick break) has been grouped up into a random container to randomize their sounds. After that, the day elements have been grouped again into a blend container so they can be played at the same time, and the same it has been done with the night elements. For the day background loop, the background audio has been blended with the wind random container and all ttogether has been integrated with the day elements to create the final day ambience.

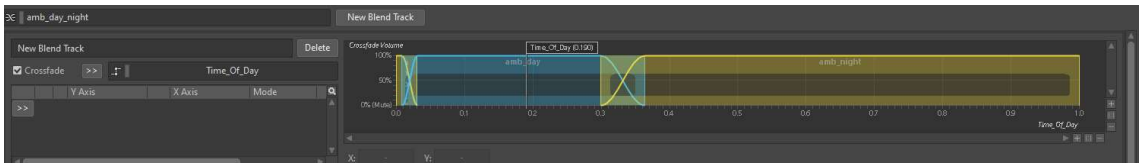
For the night ambience has been done pretty much the same but with different animals understanding that there are not the same animals during the day and the night. Some research has been done to find which animals can live in arid areas. To finish the night ambience, the background loop and all the elements blended together have been grouped again into a blender container.

To make the ambience sound continuously it has to be checked the Loop box in the background sounds, also checking the infinite option. On the blend and random containers, the play mode has to be continuous and looping infinitely. It

has to be checked in all the containers, so it will be no sound gaps while playing the game after a while. Also, some silences have been added in different random containers to make it not sound all together and space up all the sounds. An example of this is attached here, and all the random containers will be pretty much the same excepting the weight, which is the percentage of probability that sound will play, the voice volume, the voice pitch or the voice Low-pass filter.



To wrap up the environment, both day and night cycles have been grouped into another blend container named **amb_day_night**. To create a full cycle, it has been created a Game Parameter in the Game Syncs tab called **Time_Of_Day** in which the range is between 0 and 1 to match the numbers in Unity, where 0 (and 1) is supposed to be midnight. To create the cycle, it has been used a crossfade and tried to match the light cycle in unity with the Wwise sounds.



Jumping to Unity, to complete the day-night cycle both in Wwise and Unity, it has been modified the sun properties Script (will be added on an Annex) to add the Wwise blended sounds to match the light level and the sun position. The line added is the number 47 and it has been added this:

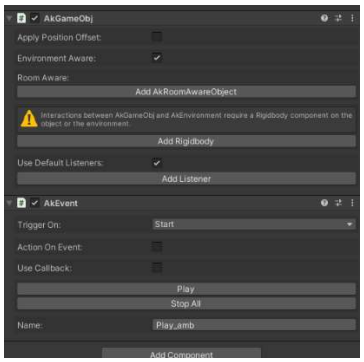
```
AkSoundEngine.SetRTPCValue("Time_Of_Day", time);
```

Event Sound Design

Play and Stop Events

Setting up any event in Wwise allows to trigger the container that hold the game sounds. If a sound wants to be played into the Unity engine, it must be activated or triggered by an event named play. It can also be created stop events which when they will be triggered the sound attached to it will stop playing. This can be useful when the character leaves an area and a sound wants to stop playing.

To be more organised and being able to select which sound wants to be played, it is suggested to create Work Units, and the number of events will depend on how many sounds have been created in the Wwise project, or how many are wanted to be into the game.

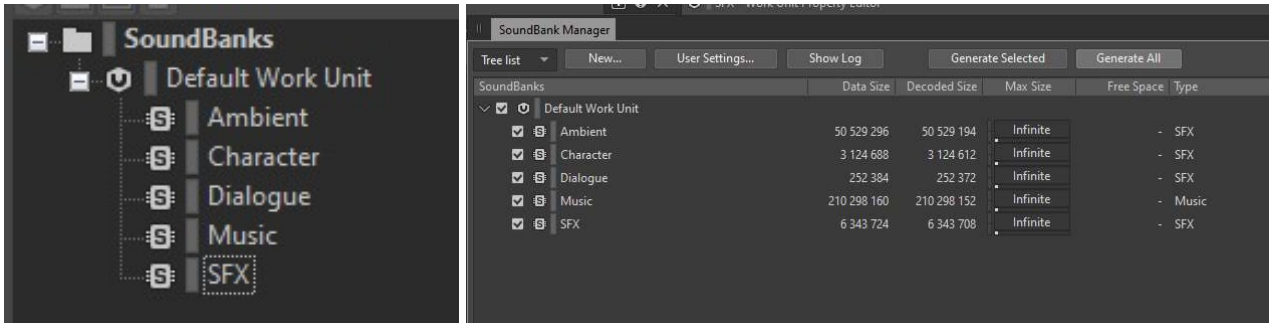


For implementing this into Unity, Ak Ambient scripts must be inserted and attached to the object desired. In the example below, it has been created an empty object named **Event** and inside it, the script has been added selecting, in this case, **Play_amb** and triggering it at the start, so when the game starts, the ambience sound will start playing regardless of the position of the character. For a sound being stop, it's the same procedure but adding a stop event.

SoundBanks

Once the events are being created, SoundBanks are essential to make the game work. SoundBanks should be named correctly in accordance with all the general elements that there are in the Wwise project and events must be inserted inside the SoundBanks so Unity will know what events can be played. To insert the events into the SoundBanks, it has to be double-clicked the SoundBank is wanted and drag the events desired into it.

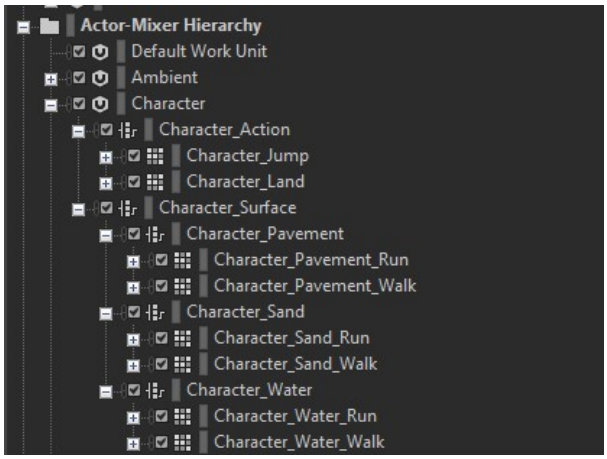
When all the play and stop events are linked with a SoundBanks, which can have more than one event inside of it, the project can be exported and linked directly with Unity. After saving the project, if Shift + B is pressed, a new window will pop-up and will let it be generated into Unity. This process must be done every time something new wants to be implemented in the game.

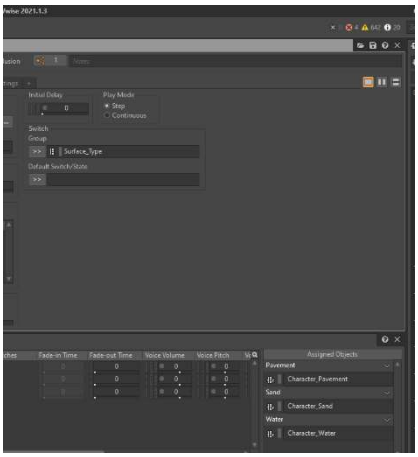


These SoundBanks are the most common to use because they will group all the general elements that it can be created into a game. Once the Generate All tab is clicked, a log status windows will pop-up and it will tell if there is any error.

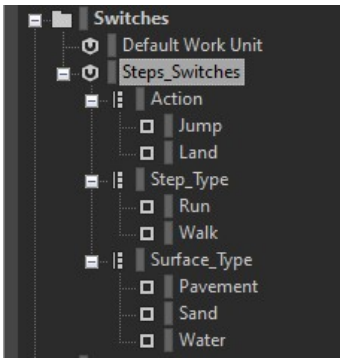
Switches and Footsteps

Footsteps and Switches are a very important part of the sound implementation and the first ones use the switch containers. It has to be a switch container for every kind of surface in the game, and inside those, one for run and another one for walk, if needed, which is the case of this game. On this project, it

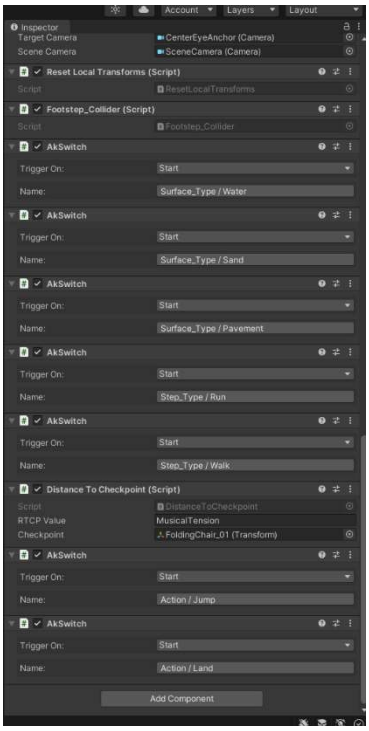




has also been implemented a new container aside of the **Character_Surface** named **Character_Action** which will contain and detect when the player jumps and lands on the floor. Inside the action container, there is two different random containers so it will sound different every time the player jumps or lands. It happens the same with the footsteps, which each random container has a good number of different footsteps to randomize and be more immersive.

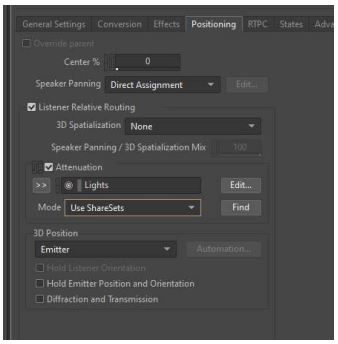


To create the proper Switches, must be head to the Game Syncs tab and under the Switches folder it must be created each of the changes that the game will notice. It is really important that each switch state is named appropriately for the different floors and surfaces to be easier in Unity with the script. Once all the switches are created, the objects have to be assigned to each group and then the containers have to be dragged properly.



In unity, each surface must be selected one by one and it has to be implemented a Script named Material Switch Controller which will let it change the surface the character is stepping on. If the name of the containers does not match the script names, they must be changed so everything can work. Once all the surfaces are linked correctly, it can be added a script into the FPSController object named **Footstep_Collider** which will detect when the character collides with the floor. Again, the name inside the script must be the same with the one in Wwise.

Attenuation and Permanent Sounds



Attenuation is a good feature to make the game look real. It will attenuate a sound over distance, and it can be used in lights, water, or any diegetic or diegetic sound. The sounds that are wanted to be attenuated, need a new parameter, in the ShareSets tab in Wwise and it has to be created an attenuation element. To link the attenuation feature into the sound, in the positioning tab of the sound desired, it can be selected the attenuation needed. If the distance wants to be edited, just editing it will let it change the distance or any other parameter.

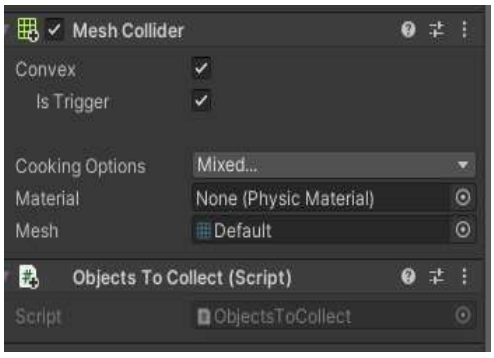
In unity, like in the surface, it has to be added to any object wanted to make a sound an Ambient Script selecting the correct play event. The same happens with the water, where another Ambient Script must be added and it will create the effect of the waves in the distance when the character is far away.

Triggered Sounds.

Another interesting element that can be created is to make a sound activate when is it triggered with the character entering a delimited zone. To make this work, a new script has to be added to the FPSController named Distance to Checkpoint and it will be needed one for every object that needs it. It can be chosen which object will have the central point, so when the character approaches this element, it will trigger it. It would be interesting to add a stop event so when the player leaves the area, it will trigger again, stopping the sound.

Collisions

Once all the ambience and character sounds are inserted and linked properly, it can be started the object collection system. Any item can be inserted as the object to collect and they can be found in the Asset Store for free. In this case, a Japanese coin has been used. In each coin it has been added a mesh collider so the player can touch the coin and trigger it to activate a script named Objects to Collect.



Once all the coins are imported, inside the canvas, which is what the player sees while playing the game, it can be added an empty object named **ObjectNum** and inside it, it can be added another script named Count Objects. This last script will show on the screen how many coins are left to pick and it will send the player to another scene if wanted when all the coins are collected.

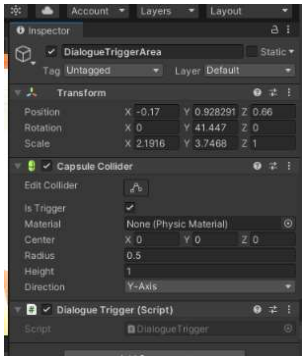
Another feature added is an NPC character that will activate the coin research and it will try to interact with the character.

Character

It has also been added an NPC character, in this case it is a Goat-Person character, which it makes the scene a little bit more alive and it will let the player interact with someone instead of playing alone. The character has been imported from Mixamo, a website where characters and animations can be downloaded for free.

Here, the NPC is just sitting on a chair having some subtle movements. When the character is in place, a capsule collider can be added and making it slightly bigger than the character and checking the trigger box will make trigger any sound when the player approach it, if it is added the correct Ambience Script. It can also be added a mesh collider so the player cannot go through the NPC.

Inside the capsule collider it can be added a script named Dialogue Trigger which will trigger the text named above with the information on how many coins are left to be pick up.



Event List

The nomenclature in the next list is the following one:

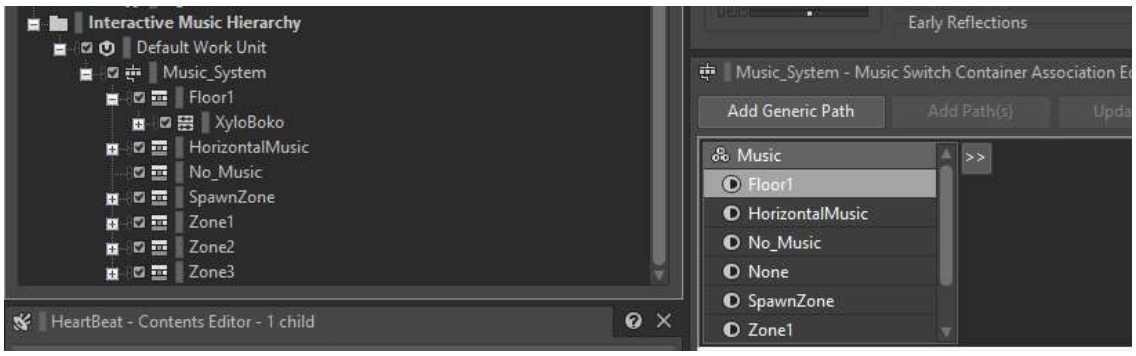
Event Name (Event type) – Container attached (Container type)

- Ambience
  - Play_amb (Play) - amb_day_night (Blender Container)
  - Play_waves (Play) - Waves (Sound SFX)
- Character
  - Play_Footstep (Play) Character_Surface (Switch Container)
  - Play_Jump (Play) - Character_Jump (Random Container)
  - Play_Land (Play) - Character_Land (Random Container)
- Dialogue
  - Play_Dialogue (Play) - GOAT (Random Container)
- Music
  - Play_Music (Play) - Music_System (Music Switch Container)
  - Stop_Music (Stop) -Music_System (Music Switch Container)
- SFX
  - Play_Heartbeat (Play) – HeartBeat (Sound SFX)

- Play_Lights (Play) – Lights (Sound SFX)
- Play_PickUp (Play) – Collect (Sound SFX)
- Play_End_Game – Congrats_UpBeat (Sound SFX)

Interactive Music

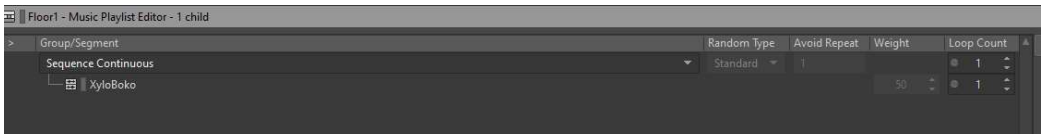
There are basically two different music approaches, vertical and horizontal, which will be explained in a moment. It is essential to create, in the Game Syncs tab, a state for each music container, and renamed appropriately. Once all the audios are imported, and the states are created, it can be linked each container with the state wanted pressing F10 to enter the Interactive Music Layout and there, the states can be seen and it can be added a path to the correspondent sound.

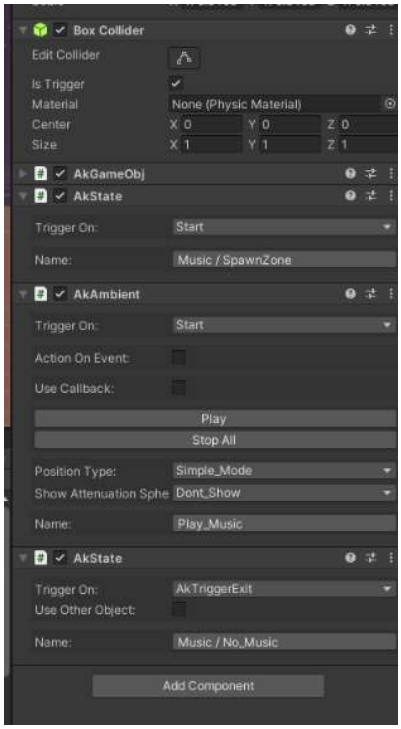


Vertical Music

Vertical Music means that all the tracks will play the same level no matter where the character is. It is useful to support the ambience. When the playlist containers are created, in each of it, it can be added one or more separate tracks which will have to be selected and imported inside the container.

Once the desired tracks are in, clicking the playlist container, and using the **Interactive Music Layout (F10),** it can be chosen between multiple options on how to reproduce the tracks. It can be either a sequence or random, and continuous and stepped. After choosing the option desired, the track has to be dragged under it so Wwise will know what to play. In this project, all the vertical music tracks have been extracted from royalty free websites like freemusicarchive.com. If it is wanted to add 2 or more tracks inside the same playlist container, it is as easier as just drag all the tracks wanted.





To integrate this into unity, it has to be created a game object named **Sound** and inside of it, it can be created as many empty objects as wanted, for as many music zones are desired.

In this case, it has been created 5 empty objects for the vertical music. In each of this it has been added a box collider, checking the trigger box, and resizing it to the zone desired. It has also need to be created an **AkState** script and selecting to trigger it on entering, it needs to be selected the proper state, or, in this case, the appropriated music container. In one of them, it has to be added an Ambient script to start running the music.

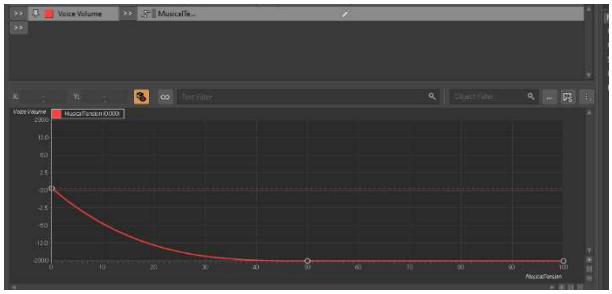
It can also be added another state script triggering it on exiting the zone, with a no music state. With all of this, it can be created as many zones as desired with as many tracks as wanted.

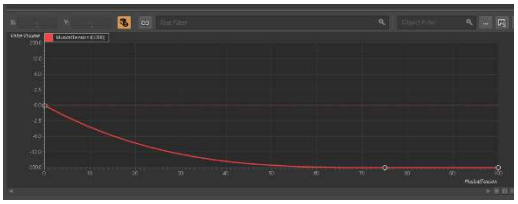
Horizontal Music

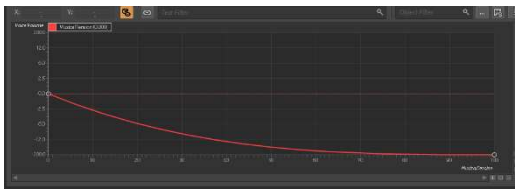
Horizontal music has a similar approach with vertical music but, in this case, the tracks can be complementary, meaning that they can be played one above the other. In this project, the tracks have been given by the module teacher, and there is a piano, a choir and a sitar.

A game parameter is needed to create this effect and it has been named Music_Tension with a range of 100. To implement this parameter, it just needs to be added in the Real Time Parameter Controller (RTPC) tab in each of the music segments. A graphic will pop-up and there it can be adjusted the distance, the line or any other parameter. To make it more realistic, instead of using a linear line, it has been used a logarithmic (Base 3).

Each instrument or track have a different starting point to they will start playing in a different distance from the character.







Once all the tracks are edited and can be exported into Unity, using the SoundBanks, another game object has to be created with the same, a box collider and a state script selecting the proper state and triggering on entering the zone.

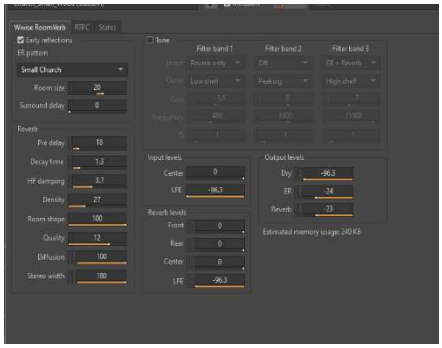
The different thing of the horizontal music is that after inserting the music into unity, it has to be created a new script named Distance to Checkpoint where it must be chosen the object where the music it is going to be in its peak. If there is more than one horizontal music track this new script has to be added again choosing the appropriate object.

Interactive Mix

RTPC

As mentioned before, RTPC stand for Real Time Parameter Controller, and what they do is control different parameters such distance, pitch or frequency and with an X-Y graphic, it can be chosen what to do, for example, if it is wanted to increase or decrease the volume as the character goes far or near from a specific object.

Reverb Zones



Wwise Engine also has a Master mixer where buses can be added. There are two kinds of buses available, which are the auxiliar bus, that is it used for the reverb zones, and the audio buses, that will be explained below. Auxiliar buses in this project have been used to create a reverb zone in specific chambers in the game. To create this effect, under the effect tab, it can be added a reverb plug-in integrated with Wwise, and it has been adjusted so it is not that subtle.

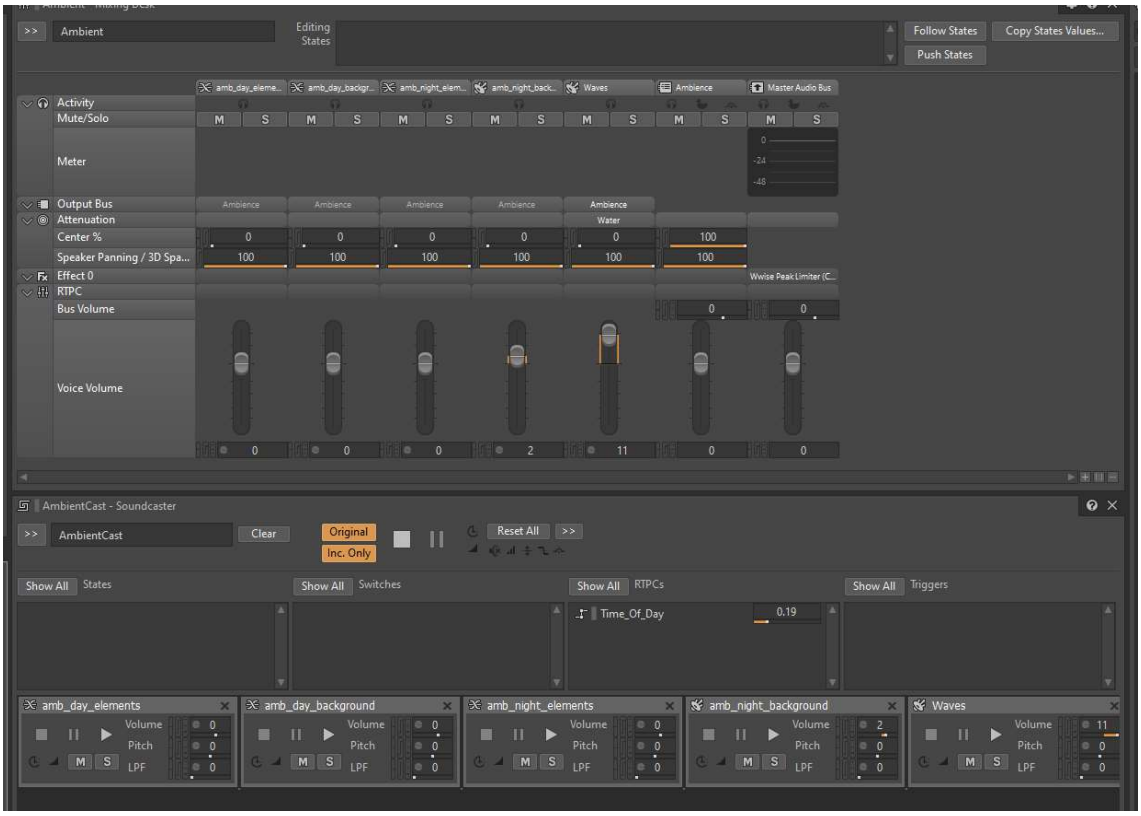
To implement this into Unity, each chamber already has a music zone box and inside these game objects, it has been added an Environment Script where it can be selected the correct auxiliary bus.

Audio Buses

In the same Master mixer folder, it can be created audio buses for each work unit if possible, and when selecting the most general container in each work unit, it can be chosen in the output bus, the audio bus desired.

When all the containers are inside an audio bus, it can be headed to the Mixing Layout (F8) where, even if it is slowly to do, it can be created as many mixing sessions as wanted. In each mixing session it has to be dragged all the containers wanted and there they can be adjusted like a mixing desk. After that, it can be done the same with a soundcaster where it will let play the sounds separately or all together.

In this project, it has been created six mixing desks, one for each work unit and a last one for all the audio buses to have the control over all of them.



In this project it has been used auto-ducking to minimize the ambience and the music sounds when the NPC is chatting.

Game Engine Integration

Scripts

All the scripts used for this project it has been given by the module teachers, and it will be an annex with all of them.

Video Captured Gameplay

The week before the deadline, a five-minutes video has been recorded to show all the features in the game. This video has been captured with OBS Recorder and the voice over it has been recorded using Pro Tools and a NT1-A Microphone. It has been a lot of problems with the recordings because the computer used is it not the best to do it. Even though the fps are not great, the video shows the work done. If after writing these technical notes, the video can be upgraded, it will be better in the blog below.

The blog is posted in my personal webpage where all the projects are shown. This is the link: https://max13torras.wixsite.com/website/intermediate-interactive-audio

Transcription

Welcome, my name is Max and this is my Intermediate Interactive Audio, integrating Wwise and Unity engines. When the game is started, it is near midnight, so we can hear a lot of animals who live in the desert. We appear in the top of a building, so we will have to go down somehow. Luckily, we can just jump down. When we arrive down, we can start hearing a heart beat and there is no sign of where it comes from. We will take care of that in a moment, but first we should talk with the character sitting in a chair. It tells us, in its language, to collect 5 coins. We do not know where are they, so we have to start exploring. While we explore, we can hear a lot of different sounds coming from various sounds. One of the most important sounds are the footsteps on the pavement, which will change if the character is running or walking. If we jump, we can hear some noises doing some effort, both jumping and landing. When we enter a chamber, the footsteps sounds will change to be with some reverb, making entering the chamber more realistic. We could find a coin here, so when we pick it up, a sound will pop telling us that we collected the coin. Some music will also be heard time by time, which is it supposed to be triggered when entering the chamber. When it is day time, we can explore outside, on the sand where we can hear a different footsteps sound, matching the proper surface. Outside, there is nothing especial, but we can see the cactus which will make the ambience sounds real because, as there is some greenery, animals can live somewhere. The ambience sounds are a mix of a background loop sounds and some animals sounds that will change during day and night. Making a whole round on the temple, will make us notice a coin at the beginning approximately, and also the heart beat again. If we go to the sound, we will see that it comes from near a chair, to make us understand that we are getting anxious. Another coin can be found on the stairs near the lake. If we want to enter the lake, which we must to collect another coin we can hear the last footsteps sounds when we walk on water. Supposedly, when we enter the lake, we should start hearing a mysterious song that will bring us the top of the pyramid, where there is a magical chair standing there. We do not know what is it, so we should better let it there. Finally, if we go on top and look around, we can see the last coin on the roof so we can try to make some parkour to get there.

First, though, it is important to point that all the artificial lights in the scene have their own sound, attenuating when the character goes further from them, and in every chamber, there is some reverb. As you can see, when we collect the last coin, we will be transported to another dimension, making and upbeat sound to congratulate that we collected all the coins. After finishing this project and recording the video there is some issues with the music that sometimes it does not trigger appropriately.

Annex 1. Library Sources

Clip Name	Source	Details	Location in Wwise	Edits	Mixing
Ambient Work Unit					
amb_wind_01	Rode NT1 - A mic, recorded myself	Wind simulation blowing the mic	amb_day_background/amb_day_wind	Weight: 25 Voice Volume: -5 dB	amb_day_background Voice Volume: -4dB
amb_wind_02		Wind simulation blowing the mic	amb_day_background/amb_day_wind	Weight: 25 Voice Volume: -3 dB	
amb_wind_03		Wind simulation blowing the mic	amb_day_background/amb_day_wind	Weight: 25 Voice Volume: -4 dB	
amb_wind_04		Wind simulation blowing the mic	amb_day_background/amb_day_wind	Weight: 25 Voice Volume: -4 dB	
amb_wind_05		Wind simulation blowing the mic	amb_day_background/amb_day_wind	Weight: 25 Voice Volume: -4 dB	
amb_day_loop	Given by module teacher	Arid ambience sound	amb_day_background		
Parrot_African_1		Bird Whistle	amb_day_elements/amb_day_elements_african_bird	Weight: 30 Voice Volume: -11 dB	amb_day_elements Voice Volume: -9 dB
Parrot_African_2		Bird Whistle	amb_day_elements/amb_day_elements_african_bird	Weight: 65 Voice Volume: -8 dB	
Parrot_African_3		Bird Whistle	amb_day_elements/amb_day_elements_african_bird	Weight: 50 Voice Volume: -13 dB	
Parrot_Cockatoo		Parrot cuacking	amb_day_elements/amb_day_elements_cockatoo	Weight: 50 Voice Volume: -13 dB	
Leopard_1		Leopard Roaring	amb_day_elements/amb_day_elements_leopard	Weight: 9	
Leopard_2		Leopard Roaring	amb_day_elements/amb_day_elements_leopard	Weight: 11	
Leopard_3		Leopard Roaring	amb_day_elements/amb_day_elements_leopard	Weight: 10	
Leopard_4		Leopard Roaring	amb_day_elements/amb_day_elements_leopard	Weight: 23	
Stick_Break_1	FreeSound.org	Stick Breaking	amb_day_elements/amb_day_elements_stickbreak		
Stick_Break_2		Stick Breaking	amb_day_elements/amb_day_elements_stickbreak		
Stick_Break_3		Stick Breaking	amb_day_elements/amb_day_elements_stickbreak		
Stick_Break_4		Stick Breaking	amb_day_elements/amb_day_elements_stickbreak		
Stick_Break_5		Stick Breaking	amb_day_elements/amb_day_elements_stickbreak		
Stick_Break_6		Stick Breaking	amb_day_elements/amb_day_elements_stickbreak		
Stick_Break_7		Stick Breaking	amb_day_elements/amb_day_elements_stickbreak		
Stick_Break_8		Stick Breaking	amb_day_elements/amb_day_elements_stickbreak		
Stick_Break_9		Stick Breaking	amb_day_elements/amb_day_elements_stickbreak		
Stick_Break_10		Stick Breaking	amb_day_elements/amb_day_elements_stickbreak		
Bat_1	Library Sounds	Bat flapping its wings	amb_night_elements/amb_night_elements_bat		amb_night_elements Voice Volume: -8 dB
Bat_2		Bat flapping its wings	amb_night_elements/amb_night_elements_bat		
Bat_3		Bat flapping its wings	amb_night_elements/amb_night_elements_bat		
Bat_4		Bat flapping its wings	amb_night_elements/amb_night_elements_bat		
BatFlap_01		Bat flapping its wings slower	amb_night_elements/amb_night_elements_bat	Weight: 46	
BatFlap_02		Bat flapping its wings slower	amb_night_elements/amb_night_elements_bat	Weight: 45	
BatFlap_03		Bat flapping its wings slower	amb_night_elements/amb_night_elements_bat	Weight: 47	
BatFlap_04		Bat flapping its wings slower	amb_night_elements/amb_night_elements_bat	Weight: 48	
Owl_1	Given by the module teachers and edited on Pro Tools	Owl Hooting	amb_night_elements/amb_night_elements_owl	Weight: 10	
Owl_2		Parrot cuacking	amb_night_elements/amb_night_elements_owl	Weight: 9 Voice Pitch: 300	
Owl_3		Leopard Roaring	amb_night_elements/amb_night_elements_owl	Weight: 10 Voice Pitch: -300	
Owl_4		Leopard Roaring	amb_night_elements/amb_night_elements_owl	Weight: 36	
Owl_5		Leopard Roaring	amb_night_elements/amb_night_elements_owl	Weight: 39	
Owl_6		Leopard Roaring	amb_night_elements/amb_night_elements_owl	Weight: 36	
amb_night_background	Given by the module teachers	Arid nightly background sounds	amb_night	Voice Volume: -4 dB	amb_night_background Voice Volume: -4 dB
Waves		Small waves crushing	Ambient Work Unit		Waves

Clip Name	Source	Details	Location in Wwise	Edits	Mixing
Character Work Unit					
Jump_01	H2n Zoom recording my voice	Simulating a jump effort sound	Character_Action/Character_Jump		Voice Volume: 4 dB
Jump_02					
Jump_03					
Jump_04					
Land_01		Simulating when you land from a place	Character_Action/Character_Land		
Land_02					
Land_03					
Land_04					
Land_05					
RunPavement_1	H5 Handy Recorder at my house in Sabadell	Footsteps in pavement	Character_Pavement/Character_Pavement_Run		
RunPavement_2					
RunPavement_3					
RunPavement_4					
RunPavement_5					
RunPavement_6					
RunPavement_7					
RunPavement_8					
RunPavement_9					
RunPavement_10					
WalkPavement_1			Character_Pavement/Character_Pavement_Walk		
WalkPavement_2					
WalkPavement_3					
WalkPavement_4					
WalkPavement_5					
WalkPavement_6					
WalkPavement_7					
WalkPavement_8					
WalkPavement_9					
WalkPavement_10					
WalkPavement_11					
WalkPavement_12					
WalkPavement_13					
WalkPavement_14					
WalkPavement_15					
RunSand_1	H5 Handy Recorder in a beach	Footsteps in the sand	Character_Sand/Character_Sand_Run		Voice Volume: 8 dB
RunSand_2					
RunSand_3					
RunSand_4					
RunSand_5					
RunSand_6					
RunSand_7					
RunSand_8					
RunSand_9					
RunSand_10					
RunSand_11					
RunSand_12					
WalkSand_1			Character_Sand/Character_Sand_Walk		Voice Volume: 6 dB
WalkSand_2					
WalkSand_3					
WalkSand_4					
WalkSand_5					
WalkSand_6					
WalkSand_7					
WalkSand_8					
WalkSand_9					
WalkSand_10					
RunWater_1	H5 Handy Recorded in my bath	Footsteps in the water	Character_Water/Character_Water_Run		
RunWater_2					
RunWater_3					
RunWater_4					
RunWater_5					
RunWater_6					
RunWater_7					
RunWater_8					
WalkWater_1			Character_Water/Character_Water_Walk		
WalkWater_2					
WalkWater_3					
WalkWater_4					
WalkWater_5					
WalkWater_6					
WalkWater_7					
WalkWater_8					
WalkWater_9					

Clip Name	Source	Details	Location in Wwise	Edits	Mixing	RTPC
Music Work Unit						
1_Choir	Given by Module teachers	Choir Singing	Music_System/ Horizontal_Music/ AmbientSegment	Volume: -4 dB	Volume: -14 dB Also there is three game intensity parameters not used.	Music_Tension: Voice Volume going from +0,0 dB to -200 dB in 50 units distance
2_Sitar		Sitar playing				Music_Tension: Voice Volume going from +0,0 dB to -200 dB in 75 units distance
3_BigPno		Piano playing				Music_Tension: Voice Volume going from +0,0 dB to -200 dB in 100 units distance
XyloBoko	Royalty Free music	African Generic Music	Music_System/ Floor1	Volume: -8 dB		
SpawnZone		Calm music but with an african genre	Music_System/SpawnZone	Volume: +3 dB		
Zone1		Tense Music with violins	Music_System/Zone1	Volume: -9 dB		
Zone2		Asian music	Music_System/Zone2	Volume: -14 dB		
Zone3		Guitar Asian Music	Music_System/Zone3	Volume: -2 dB		

Clip Name	Source	Details	Location in Wwise	Edits	Mixing	RTPC
SFX & Dialogue Work Unit						
Goat_01	Freesounds	Goat Screaming	Dialogue/GOAT		Voice Volume: -14 dB	
Goat_02						
Goat_03						
Collect		Money Picking up	SFX	Voice Volume: -24 dB		
Congrats_UpBeat		Upbeat tone	SFX	Voice Volume: -10 dB		
Lights		Fluorescent light clipping	SFX	Light Attenuation: Output Volume decreasing from 15.0 to 0.0 in a Logarithmic (Base 3) curve		
HeartBeat	Given by module teacher	Just a heartbeat	SFX			Time stretch 50 - 100 Output Gain +48dB - 0 dB

Annex 2. Scripts

```
1  using System;
2  using UnityEngine;
3  using UnityStandardAssets.CrossPlatformInput;
4  using UnityStandardAssets.Utility;
5  using Random = UnityEngine.Random;
6
7  namespace UnityStandardAssets.Characters.FirstPerson
8  {
9      [RequireComponent(typeof (CharacterController))]
10     [RequireComponent(typeof (AudioSource))]
11     public class FirstPersonController : MonoBehaviour
12     {
13         [SerializeField] private bool m_IsWalking;
14         [SerializeField] private float m_WalkSpeed;
15         [SerializeField] private float m_RunSpeed;
16         [SerializeField] [Range(0f, 1f)] private float m_RunstepLenghten;
17         [SerializeField] private float m_JumpSpeed;
18         [SerializeField] private float m_StickToGroundForce;
19         [SerializeField] private float m_GravityMultiplier;
20         [SerializeField] private MouseLook m_MouseLook;
21         [SerializeField] private bool m_UseFovKick;
22         [SerializeField] private FOVKick m_FovKick = new FOVKick();
23         [SerializeField] private bool m_UseHeadBob;
24         [SerializeField] private CurveControlledBob m_HeadBob = new
             CurveControlledBob();
25         [SerializeField] private LerpControlledBob m_JumpBob = new
             LerpControlledBob();
26         [SerializeField] private float m_StepInterval;
27         [SerializeField] private AudioClip[] m_FootstepSounds;    // an array
             of footstep sounds that will be randomly selected from.
28         [SerializeField] private AudioClip m_JumpSound;           // the sound
              played when character leaves the ground.
29         [SerializeField] private AudioClip m_LandSound;           // the sound
              played when character touches back on ground.
30
31         private Camera m_Camera;
32         private bool m_Jump;
33         private float m_YRotation;
34         private Vector2 m_Input;
35         private Vector3 m_MoveDir = Vector3.zero;
36         private CharacterController m_CharacterController;
37         private CollisionFlags m_CollisionFlags;
38         private bool m_PreviouslyGrounded;
39         private Vector3 m_OriginalCameraPosition;
40         private float m_StepCycle;
41         private float m_NextStep;
42         private bool m_Jumping;
43         private AudioSource m_AudioSource;
44
45         // Use this for initialization
46         public void Start()
47         {
48             m_CharacterController = GetComponent<CharacterController>();
49             m_Camera = Camera.main;
50             m_OriginalCameraPosition = m_Camera.transform.localPosition;
51             m_FovKick.Setup(m_Camera);
```

```
52              m_HeadBob.Setup(m_Camera, m_StepInterval);
53              m_StepCycle = 0f;
54              m_NextStep = m_StepCycle/2f;
55              m_Jumping = false;
56              m_AudioSource = GetComponent<AudioSource>();
57              m_MouseLook.Init(transform , m_Camera.transform);
58          }
59
60
61          // Update is called once per frame
62          private void Update()
63          {
64              RotateView();
65              // the jump state needs to read here to make sure it is not missed
66              if (!m_Jump)
67              {
68                  m_Jump = CrossPlatformInputManager.GetButtonDown("Jump");
69              }
70
71              if (!m_PreviouslyGrounded && m_CharacterController.isGrounded)
72              {
73                  StartCoroutine(m_JumpBob.DoBobCycle());
74                  PlayLandingSound();
75                  m_MoveDir.y = 0f;
76                  m_Jumping = false;
77              }
78              if (!m_CharacterController.isGrounded && !m_Jumping &&
                    m_PreviouslyGrounded)
79              {
80                  m_MoveDir.y = 0f;
81              }
82
83              m_PreviouslyGrounded = m_CharacterController.isGrounded;
84          }
85
86
87          private void PlayLandingSound()
88          {
89              //m_AudioSource.clip = m_LandSound;
90              //m_AudioSource.Play();
91              AkSoundEngine.PostEvent ("Play_Land", gameObject);
92              //m_NextStep = m_StepCycle + .5f;
93          }
94
95
96          private void FixedUpdate()
97          {
98              float speed;
99              GetInput(out speed);
100             // always move along the camera forward as it is the direction
                    that it being aimed at
101             Vector3 desiredMove = transform.forward*m_Input.y +
                    transform.right*m_Input.x;
102
103             // get a normal for the surface that is being touched to move
                    along it
```

```
104             RaycastHit hitInfo;
105             Physics.SphereCast(transform.position,
                  m_CharacterController.radius, Vector3.down, out hitInfo,
106                                m_CharacterController.height/2f,
                        Physics.AllLayers, QueryTriggerInteraction.Ignore);
107             desiredMove = Vector3.ProjectOnPlane(desiredMove,
                  hitInfo.normal).normalized;
108
109             m_MoveDir.x = desiredMove.x*speed;
110             m_MoveDir.z = desiredMove.z*speed;
111
112
113             if (m_CharacterController.isGrounded)
114             {
115                 m_MoveDir.y = -m_StickToGroundForce;
116
117                 if (m_Jump)
118                 {
119                     m_MoveDir.y = m_JumpSpeed;
120                     PlayJumpSound();
121                     m_Jump = false;
122                     m_Jumping = true;
123                 }
124             }
125             else
126             {
127                 m_MoveDir +=
                      Physics.gravity*m_GravityMultiplier*Time.fixedDeltaTime;
128             }
129             m_CollisionFlags = m_CharacterController.Move
                  (m_MoveDir*Time.fixedDeltaTime);
130
131             ProgressStepCycle(speed);
132             UpdateCameraPosition(speed);
133
134             m_MouseLook.UpdateCursorLock();
135         }
136
137
138         private void PlayJumpSound()
139         {
140             //m_AudioSource.clip = m_JumpSound;
141             //m_AudioSource.Play();
142             AkSoundEngine.PostEvent ("Play_Jump", gameObject);
143         }
144
145
146         private void ProgressStepCycle(float speed)
147         {
148             if (m_CharacterController.velocity.sqrMagnitude > 0 &&
                  (m_Input.x != 0 || m_Input.y != 0))
149             {
150                 m_StepCycle += (m_CharacterController.velocity.magnitude +
                      (speed*(m_IsWalking ? 1f : m_RunstepLenghten)))*
151                                Time.fixedDeltaTime;
152             }
```

```
153
154             if (!(m_StepCycle > m_NextStep))
155             {
156                 return;
157             }
158
159             m_NextStep = m_StepCycle + m_StepInterval;
160
161             PlayFootStepAudio();
162         }
163
164
165         private void PlayFootStepAudio()
166         {
167             if (!m_CharacterController.isGrounded)
168             {
169                 return;
170             }
171
172     AkSoundEngine.PostEvent ("Play_Footstep", gameObject);
173             // pick & play a random footstep sound from the array,
174             // excluding sound at index 0
175             //int n = Random.Range(1, m_FootstepSounds.Length);
176             //m_AudioSource.clip = m_FootstepSounds[n];
177             //m_AudioSource.PlayOneShot(m_AudioSource.clip);
178             // move picked sound to index 0 so it's not picked next time
179             //m_FootstepSounds[n] = m_FootstepSounds[0];
180             //m_FootstepSounds[0] = m_AudioSource.clip;
181         }
182
183
184         private void UpdateCameraPosition(float speed)
185         {
186             Vector3 newCameraPosition;
187             if (!m_UseHeadBob)
188             {
189                 return;
190             }
191             if (m_CharacterController.velocity.magnitude > 0 &&
                  m_CharacterController.isGrounded)
192             {
193                 m_Camera.transform.localPosition =
194                     m_HeadBob.DoHeadBob
                        (m_CharacterController.velocity.magnitude +
195                                         (speed*(m_IsWalking ? 1f :
                        m_RunstepLenghten)));
196                 newCameraPosition = m_Camera.transform.localPosition;
197                 newCameraPosition.y = m_Camera.transform.localPosition.y -
                      m_JumpBob.Offset();
198             }
199             else
200             {
201                 newCameraPosition = m_Camera.transform.localPosition;
202                 newCameraPosition.y = m_OriginalCameraPosition.y -
                      m_JumpBob.Offset();
203             }
```

```
204             m_Camera.transform.localPosition = newCameraPosition;
205         }
206 
207 
208         private void GetInput(out float speed)
209         {
210             // Read input
211             float horizontal = CrossPlatformInputManager.GetAxis
                  ("Horizontal");
212             float vertical = CrossPlatformInputManager.GetAxis("Vertical");
213 
214             bool waswalking = m_IsWalking;
215 
216 #if !MOBILE_INPUT
217             // On standalone builds, walk/run speed is modified by a key
                  press.
218             // keep track of whether or not the character is walking or
                  running
219             m_IsWalking = !Input.GetKey(KeyCode.LeftShift);
220 #endif
221             // set the desired speed to be walking or running
222             speed = m_IsWalking ? m_WalkSpeed : m_RunSpeed;
223             m_Input = new Vector2(horizontal, vertical);
224 
225             // normalize input if it exceeds 1 in combined length:
226             if (m_Input.sqrMagnitude > 1)
227             {
228                 m_Input.Normalize();
229             }
230 
231             // handle speed change to give an fov kick
232             // only if the player is going to a run, is running and the
                  fovkick is to be used
233             if (m_IsWalking != waswalking && m_UseFovKick &&
                  m_CharacterController.velocity.sqrMagnitude > 0)
234             {
235                 StopAllCoroutines();
236                 StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() :
                      m_FovKick.FOVKickDown());
237             }
238         }
239 
240 
241         private void RotateView()
242         {
243             m_MouseLook.LookRotation (transform, m_Camera.transform);
244         }
245 
246 
247         private void OnControllerColliderHit(ControllerColliderHit hit)
248         {
249             Rigidbody body = hit.collider.attachedRigidbody;
250             //dont move the rigidbody if the character is on top of it
251             if (m_CollisionFlags == CollisionFlags.Below)
252             {
253                 return;
```

```
254             }
255 
256             if (body == null || body.isKinematic)
257             {
258                 return;
259             }
260             body.AddForceAtPosition(m_CharacterController.velocity*0.1f,
                  hit.point, ForceMode.Impulse);
261         }
262     }
263 }
264 
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class ObjectsToCollect : MonoBehaviour
6  {
7      public static int objects = 0;
8      // Use this for initialization
9      // let the count objects script know that this object is
10     //  part of the collection and should be counted
11     void Awake()
12     {
13         objects++;
14     }
15
16     // Update is called once per frame
17     void OnTriggerEnter(Collider plyr)
18     {
19         //if the tagged FPSController 'Player' collides with an object, take it
               away from the total
20         if (plyr.gameObject.tag == "Player"){
21             objects--;
22             AkSoundEngine.PostEvent("Play_PickUp", gameObject);
23             gameObject.SetActive(false);
24         }
25     }
26 }
27
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class MaterialSwitchController : MonoBehaviour {
6
7
8      public enum Mode {Pavement, Sand, Water}
9      public Mode terrainType;
10
11
12     // Use this for initialization
13     void Start () {
14
15     }
16
17     // Update is called once per frame
18     void Update () {
19
20     }
21
22     public string GetTerrainType(){
23
24         string typeString = "";
25
26         switch (terrainType) {
27
28         case Mode.Pavement:
29             typeString = "Pavement";
30             break;
31         case Mode.Sand:
32             typeString = "Sand";
33             break;
34         case Mode.Water:
35             typeString = "Water";
36             break;
37
38         }
39
40         Debug.Log (typeString);
41         return typeString;
42
43     }
44 }
45
46
47
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityStandardAssets.CrossPlatformInput;
5  using UnityStandardAssets.Utility;
6  using Random = UnityEngine.Random;
7
8  public class DistanceToCheckpoint : MonoBehaviour
9  {
10     public string RTCPValue = "HeartBeat";
11
12     // Reference to checkpoint position
13     [SerializeField]
14     private Transform checkpoint;
15
16
17     //Serialization is the process of taking an object in ram (classes, fields,
          etc...)
18     //and making a disk representation of it which can be recreated at any
         point in the future.
19
20
21
22     // Calculated distance value
23     private float distance;
24
25
26
27     // Update is called once per frame
28     void Update ()
29     {
30         // calculate distance value between character and checkpoint
31         distance = (checkpoint.transform.position -
             transform.position).magnitude;
32
33         // set parameter from Wwise game parameter to scaled distance value
34         AkSoundEngine.SetRTPCValue(RTCPValue, distance);
35
36         Debug.Log(message: RTCPValue + distance);
37
38     }
39 }
40
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class DialogueTrigger : MonoBehaviour
6  {
7      GameObject objUI2;
8
9      private bool audioIsPlaying = false;
10     // Start is called before the first frame update
11     void Start()
12     {
13         objUI2 = GameObject.Find("ObjectNum");
14         //objUI2.SetActive(false);
15     }
16
17     // Update is called once per frame
18     void Update()
19     {
20
21     }
22
23     void OnTriggerEnter(Collider plyr)
24     {
25         if (plyr.gameObject.tag == "Player" && !audioIsPlaying){
26             AkSoundEngine.PostEvent("Play_dialogue", gameObject);
27             audioIsPlaying = true;
28             objUI2.SetActive(true);
29         }
30     }
31 }
32
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class CountObjects : MonoBehaviour
7  {
8      public string nextLevel;
9      public GameObject objToDestroy;
10     GameObject objUI;
11
12     // Use this for initialization
13     void Start()
14     {
15         //look for the text object in the UI called ObjectNum
16         objUI = GameObject.Find("ObjectNum");
17         objUI.SetActive(false);
18         objUI.GetComponent<Text>().text = (ObjectsToCollect.objects.ToString())
               + " coins left to pick up";
19     }
20     // Update is called once per frame
21     void Update()
22     {
23         //convert the numbers to string and send to the text object to update
24         objUI.GetComponent<Text>().text = (ObjectsToCollect.objects.ToString())
               + " coins left to pick up";
25
26         if (ObjectsToCollect.objects == 0)
27         {
28
29             //load a new level once all objects have been picked up
30             Application.LoadLevel(nextLevel);
31             //destroy the chosen object once the total reaches 0
32             Destroy(objToDestroy);
33             objUI.GetComponent<Text>().text = "All coins collected.";
34
35         }
36
37     }
38 }
39
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Footstep_Collider : MonoBehaviour {
6
7
8      private string colliderType;
9
10
11     // Use this for initialization
12     void Start () {
13
14         AkSoundEngine.SetSwitch ("Surface_Type", "Pavement", gameObject);
15     }
16
17     // Update is called once per frame
18     void Update () {
19
20     }
21
22     //this function dectects if there is a collision between the player
         controller and a game object
23     //and calls the function GetTerrainType which retairns the terrain type
         that has been set in the enumaration mode of that object.
24     //then it calls the PlayStepSoundMaterial method which is using a switch
         stament to set the Wwise Switches.
25
26     void OnControllerColliderHit (ControllerColliderHit col){
27         if (col.gameObject.GetComponent<MaterialSwitchController>()) {
28
29             //Store what the GetTerrainType returns and store is in the
                 variable collider type.
30             colliderType =
                 col.gameObject.GetComponent<MaterialSwitchController>
                 ().GetTerrainType ();
31         }
32         // calling the PlayStepSoundMaterialType function
33         PlayStepSoundMaterialType();
34
35         //print in the console the returned value of the
             MaterialSwitchController
36         //Debug.Log (colliderType);
37
38     }
39
40
41     void PlayStepSoundMaterialType()
42     {
43         //checks the content of the colliderType variable and depending on the
             value of the variable we switch the surface type switch
44         //group to the appropriate switch type
45         switch (colliderType) {
46         case "Pavement":
47             AkSoundEngine.SetSwitch ("Surface_Type", "Pavement", gameObject);
48             //Debug.Log (colliderType);
```

```
49              break;
50          case "Water":
51              AkSoundEngine.SetSwitch ("Surface_Type", "Water", gameObject);
52              //Debug.Log (colliderType);
53              break;
54          case "Sand":
55              AkSoundEngine.SetSwitch ("Surface_Type", "Sand", gameObject);
56              //Debug.Log (colliderType);
57              break;
58          }
59      }
60
61  }
62
```