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
Library SFX3
Day Ambience3
Night Ambience4
Pavement Footsteps4
Water Footsteps5
Sand Footsteps6
Artificial Lake Shore6
Dialogue6
Sound FX 6
Environment Sound Design7
Event Sound Design9
Play and Stop Events9
SoundBanks9
Switches and Footsteps
Attenuation and Permanent Sounds11
Triggered Sounds12
Collisions12
Character
Event List
Interactive Music
Vertical Music14
Horizontal Music15
Interactive Mix
RTPC
Reverb Zones16
Audio Buses
Game Engine Integration
Scripts
Video Captured Gameplay 17
Transcription
Annex 1. Library Sources
Annex 2. Scripts

## 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.

Pro Tools	AudioSuite Options Setup Window Avid Link Hel		- 0
Edit: Intermediate Interactive Audio	Protectorie Obseus Serah Wingon, was ruik Del		0.0
TRACKS Mile Sees TRACKS Mile Sees Transcode Transcode Markers Transcode Markers Transcode Markers Transcode Markers Transcode Markers Transcode Markers Transcode Transcode Markers Transcode			2 50 2.55 3.00 T 10.01.56.00 10.02 10.01.56.00 10.02 10.02 10.01 5.00 10.02 10.0
	voj 0.0 		5 + 1 for +
the rose     Cont_Oy_Lpudp1     Soft     So	Output 12 11	PL4 with days because down of 21	
0	* 60 2-2004 * 60 2-2004 ***********************************		+ 1 For + 1 For
Day Elements 1			+ + 5 + + 10 + 10
DyElmatMONO1	Cupue 13		+ + P, + + P + > 10 + >
O - Day Elements 2	Trout 12		+ > 53 + > 57 + > 57 + > 57 + > 58 + > 53 + > 53 + > 53
CROUPS C L 1 MLD C L 5 And De L 6 C 1 C 1 C 1 C 1 C 1 C 1 C 1 C 1 C 1 C	V0 00 +101 100		

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



Niak	<b>.</b> + Λ.	mhia	
INICI	IL AI	mbie	ence
J			

Pro Tools		- B X
Elle Edit View Jisck Clip Event AudioSuite Options Setup Window Avid Link He Edit Intermediate Interactive Audio	49	
		•••

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.



	×
Track         Preset         Auto           Footung: STE 3         - detery default         BYM355           RR1 Weise Demoise         - + -5         COMPARE SUFE	
🛞 RX Voice De-noise	
Learn Adaptive mode Optimize for: Dialogue Filter type: Surgical	3,2 2,7 0 0
dB Threshold [dB] Reduction [dB]	
100 1k 10k Hz 00 120	
Bypass	CD .

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.

Audio Events SoundBanks Game Syncs ShareSets Sessions Queries	in and an and a state state state of the st
◎■■☆田田田米※約●●中田田田	■ III X amb_night_elements
🚍 🖾 🔿 🛛 Ambient*	amb_night_elements_bat
🚊 🛛 🔆 🛛 amb_day_night	⊂r≪y Bat_1 ⊂r≪y Bat_2
■ 🖾 🛠 🛛 amb_day	
🔳 🖾 🛠 🛛 amb_day_background	
😑 🖾 🏭 amb_day_wind	
🖾 🕵 📗 Silence	a a a a a a a a a a a a a a a a a a a
- I <b>⊠ ∰</b> Wind_01	⊡ tott hap_oc
	a ar but top_oo
	Silence
🖾 🕵 🛛 Wind_04	Silence2
- u 🛠    Wind_05	amb_night_elements_owl
amb_day_loop	
amb_day_elements	
🚍 🖾 🏭 amb_day_elements_african_bird	
🔤 🛛 🛠 🛛 Parrot_African_1	
🔤 🖉 💕 🛛 Parrot_African_2	
🛛 🛛 🖋 🛛 Parrot_African_3	
🖾 🛱 Silence	🔤 🗹 发 🛛 Silence
amb_day_elements_cockatoo	
Parrot_Cockatoo	🔤 🕰 🕷 Waves
🛛 🛛 🎸 🛛 Silence	
🔤 🛛 🕵 🛛 Silence2	
🚍 🗳 🏭 amb_day_elements_leopard	
🔤 🛛 🛠 🛛 Leopard_2	
🖾 🛠 🛛 Leopard_3	
l⊂a 🛠 🛛 Leopard_4	
🖓 🛠 📕 Silence	
🔤 💥 🖌 Silence2	
🚊 🖾 🗰 amb_day_elements_stickbreak	
🖓 🕊 silence	
- 🖾 🕵 🛛 silence2	
Stick_Break_01	
□ 🖉 🖋 🛛 Stick_Break_2	
CI 🛠 Stick_Break_3	
- 🖾 🕵 🖌 Stick_Break_4	
□ 式 💕 Stick_Break_5	
Stick_Break_7	
□ 🛫 Stick_Break_10	
These are all the evolie align that	, have been used to such the day winds

## **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.

	In	teractiveAudio2* - Wwise	2021.1.3			
0:00:00.000						× 0 🙁 4 🛕 642 🚯 20
🖽 🛙 amb_day_wind - Cor	ntainer Property Editor					≈ B 0 ×
amb_day_wind		M S 🗹 Inclusion	n 🔍 0 Notes			
General Settings Com Voice 0 Volume Pitch 0 Low-pass filter 0 High-pass filter 0 Make-Up Gain 0	version Effects Positioning RTPC S Output Bus Output Bus Ambience Volume Case-Defined Auxiliary Sends Output Bus Game-Defined Auxiliary Sends User-Defined Auxiliary Sends User-Defined Auxiliary Sends Output Barent D Auxiliary Bus O I I S Barly Reflections Case-Barent Auxiliary Bus O I I I I I I I I I I I I I I I I I I	kates Advanced Settings	Initial Delay 0 Play Type 0 Randorn 0 Standard 0 Shuffle 2 Avoid repeating la Play Mode 0 Step 0 Continue 2 Avoid repeating la 1 Continue	ous		
iii amb_day_wind - Co						e ×
Name           0         %         Silence           0         %         Silence           0         %         Wind_01           0         %         Wind_02           0         %         Wind_03           0         %         Wind_04           1         1         1	75	-5 0 -3 0 -4 0 -4 0	Voice Low-pass Filter 0 0 0 0 0 0 0 0 0		· · · · · · · · · · · · · · · · · · ·	Ĵ

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.

SoundBanks	Tree list 👻 New User Settings			Generate All
🚊 🙂 🛛 Default Work Unit	SoundBanks	Data Size	Decoded Size Max Size	Free Space Type
G Ambient G Character J Dialogue G Music SFX	V     Image: Constraint Work Unit       Image: Constraint Work Unit <t< th=""><th>50 529 296 3 124 688 252 384 210 298 160 6 343 724</th><th>50 529 194 Infinite 3 124 612 Infinite 252 372 Infinite 210 298 152 Infinite 6 343 708 Infinite</th><th>- SFX - SFX - SFX - Music - SFX</th></t<>	50 529 296 3 124 688 252 384 210 298 160 6 343 724	50 529 194 Infinite 3 124 612 Infinite 252 372 Infinite 210 298 152 Infinite 6 343 708 Infinite	- SFX - SFX - SFX - Music - SFX

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.

Inspector	📥 Account 🕶 Layers 👻 La		а
	CenterEyeAnchor (Camera)		
	SceneCamera (Camera)		
🗿 🖌 Reset Local Transfo	orms (Script)		
🖉 🗹 Footstep_Collider (!	Script)		
🕷 🛩 AkSwitch		0	
Trigger On:	Start		
Name:	Surface_Type / Water		
	Surface_Lype / water		
🖷 🗹 AkSwitch		0	
	Start		
	Surface_Type / Sand		
# AkSwitch			
	Start		
	Surface_Type / Pavement		
🕷 🖉 AkSwitch			
Trigger On:	Start		
Name:	Step_Type / Run		
	Covery 1 Alice 1 Maria		
🗰 🗹 AkSwitch			
	Start		
	Step_Type / Walk		
🕐 🛩 Distance To Checkp	voint (Scrint)		
Script	DistanceToCheckpoint		
	MusicalTension		
	FoldingChair_01 (Transform)		
# 🖌 AkSwitch			
Trigger On:	Start		
Name:	Action / Jump		
	Action ( Jump		
🛱 🖌 AkSwitch			
	Start		
	Action / Land		

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.

💼 🗹 Material Switch C	ontrolier (Script)	0 ‡ ∶
	MaterialSwitchController	
Terrain Type	Pavement	

#### 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.

🖽 🗹 Mesh Collid	er	0	#	
Convex	¥			
ls Trigger	~			
Cooking Options	Mixed			
Material	None (Physic Material)			
Mesh	Default			
🛃 🛛 Objects To	Collect (Script)	0	Ŧ	
	DiectsToCollect			

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.

😭 🤟 DialogueTri	ggerArea			itati	c •
Tag Untagged		Layer Defaul			
🛃 Transform			0		
	X -0.17	Y 0.928291	0.6	6	
	X 2.1916	Y 3.7468			
🔒 🖌 Capsule Colii	ider		0		
	2				
	None (Phy	sic Material)			
🖷 🔿 Dialogue Trig	ger (Script)		0		
	Dialogue				

# **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.

🖀 📓 🛛 Interactive Music Hierarchy	Early Reflections				
■ © ① Default Work Unit ■ © 韓 Music_System ■ © 韓 Floor1	백 Music_System - Music Switch Container Associati Add Generic Path Add Path(s)				
	o‰ Music				
u u m Zone2 u u m Zone3 ≰ HeartBeat - Contents Editor - 1 child	O No_Music O None O SpawnZone O Zone1				

#### 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.

Sequence Continuous				
- 🖽 🛛 XyloBoko				

🥆 🍞 🗹 Box (	ollider				a	
		Δ				
ls Trigger		2				
Material		None (P	hysic Materia	0		6
		хo	Y O			
		X 1				
🗏 🖌 AkGa	meObj				0	
🖷 🖌 AkSt	ate				0	
Trigger On:		Start				
Name:		Music /	(SpawnZone			
🕈 🗃 🛩 AkAr	nbient				0	
Trigger On:		Start				
Use Callbac	k:					
		Pla				
		Stop	All			
Position Typ		Simple	Mode			
Show Atten	sation Sph	e Dont_S	how			
Nam <del>a</del> :		Play_M	lusic			
🗉 🛃 🛩 AkSt	ate				0	
Trigger On:		AkTrigg	jerExit			
Use Other O	Use Other Object:					
		Music /	No_Music			
		Add Comp				

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: <u>https://max13torras.wixsite.com/website/intermediate-interactive-audio</u>

#### 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		Wind simulation blowing the mic	amb_day_background/amb_day_wind	Weight: 25 Voice Volume: -5 dB	
amb_wind_02		Wind simulation blowing the mic	amb_day_background/amb_day_wind	Weight: 25 Voice Volume: -3 dB	
amb_wind_03	Rode NT1 - A mic, recorded	Wind simulation blowing the mic	amb_day_background/amb_day_wind	Weight: 25	amb_day_background
amb_wind_04	- myself	Wind simulation blowing the	amb_day_background/amb_day_wind	Voice Volume: -4 dB Weight: 25	Voice Volume: -4dB
amb_wind_05		mic Wind simulation blowing the	amb day background/amb day wind	Voice Volume: -4 dB Weight: 25	
amb_day_loop		mic Arid ambience sound	amb_day_background	Voice Volume: -4 dB	
Parrot_African_1		Bird Whistle	amb_day_elements/amb_day_elements_af rican_bird	Weight: 30 Voice Volume: -11 dB	
Parrot_African_2		Bird Whistle	amb_day_elements/amb_day_elements_af rican_bird	Weight: 65 Voice Volume: -8 dB	
Parrot_African_3		Bird Whistle	amb_day_elements/amb_day_elements_af rican_bird	Weight: 50 Voice Volume: -13 dB	
Parrot_Cockatoo	Given by module teacher	Parrot cuacking	amb_day_elements/amb_day_elements_c ockatoo	Weight: 50 Voice Volume: -13 dB	
Leopard_1	leacher	Leopard Roaring	amb_day_elements/amb_day_elements_le opard	Weight: 9	
Leopard_2		Leopard Roaring	amb_day_elements/amb_day_elements_le opard	Weight: 11	
Leopard_3		Leopard Roaring	amb_day_elements/amb_day_elements_le opard	Weight: 10	
Leopard_4		Leopard Roaring	amb_day_elements/amb_day_elements_le opard	Weight: 23	amb_day_elements Voice Volume: -9 dB
Stick_Break_1		Stick Breaking	amb_day_elements/amb_day_elements_st ickbreak		
Stick_Break_2		Stick Breaking	amb_day_elements/amb_day_elements_st ickbreak		
Stick_Break_3		Stick Breaking	amb_day_elements/amb_day_elements_st ickbreak		
Stick_Break_4		Stick Breaking	amb_day_elements/amb_day_elements_st ickbreak		
Stick_Break_5		Stick Breaking	amb_day_elements/amb_day_elements_st ickbreak		
Stick_Break_6	FreeSound.org	Stick Breaking	amb_day_elements/amb_day_elements_st ickbreak		
Stick_Break_7		Stick Breaking	amb_day_elements/amb_day_elements_st ickbreak		
Stick_Break_8		Stick Breaking	amb_day_elements/amb_day_elements_st ickbreak		
Stick_Break_9	-	Stick Breaking	amb_day_elements/amb_day_elements_st ickbreak		
Stick_Break_10		Stick Breaking	amb_day_elements/amb_day_elements_st ickbreak		
			lower		
Bat_1		Bat flapping its wings	amb_night_elements/amb_night_elements		
Bat_2		Bat flapping its wings	_bat amb_night_elements/amb_night_elements		
Bat_3		Bat flapping its wings	_bat amb_night_elements/amb_night_elements		
Bat_4	-	Bat flapping its wings	_bat amb_night_elements/amb_night_elements		
BatElap_01	Library Sounds	Bat flapping its wings slower	_bat amb_night_elements/amb_night_elements	Weight: 46	
BatFlap_02	-	Bat flapping its wings slower	_bat amb_night_elements/amb_night_elements	Weight: 45	
BatFlap_03	-	Bat flapping its wings slower	_bat amb_night_elements/amb_night_elements	Weight: 45	
BatFlap_04	-	Bat flapping its wings slower	_bat amb_night_elements/amb_night_elements	Weight: 47	amb_night_elements Voice Volume: -8 dB
Owl_1		Owl Hooting	_bat amb_night_elements/amb_night_elements	Weight: 48	
	-		_owl amb_night_elements/amb_night_elements	Weight: 9	
Owl_2	Given by the	Parrot cuacking	_owl amb_night_elements/amb_night_elements	Voice Pitch: 300 Weight: 10	
Owl_3	module teachers and edited on	Leopard Roaring	_owl amb_night_elements/amb_night_elements	Voice Pitch: -300	
Owl_4	Pro Tools	Leopard Roaring	_owl amb_night_elements/amb_night_elements	Weight: 36	
Owl_5	-	Leopard Roaring	amb_night_elements/amb_night_elements	Weight: 39	
Owl_6	_	Leopard Roaring	_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		Location in Wwise	Edits	Mixing
	1	Ch	aracter Work Unit	1	
Jump_01					
Jump_02		Simulating a jump effort sound	Character_Action/Character_Jump		Voice Volume: 4 dB
Jump_03			onaracter_Action/onaracter_bump		
Jump_04	H2n Zoom recording				
Land_01	my voice				
Land_02		Simulating when you land from a			
Land_03		place	Character_Action/Character_Land		
Land 04	1	piace			
Land 05					
RunPavement 1					
RunPavement 2					
RunPavement 3	1				
RunPavement 4					
RunPavement 5			Character_Pavement/Character_Pavement_Ru		
RunPavement 6					
RunPavement 7					
RunPavement 8					
RunPavement 9	-				
RunPavement 10					
WalkPavement 1				1	
WalkPavement_2	H5 Handy Recorder at	Eastatopa in neuroment			
WalkPavement_3	my house in Sabadell				
WalkPavement_4					
WalkPavement_5					
WalkPavement_6					
WalkPavement_7			Character_Pavement/Character_Pavement_W		
WalkPavement_8			alk		
WalkPavement_9			ain		
WalkPavement_10	-				
WalkPavement_11					
WalkPavement_12					
WalkPavement 13					
WalkPavement 14					
WalkPavement 15					
RunSand 1					
RunSand 2					
RunSand 3					
RunSand 4					
RunSand_4			Character_Sand/Character_Sand_Run		
RunSand_6					Voice Volume: 8 dB
RunSand_7					
RunSand_8					
RunSand_9					
RunSand_10					
RunSand_11	H5 Handy Recorder in	Footsteps in the sand			
RunSand_12	a beach				
WalkSand_1					
WalkSand_2	1				
WalkSand_3	1				
WalkSand_4					
WalkSand_5			Character_Sand/Character_Sand_Walk		Voice Volume: 6 dB
WalkSand_6			Character_Janu/Charactel_Jahu_Walk		voice volume. 0 ub
WalkSand_7	]				
WalkSand_8	1				
WalkSand_9	1				
WalkSand 10	1				
RunWater_1					
RunWater_2					
RunWater_3					
RunWater_4					
RunWater_5			Character_Water/Character_Water_Run		
RunWater_6					
RunWater_7					
RunWater_8	H5 Handy Recorded in	<b>F</b>			
WalkWater_1	my bath	Footsteps in the water			
WalkWater_2	,				
WalkWater_3					
WalkWater_4					
WalkWater_5			Character_Water/Character_Water_Walk		
WalkWater_6					
WalkWater_7					
WalkWater_8	1				
	4				
WalkWater_9					

Clip Name	Source	Details	Location in Wwise	Edits	Mixing	RTPC
			Music Work Unit			
1_Choir		Choir Singing				Music_Tension: Voice Volume going from +0,0 dB to -200 dB in 50 units distance
2_Sitar	Given by Module teachers	e Sitar playing Music_System/ Horizontal_Music/ AmbientSegment		Volume: -4 dB		Music_Tension: Voice Volume going from +0,0 dB to -200 dB in 75 units distance
3_BigPno		Piano playing			Also there is three	Music_Tension: Voice Volume going from +0,0 dB to -200 dB in 100 units distance
XyloBoko		African Generic Music	Music_System/ Floor1	Volume: -8 dB	used.	
SpawnZone	Royalty Free music	lan atrican denre	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 & I	Dialogue Work Unit		•	•
Goat_01 Goat_02 Goat_03	Freesounds	Goat Screaming	Dialogue/GOAT		Voice Volume: -14 dB	
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

...s\FirstPersonCharacter\Scripts\FirstPersonController.cs

```
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))]
        public class FirstPersonController : MonoBehaviour
11
12
        {
            [SerializeField] private bool m IsWalking;
13
14
            [SerializeField] private float m_WalkSpeed;
            [SerializeField] private float m_RunSpeed;
15
            [SerializeField] [Range(0f, 1f)] private float m_RunstepLenghten;
16
            [SerializeField] private float m_JumpSpeed;
17
            [SerializeField] private float m_StickToGroundForce;
18
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();
            [SerializeField] private LerpControlledBob m JumpBob = new
25
                                                                                    P
              LerpControlledBob();
            [SerializeField] private float m_StepInterval;
26
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.
            [SerializeField] private AudioClip m LandSound;
                                                                       // the sound >
29
               played when character touches back on ground.
30
31
           private Camera m_Camera;
           private bool m_Jump;
32
33
           private float m_YRotation;
34
           private Vector2 m_Input;
35
           private Vector3 m_MoveDir = Vector3.zero;
36
           private CharacterController m CharacterController;
           private CollisionFlags m CollisionFlags;
37
           private bool m PreviouslyGrounded;
38
39
           private Vector3 m OriginalCameraPosition;
40
           private float m StepCycle;
           private float m NextStep;
41
42
           private bool m_Jumping;
43
            private AudioSource m AudioSource;
44
45
           // Use this for initialization
           public void Start()
46
47
            {
                m CharacterController = GetComponent<CharacterController>();
48
49
                m Camera = Camera.main;
50
                m_OriginalCameraPosition = m_Camera.transform.localPosition;
51
                m FovKick.Setup(m Camera);
```

...s\FirstPersonCharacter\Scripts\FirstPersonController.cs 2 52 m HeadBob.Setup(m Camera, m StepInterval); 53 m StepCycle = 0f; 54 m\_NextStep = m\_StepCycle/2f; 55 m\_Jumping = false; m AudioSource = GetComponent<AudioSource>(); 56 m\_MouseLook.Init(transform , m\_Camera.transform); 57 } 58 59 60 // Update is called once per frame 61 private void Update() 62 63 { RotateView(); 64 // the jump state needs to read here to make sure it is not missed 65 66 if (!m\_Jump) 67 { m\_Jump = CrossPlatformInputManager.GetButtonDown("Jump"); 68 69 } 70 71 if (!m\_PreviouslyGrounded && m\_CharacterController.isGrounded) 72 { 73 StartCoroutine(m JumpBob.DoBobCycle()); 74 PlayLandingSound(); m\_MoveDir.y = 0f; 75 m\_Jumping = false; 76 77 } if (!m CharacterController.isGrounded && !m Jumping && 78 P m\_PreviouslyGrounded) 79 { m\_MoveDir.y = 0f; 80 81 } 82 83 m\_PreviouslyGrounded = m\_CharacterController.isGrounded; 84 } 85 86 private void PlayLandingSound() 87 88 { 89 //m\_AudioSource.clip = m\_LandSound; 90 //m\_AudioSource.Play(); AkSoundEngine.PostEvent ("Play Land", gameObject); 91 //m NextStep = m StepCycle + .5f; 92 93 } 94 95 private void FixedUpdate() 96 97 { 98 float speed; 99 GetInput(out speed); 100 // always move along the camera forward as it is the direction P that it being aimed at Vector3 desiredMove = transform.forward\*m Input.y + 101 P transform.right\*m\_Input.x; 102 103 // get a normal for the surface that is being touched to move P along it

104	tPersonCharacter\Scripts\FirstPersonController.cs RaycastHit hitInfo;	
105	Physics.SphereCast(transform.position,	P
	<pre>m_CharacterController.radius, Vector3.down, out hitInfo,</pre>	
106	<pre>m_CharacterController.height/2f,</pre>	P
100	Physics.AllLayers, QueryTriggerInteraction.Ignore);	
107	desiredMove = Vector3.ProjectOnPlane(desiredMove,	P
107	hitInfo.normal).normalized;	-
108		
	<pre>m MoveDir.x = desiredMove.x*speed;</pre>	
109	-	
110	<pre>m_MoveDir.z = desiredMove.z*speed;</pre>	
111		
112	if (m. Channachan Cambra 11 an dia Canaun da di)	
113	<pre>if (m_CharacterController.isGrounded)</pre>	
114	{	
115	<pre>m_MoveDir.y = -m_StickToGroundForce;</pre>	
116		
117	<pre>if (m_Jump)</pre>	
118	{	
119	<pre>m_MoveDir.y = m_JumpSpeed;</pre>	
120	PlayJumpSound();	
121	m_Jump = false;	
122	m_Jumping = true;	
123	}	
124	}	
125	else	
126	{	
127	m_MoveDir +=	₽
128	}	
129	<pre>m_CollisionFlags = m_CharacterController.Move    (m_MoveDir*Time.fixedDeltaTime);</pre>	₽
130		
131	<pre>ProgressStepCycle(speed);</pre>	
132	<pre>UpdateCameraPosition(speed);</pre>	
133		
134	<pre>m_MouseLook.UpdateCursorLock();</pre>	
135	}	
136		
137		
138	<pre>private void PlayJumpSound()</pre>	
139	{	
140	//m AudioSource.clip = m JumpSound;	
141	<pre>//m_AudioSource.Play();</pre>	
142	AkSoundEngine.PostEvent ("Play_Jump", gameObject);	
143	}	
144	ſ	
144		
145	<pre>private void ProgressStepCycle(float speed)</pre>	
146		
	$\begin{cases} if (m CharacterController velocity conMagnitude ) 0.88 \end{cases}$	_
148	<pre>if (m_CharacterController.velocity.sqrMagnitude &gt; 0 &amp;&amp;     (m_Input.x != 0    m_Input.y != 0)) </pre>	P
149	{	
	<pre>m_StepCycle += (m_CharacterController.velocity.magnitude +</pre>	P
150	<pre>(speed*(m_IsWalking ? 1f : m_RunstepLenghten)))*</pre>	
150 151	<pre>(speed*(m_IsWalking ? 1f : m_RunstepLenghten)))*         Time.fixedDeltaTime;</pre>	

s\F	irstPersonCharacter\Scripts\FirstPersonController.cs	4
153		
154	<pre>if (!(m_StepCycle &gt; m_NextStep))</pre>	
155	{	
156	return;	
157	}	
158		
159	<pre>m_NextStep = m_StepCycle + m_StepInterval;</pre>	
160		
161	<pre>PlayFootStepAudio();</pre>	
162	}	
163		
164		
165	<pre>private void PlayFootStepAudio()</pre>	
166	{	
167	<pre>if (!m_CharacterController.isGrounded)</pre>	
168	{	
169	return;	
170	}	
171		
172	AkSoundEngine.PostEvent ("Play_Footstep", gameObject);	
173	// pick & play a random footstep sound from the array,	
174	<pre>// excluding sound at index 0</pre>	
175	<pre>//int n = Random.Range(1, m_FootstepSounds.Length);</pre>	
176	<pre>//m_AudioSource.clip = m_FootstepSounds[n];</pre>	
177	<pre>//m_AudioSource.PlayOneShot(m_AudioSource.clip);</pre>	
178	<pre>// move picked sound to index 0 so it's not picked next time</pre>	
179	<pre>//m_FootstepSounds[n] = m_FootstepSounds[0];</pre>	
180	<pre>//m_FootstepSounds[0] = m_AudioSource.clip;</pre>	
181	}	
182		
183		
184	<pre>private void UpdateCameraPosition(float speed)</pre>	
185	{	
186	Vector3 newCameraPosition;	
187	<pre>if (!m_UseHeadBob)</pre>	
188	{	
189	return;	
190	}	
191	<pre>if (m_CharacterController.velocity.magnitude &gt; 0 &amp;&amp;</pre>	P
	<pre>m_CharacterController.isGrounded)</pre>	
192	{	
193	<pre>m_Camera.transform.localPosition =</pre>	
194	m_HeadBob.DoHeadBob	P
	<pre>(m_CharacterController.velocity.magnitude +</pre>	
195	<pre>(speed*(m_IsWalking ? 1f :</pre>	P
	<pre>m_RunstepLenghten)));</pre>	
196	<pre>newCameraPosition = m_Camera.transform.localPosition;</pre>	
197	<pre>newCameraPosition.y = m_Camera.transform.localPosition.y -</pre>	P
	<pre>m_JumpBob.Offset();</pre>	
198	}	
199	else	
200	{	
201	<pre>newCameraPosition = m_Camera.transform.localPosition;</pre>	
202	<pre>newCameraPosition.y = m_OriginalCameraPosition.y -</pre>	P
	<pre>m_JumpBob.Offset();</pre>	
203	}	

<pre>265 } 266 267 268 private void GetInput(out float speed) 269 { 269 / 269 / 269 / 269 / 269 / 260 // Read input 261 float horizontal = CrossPlatformInputManager.GetAxis P 261 // Read input 262 float vertical = CrossPlatformInputManager.GetAxis("Vertical"); 263 // float vertical = CrossPlatformInputManager.GetAxis("Vertical"); 264 // float vertical = CrossPlatformInputManager.GetAxis("Vertical"); 275 // On standalone builds, walk/run speed is modified by a key 276 press. 277 // On standalone builds, walk/run speed is modified by a key 277 // On standalone builds, walk/run speed is modified by a key 278 press. 278 // keep track of whether or not the character is walking or 279 m_IsWalking = IInput.GetKey(KeyCode.LeftShift); 270 #endif 271 // set the desired speed to be walking or running 272 speed = m_IsWalking ? m_WalkSpeed : m_RunSpeed; 273 m_Input = new Vector2(horizontal, vertical); 274 // normalize input if it exceeds 1 in combined length: 275 if (m_Input.sqrMagnitude &gt; 1) 277 { 278 m_Input.Normalize(); 279 } 279 } 279 // handle speed change to give an fov kick 270 // only if the player is going to a run, is running and the 273 ff (m_IsWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp; 273 m_CharacterController.velocity.sqrMagnitude &gt; 0) 274 { 275 StartCoroutines(); 276 StartCoroutines(); 277 } 277 } 277 }</pre>	s	\FirstPersonCharacter\Scripts\FirstPersonController.cs	5
<pre>266 277 278 278 279 279 270 271 271 271 272 273 273 274 275 275 277 277 277 277 277 277</pre>	204	<pre>m_Camera.transform.localPosition = newCameraPosition;</pre>	
<pre>286 287 288 private void GetInput(out float speed) 289 289 289 289 280 280 281 281 282 282 282 282 282 282 283 284 282 28 28 28 28 28 28 28 28 28 28 28 28</pre>	205	}	
<pre>297 208 private void GetInput(out float speed) 209 { 209 // Read input 210 float horizontal = CrossPlatformInputManager.GetAxis P 212 float vertical = CrossPlatformInputManager.GetAxis("Vertical"); 213 214 bool waswalking = m_ISWalking; 216 fif !NOBILE_INPUT 217 // On standalone builds, walk/run speed is modified by a key 217 // keep track of whether or not the character is walking or 218 m_rouning 220 fendif 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 // nonmalize input if it exceeds 1 in combined length: 225 // normalize input if it exceeds 1 in combined length: 226 if (m_Input.sqrMagnitude &gt; 1) 227 { 238 m_Input if the player is going to a run, is running and the 239 forwick is to be used 231 if (m_ISWalking != waswalking &amp;&amp; m_USeFovKick &amp;&amp; P 234 m_Input.Normalize(); 235 StopAllCoroutines(); 236 StartCoroutine(!m_ISWalking ? m_FovKick.FOVKickUp() : P 237 } 238 } 240 241 private void RotateView() 242 { 243 m_MouseLook.LookRotation (transform, m_Camera.transform); 244 } 245 private void OnControllerColliderHit(ControllerColliderHit hit) 246 // dont move the rigidbody if the character is on top of it 247 if (m_CollisionFlags == CollisionFlags.Below) 248 // dont move the rigidbody if the character is on top of it 249 // dont move the rigidbody if the character is on top of it 240 // dont move the rigidbody if the character is on top of it 241 if (m_CollisionFlags == CollisionFlags.Below) 242 // dont move the rigidbody if the character is on top of it 243 // dont move the rigidbody if the character is on top of it 244 // dont move the rigidbody if the character is on top of it 245 // dont move the rigidbody if the character is on top of it 245 // dont move the rigidbody if the character is on top of it 245 // dont move the rigidbody if the character is on top of it 245 // dont move the rigidbody if the character is on top of it 245 // dont move the rigidbody if the character is on top of it 245</pre>	206		
<pre>288     private void GetInput(out float speed) 289     { 289     // Read input 210     // Read input 211     float horizontal = CrossPlatformInputManager.GetAxis</pre>	207		
<pre>299 { 210 // Read input 211 float horizontal = CrossPlatformInputManager.GetAxis 212 float vertical = CrossPlatformInputManager.GetAxis("Vertical"); 213 bool waswalking = m_IsWalking; 214 bool waswalking = m_IsWalking; 215 // On standalone builds, walk/run speed is modified by a key 216 field in the provided of the provided of the character is walking or 217 // Keep track of whether or not the character is walking or 218 // keep track of whether or not the character is walking or 219 m_IsWalking = !Input.GetKey(KeyCode.LeftShift); 220 flendif 221 speed = m_IsWalking ? m_WalkSpeed : m_RunSpeed; 222 m_Input = new Vector2(horizontal, vertical); 223 // normalize input if it exceeds 1 in combined length: 226 if (m_Input.sqrMagnitude &gt; 1) 227 { 228 m_Input.Normalize(); 229 } 230 // handle speed change to give an fov kick 232 // only if the player is going to a run, is running and the 233 for (m_IsWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp; p     m_CharacterController.velocity.sqrMagnitude &gt; 0) 234 { 235 StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() : 236 m_FovKick.FOVKickDown()); 237 } 238 } 239 240 241 private void RotateView() 242 { 243 m_MouseLook.LookRotation (transform, m_Camera.transform); 243 { 244 244 244 private void OnControllerColliderHit(ControllerColliderHit hit) 245 { 246 247 private void OnControllerColliderHit(ControllerColliderHit hit) 248 { 249 Private void OnControllerColliderHit(ControllerColliderHit hit) 249 { 240 Private void OnControllerColliderHit(ControllerColliderHit hit) 241 { 242 Private void OnControllerColliderHit(ControllerColliderHit hit) 242 { 243 Private void OnControllerColliderHit(ControllerColliderHit hit) 244 { 244 Private void OnControllerColliderHit(ControllerColliderHit hit) 245 // dont move the rigidbody if the character is on top of it 245 // dont move the rigidbody if the character is on top of it 245 // dont move the rigidbody if the character is on top of it 245 // dont move the rigidbody if the character is on top of it 245 // dont move the rigidbody if the chara</pre>		private void GetInput(out float speed)	
<pre>210</pre>			
<pre>211 float horizontal = CrossPlatformInputManager.GetAxis</pre>			
<pre>("Horizontal"); float vertical = CrossPlatformInputManager.GetAxis("Vertical"); lat bool waswalking = m_IsWalking; lf if if IMOBILE_INPUT // On standalone builds, walk/run speed is modified by a key press. // keep track of whether or not the character is walking or running m_ISWalking = !Input.GetKey(KeyCode.LeftShift); if effendif // set the desired speed to be walking or running speed = m_ISWalking ? m_WalkSpeed : m_RunSpeed; m_Input = new Vector2(horizontal, vertical); // normalize input if it exceeds 1 in combined length: if (m_Input.sqrMagnitude &gt; 1) { m_Input.Normalize(); } // only if the player is going to a run, is running and the fovkick is to be used m_CharacterController.velocity.sqrMagnitude &gt; 0) f(m_ISWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp; m_CharacterController.velocity.sqrMagnitude &gt; 0) { fun_IsWalking != waswalking ? m_FovKick.FOVKickUp() : m_FovKick.FOVKickDowm()); } } } // private void RotateView() fm_figidbody body = hit.collider.attachedRigidbody; //dont move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) fm_CollisionFlags == CollisionFlags.Below) fm_ColliserControllers(); } // on move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) fm_ColliserControllerCollider.attachedRigidbody; //dont move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) fm_ColliserControllerCollider.attachedRigidbody; //dont move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below)</pre>		•	P
<pre>float vertical = CrossPlatformInputManager.GetAxis("Vertical"); dool waswalking = m_IsWalking; fif loof waswalking = m_IsWalking; // On standalone builds, walk/run speed is modified by a key</pre>			
<pre>bool waswalking = m_IsWalking; bool waswalking = m_IsWalking; fif !MOBILE_INPUT // On standalone builds, walk/run speed is modified by a key press. // keep track of whether or not the character is walking or running m_IsWalking = !Input.GetKey(KeyCode.LeftShift); #endif // set the desired speed to be walking or running speed = m_IsWalking ? m_WalkSpeed : m_RunSpeed; m_Input = new Vector2(horizontal, vertical); // normalize input if it exceeds 1 in combined length: if (m_Input.sqrMagnitude &gt; 1) // handle speed change to give an fov kick // noly if the player is going to a run, is running and the fovkick is to be used // handle speed change to give an fov kick // only if the player is going to a run, is running and the fovkick is to be used // handle speed change to give an fov kick &amp;&amp; m_CharacterController.velocity.sqrMagnitude &gt; 0) // formative(!m_IsWalking ? m_FovKick.FOVKickUp() : m_FovKick.FOVKickDown()); // a // private void RotateView() // figidbody body = hit.collider.attachedRigidbody; //dont move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) // formative collise for the character is on top of it // formative coll startComplex for the character is on top of it // formative collisionFlags == CollisionFlags.Below) // formative collisionFlags == CollisionFlags.Below)</pre>	212	• • •	
<pre>214 bool waswalking = m_ISWalking; 215 216 #if !MOBILE_INPUT 217</pre>			
<pre>215 216 #if !MOBILE_INPUT 217</pre>		hool waswalking = m TsWalking:	
<pre>216 #if IMOBILE_INPUT 217</pre>		boor waswarking - m_iswarking,	
<pre>217  // On standalone builds, walk/run speed is modified by a key</pre>		#if IMORTLE INDUT	
<pre>press. // keep track of whether or not the character is walking or running m_ISWalking = lInput.GetKey(KeyCode.LeftShift); // set the desired speed to be walking or running speed = m_ISWalking ? m_WalkSpeed : m_RunSpeed; m_Input = new Vector2(horizontal, vertical); // normalize input if it exceeds 1 in combined length: if (m_Input.sqrMagnitude &gt; 1) { m_Input.Normalize(); } // handle speed change to give an fov kick // only if the player is going to a run, is running and the fovkick is to be used // handle speed change to give an fov kick // only if the player is going to a run, is running and the fovkick is to be used ff (m_ISWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp; m_CharacterController.velocity.sqrMagnitude &gt; 0) { StartCoroutine(!m_ISWalking ? m_FovKick.FOVKickUp() : m_FovKick.FOVKickDown()); } } } // private void RotateView() { m_MouseLook.LookRotation (transform, m_Camera.transform); } } } } // private void OnControllerColliderHit(ControllerColliderHit hit) { Rigidbody body = hit.collider.attachedRigidbody; //dont move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) { // cont move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) } // cont move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) }</pre>			-
<pre>218 // keep track of whether or not the character is walking or</pre>	21/		*
<pre>running m_TsWalking = !Input.GetKey(KeyCode.LeftShift); m_TsWalking = !Input.GetKey(KeyCode.LeftShift); rendif // set the desired speed to be walking or running speed = m_TsWalking ? m_WalkSpeed : m_RunSpeed; m_Input = new Vector2(horizontal, vertical); // normalize input if it exceeds 1 in combined length: if (m_Input.sqrMagnitude &gt; 1) // normalize input.sqrMagnitude &gt; 1) // handle speed change to give an fov kick // only if the player is going to a run, is running and the fovkick is to be used // only if the player is going to a run, is running and the fovkick is to be used // only if the player is going to a run, is running and the fovkick is to be used // only if the player is going to a run, is running and the fovkick. is to be used // only if the player is going to a run, is running and the fovkick. is to be used // only if the player is going to a run, is running and the fovkick. is to be used // only if the player is going to a run, is running and the fovkick. is to be used // only if the player is going to a run, is running and the fovkick. is to be used // for the player is going to a run, is running and the fovkick. is to be used // for the player is going to a run, is running and the fovkick.FOVKickDown()); // a for the running the player is going to a run, is running and the for the void RotateView() // a for the running the player is going to a run, is running and the for private void OnControllerColliderHit(ControllerColliderHit hit) // a figidbody body = hit.collider.attachedRigidbody; // dont move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) // a full figure for the figidbody if the character is on top of it for for the player for th</pre>	210		_
<pre>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 // normalize input if it exceeds 1 in combined length: 225 // normalize input.sqrMagnitude &gt; 1) 226 if (m_Input.sqrMagnitude &gt; 1) 227 { 228 m_Input.Normalize(); 229 } 230 // handle speed change to give an fov kick 232 // only if the player is going to a run, is running and the 233 if (m_IsWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp; 234 m_CharacterController.velocity.sqrMagnitude &gt; 0) 234 { 235 StartCoroutine(); 236 StartCoroutine(); 237 } 238 } 239 240 240 24 { 241 private void RotateView() 242 { 243 m_MouseLook.LookRotation (transform, m_Camera.transform); 244 } 245 24 246 25 // dont move the rigidbody if the character is on top of it 247 if (m_CollisionFlags == CollisionFlags.Below) 248 { 249 // dont move the rigidbody if the character is on top of it 250 // toten to the rigidbody if the character is on top of it 251 if (m_CollisionFlags == CollisionFlags.Below) 252 { 253 // dont move the rigidbody if the character is on top of it 254 // toten to the rigidbody if the character is on top of it 255 // toten to the rigidbody if the character is on top of it 256 // toten to the rigidbody if the character is on top of it 252 // toten to the rigidbody if the character is on top of it 254 // toten to the rigidbody if the character is on top of it 255 // toten to the rigidbody if the character is on top of it 256 // toten to the rigidbody if the character is on top of it 257 // toten to the rigidbody if the character is on top of it 258 // toten to the rigidbody if the character is on top of it 259 // toten to the rigidbody if the character is on top of it 250 // toten to the rigidbody if the character is on top of it 250 // toten to the rigidbody if the character is on top of it 258 // toten to the rigidbody if the character is on top of it 259 // tot the the the the the the the the the th</pre>	219	· ·	•
<pre>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.sqnMagnitude &gt; 1) 227 { 228 m_Input.Normalize(); 239 } 230 231 // handle speed change to give an fov kick 232 // only if the player is going to a run, is running and the 233 if (m_ISWalking != waswalking &amp;&amp; m_USeFovKick &amp;&amp; 234 { 235 StopAllCoroutines(); 236 StartCoroutine(!m_ISWalking ? m_FovKick.FOVKickUp() : 237 } 238 } 239 240 240 241 private void RotateView() 242 { 243 m_MouseLook.LookRotation (transform, m_Camera.transform); 244 } 245 246 246 247 private void OnControllerColliderHit(ControllerColliderHit hit) 248 { 249 Rigidbody body = hit.collider.attachedRigidbody; 244 // dont move the rigidbody if the character is on top of it 250 // dont move the rigidbody if the character is on top of it 251 if (m_CollisionFlags == CollisionFlags.Below) 252 { 253 // dont move the rigidbody if the character is on top of it 254 // dont move the rigidbody if the character is on top of it 255 // dont move the rigidbody if the character is on top of it 255 // dont move the rigidbody if the character is on top of it 252 // dont move the rigidbody if the character is on top of it 254 // dont move the rigidbody if the character is on top of it 255 // dont move the rigidbody if the character is on top of it 255 // dont move the rigidbody if the character is on top of it 255 // dont move the rigidbody if the character is on top of it 255 // dont move the rigidbody if the character is on top of it 255 // dont move the rigidbody if the character is on top of it 255 // dont move the rigidbody if the character is on top of it 255 // dont move the rigidbody if the character is on top of it 255 // dont move the rigidbody if the character is on top of it 255 // dont move the rigidbody if the character is on top of it 256 // dont move the rigidbody if the character is on top of it 257 // do</pre>	210		
<pre>// set the desired speed to be walking or running speed = m_ISWalking ? m_WalkSpeed : m_RunSpeed; m_Input = new Vector2(horizontal, vertical); // normalize input if it exceeds 1 in combined length: if (m_Input.sqrMagnitude &gt; 1) { m_Input.Normalize(); } // handle speed change to give an fov kick // only if the player is going to a run, is running and the fovkick is to be used if (m_ISWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp; m_CharacterController.velocity.sqrMagnitude &gt; 0) { StartCoroutine(!m_ISWalking ? m_FovKick.FOVKickUp() : P m_FovKick.FOVKickDown()); } } // private void RotateView() // m_MouseLook.LookRotation (transform, m_Camera.transform); // and MouseLook.LookRotation (transform, m_Camera.transform); // and f // private void OnControllerColliderHit(ControllerColliderHit hit) // and f // gigidbody body = hit.collider.attachedRigidbody; //dont move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) // and the start of the start</pre>			
<pre>speed = m_ISWalking ? m_WalkSpeed : m_RunSpeed; m_Input = new Vector2(horizontal, vertical); // normalize input if it exceeds 1 in combined length: if (m_Input.sqrMagnitude &gt; 1) { m_Input.Normalize(); } // handle speed change to give an fov kick // only if the player is going to a run, is running and the fovkick is to be used if (m_IsWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp; m_CharacterController.velocity.sqrMagnitude &gt; 0) { StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() : P m_FovKick.FOVKickDown()); } } private void RotateView() { m_MouseLook.LookRotation (transform, m_Camera.transform); } } f if (m_CollisionFlags == CollisionFlags.Below) { // dont move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) } // not move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) } // not move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) } // m_Forkick.Forkick.ForkickDow // manual for the start f</pre>			
<pre>223 m_Input = new Vector2(horizontal, vertical); 224 225 // normalize input if it exceeds 1 in combined length: 226 if (m_Input.sqrMagnitude &gt; 1) 227 { 228 m_Input.Normalize(); 239 } 230 231 // handle speed change to give an fov kick 232 // only if the player is going to a run, is running and the 233 if (m_IsWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp; 234 m_CharacterController.velocity.sqrMagnitude &gt; 0) 234 { 235 StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() : 236 StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() : 237 } 238 } 239 240 240 private void RotateView() 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 { 352 { 353 { 354 { 355 { 355</pre>			
<pre>224 225  // normalize input if it exceeds 1 in combined length: 226  if (m_Input.sqrMagnitude &gt; 1) 227  { 228  m_Input.Normalize(); 239  } 230 231  // handle speed change to give an fov kick 232  // only if the player is going to a run, is running and the 233  if (m_ISWalking != waswalking &amp;&amp; m_USeFovKick &amp;&amp; 234</pre>			
<pre>225</pre>		<pre>m_Input = new Vector2(horizontal, vertical);</pre>	
<pre>226 if (m_Input.sqrMagnitude &gt; 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 &amp;&amp; m_UseFovKick &amp;&amp;         m_CharacterController.velocity.sqrMagnitude &gt; 0) 234 { 235 StopAllCoroutines(); 236 StopAllCoroutines(); 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 { 26 27 28 29 29 29 29 20 20 20 20 20 20 20 20 20 20 20 20 20</pre>			
<pre>227 {</pre>		· · · · ·	
<pre>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 &amp;&amp; m_UseFovKick &amp;&amp; m_CharacterController.velocity.sqrMagnitude &gt; 0) 234 { 235 StopAllCoroutines(); 236 StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() : P 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 { 252 } 253 } 254 } 255 } 256 // CollisionFlags == CollisionFlags.Below) 252 } 256 // CollisionFlags == CollisionFlags.Below) 257 } 258 // CollisionFlags == CollisionFlags.Below) 259 // CollisionFlags == CollisionFlags.Below) 250 // CollisionFlags == CollisionFlags.Below) 251 // CollisionFlags == CollisionFlags.Below) 252 // CollisionFlags == CollisionFlags.Below) 251 // CollisionFlags == CollisionFlags.Below) 252 // CollisionFlags == CollisionFlags.Below) 252 // CollisionFlags == CollisionFlags.Below) 251 // CollisionFlags == CollisionFlags.Below) 252 // CollisionFlags == CollisionFlags.Below}</pre>			
<pre>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 &amp;&amp; m_USeFovKick &amp;&amp; m_CharacterController.velocity.sqrMagnitude &gt; 0) 234 { 235 StopAllCoroutines(); 236 StartCoroutine(!m_ISWalking ? m_FovKick.FOVKickUp() : m_FovKick.FOVKickDown()); 237 } 238 } 240 240 241 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 151 if (m_CollisionFlags == CollisionFlags.Below) 252 { 261 } 272 } 283 } 294 } 295 ] 295 } 295 ] 295 ] 29</pre>	227		
<pre>230 231 // handle speed change to give an fov kick 232 // only if the player is going to a run, is running and the</pre>	228	<b>—</b> · · · · ·	
<pre>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 &amp;&amp; m_UseFovKick &amp;&amp; m_CharacterController.velocity.sqrMagnitude &gt; 0) 234  { 235        StopAllCoroutines(); 236        StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() : P m_FovKick.FOVKickDown()); 237  } 238  } 239  } 240 240 241  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       {</pre>	229	}	
<pre>232 // only if the player is going to a run, is running and the fovkick is to be used 233 if (m_IsWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp; m_CharacterController.velocity.sqrMagnitude &gt; 0) 234 { 235 StopAllCoroutines(); 236 StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() : m_FovKick.FOVKickDown()); 237 } 238 } 239 240 241 private void RotateView() 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 { 251 } 252 { 252 } 253 } 254 } 255 <math>255 </math>255 <math>255 </math>255</pre>	230		
<pre>fovkick is to be used if (m_IsWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp; m_CharacterController.velocity.sqrMagnitude &gt; 0) { StopAllCoroutines(); StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() : m_FovKick.FOVKickDown()); } } } } private void RotateView() { m_MouseLook.LookRotation (transform, m_Camera.transform); } m_MouseLook.LookRotation (transform, m_Camera.transform); } f m_MouseLook.LookRotation (transform, m_Camera.transform); } { m_MouseLook.LookRotation (transform, m_Camera.transform); } rivate void OnControllerColliderHit(ControllerColliderHit hit) { Rigidbody body = hit.collider.attachedRigidbody; //dont move the rigidbody if the character is on top of it if (m_CollisionFlags == CollisionFlags.Below) { } } </pre>	231	<pre>// handle speed change to give an fov kick</pre>	
<pre>233 if (m_IsWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp;</pre>	232	<pre>// only if the player is going to a run, is running and the</pre>	₽
<pre>m_CharacterController.velocity.sqrMagnitude &gt; 0) {     StopAllCoroutines();     StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() :</pre>		fovkick is to be used	
<pre>234 { 235 StopAllCoroutines(); 236 StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() :</pre>	233	<pre>if (m_IsWalking != waswalking &amp;&amp; m_UseFovKick &amp;&amp;</pre>	₽
<pre>235 StopAllCoroutines(); 236 StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() :</pre>		<pre>m_CharacterController.velocity.sqrMagnitude &gt; 0)</pre>	
<pre>235 StopAllCoroutines(); 236 StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() :</pre>	234	{	
<pre>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 { </pre>	235		
<pre>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 { </pre>	236	<pre>StartCoroutine(!m IsWalking ? m FovKick.FOVKickUp() :</pre>	P
<pre>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 { </pre>			
<pre>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 { </pre>	237	<b>_</b>	
<pre>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 { </pre>	238		
<pre>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 {</pre>		,	
<pre>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 { </pre>			
<pre>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 { </pre>		private void RotateView()	
<pre>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 { </pre>			
<pre>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 { </pre>			
<pre>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 { </pre>			
<pre>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 {</pre>		L	
<pre>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 { </pre>			
<pre>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 {</pre>		nrivate void OnControllenColliderHit(ControllenColliderHit bit)	
249Rigidbody body = hit.collider.attachedRigidbody;250//dont move the rigidbody if the character is on top of it251if (m_CollisionFlags == CollisionFlags.Below)252{			
<pre>250 //dont move the rigidbody if the character is on top of it 251 if (m_CollisionFlags == CollisionFlags.Below) 252 {</pre>			
<pre>251 if (m_CollisionFlags == CollisionFlags.Below) 252 {</pre>			
252 {			
253 return;		•	
	253	recum;	

•••s	\Fi	rstP	erso	nCharacter\Scripts\FirstPersonController.cs	6
254				}	
255					
256				<pre>if (body == null    body.isKinematic)</pre>	
257				{	
258				return;	
259				}	
260				<pre>body.AddForceAtPosition(m_CharacterController.velocity*0.1f, hit.point, ForceMode.Impulse);</pre>	Ş
261			}		
262		}			
263	}				
264					

...madiate Interactive Audio\TechNotes\ObjectsToCollect.cs

```
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;
       // Use this for initialization
 8
       // let the count objects script know that this object is
 9
10
       // part of the collection and should be counted
       void Awake()
11
12
       {
13
            objects++;
14
       }
15
       // Update is called once per frame
16
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"){
               objects--;
21
               AkSoundEngine.PostEvent("Play_PickUp", gameObject);
22
23
               gameObject.SetActive(false);
24
            }
25
       }
26 }
27
```

...Interactive Audio\TechNotes\MaterialSwitchController.cs

```
1 using System.Collections;
 2 using System.Collections.Generic;
 3 using UnityEngine;
 4
 5 public class MaterialSwitchController : MonoBehaviour {
 6
 7
       public enum Mode {Pavement, Sand, Water}
 8
 9
       public Mode terrainType;
10
11
       // Use this for initialization
12
13
       void Start () {
14
15
       }
16
17
       // Update is called once per frame
18
       void Update () {
19
20
       }
21
22
       public string GetTerrainType(){
23
           string typeString = "";
24
25
           switch (terrainType) {
26
27
28
           case Mode.Pavement:
               typeString = "Pavement";
29
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
```

...e Interactive Audio\TechNotes\DistanceToCheckpoint-2.cs

```
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
        // Reference to checkpoint position
12
        [SerializeField]
13
14
        private Transform checkpoint;
15
16
        //Serialization is the process of taking an object in ram (classes, fields, \rightarrow
17
           etc...)
18
        //and making a disk representation of it which can be recreated at any
                                                                                     P
          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
        {
            // calculate distance value between character and checkpoint
30
31
            distance = (checkpoint.transform.position -
                                                                                     P
              transform.position).magnitude;
32
            // set parameter from Wwise game parameter to scaled distance value
33
34
            AkSoundEngine.SetRTPCValue(RTCPValue, distance);
35
36
            Debug.Log(message: RTCPValue + distance);
37
38
        }
39 }
40
```

1

... rmadiate Interactive Audio\TechNotes\DialogueTrigger.cs

```
1 using System.Collections;
 2 using System.Collections.Generic;
 3 using UnityEngine;
 4
 5 public class DialogueTrigger : MonoBehaviour
 6
   {
 7
       GameObject objUI2;
 8
       private bool audioIsPlaying = false;
9
10
       // Start is called before the first frame update
       void Start()
11
12
       {
            objUI2 = GameObject.Find("ObjectNum");
13
            //objUI2.SetActive(false);
14
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){
                AkSoundEngine.PostEvent("Play_dialogue", gameObject);
26
27
                audioIsPlaying = true;
28
                objUI2.SetActive(true);
29
            }
30
       }
31 }
32
```

...ntermadiate Interactive Audio\TechNotes\CountObjects.cs

```
1 using System.Collections;
 2 using System.Collections.Generic;
 3 using UnityEngine;
 4 using UnityEngine.UI;
 5
 6 public class CountObjects : MonoBehaviour
 7
   {
       public string nextLevel;
 8
 9
       public GameObject objToDestroy;
10
       GameObject objUI;
11
       // Use this for initialization
12
13
       void Start()
14
       {
            //look for the text object in the UI called ObjectNum
15
16
            objUI = GameObject.Find("ObjectNum");
            objUI.SetActive(false);
17
            objUI.GetComponent<Text>().text = (ObjectsToCollect.objects.ToString()) →
18
               + " coins left to pick up";
19
       }
       // Update is called once per frame
20
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
           if (ObjectsToCollect.objects == 0)
26
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

...adiate Interactive Audio\TechNotes\Footstep\_Collider.cs

```
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
        // Use this for initialization
11
        void Start () {
12
13
14
            AkSoundEngine.SetSwitch ("Surface_Type", "Pavement", gameObject);
        }
15
16
        // Update is called once per frame
17
18
       void Update () {
19
20
        }
21
22
        //this function dectects if there is a collision between the player
                                                                                      P
          controller and a game object
23
        //and calls the function GetTerrainType which retairns the terrain type
                                                                                      P
          that has been set in the enumaration mode of that object.
        //then it calls the PlayStepSoundMaterial method which is using a switch
24
                                                                                      P
          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
                                                                                      P
                  variable collider type.
30
                colliderType =
                                                                                      P
                  col.gameObject.GetComponent<MaterialSwitchController>
                                                                                      Þ
                  ().GetTerrainType ();
            }
31
32
            // calling the PlayStepSoundMaterialType function
            PlayStepSoundMaterialType();
33
34
            //print in the console the returned value of the
35
                                                                                      ₽
              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 \rightarrow
              value of the variable we switch the surface type switch
44
            //group to the appropriate switch type
            switch (colliderType) {
45
46
            case "Pavement":
                AkSoundEngine.SetSwitch ("Surface_Type", "Pavement", gameObject);
47
48
                //Debug.Log (colliderType);
```

...adiate Interactive Audio\TechNotes\Footstep\_Collider.cs

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