The Viking Village

Technical Notes

Contents

Library SFX

For importing the sounds in Wwise it has been created different type of containers that are mentioned below and in each of these, sounds have been imported either pressing Shift + I or in the Project tab.

Blend Container (Blends all the sounds together)

- Ambient_Day_Night
- amb_day_elements_sum
- amb_night_elements_sum

Random Container (Randomize the sounds inside)

- amb_forest_day_background_loop
- amb_forest_night_background_loop
- amb_day_elements_birds
- amb_day_elements_fauna
- amb_night_elements_atmos
- amb_night_elements_crickets
- Hero_Up
- Hero_Down
- Hero_Grass_Run
- Hero_Grass_Walk
- Hero_Sand_Run
- Hero_Sand_Walk
- Hero_Water_Run
- Hero_Water_Walk
- Hero_Wood_Run
- Hero_Wood_Walk
- Voices_Archer

Sequence Container (Plays the sounds consequently)

- Not used

Switch Container (The game will detect when they have to switch the sound)

- Hero_Jump
- Hero_Land
- Hero_Surface
- Hero_Grass_Step_Type
- Hero_Sand_Step_Type
- Hero_Water_Step_Type
- Hero_Wood_Step_Type

In Wwise we also need to create, in the GameSyncs tab, footsteps switches classifying them in Step_Type and Surface_Type. In the first one we will add a Run switch and a Walk Switch and in the other one we will add one for each type of surface. Then, we need to assign this switched to every switch container dragging them in the correct place. We can find this in the audio type and clicking the switch containers.

Sound SFX (Where most of the sounds are imported. No specific feature)

Music Switch Container (For the game to detect when it is needed music)

- Music_System

Music Playlist Container (If there is more than one music and it is needed to be played sequentially or randomly)

- Horizontal_Music
- House1
- Zone1
- Zone2

Music Segment (Where the music is going to be displayed once it is imported)

- Ambient_Horizontal
- Fantasy_Music_House
- Music_Ambient_Mix
- Battle_Flutes
- EvilHead_Tuba
- Music_Volcanic_Mix

Music Track (The music track itself)

For all the detailed sound effects, it will be attached a document at the end of the technical notes.

Environment Sound Design

For the day – night cycle in Wwise, it has been used a blend track to mix all the sounds from the day ambience and night ambience. Once all the sounds have been imported, we need to create, in the Game Syncs tab, under Game Parameters a Game Parameter and set a range between 0 and 24 simulating the 24h day. Then, in the blend tracks checking the crossfade box and locating the Game Parameter which in this case it is been called Time_Of_Day, we can adjust when every ambience sound it is going to play. We have to do this with both background and elements and it is recommended to adjust in the same way both of them.

Then, in Unity, there is a Game Object called Directional Light. We have to add a new component and insert the Time Controller Script. If we edit the script, we can control the velocity of the day passing or when it starts. We also added an Ak Ambient Script inside the AreaSoundSources object with the ambient event.

At the end of the technical notes, all the scripts will be attached.

Event Sound Design

Play and Stop Events

The events are used to notice the sounds and target which sounds are needed in Unity. When an Event is created, it is suggested to create different Work Units to classify each one. In this case, there are four Work Units.

It has been used two different parameters in events. The first one is the Play event which will play the sound attached to it when it is required. It has been used the Stop event which stops the target chosen.

Below will be displayed which events have been created and what sounds are attached to them. (Event Name (Event) – Sound Attached).

- Ambient
    - Play_Ambient_day_night (Play) – Ambient_Day_Night
    - Stop_Ambient_day_night (Stop) – Ambient_Day_Night
- Character
    - Play_Footstep (Play) – Hero_Surface
    - Play_Footsteps2 (Play) – Hero_Surface
    - Play_Jump (Play) – Hero_Up
    - Play_Jumps2 (Play) – Hero_Up
    - Play_Land (Play) – Hero_Down
    - Play_Lands2 (Play) – Hero_Down
- Music
    - Play_Music (Play) – Music_System
    - Stop_Music (Stop) – Music_System
- SFX
    - Play_Danger (Play) – heart_beat
    - Play_Destroy (Play) – end_game
    - Play_Magic (Play) – magical_sound
    - Play_PickUp (Play) – collect_item
    - Play_torch_sounds (Play) – BAS_amb_torch
    - Play_Voices (Play) – Voices_Archer
    - Play_water_sounds (Play) – Lake_Shore_Waves
    - Play_Welcome (Play) – Paladin3
    - Stop_Danger (Stop) – heart_beat

- Stop_Magic (magical_sound)

SoundBanks

Once all the events are created, we have to create SoundBanks and insert each event inside the SoundBanks renamed accordingly to each work unit. If all the SoundBanks are fulfilled, we can save the project and press Shift + B to generate the SoundBanks, so they can be used in Unity. Select all the Banks we need to generate and click generate selected.

If all this is done, we can go to Unity and add an Empty Game Object that it is going to be called SoundBanks. Here, we can add components called Ak Bank Script and insert on the Bank Name, all the SoundBanks. It is necessary to add a different component every time we want to add a different SoundBank. Once all the SoundBanks are in Unity, we can switch to the FPSController to start adding scripts.

Every time it is changed something in Wwise and it needs to be implemented in Unity, the project has to be saved and SoundBanks have to be generated pressing Shift + B

Switches and Footsteps

The first script we need to add is the FirstPersonController. If we edit the script we can locate, were we need to add this line: AKSoundEngine.PostEvent ("*Event*", gameObject); We will need to add this three times, one for footsteps, one for jumping if we needed and one for landing. Once the lines are added, we can save the script and close the tab. We also added an AK Switch script for every different footstep that we have (In this case we have 8 in total, Run and Walk for every kind of surface). Then, we just need to select all the switches (One on each script). Then, we need to select one by one, all the surface the character will step on, and add a script called Material Switch Controller. We can edit it to change the name of the surfaces to match the ones we named in Wwise and choose which terrain it is every object.

When all the surface is linked, we can add a script in the FPSController named Footsep_Collider which will simply detect when the Character collides with the surface. Inside this script we also need to change the names to match the names with Wwise.

Attenuations and Permanent Sounds

After that, we can start adding specific sounds for objects such torches, water, magical sounds or dangerous heart beats. If we start with the torches, we need to go torch by torch adding and Ak Ambient Script and choosing the correspondent event.

This sounds have an attenuation parameter which is used to decrease the volume the further you are. We did this in Wwise in the ShareSets tab, under the attenuations folder, adding a named attenuation and then in the torch sound in the positioning tab, under attenuation we just selected the attenuation that we need. To edit the distance, we just clicked edit and we can adjust the distance or any other parameter needed.

For the water sounds, we created two Empty Game Objects and we just added an Ak Ambient Script with the water sounds event. We located the game objects in different positions so we can hear the sea from a wider range. Water sounds also have attenuation but, in this case, the distance is bigger, but we followed the same steps as before.

Collisions and Triggered Sounds

We can also use other sounds that will only trigger when the player is near them. To do this, we need to add an object, in this project, it has been used 4 different object that will trigger sounds when the player is near them. For each one of these objects, we need to add a script called Distance to Checkpoint inside the FPSController. (We can't use the same one for each object so we need to duplicate them and edit it so the public class matches with the name of the script.) We have the option to choose the object that will have the central point of the sound. In this case, it is used a Bear Statue that will trigger the magical sound when the player is quite near and the frequency will decrease as the player approaches the object. In the Bear Statue object, we added two Ak Ambient Scripts, one used to Play the sound, selecting to trigger on when it the player enters a sphere collider quite big, and another Ak Ambient script to Stop the sound when the Player leaves the area. We selected the event name correspondent. Another object used is a stone with a thrust sword and we did the same process as before, just changing which sound will play, in this case, the heartbeat, increasing the rate when the Player is approaching the stone.

The other two Distance to Checkpoints are used in music and they will be explained below.

The FPSController does not have more scripts so we can start with objects to collect. It has been added 8 different skulls all around the map with a red light coming from them so they can be detected. Each of them has an Ak Ambient script with the Ak Ambient Trigger on and playing the pickup sound. It is added also an Objects to Collect script so it will detect this object as a one that it is needed to be collected and it will disappear after the player touches it. This it is been done 8 times in total so we have eight items to collect.

After that, an Empty Object is been created renamed as EventSystem, and it is added a Count Objects Script where it can be chosen a next level, in this case, it is been chosen a room, where the game ends, and an object can be destroyed. In this project it is destroyed a Shed and it triggers a sound using the Ak Ambient Script, triggering on when it is destroyed. If the script is edited, the text that will be displayed in the screen when the game is played, can be changed.

Characters

It is been added also some other characters to make the village alive, and it is added an archer called Erika using Mixamo website and importing the character and some animation in unity. In this case, is used a walking in circles animation.

Once the character is in place, it can be used the same method used with the objects with sound, so it can be added a sphere collider, checking the “Is trigger” box and it can be added a mesh collider so the player can not go through the character. If the character needs some phrases or words, they can be added by using again and Ak Ambient Script. In Erika it is used the Play_Voices event. Another character is implemented in the game, and it is a Paladin in front of the front door with an idle animation so he is not static. A phrase it is been recorded so when the player approaches the Paladin, he explains what the player needs to do.

All the scripts will be available at the end of the technical notes

Interactive Music

Before starting with vertical or horizontal music, is required to create some states in the Game Syncs tab. The game will need one for each Music Playlist, no matter if it is vertical music or horizontal. To do this, it is necessary a new state group, and inside of it, create different states renamed the same as our Playlist containers. Then, on the audio track, if the Music System is selected, paths can be added so this is what it is mandatory to do it, one for each state.

Vertical Music

For the vertical music it is been used two ways to implement them, one for just a single track and another to randomize different tracks in the same space. In Wwise, it has been created inside the Music System two different track Playlist containers. In the first one it is just added a single track but it is imported inside a music segment. This segment, once the track is in it, has to be implemented in the Playlist Container named, in this case, Zone1. To do that, it is necessary to use the Interactive Music layout which can be selected pressing F10 or going to the Layouts tab. Once there, under the General Setting, if Zone1 is selected, can be added a New Group and inside Sequence Continuous can be chosen the Music Segment. The sequence continuous can be changed to other parameters but this is used in the next vertical music playlist. It is created another Playlist called Zone2 where three tracks have been implemented and, because the Interactive Music layout is active, the Music segments can be inserted in a new group but, this time. It is used a Random Continuous so the three tracks will be played randomly. After all the tracks are imported, in the Music System switch container, and with the Interactive Music Layout selected, the playlist containers must be dragged in their correspondent path where the object column is. This it can be done either dragging the playlist or choosing the patch via the three dots on the right of each path.

To implement this into Unity, a GameObject has to be created called MusicSystem and inside it can be created, in this project, two new game objects, one called MusicZone1 and the other one MusicZone2. Both of them will need a Box collider with the trigger option selected and an Ak State Script triggering on when entering and the correspondent state for each zone. The MusicZone1 will need an Ak Ambient Script so Music will be played when the game starts.

Horizontal Music

The horizontal music is quite similar to the vertical one but it has a huge different which is that music will change depending on when the player is. In this project, this is used twice. In the first one, back in Wwise, it is created a new playlist container with a music segment and inside it three tracks are imported. Before doing any modification, a Game Parameter is created named Music_Tension with a fifty range. Once this is done, it will be implemented this game parameter in every track and this can be done clicking the track and going to the RTPC track. Under the graphic, if the two arrows are selected the Voice Volume option can be chosen and also the Game Parameter. After, it is time to adjust the game parameter using the graphic above. This is used in all three different tracks just changing the distance from the player where it can be heard.

The other horizontal music is quite different because it only has one track but the process is similar because it is also implemented a game parameter called HouseMusic. This time, the RTPC it is not in a linear line but in a logarithmic curve so the effect is more realistic. After this, it has to be done the same as before, adding the segments in a new group inside the playlists and creating two paths for the horizontal music and for the House Music and adding the playlists track inside the paths.

After generating the SoundBanks and jumping back to Unity it can be created two Game Objects with a box collider with the trigger box selected and an Ak State script triggering it when entering and the State Name using the different horizontal music created for each game object. For the horizontal music zone, which will be located over a mountain with a tower on top, the FPSController will need another Distance to Checkpoint and now it will be selected the ruined tower so when the player approaches it the music will start playing, and as near the tower the player is, music will sound louder and instruments will come in. Another Distance to Checkpoint is necessary and now a house where the Game Object with the House state is has to be selected so, again, when the player is near this house, some music will start playing, simulating it comes from inside the house.

Interactive Mix

RTPC

RTPC stands for Real Time Parameter Controller and they have been implemented before explaining their use and how do they work but, to summarize, the RTPC are used to control different parameters such distance which is the only one used in this project and this can be used to decrease volume or change other parameters when the player is approaching or leaving.

Reverb Zones

Wwise also has a Master-Mixer so buses can be added and, in fact, there are two different buses that can be added. One is the Auxiliar Bus, which is the one it is used now, and the other one is the Audio Bus, which will be explained how it has been implemented after. The Auxiliar bus used in this project is for a reverb effect in a specific zone so it is created an auxiliar bus and in the effects tab, it has been implemented a Wwise RoomVerb, very subtle, for when the player will enter the ruined tower. It is chosen an exterior preset for it. To implement this into Unity, the Ruined Tower has to be selected and it is necessary to add a capsule collider (to match the tower shape), and an Ak Environment Script where it can be selected the AuxBus. The capsule collider trigger box needs to be selected to be sure it will work.

Audio Buses

Audio buses are a longer process but are very useful to control different tracks at the same volume. It can be created as many Audio Buses as wanted and, in this project, there are 5 different Buses with tracks inside of them. All the buses, including the auxiliar one, are inside a Master Audio Bus. To insert the tracks or the containers inside each bus, it has to be selected the container it is wanted to be sent to the bus, and on the output bus, in General Settings, just path it to the bus wanted.

- Ambient Bus
    - Ambient_day_night
- Character Bus
    - Hero_Jump
    - Hero_Land
    - Hero_Surface
- MusicBus
    - Music_System
- SFXBus
    - BAS_amb_torch
    - collect_item
    - end_game
    - heart_beat
    - Lake_Shore_Waves
    - magical_sound
- VoiceBus
    - Voices_Archer
    - Paladin3

Once all the tracks and containers are in the busses wanted if it is pressed F8 or, in the layout tab it is selected the mixer layout, it can be created a new mix session just selecting the two arrows on the top. Once the mix session is named, it is recommended to create a Soundcaster (below the Mixing Desk) with the same name. Once both are created, for example, under the Bus_Mix and Bus_Cast names, all the busses can be dragged in both spaces so they can be controlled all together. It can be also dragged the Master Audio Bus to control everything. In this project, this process it is been done 5 times, to control: Bus, Character, Ambient, Music and SFX. This is helpful to level every sound so there is none louder or quieter.

In this project it is not used the auto-ducking but it is worth it to know how it works because it is similar to Side-Chain in another DAWs because it lows the level when another sound is playing.

Game Engine Integration

Scripts

A lot of scripts are used in the unity and they are explained during the technical notes but as said before, all the scripts will be attached at the end of this document.

Video Captured Gameplay

After all the details polished and touch up some things that need some changes or some improvements in the sound, the video gameplay is captured with OBS recorder and then a Voice Over using ProTools and a NT1-A Microphone. The computer where is recorded is not the best at all, and there are a lot of problems with player movement because the PC is really bad. It is done the best way and it has been recorded multiple times until the final one which is the cleanest. It is not the best quality but it is the best it can be done in this situation. Sorry for the inconveniences.

The blog is created inside my own webpage where I post all my projects, either if they are from university or personal projects, and I just wanted to attach this project inside the web.

BlogSite

Transcription

Also, here it is the transcribed text from the video:

Hello, I'm Max, and this is my Interactive Audio Project, integrating Wwise and Unity. It is midnight so we can here some hostile animals in the forest. And before entering we can see we are coming from a tent outside the village.

This knight, introduce us what we have to look for inside the village and on the top-left of the screen we can see that we have 8 skulls left to pick up. As we can see, while we hear the waves on the shore, we have 2 skulls right in front of the front door and the sound used is a neutral sound because I did not want to use a bright sound to pick up skulls.

Also, as we heard before, this woman has four different introducing phrases that will play randomly. The day is starting and the sun is rising so we can start hearing the sound of the birds and other animals in the nature

If we talk about the character steps, we will hear four different kind of steps which are, wood steps, sand steps, grass steps, and water steps, which we will hear later on. We can also hear when the player jumps and lands, with for different sounds for each one. Still talking about the character, each surface has 8 different footsteps sounds, 4 for walking and 4 for jumping, which are a little bit faster. It is worth mentioning that mostly every surface has its own surface change.

If we listen carefully, each torch has it's own sound at you will hear them louder as near as you are from one torch.

Here I have some problems, but when we pass this point, as we can hear, another music will play because we are entering another part of the village.

Music stops abruptly here, and I could not fix this problem, but we hear a calm song instead. We also hear a weird but magical sound that we don’t know where it comes from. But when we are approaching the bear, we notice the frequencies are going higher. And we pick up our 5th skull. 3 to go.

At this point, my character starts doing weird stuff, and I could not fix it, but it depends on the gameplay. It is not a box collider problem because we will see the character will go through without any problem. We hear the music going louder as we approach the house and we will verify that when we pass this point. Now we can confirm the music is coming from inside the house and heartbeat starts appearing, when we look for one of the last skulls, we will find a sword on a rock, and our heartbeat will increase it’s rate.

Here we will here when the character starts running. This time, the woman didn’t say anything to us because we didn’t go close enough.

Before picking up the last skull, we will have to do some unnecessary parkour because you can walk on water. And after climbing this mountain, where you can hear clearly all the jumping and landing sounds, which seems that the player is doing a lot of effort, we arrive to the last skull. We can also look the entire village. The animals start to mix with some night animals because the sun is setting. And when we enter the ruined tower, we can take the last skull.

Finally, you heard a phone ringing, and that is because it simulates that the person playing the Viking village is inside a virtual reality, and he stops playing, inside the game.

Clip Name	Source	Details	Location in Wwise	Edits	Mixing	RTPC
Music Work Unit						
1_Choir	Given by John McGowan	Choir Singing	Music_System/ Horizontal_Music1/ Ambient_Horizontal			Music_Tension: Voice Volume going from +6,0 dB to -200 dB in 40 units distance
2_Sitar		Sitar playing				Music_Tension: Voice Volume going from +6,0 dB to -200 dB in 14 units distance
3_BigPno		Piano playing				Music_Tension: Voice Volume going from +6,0 dB to -200 dB in 25 units distance
Fantasy_Music_House	FreeMusicArchive	Guitar song	Music_System/ House1/ Music_House		Wwise Parametric EQ: Low Pass Filter_20 kHz	HouseMusic: Voice Volume going from 2,5 dB to -200 dB in 50 units distance
Music_Ambient_Mix	Given by John McGowan	Tense Music	Music_System/ Zone1/ Music_Ambient_Mix	Volume: -6 dB		
Battle_Flutes	Given by John McGowan	Calm Music	Music_System/ Zone2/ Music_Ambient_Mix			
EvilHead_Tuba		Battle Music				
Music_Volcanic_Mix		Epic Music				

Clip Name	Source	Details	Location in Wwise	Edits	Mixing	RTPC
SFX Work Unit						
vo_archer_hello1	Freesound.org	Women saying hello in different ways	Voices_Archer	**Volume:** -5 dB		
vo_archer_hello2						
vo_archer_hello3						
vo_archer_hello4						
BAS_amb_torch	Given by John McGowan	Torch burning	SFX	**Torch_Attenuation:** Output volume decreasing from distance 0,0 to 15,0 in a Logarithmic(Base 3) curve		
collect_item	Freesound.org	Someone picking an object				
end_game		A telephone ringing		**Volume:** -11 dB		
heart_beat		Heart beating			**Wwise Time Stretch** 112-47	Danger: Range 50-0
Lake_Shore_Wave	Given by John McGowan	Sea waves sound in the shore		**Water_attenuation:** Outpus volume decreasing from distance 0,0 to 50,0 in a Logarithmic(Base 3) curve **Volume:** -5 dB		
magical_sound		mysterious sound		**Volume:** -5 dB	**Wwise Flanger:** Flanging/Deep. LFO Frequency 20 - 0,002	Magic: Range 50-0
Paladin3	Recorded myself with H2n Zoom	Welcome phrase			**Wwise Parametric EQ:** Static_Sibilance_Remover_Male **Wwise_Pitch_Shifter:** Octave_Down	

Clip Name	Source	Details	Location in Wwise	Edits	Mixing
Character Work Unit					
Jumps	H2n Zoom recording my voice	Simulating a jump effort sound	Hero_Jump/Hero_Up		
Jumps_01					
Jumps_02					
Jumps_03					
Land_01		Simulating when you land from a place	Hero_Land/Hero_Down		
Land_02					
Land_03					
Land_04					
Land_05					
Footsteps_Grass	Park called "Parc de la Oreneta" in Barcelona	Footsteps in a grassy surface	Hero_Surface/ Hero_Grass_Step_Type/ Hero_Grass_Run		Low-Pass filter: 29 High-Pass filter: 36
Footsteps_Grass_01					
Footsteps_Grass_02					
Footsteps_Grass_03					
Footsteps_Grass_04			Hero_Surface/ Hero_Grass_Step_Type/ Hero_Grass_Walk		Low-Pass filter: 29 High-Pass filter: 36
Footsteps_Grass_05					
Footsteps_Grass_06					
Footsteps_Grass_07					
Footsteps_Sand	Freesound.org	Footsteps in a sandy surface	Hero_Surface/ Hero_Sand_Step_Type/ Hero_Sand_Run	Volume: -6 dB	
Footsteps_Sand_01					
Footsteps_Sand_02					
Footsteps_Sand_03					
Footsteps_Sand_04			Hero_Surface/ Hero_Sand_Step_Type/ Hero_Sand_Walk		
Footsteps_Sand_05					
Footsteps_Sand_06					
Footsteps_Sand_07					
Footsteps_Water	Freesound.org	Footsteps in a swamp simulating when you walk on water	Hero_Surface/ Hero_Water_Step_Type/ Hero_Water_Run	Volume: -14 dB	
Footsteps_Water_01					
Footsteps_Water_02					
Footsteps_Water_03					
Footsteps_Water_04			Hero_Surface/ Hero_Water_Step_Type/ Hero_Water_Walk		
Footsteps_Water_05					
Footsteps_Water_06					
Footsteps_Water_07					
Footsteps_Wood	Wood planks in "Sant Feliu del Racó", near Barcelona	Footsteps in a woody surface on exterior	Hero_Surface/ Hero_Wood_Step_Type/ Hero_Wood_Run	Volume: -11 dB	
Footsteps_Wood_01					
Footsteps_Wood_02					
Footsteps_Wood_03					
Footsteps_Wood_04			Hero_Surface/ Hero_Wood_Step_Type/ Hero_Wood_Walk		
Footsteps_Wood_05					
Footsteps_Wood_06					
Footsteps_Wood_07					
All my recordings are recorded using a H2n Zoom handy recorder					

Clip Name	Source	Details	Location in Wwise	Edits
Ambient Work Unit				
amb_forest_day_background	Given by John McGowan	Forest ambience with some birds in the background	Ambient_day_night/ amb_forest_dat_background_loop	Volume: -9 dB
amb_forest_night_background		Forest ambience at night with crickets and wind	Ambient_day_night/ amb_forest_night_background_loop	Volume: - 14 dB
amb_day_element_02_01		Bird sound	amb_day_elements_sum/ amb_day_elements_birds	Pitch randomizer between -51 and 101 Volume: -9 dB
amb_day_element_02_02		Bird sound	amb_day_elements_sum/ amb_day_elements_birds	
amb_day_element_02_03		Bird sound	amb_day_elements_sum/ amb_day_elements_birds	
amb_day_element_02_04		Bird sound	amb_day_elements_sum/ amb_day_elements_birds	
amb_day_element_01_01		Tree branch getting broke	amb_day_elements_sum/ amb_day_elements_fauna	
amb_day_element_01_02		Tree branch getting broke	amb_day_elements_sum/ amb_day_elements_fauna	
amb_day_element_01_03		Tree branch getting broke	amb_day_elements_sum/ amb_day_elements_fauna	
amb_night_element_03_01		Howling wolf	amb_night_elements_sum/ amb_night_elements_atmos	Volume -11 dB
amb_night_element_03_02		Some animal screaming	amb_night_elements_sum/ amb_night_elements_atmos	
amb_night_element_03_03		Animal doing sounds	amb_night_elements_sum/ amb_night_elements_atmos	
amb_night_element_03_04		Some animal screaming	amb_night_elements_sum/ amb_night_elements_atmos	
amb_night_element_01_01		Cricket sound	amb_night_elements_sum/ amb_night_elements_crickets	Volume: +12 dB
amb_night_element_01_02		Cricket sound	amb_night_elements_sum/ amb_night_elements_crickets	
amb_night_element_01_03		Cricket sound	amb_night_elements_sum/ amb_night_elements_crickets	
amb_night_element_01_04		Cricket sound	amb_night_elements_sum/ amb_night_elements_crickets	
Added a Wwise Silence in every group of elements with a 5 seconds durantion by default and randomizing it betweem 2 and 6,5 secs.				

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class CountObjects : MonoBehaviour
{
    public string nextLevel;
    public GameObject objToDestroy;
    GameObject objUI;

    // Use this for initialization
    void Start()
    {
        //look for the text object in the UI called ObjectNum
        objUI = GameObject.Find("ObjectNum");
    }
    // Update is called once per frame
    void Update()
    {
        //convert the numbers to string and send to the text object to update
        objUI.GetComponent<Text>().text = (ObjectsToCollect.objects.ToString()) ↩
           + " Skulls left to pick up";

        if (ObjectsToCollect.objects == 0)
        {

            //load a new level once all objects have been picked up
            Application.LoadLevel(nextLevel);
            //destroy the chosen object once the total reaches 0
            Destroy(objToDestroy);
            objUI.GetComponent<Text>().text = "All objects collected.";

        }

    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ObjectsToCollect : MonoBehaviour
{
    public static int objects = 0;
    // Use this for initialization
    // let the count objects script know that this object is
    //  part of the collection and should be counted
    void Awake()
    {
        objects++;
    }

    // Update is called once per frame
    void OnTriggerEnter(Collider plyr)
    {
        //if the tagged FPSController 'Player' collides with an object, take it
           away from the total
        if (plyr.gameObject.tag == "Player")
            objects--;
        gameObject.SetActive(false);
    }
}

```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;
using UnityStandardAssets.Utility;
using Random = UnityEngine.Random;

public class DistanceToCheckpoint : MonoBehaviour
{
    // Reference to checkpoint position
    [SerializeField]
    private Transform checkpoint;


    //Serialization is the process of taking an object in ram (classes, fields, 
       etc...)
    //and making a disk representation of it which can be recreated at any 
      point in the future.



    // Calculated distance value
    private float distance;



    // Update is called once per frame
    void Update ()
    {
        // calculate distance value between character and checkpoint
        distance = (checkpoint.transform.position - 
          transform.position).magnitude;

        // set parameter from Wwise game parameter to scaled distance value
        AkSoundEngine.SetRTPCValue("Magic", distance);

        //Debug.Log(message: "Distance to checkpoint is " + distance);

    }
}

```

```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Footstep_Collider : MonoBehaviour {


    private string colliderType;


    // Use this for initialization
    void Start () {

        AkSoundEngine.SetSwitch ("Surface_Type", "Grass", gameObject);
    }

    // Update is called once per frame
    void Update () {

    }

    //this function dectects if there is a collision between the player controller and a game object
    //and calls the function GetTerrainType which retairns the terrain type that has been set in the enumaration mode of that object.
    //then it calls the PlayStepSoundMaterial method which is using a switch stament to set the Wwise Switches.

    void OnControllerColliderHit (ControllerColliderHit col){
        if (col.gameObject.GetComponent<MaterialSwitchController>()) {

            //Store what the GetTerrainType returns and store is in the variable collider type.
            colliderType = col.gameObject.GetComponent<MaterialSwitchController>().GetTerrainType ();
        }
        // calling the PlayStepSoundMaterialType function
        PlayStepSoundMaterialType();

        //print in the console the returned value of the MaterialSwitchController
        //Debug.Log (colliderType);

    }


    void PlayStepSoundMaterialType()
    {
        //checks the content of the colliderType variable and depending on the value of the variable we switch the surface type switch
        //group to the appropriate switch type
        switch (colliderType) {
        case "Grass":
            AkSoundEngine.SetSwitch ("Surface_Type", "Grass", gameObject);
            //Debug.Log (colliderType);
```

```
                break;
            case "Wood":
                AkSoundEngine.SetSwitch ("Surface_Type", "Wood", gameObject);
                //Debug.Log (colliderType);
                break;
            case "Water":
                AkSoundEngine.SetSwitch ("Surface_Type", "Water", gameObject);
                //Debug.Log (colliderType);
                break;
            case "Sand":
                AkSoundEngine.SetSwitch ("Surface_Type", "Sand", gameObject);
                //Debug.Log (colliderType);
                break;

            }
        }

}

```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MaterialSwitchController : MonoBehaviour {


    public enum Mode {Grass, Wood, Water, Sand}
    public Mode terrainType;


    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }

    public string GetTerrainType(){

        string typeString = "";

        switch (terrainType) {

        case Mode.Grass:
            typeString = "Grass";
            break;
        case Mode.Wood:
            typeString = "Wood";
            break;
        case Mode.Water:
            typeString = "Water";
            break;
        case Mode.Sand:
            typeString = "Sand";
            break;

        }

        return typeString;
        //Debug.Log (typeString);
    }
}



```

```csharp
using System;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;
using UnityStandardAssets.Utility;
using Random = UnityEngine.Random;

namespace UnityStandardAssets.Characters.FirstPerson
{
    [RequireComponent(typeof (CharacterController))]
    [RequireComponent(typeof (AudioSource))]
    public class FirstPersonController : MonoBehaviour
    {
        [SerializeField] private bool m_IsWalking;
        [SerializeField] private float m_WalkSpeed;
        [SerializeField] private float m_RunSpeed;
        [SerializeField] [Range(0f, 1f)] private float m_RunstepLenghten;
        [SerializeField] private float m_JumpSpeed;
        [SerializeField] private float m_StickToGroundForce;
        [SerializeField] private float m_GravityMultiplier;
        [SerializeField] private MouseLook m_MouseLook;
        [SerializeField] private bool m_UseFovKick;
        [SerializeField] private FOVKick m_FovKick = new FOVKick();
        [SerializeField] private bool m_UseHeadBob;
        [SerializeField] private CurveControlledBob m_HeadBob = new CurveControlledBob();
        [SerializeField] private LerpControlledBob m_JumpBob = new LerpControlledBob();
        [SerializeField] private float m_StepInterval;
        [SerializeField] private AudioClip[] m_FootstepSounds;    // an array of footstep sounds that will be randomly selected from.
        [SerializeField] private AudioClip m_JumpSound;           // the sound played when character leaves the ground.
        [SerializeField] private AudioClip m_LandSound;           // the sound played when character touches back on ground.

        private Camera m_Camera;
        private bool m_Jump;
        private float m_YRotation;
        private CameraRefocus m_CameraRefocus;
        private Vector2 m_Input;
        private Vector3 m_MoveDir = Vector3.zero;
        private CharacterController m_CharacterController;
        private CollisionFlags m_CollisionFlags;
        private bool m_PreviouslyGrounded;
        private Vector3 m_OriginalCameraPosition;
        private float m_StepCycle;
        private float m_NextStep;
        private bool m_Jumping;
        private AudioSource m_AudioSource;

        // Use this for initialization
        private void Start()
        {
            m_CharacterController = GetComponent<CharacterController>();
            m_Camera = Camera.main;
            m_OriginalCameraPosition = m_Camera.transform.localPosition;
            m_CameraRefocus = new CameraRefocus(m_Camera, transform, 
              m_Camera.transform.localPosition);
            m_FovKick.Setup(m_Camera);
            m_HeadBob.Setup(m_Camera, m_StepInterval);
            m_StepCycle = 0f;
            m_NextStep = m_StepCycle/2f;
            m_Jumping = false;
            m_AudioSource = GetComponent<AudioSource>();
            m_MouseLook.Init(transform , m_Camera.transform);
        }


        // Update is called once per frame
        private void Update()
        {
            RotateView();
            // the jump state needs to read here to make sure it is not missed
            if (!m_Jump)
            {
                m_Jump = CrossPlatformInputManager.GetButtonDown("Jump");
            }

            if (!m_PreviouslyGrounded && m_CharacterController.isGrounded)
            {
                StartCoroutine(m_JumpBob.DoBobCycle());
                PlayLandingSound();
                m_MoveDir.y = 0f;
                m_Jumping = false;
            }
            if (!m_CharacterController.isGrounded && !m_Jumping && 
              m_PreviouslyGrounded)
            {
                m_MoveDir.y = 0f;
            }

            m_PreviouslyGrounded = m_CharacterController.isGrounded;
        }


        private void PlayLandingSound()
        {
            //m_AudioSource.clip = m_LandSound;
            //m_AudioSource.Play();
            //m_NextStep = m_StepCycle + .5f;
            AkSoundEngine.PostEvent ("Play_Lands2", gameObject);
        }


        private void FixedUpdate()
        {
            float speed;
            GetInput(out speed);
            // always move along the camera forward as it is the direction 
              that it being aimed at
            Vector3 desiredMove = m_Camera.transform.forward*m_Input.y + 
              m_Camera.transform.right*m_Input.x;

            // get a normal for the surface that is being touched to move along it
            RaycastHit hitInfo;
            Physics.SphereCast(transform.position, m_CharacterController.radius, Vector3.down, out hitInfo,
                               m_CharacterController.height/2f);
            desiredMove = Vector3.ProjectOnPlane(desiredMove, hitInfo.normal).normalized;

            m_MoveDir.x = desiredMove.x*speed;
            m_MoveDir.z = desiredMove.z*speed;


            if (m_CharacterController.isGrounded)
            {
                m_MoveDir.y = -m_StickToGroundForce;

                if (m_Jump)
                {
                    m_MoveDir.y = m_JumpSpeed;
                    PlayJumpSound();
                    m_Jump = false;
                    m_Jumping = true;
                }
            }
            else
            {
                m_MoveDir += Physics.gravity*m_GravityMultiplier*Time.fixedDeltaTime;
            }
            m_CollisionFlags = m_CharacterController.Move(m_MoveDir*Time.fixedDeltaTime);

            ProgressStepCycle(speed);
            UpdateCameraPosition(speed);
        }


        private void PlayJumpSound()
        {
            AkSoundEngine.PostEvent ("Play_Jumps2", gameObject);
            //m_AudioSource.clip = m_JumpSound;
            //m_AudioSource.Play();
        }



        private void ProgressStepCycle(float speed)
        {
            if (m_CharacterController.velocity.sqrMagnitude > 0 && (m_Input.x != 0 || m_Input.y != 0))
            {
                m_StepCycle += (m_CharacterController.velocity.magnitude + (speed*(m_IsWalking ? 1f : m_RunstepLenghten)))*
                             Time.fixedDeltaTime;
            }

            if (!(m_StepCycle > m_NextStep))
            {
                return;
            }

            m_NextStep = m_StepCycle + m_StepInterval;

            PlayFootStepAudio();
        }


        private void PlayFootStepAudio()
        {
            if (!m_CharacterController.isGrounded)
            {
                return;
            }

            AkSoundEngine.PostEvent ("Play_Footsteps2", gameObject);


            // pick & play a random footstep sound from the array,
            // excluding sound at index 0
            //int n = Random.Range(1, m_FootstepSounds.Length);
            //m_AudioSource.clip = m_FootstepSounds[n];
            //m_AudioSource.PlayOneShot(m_AudioSource.clip);
            // move picked sound to index 0 so it's not picked next time
            //m_FootstepSounds[n] = m_FootstepSounds[0];
            //m_FootstepSounds[0] = m_AudioSource.clip;
        }


        private void UpdateCameraPosition(float speed)
        {
            Vector3 newCameraPosition;
            if (!m_UseHeadBob)
            {
                return;
            }
            if (m_CharacterController.velocity.magnitude > 0 && m_CharacterController.isGrounded)
            {
                m_Camera.transform.localPosition =
                    m_HeadBob.DoHeadBob(m_CharacterController.velocity.magnitude +
                                      (speed*(m_IsWalking ? 1f : m_RunstepLenghten)));
                newCameraPosition = m_Camera.transform.localPosition;
                newCameraPosition.y = m_Camera.transform.localPosition.y - m_JumpBob.Offset();
            }
            else
            {
                newCameraPosition = m_Camera.transform.localPosition;
                newCameraPosition.y = m_OriginalCameraPosition.y - m_JumpBob.Offset();
            }
            m_Camera.transform.localPosition = newCameraPosition;

            m_CameraRefocus.SetFocusPoint();
        }


        private void GetInput(out float speed)
        {
            // Read input
            float horizontal = CrossPlatformInputManager.GetAxis("Horizontal");
            float vertical = CrossPlatformInputManager.GetAxis("Vertical");

            bool waswalking = m_IsWalking;

#if !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 > 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 && m_UseFovKick && m_CharacterController.velocity.sqrMagnitude > 0)
            {
                StopAllCoroutines();
                StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp() : m_FovKick.FOVKickDown());
            }
        }


        private void RotateView()
        {
            m_MouseLook.LookRotation (transform, m_Camera.transform);
            m_CameraRefocus.GetFocusPoint();
        }


        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)
            {
                return;
            }

            if (body == null || body.isKinematic)
            {
                return;
            }
            body.AddForceAtPosition(m_CharacterController.velocity*0.1f,
              hit.point, ForceMode.Impulse);
        }
    }
}
```