
BACHELOR THESIS

Mr.
Omar Sharaki

VR Learning Game for Identifying Bird Vocalizations

Ithaca, NY, 2017

BACHELOR THESIS

VR Learning Game for Identifying Bird Vocalizations

Author:

Mr. Omar Sharaki

Course of Study:

Applied Computer Science

Seminar Group:

IF14wS-B

First examiner:

Prof. Dr. Marc Ritter

Second examiner:

Dr. Holger Klinck

Submission:

Mittweida, 21.12.2017

Defense:

Mittweida, 2018

Bibliographic Description:

Sharaki, Omar:

VR Learning Game for Identifying Bird Vocalizations. - 2017. - 77 pages content.

Mittweida, Hochschule Mittweida – University of Applied Sciences, Faculty of Applied Computer Sciences & Biosciences, Bachelor thesis, 2017

Abstract:

The following is a description and outline of the work done at the Cornell Lab of Ornithology developing Nation Feathers VR, a virtual reality game for learning about bird calls and songs. The goal was to develop a game which is intuitive, educational and entertaining. Furthermore, the software needed to be structured in a way that allows for feasible future expansion. This required careful data saving and retrieval. The game gives the player an opportunity to learn and apply that knowledge, all while maintaining a shorter runtime in order to reduce the total time spent in the virtual world. This is meant to prevent any discomfort to the player that may result from extended use of the VR headset.

Contents

Contents	I
List of Figures	III
List of Tables	VII
Abbreviations	VIII
1. Motivation.....	1
1.1. About Cornell Lab of Ornithology and Bioacoustics Research Program	4
1.2. Task.....	8
1.3. Structure of this Thesis	9
2. State of the Art	11
2.1. Virtual Reality.....	11
2.1.1. History of Virtual Reality	11
2.1.2. Technical details.....	14
2.1.3. Immersion.....	17
2.1.4. Ongoing research	20
2.1.5. VR in Game Engines	21
2.2. Tools for VR development in Unity	22
2.3. Spectrograms	24
2.4. Visualization and Sound	25
2.4.1. 3D Models.....	26
2.4.2. Materials, Shaders and Textures	28
2.4.3. 3D Sounds	30
2.5. Conclusion.....	33
3. Game Concept	35
3.1. Scenes.....	35
3.1.1. Menu Scene	35
3.1.2. Intro Scene.....	35
3.1.3. Learn Scene.....	37
3.1.4. Challenge Scene	38
3.2. System Structure.....	38
3.3. Birds.....	40
3.4. Conclusion.....	44
4. Implementation Details.....	45

4.1. Database	45
4.2. User Interface.....	47
4.2.1. Infobox.....	47
4.2.2. Signpost	49
4.2.3. Reticle and Player Gaze	49
4.2.4. Navigation Hint	52
4.3. Environment.....	55
4.3.1. Grass	55
4.3.2. Trees	56
4.3.3. Boundaries	58
4.4. Animation.....	59
4.5. Creating the Spectrograms	60
4.6. Playing the Spectrograms	63
4.7. Custom Inspector	64
4.8. Lighting	65
5. Evaluation.....	67
5.1. Training Effectiveness and Enjoyment	67
5.2. Success Verdict.....	71
6. Conclusion	75
Bibliography.....	79
Installation Instructions	82
Declaration of Authorship.....	83

List of Figures

Figure 1. Left: virtual endoscopy with internal view of the transverse colon (Satava & Jones, 1998). Middle: abdomen simulator (Székely & Satava, 1999). Right: cross section of skull – Visible Human Male (Waldby, 1997).	4
Figure 2. View of Cornell Lab of Ornithology from across the pond (Arch Daily).	7
Figure 3. Left: disassembled ARU. Middle: MARUs in transit to a deployment location. Right: Swift unit strapped to a tree. (Bioacoustics Research Program)	8
Figure 4. Battle of Borodino by Franz Roubaud.....	11
Figure 5. Top: Brewster-type stereoscopic viewer. Bottom: Stereoscopic image of Boston.....	12
Figure 6. Top row (Left to right): The Sword of Damocles, VCASS, CAVE. Bottom row (Left to right): BOOM, Sensorama, VVED, Videoplace.....	14
Figure 7. Shows six degrees of freedom.	15
Figure 8. FOV in humans. (a) vertical, (b) horizontal. (Mazuryk & Gervautz, 1996, p. 16)	16
Figure 9. Two feeds sent to a single mobile device screen.....	17
Figure 10. Modified Necker Cube resembling a house. It is much more difficult to see the house from below than from above, as in normal experiences houses are viewed from above or from straight on. (Pspotka, 1995, p. 411)	19
Figure 11. The profiler window. The window’s upper section shows the resources (CPU, Rendering, etc.) being used by the game on a timeline containing a graph which shows data usage. The data at the bottom of the window correspond to a specific frame. The frame can be changed by moving the white vertical line across the timeline.....	23
Figure 12. Labeled image of a northern cardinal’s spectrogram showing how spectrograms are to be interpreted. (Bird Academy).....	25
Figure 13. Significance of each sense for experiencing VR. (Mazuryk & Gervautz, 1996, p. 15).....	26
Figure 14. A visualization of linear perspective. The colored bars are for measurement, while the red line represents the horizon line (i.e. the viewer’s eye level). VP denotes the vanishing point (i.e. the point on the horizon line where parallel lines seem to converge). The circle represents the field of view (FOV). The points where the lines meet the FOV are diagonal vanishing points; their effect can be perceived, for example, when the viewer is facing the corner of a building and its sides seem to get smaller or	

more elongated as they recede towards the vanishing points. (Paquette, 2008, p. xiii)	27
Figure 15. Vertices, edges, and the resulting triangular face represented in 3D space. (Paquette, 2008, p. 7)	28
Figure 16. A scene with 3D ground and wall objects. The left image shows the scene before texturing with the default material applied, while the right image shows what the scene looks like after texturing. (Hocking, 2015, p. 75)	30
Figure 17. A source-medium-receiver model. The dotted line sections numbered 1 through n represent the physical, neurological, or perceptual transformations that each element contains.in the communication of the spatial location of a sound source. For a 3D sound system, the question is to what degree the 1 st element of the source equals the n th element of the receiver (Begault, 2000, p. 5)	31
Figure 18. Highly simplified schematic overview of the auditory system. (A) Sound. (B) Pinna. (C) Meatus. (D) Eardrum. (E) Ossicles. (F) Oval window. (G) Cochlea. (H) Basilar membrane. (Begault, 2000, p. 7)	32
Figure 19. Screenshot from main menu	36
Figure 20. From left to right: 3D object text, world space text, screen space text	37
Figure 21. From left to right: Screenshots of Intro, Learn and Challenge Scenes	38
Figure 22. Overview of the game's scene structure and its scripts. The arrows show how the player can move between scenes in the game. On the right is shown how the scripts are assigned to the specific scenes. <i>GameController</i> for example, the most general script, is assigned to all scenes. <i>AudioController</i> on the other hand is very specialized and is assigned only to the Learn scene.	39
Figure 23. Range map of the Dark-eyed Junco showing year-round, winter, and summer presence of the species. (All About Birds)	42
Figure 24. Birds featured in the game and their real attributes. Column T.o.Y. shows the time of year when the bird species can be found in the Northeastern US. T.o.D. shows whether the bird is diurnal or nocturnal (disregarding migration patterns). Habitat is a strict indication of the bird's preferred habitat; in reality it is possible that the bird is present in other environments. (All About Birds)	43
Figure 25. Depiction of a more diverse landscape that could make full use of the database's bird attributes. Player shown as purple capsule.	47

Figure 26. Screenshots from Challenge Scene. From left to right: Image of silhouetted, generic bird before correct identification and full Infobox featuring a tufted titmouse. Images also show animated reticle and partially visible signpost.	48
Figure 27. Screenshot of signpost from Challenge Scene.	50
Figure 28. Different pointers used in the Gear VR. Left to right: Normal reticle used in the game, animated reticle used in the game, particles used in game’s first iteration, pointer used in Oculus Store.	51
Figure 29. Algorithm to determine shortest path for footprints. α : Initial direction, β : Final direction, μ : smallest difference between α and β	52
Figure 30. Screenshots from editor view that demonstrate how footprints take shortest path. Blue arrow shows direction where player is facing. Tree shown in bottom right corner is target tree.....	54
Figure 31. Left: interpolation path if $ \alpha - \beta < 180^\circ$, right: interpolation paths if $ \alpha - \beta > 180^\circ$ (Green is correct, red is wrong). Red and blue dots denote α and β respectively. α : Initial direction, β : Final direction	54
Figure 32. Angles as they are interpreted by Unity in the algorithm’s use case.....	54
Figure 33. Left two images: Grass blade; mesh shown left. Right two images: Double plane grass; mesh shown left	56
Figure 34. Demonstration of the main stages in the trees’ progression. Meshes shown in upper row. From left to right: Stylistic opaque tree, tree made up of transparent planes, semi-transparent tree with cutout edges and opaque blob (currently in use)..	57
Figure 35. Barriers as they are used in different scenes. Top to bottom: Fence used in first iteration’s Learn Scene, terrain barrier currently used in Learn and Challenge Scenes, Cliffs used in Intro Scene.....	59
Figure 36. Part of the image sequence of a flying blue jay.	60
Figure 37. The effects of discrete Fourier transform (DFT) and the window function. Each panel shows a signal on the left and its spectrum on the right. (a) A single record of an untapered sinusoidal signal flanked by sidelobes. (b) A single record of a sinusoidal signal multiplied by a “taper” or window function has smaller sidelobes. (Charif, Waack, & Stickman, 2010, p. 343)	61
Figure 38. Close-up of portion of two spectrogram views of a signal digitized at a sample rate of 44.1 kHz. The upper view is smoothed while the lower view is unsmoothed. For both views, window type = Hann, window size = 512 samples, and overlap = 50%. (Charif, Waack, & Stickman, 2010, p. 135)	62

Figure 39. Part of the inspector window of an object with the Mesh Combiner editor script attached to it. If the script is left unaltered, this part of the inspector window would contain nothing but the name of the script. However, for the purposes of combining game object meshes in edit mode, the editor script was altered to add two functional buttons which, when clicked, combine the meshes of all the game object's children into a single mesh that is added to the game object. 65

Figure 40. Values from **Table 6** represented as graphs showing potential correlations. Each bar represents a participant. Top to bottom: 1) How birding experience affects training effectiveness. 2) How enjoyment varies across age groups. 3) How ease of use varies across age groups. 4) How ease of use affects enjoyment. Number labels indicate duplicate values. In the bar graphs, group values are arranged horizontally in ascending order. 72

List of Tables

Table 1. Timeline of advances in VR from the 60s to the 90s. (Mazuryk & Gervautz, 1996, pp. 2-3)	13
Table 2. A verbal illustration of the differences between NSH and VSH in the source-medium-receiver model. (Begault, 2000, pp. 4-5).....	31
Table 3. Birds database table, red denotes primary key	46
Table 4. Sounds database table, red denotes primary key, blue denotes foreign key ..	46
Table 5. Sidelobe rejection for Raven’s five window types. The sidelobe rejection for each window type is expressed as the height of the highest sidelobe relative to the peak of the main lobe. (Charif, Waack, & Stickman, 2010, pp. 343-344)	62
Table 6. Player feedback and performance. Each row represents a player. Main attributes of success are given by <i>training effectiveness</i> , <i>entertainment</i> , and <i>ease of use</i> . <i>Training effectiveness</i> is the percentage of correct answers from the total questions attempted rounded to the nearest 10. <i>Enjoyment</i> and <i>ease of use</i> are values on a scale of 1-10 (10 being most positive) provided by the players.	70

Abbreviations

6DoF: 6 Degrees of Freedom
AR: Augmented Reality
ARU: Autonomous Recording Unit
ASR: Automatic Speaker Recognition
BRP: Bioacoustics Research Program
CAD: Computer Aided Design
CHAR(n): String of n characters
CLO: Cornell Lab of Ornithology
CT: Computer Tomography
DB: Database
dB: Decibel
DFT: Discrete Fourier Transform
ELP: Elephant Listening Project
FGS: Frequency Grid Spacing
FOV: Field of View
GPU: Graphics Processing Unit
HMD: Head Mounted Device
HMD: Head Mounted Display
HRTF: Head Related Transfer Function
HST: Hubble Space Telescope
HUD: Heads Up Display
Hz: Hertz
MARU: Marine Autonomous Recording Unit
mm: Millimeters
MRI: Magnetic Resonance Imaging
ms: Milliseconds
NSH: Natural Spatial Hearing
SDK: Software Development Kit
Tris: Triangles
UI: User Interface
VCASS: Visually Coupled Airborne Systems Simulator
Verts: Vertices
VHP: Visible Human Project
VR: Virtual Reality
VSH: Virtual Spatial Hearing
VVED: Virtual Visual Environment Display

1. Motivation

Few technologies in the world of digital entertainment today draw as much attention and create as much enthusiasm as virtual reality (VR) and augmented reality (AR). And with the gaming industry growing as fast as it is, the drive to venture from traditional console and PC games into the world of VR/AR gaming has never been greater. Although not yet capable of delivering the same experience across all gaming genres as console and PC games, there is no doubt that VR/AR gaming has come a long way. From its humble beginnings in Nintendo's Virtual Boy™ to the PlayStation VR™, the rate at which this technology is growing is astounding and it is becoming more accessible than ever with major software and hardware companies developing not only high-end VR/AR products, but also more affordable mobile based options.

Where VR especially still lags behind, however, is with action and adventure games requiring lots of movement. One of the main reasons for this is that VR, even in its mildest forms, is an uncomfortable experience for some, at times even inducing dizziness and nausea. This is further exacerbated by experiences involving lots of looking or moving around. How to tackle this problem is still a topic of ongoing research and while many games have approached it differently, a universal solution still remains elusive (Rubin, 2016). Aside from gaming, some of the main areas where VR has constantly been put to great use have been medicine, engineering and aviation.

In medicine, simulators can be used to assist with surgical planning, noninvasive screening and with medical training. Virtual endoscopy is a clinically acceptable form of noninvasive screening for the diagnosis of structural abnormalities in internal organs. The concept is based on the 3D reconstruction of computed tomography (CT) or magnetic resonance imaging (MRI) datasets of a specific patient's internal organ of interest. This is followed by a visualization flythrough (Arvanitis, 2006, p. 62).

A visualization flythrough uses flight path algorithms derived from terrain following algorithms for military aircraft to allow the user to "fly through" the organ similar to performing the examination with a video endoscope (Satava & Jones, 1998, p. 487). This process avoids the need to enter the body through an incision or natural opening, such as the mouth, found in traditional endoscopy. **Figure 1** shows an example of virtual endoscopy.

Closely related to virtual endoscopy is surgical planning wherein a surgeon may utilize custom, patient-specific 3D models to rehearse or practice an operation, testing different variants and approaches well before the actual procedure takes place. Coupled with realistic data of the organs and haptic feedback devices, the surgeon will be able to see how each and every action affects the model and thus the patient. An educational example is the virtual abdomen (see **Figure 1**), which is an interactive system allowing the user to interact with an anatomically accurate 3D torso created with computer aided design (CAD), containing stomach, pancreas, liver, biliary tree, gallbladder, and colon. In the simulation, the user can make use of virtual surgical instruments to interact with the organs. Allowing the organs to have accurate physiologic parameters or physical tissue properties is important as it allows for realistic feedback when the organs are grabbed, moved or opened. This also adds to the immersion effect when paired with input devices with force feedback capabilities, meaning that a realistic reaction to every action performed by the user will be felt through the input devices. (Satava, Virtual Reality Surgical Simulator, 1993)

On the other hand, photorealistic representations of the organs are also an important aspect when it comes to the immersiveness of the experience. However, due to finite computational capabilities, it is more often than not the case, where a tradeoff needs to be made between visuals and attributes, such that the system either looks very realistic or responds in a very realistic manner. An example would be having a system with very realistic graphics and few implemented physical attributes versus having a system with realistic physical attributes, but low quality graphics. As computers continue to improve, however, the need for compromise will decrease while both visual and functional qualities increase. Systems such as these reduce the amount of time spent by trainees practicing on animals or patient training. These innovative technologies in medicine are made possible in large part due to the so called Visible Human Project (VHP). (Satava & Jones, 1998, p. 484)

VHP is a computerized medical vision project developed at the National Library of Medicine in Baltimore, USA. Making use of a computer's ability to render information as visual data, medical and anatomic information included, CT scans or MRI are used to render informative images of a body's interior. To do this, a body is frozen in gel at minus 70°C as soon after death as possible. A cryomacrotome, a laser dissection device, is then used to plane the body at very fine intervals transversely from head to

foot. Each “slice” is then digitally photographed (see **Figure 1**), where each photograph effectively represents a move through the body’s mass. The datasets are supplemented by axial sections of the whole body obtained by CT, axial sections of the head and neck obtained by MRI and coronal sections of the rest of the body also obtained by MRI. The result means that blood vessels can be isolated and tracked through virtual space, and the body can be opened out in any direction, viewed from any angle and at any level of corporeal depth. The virtual corpse can also be animated and programmed for interactive simulations of trauma, of human movement, and of surgery (Waldby, 1997, pp. 234-236).

VR was also used in 1993 by the Software Technology Branch of the Johnson Space Center of NASA for the Hubble Space Telescope (HST) Repair and Servicing Mission. Regarded as the first effort of its kind in training large numbers of people naive to VR, including flight controllers, engineers and technicians, the project was motivated by the limited access to training facilities by anyone other than astronaut personnel. The training goal was to familiarize them with the location, appearance, and operability of the different components on the HST, as well as the maintenance components in the shuttle cargo bay. Using VR simulation, the trainees were provided with firsthand insights into the actions astronauts performed in an open shuttle bay in space. Each session, which focused on one training module, an average of three trainees, monitored by one trainer, took turns performing a virtual task while observing on a large monitor the remainder of the time. Sequence management, blinking objects for next step prompting, and audio identification of object and error messages were implemented in order to incorporate intelligent training capabilities. Following training, participants responded with overwhelming positivity and enthusiasm regarding this (at that time) relatively new training method with the majority citing significant knowledge gains (Psocka, 1995, pp. 424-425).

It seems clear then, that VR has, for the past two decades now, had an excellent track record as an educational and training tool. This has not changed for the current generation of VR technology. This idea of a fun, educational learning tool is also one that seems to lend itself well to the gaming genre, thus catering to a much wider audience and not only to specialists of certain industries. Furthermore, such a game should not require intense movement and as such should offer a relaxed, yet engaging, environment for learning and playing.

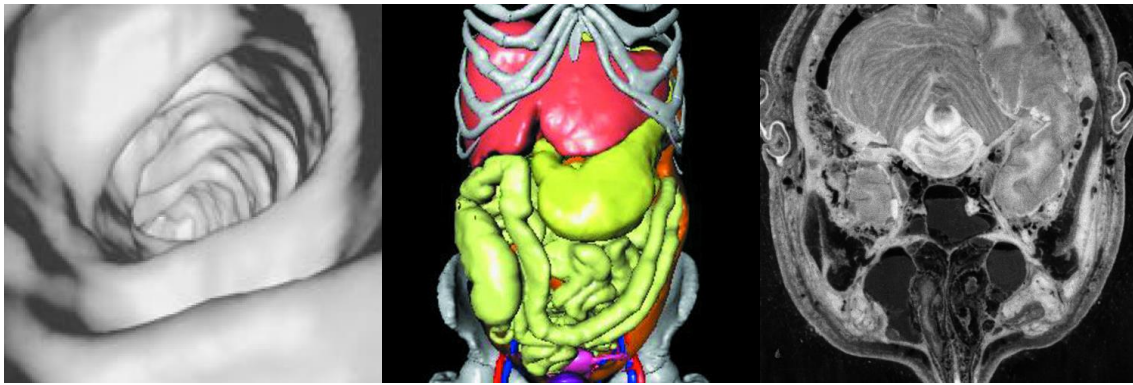


Figure 1. Left: virtual endoscopy with internal view of the transverse colon (Satava & Jones, 1998). Middle: abdomen simulator (Székely & Satava, 1999). Right: cross section of skull – Visible Human Male (Waldby, 1997).

Building upon the work which took place from April to June 2017 at Cornell Lab of Ornithology (Sharaki, 2017), this second iteration of the bird identification game aims to expand as well as enrich the existing experience while maintaining the essence and feel the game has thus far come to acquire. The aim is that the result will urge players to learn, explore and wonder as they are introduced to, or reacquainted with, the birds of North America as they have never experienced them before.

1.1. About Cornell Lab of Ornithology and Bioacoustics Research Program

Cornell Lab of Ornithology is a nonprofit organization, research institute and one of the many units of Cornell University in Ithaca, NY. Founded in 1915, the lab focuses on the study, appreciation and conservation of birds as well as other wildlife, all while seeking to engage and motivate as many people as possible to take part in, as well as learn more about the natural world driven by the many citizen science projects it leads (Cornell Lab of Ornithology). CLO is home to many departments where around 300 interdisciplinary faculty and staff members strive to achieve the lab's message of research, education, citizen science and conservation in many different ways. A few of these departments will be introduced in this section.

For instance, *Bird Academy*, brought forth by the lab's education program, focuses on providing high quality educational content about birds in the form of videos, interactives and games. It is one of the many sites found on *All About Birds*¹, a website created by the lab where users have access to an all-encompassing bird guide, bird

¹ www.allaboutbirds.org

cams all over the world, the lab's very own *Living Bird* magazine and many other features.

Macaulay library is the world's foremost scientific archive of natural history audio, video, and photographs. Its mission is to facilitate the preservation and collection of these recordings as well as promote their use for scientific research, education, conservation and the arts. The library's collection not only features birds, but also amphibians, fishes, and mammals. Currently, Macaulay library hosts more than 4 million media items featuring about 12 thousand species in total, including almost 10 thousand bird species.

The Bioacoustics Research Program, or BRP, the department where this project was developed, specializes in the development and application of many innovative technologies to capture, interpret and analyze voices and sounds from nature, on land and at sea. Its mission is to inspire and inform conservation of wildlife and habitats. BRP's area of interests extends from the woods of North America, to the depths of the Atlantic and Pacific oceans, all the way to the forests of Africa.

BRP's Elephant Listening Project (ELP), studies forest elephants in many sites in Africa including Gabon and Cameroon in the west, to the Central African Republic and the Republic of Congo to the east. The project aims to learn more about these creatures and how to increase awareness as well as fight illegal poaching that targets them for their tusks and organs. ELP utilizes recording devices called ARUs (Autonomous Recording Units), built to withstand climates in Africa's tropical rainforests. These units are hung up on trees far from the reach of elephants, yet susceptible to occasional, inquisitive chimpanzees that damage power cables. From these vantage points, the ARUs collect long recordings of elephant vocalizations, other vocal species and gunshots. These recordings are then analyzed by the ELP team at CLO with the help of many volunteers.

BRP has also developed so called Marine Autonomous Recording Units, or *MARUs*, that are dropped in the ocean at depths of up to 3.7 miles, where pressure is about 600 times that of the atmosphere, and record sounds from the seabed till it is time for them to be retrieved, whereupon an acoustic signal is beamed into the water that causes the unit to sever its tie to the anchor holding it in place at the bottom causing it to float up to the surface for retrieval. Once retrieved and back in the lab, the *MARUs*

are opened up and their collected sounds analyzed. The MARUs are deployed in many spots around the world, but when deployed or retrieved along the eastern seaboard of the United States, the R/V Jaeger is used. The R/V Jaeger is a 42-foot former commercial fishing boat built in 1980 and purchased by CLO in 2016 to support the research efforts of BRP. The MARU will soon be replaced by another marine autonomous recording unit called Rockhopper, as soon as its development is finalized.

The *Swift* is BRP's terrestrial autonomous recording unit. Developed completely in-house, the Swift is able to continuously collect data for upwards of three weeks at a sampling rate of 48 kHz. Swift units are being used worldwide by students and researchers to capture sounds from nature. **Figure 3** shows what these three recording devices look like. Furthermore, BRP holds an annual sound analysis workshop that teaches participants about bioacoustic principles and methods, as well as working with different tools.

However, BRP's technical involvement is not limited to hardware. To manage the huge amount of metadata gathered by their many deployments, BRP has developed a FileMaker relational database called BRP Projects and Archives. The database is accessible by staff members from anywhere and holds information about each and every deployment BRP has ever done such as equipment, locations or general project information.

When it comes to software, BRP also focuses on providing many cutting-edge, yet cost-effective, software solutions and making them available to researchers to support them in their conservation efforts. To help with the acquisition, visualization and measurement of sound, BRP developed a software program called Raven². Raven comes in three different versions, namely Pro, Lite and Exhibit. Each package offers different features depending on the needs of the user. Raven-X on the other hand, offers a hardware independent solution for processing large acoustic data sets. It complements Raven Pro's acoustic analysis capabilities. The tool set enables commonly available multi-core computers to process large multi-channel sound archives at accelerated rates, for example, to detect and classify animal sounds or to extract noise statistics. (Bioacoustics Research Program)

² <http://www.birds.cornell.edu/brp/raven/RavenOverview.html>



Figure 2. View of Cornell Lab of Ornithology from across the pond (Arch Daily).

As can be imagined, using audio recorders such as Swifts, ARUs, MARUs and other similar equipment that is deployed for extended periods of time, yields an extraordinary amount of data that needs to be scanned to locate sounds of interest within the matrix of background sounds.

However, to do this manually by experts or individuals trained for the task is a huge time investment. That is why one of BRP's main focus areas has been the development of automated detectors that can expedite this process to a great degree and reduce the amount of time spent by humans verifying these recordings. This means that instead of experts having to review the entirety of a recording, an automated detector goes through them first, marking where in the recording they registered sounds of a specific type, for example a cerulean warbler or other wildlife species. The experts then only need to go over those areas marked by the detectors to check their validity or note false detections (Charif & Pitzrick, 2008, p. 1). All of these innovations have been put to great use by the lab and made available to students and researchers worldwide to drive conservation efforts. In order to deliver its message, the lab is constantly on the lookout for other projects and collaborations.



Figure 3. Left: disassembled ARU. Middle: MARUs in transit to a deployment location. Right: Swift unit strapped to a tree. (Bioacoustics Research Program)

1.2. Task

Combining two of the lab's main focus areas, namely education and birds, Nation Feathers VR attempts to present this concept in a unique way in the form of a virtual reality game. This section gives a brief overview of some of the challenges faced and design steps taken during development.

Other games created by the lab's education program feature traditional learning tools that utilize sounds, text and images displayed on a two dimensional screen. This can be seen in games such as *Bird Song Hero*, a quiz game that asks players to guess which spectrogram represents a given sound, and *Flap to the Future*, a side-scroller which gives players a sense of how bird flight capabilities evolved over time. And while it is possible to simply create a modified imitation of that sort of interface to run on VR platforms, the goal for this project was to create a genuine, immersive VR experience, one that can replicate the proven potency of traditional 2D learning methods, but at the same time is markedly different. This approach presented the first design challenge; how can an educational game be designed to incorporate traditional VR methods, such as 3D objects, movement and animation, while at the same time offer entertaining content and stay true to its educational goals? This is addressed in chapter 3.

Another important step was deciding how data was to be saved and retrieved. By far the most important pieces of data in the game are the birds. So this process included thoughts regarding the functional capacity of these birds and the role they played in the game. This helped decide what attributes needed to be assigned to these birds in order for the game to function. In other words, what does the game need to know

about each bird and how and when that piece of information will be put to use as the game progresses. Related to this point is the thought process behind choosing the birds featured in the game. Which birds should be featured and based on what criteria? Furthermore, in what order are these birds to be presented to the player? These two issues are discussed in sections 4.1 and 3.3 respectively.

The artistic style chosen for the game and how it impacted performance was a decisive factor when it came to choosing what the game will look like and what assets to use. Some styles, usually one's that are more detailed, are more computationally expensive than those with less emphasis on detail. Finding that balance between visuals and performance was necessary to the improvement of the overall experience. How this balance was achieved is explained in section 4.3. Finally, the game needs to meet its goal of igniting the interest of players in birds and creating a connection between the visible and audible, i.e. guide players to associate bird calls used in the game with their corresponding bird species. How well this goal was met is reviewed in the evaluation in chapter 5.

1.3. Structure of this Thesis

Before getting into the main contents of the thesis, this section illustrates briefly how it is structured. Chapter 2 discusses different technological aspects that are utilized in the game and attempts to draw comparisons to other versions of these technologies used elsewhere. It starts off with a look at virtual reality as a technology; how it functions, how it has changed over time, different forms of VR hardware and software, as well as other aspects and characteristics of the technology.

This is followed by an overview of the software tools used for development, including libraries, utilities and packages. Spectrograms, ingame visualization, and other game models are then explained to conclude the chapter. The game concept is explained in chapter 3. This chapter explores what exactly the game is and how it achieves its goals. It also discusses the game flow of the specific scenes and how the scenes differ.

Chapter 4 discusses implementation details of many features in the game. It goes into greater technical detail to explain topics visited in previous chapters as well as topics not yet discussed. The evaluation in chapter 5 assesses the resulting project in terms of how well it achieved its goals and also assesses how it performs on the technical level. Chapter 6, the final chapter, looks back on the work as a whole and provides a

summary of what has been discussed. Possible future continuations and expansions are contemplated and improvements to the system proposed.

2. State of the Art

This chapter takes a deeper dive into virtual reality technology and other innovations and research relating to this work. The concept of virtual reality and how it functions will be explored, followed by a brief history of its progression. Existing innovations, including software and hardware, and how they have affected this work will be discussed. Specific interdisciplinary aspects such as immersion and visualization will be outlined before finishing with a discussion on examples of other educational games as well as the thinking and planning that need to go into the design of such games.

2.1. Virtual Reality

Starting off the discussion is an overview of the technology that made the development of Nation Feathers VR both challenging and different. As interesting as the end product when using VR is, it also requires different steps to be taken during development and a different thought process and approach to design. To better put things into perspective in regards to how far this technology has come and why it differs so much from other similar technologies, the following section takes a look at its history.

2.1.1. History of Virtual Reality

For as long as humans have been expressing themselves and the world around them, new ways of conveying deeper emotion and allowing observers to connect more intimately with the source material have been sought after. This can even be seen in the ever varying art forms throughout time, even before the advent of electronic media. For instance, the use of different, perhaps more vibrant colors in paintings, different materials and levels of detail used in sculpting or the inclusion of more subjects with increasing levels of visual complexity in a piece of art are all steps taken towards a more comprehensive experience that aims to engage the senses and trigger



Figure 4. Battle of Borodino by Franz Roubaud.

more emotions than previously achieved. Panoramic paintings are one such early attempt, a prime example of which is the Battle of Borodino by Russian artist Franz Roubaud, first showcased in 1912. The 115 meter long artwork depicts the 1812 battle of Borodino fought between Napoleon's Grande Armée and Russian forces in 360° fashion (see **Figure 4**). Another attempt, which gained popularity starting in the 1800s, and that is perhaps more similar to modern VR than panoramic paintings is stereoscopic photos and stereoscopic viewers. This method involves taking two pictures of the same object or scene, each picture slightly offset. The baseline, which is the amount by which the camera is offset, or the distance between each camera if two cameras are used, is typically equal to the distance between the left and right eyes in humans. This, however, can vary depending on the viewing method and the purpose of the picture. To view a stereoscopic image, a stereoscopic viewer is used to combine the left and right images into a single image that can be processed by the brain to give the feeling of depth. **Figure 5** shows a stereoscope, or stereoscopic viewer, and a stereoscopic image.

Moving closer to the digital age, more sophisticated electronic devices start to appear that much more closely resemble current day VR technology. A brief overview of how these technologies evolved in the second half of the 20th century is given in **Table 1**.



Figure 5. Top: Brewster-type stereoscopic viewer. Bottom: Stereoscopic image of Boston.

Table 1. Timeline of advances in VR from the 60s to the 90s. (Mazuryk & Gervautz, 1996, pp. 2-3)

Name and year	Developer	Description
Sensorama™ 1960-1962	Morton Heilig	A non-interactive, multi-sensory virtual reality environment in the form of a prerecorded film augmented by scent, wind, vibrations, and binaural sound, i.e. sound heard by both ears to give the feeling of presence.
The Sword of Damocles™ 1968	Ivan Sutherland	Widely considered to be the first VR head mounted device (HMD), this device utilized appropriate head tracking and stereo view that was updated according to the user's head position.
Videoplace™ 1975	Myron Krueger	Users' silhouettes are grabbed by cameras and projected on a large screen. Their positions are determined using image processing techniques, thus enabling users to interact with each other.
Visually Coupled Airborne Systems Simulator (VCASS™) 1982	Thomas Furness	An advanced flight simulator, where pilots wore head mounted devices that augmented their view to display more information, such as targeting or optimal flight paths.
Virtual Visual Environment Display (VVED™) 1984	NASA Ames	A head-mounted, wide-angle, stereoscopic display system controlled by operator position, voice and gesture developed for use as a multipurpose interface environment.
DataGlove™ and Eyephone HMD™ 1985 and 1988	VPL Company	The first commercially available VR devices.
BOOM™ 1989	Fake Space Labs	The user gazes through a movable box containing eye holes showing two CRT monitors. A mechanical arm measures the position and orientation of the box as the user moves it.
Virtual Wind Tunnel 1990s	NASA Ames	Utilized <i>BOOM</i> and <i>DataGlove</i> to allow the observation and investigation of flow fields.
CAVE 1992	EVL, University of Illinois at Chicago	Doing away with the use of a head mounted display, <i>CAVE</i> projects stereoscopic images onto the walls of a room. Shutter glasses are used to regulate the viewing of the right and left images projected.



Figure 6. Top row (Left to right): The Sword of Damocles, VCASS, CAVE. Bottom row (Left to right): BOOM, Sensorama, VVED, Videoplacé.

2.1.2. Technical details

Regardless of the specific version of VR headset or technology, they all share many similarities. They offer an interactive and immersive experience in a simulated 3D world. The illusion of participation in this environment rather than external observation is achieved in part by head-tracking capabilities, haptic feedback, and binaural sound. This chapter will go over the basic functionality of virtual reality as a technology and what allows it to achieve what it does.

Before getting into the fundamentals of VR technology, some requirements and baselines need to be laid out. VR input devices and output devices need to fulfill certain criteria in order for the experience to be immersive. When it comes to input devices, the most critical one is the headset. Headsets act as the gateways into the virtual world and are thus decisive when it comes to believability and enjoyment. Typically, headsets use a system called six Degrees of Freedom (6DoF) to track the user's head movement. This system allows for movement plotting on the x, y, and z axes; this movement is also known as pitch, yaw, and roll (see **Figure 7**). As users look around the environment, the headset tracks this movement and updates the image accordingly. There are many methods of tracking movement in headsets; these include accelerometers, gyroscopes and magnetometers. Other methods involve the use of LEDs positioned in various places on the headset, which are captured and tracked via an external camera. Advances are also being made in headsets that can also track eye movement and change the scene accordingly.

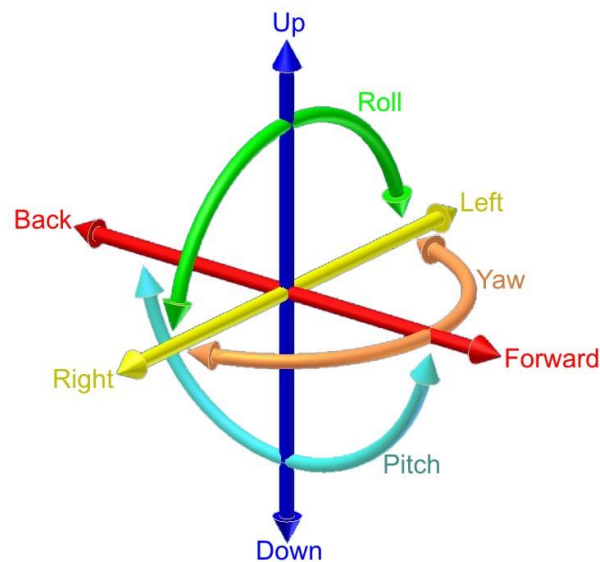


Figure 7. Shows six degrees of freedom.

What follows is a rundown of some of the factors affecting these input devices, the headset most of all, in order to ensure a smooth VR experience (Mazuryk & Gervautz, 1996, pp. 19-20).

- **Update rate (Hz):** Measurements per second. Higher update rates require more processing, but deliver smoother tracking.
- **Latency (ms):** Amount of time between the user's action and the actual transmission of the action report.
- **Accuracy (mm for position/degrees for orientation):** Measure of error in the reported position and orientation. Defined generally in absolute values, with smaller values reflecting better accuracy.
- **Resolution:** Smallest change in position and orientation that can be detected. Measured in absolute values, with smaller values reflecting better resolution.
- **Range:** Working volume, within which the tracker can measure position and orientation with its specified accuracy and resolution, and the angular coverage of the tracker.

The linking of input sensors, effectors such as a stereoscopic display, and special-purpose hardware that links all the sensors and effectors to produce sensory experiences resembling those in physical environments, creates the illusory virtual

environment (Ellis, 1994, p. 18). It has been estimated that nearly a billion polygons per second may be needed for near realism. These limitations not only lead to low resolution and cartoon-like shapes, they also lead to long lags between changes in the head position and updates of the display (Psootka, 1995, p. 407). Currently, however, the Samsung Gear VR™, which is considered a lower tier device in terms of power, is able to reasonably generate 50000 to 100000 polygons per frame (Pruett, 2015).

The human vertical field of view (FOV), which is limited by the cheeks and eyebrows, is about 150° . The horizontal FOV, on the other hand, can be sectioned into 60° towards the nose and 90° towards the temple. Taking both eyes into consideration, this allows for about 180° of total horizontal viewing range with a 120° binocular overlap when focused at infinity (Mazuryk & Gervautz, 1996, p. 16). **Figure 8** illustrates this. It has been shown that immersion can only be achieved when the headset or viewing device's FOV is greater than 60° . It is still unclear why this is so (Psootka, 1995, p. 8). Luckily, this range is easily attainable and is being surpassed more often than not by most consumer headsets on the market today. The Samsung Gear VR, for example, has a variable FOV which depends on the handheld device being used to operate it. According to the manufacturer, however, it offers a 101° FOV. Other more advanced options such as the HTC Vive™ and the Oculus Rift™ both offer about 110° .

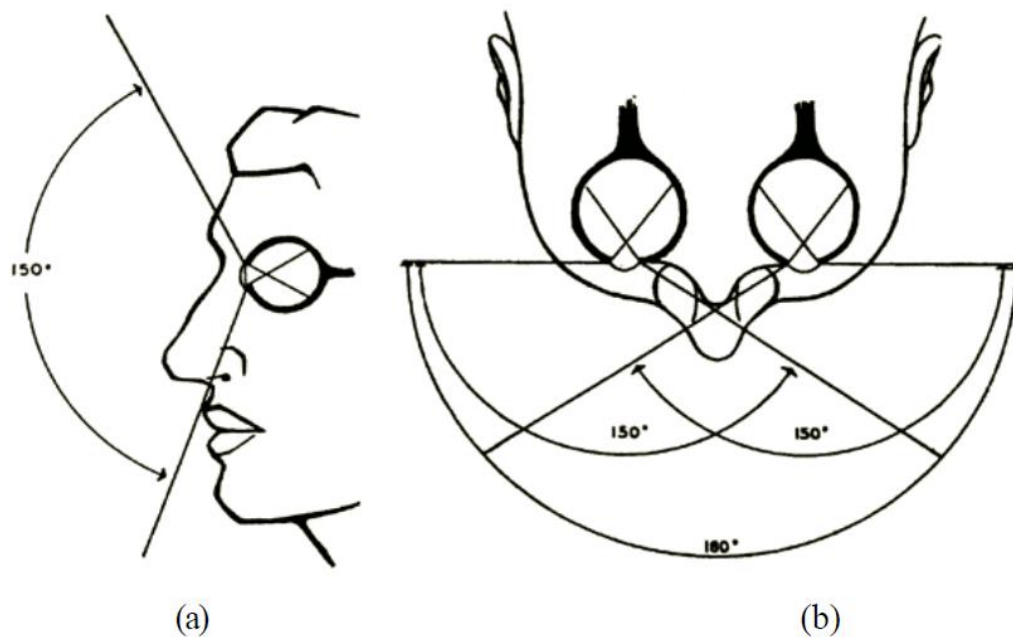


Figure 8. FOV in humans. (a) vertical, (b) horizontal. (Mazuryk & Gervautz, 1996, p. 16)



Figure 9. Two feeds sent to a single mobile device screen.

VR headsets use two feeds sent to either one or two 2D displays, placed either as part of the headset or as part of the screen of a mobile device attached to the headset. **Figure 9** shows this phenomenon. Lenses placed between the screen and the user's eyes, focus and reshape these two 2D images creating a stereoscopic 3D image. This is an attempt to mimic how our left and right eyes view the world slightly differently under normal circumstances; an effect which can be witnessed when closing and opening the eyes alternately to see how the perspective shifts and the objects in view move around.

2.1.3. Immersion

The term immersion has been used a lot in previous chapters. This section aims to take a closer look at the concept of immersion. What it is, its physical and psychological effects, possible side effects, and its overall importance to the virtual reality experience will be discussed.

Immersion in a VR sense is the phenomenon of being placed in a simulated environment that looks and feels like the real world. This look and feel is achieved both by how the environment is portrayed to the user and how it responds to the user's actions. Movement and a sense of self-location are also great contributors to this effect. (Psootka, 1995, p. 406)

The level of immersion in a VR system is determined by the type and quality of sensory impressions generated and delivered by the computer to the human senses. Ideally, high quality and consistent information should be delivered to all the user's senses. In practice however, most applications simulate only one or a few senses, thus delivering different levels of immersion. (Mazuryk & Gervautz, 1996, p. 5)

Visual acuity, field of view, temporal resolution, luminance and color, and depth perception are some of the factors affecting immersion. (Mazuryk & Gervautz, 1996, pp. 17-18)

- **Visual acuity:** The sharpness of viewing, measured as the fraction of a pixel which spans one minute of arc horizontally.
- **Field of view:** Extent of the observable game world that is seen on the display at any given moment. Section 2.1.2 talks about this in more depth.
- **Temporal resolution:** Temporal resolution of the eye refers to the flickering phenomena perceived by humans, when watching a screen, e.g. CRT, that is updated by repeated impulses. Low refresh rates cause flickering.
- **Luminance and color:** Apparent brightness and color. Currently, no output devices can match either the human eye's dynamic range, i.e. the ratio of the brightest luminance to the darkest luminance, or cover the whole color gamut. Color mapping techniques must therefore be used to achieve the best picture quality.
- **Depth perception:** The perception of the virtual world around the user in three dimensions. This has been explained in greater detail in section 2.1.2.

Immersion seems to be facilitated by the ability to control attention and focus on the new virtual reality to the exclusion of the real world. Using an avatar to simulate one's own body or part of it, even in cartoon form, adds to the experience. Temperamental differences among users and the use of a good visual imagination also play a large role in how immersive the VR experience is. These temperamental differences also determine how users react to limitations in the technology being used, such as low quality displays, latency, or intrusive hardware such as heavy headsets. (Psothka, 1995, p. 409)

One of the strongest benefits of immersion to the user is a reduction in conceptual load. What this means is that in the virtual world the directness of perception is simplified. Take for instance a simulation or a picture; in such representations, the human observer constructs a virtual self or a viewpoint that enters the space of the representation as if a human observer were there. It is also very rare when the real self and the virtual self are in the same perspective location. **Figure 10** helps visualize this phenomenon. The image of the modified Necker Cube can be held above or below the viewer and yet the viewer will most likely continue to visualize it as if he or she is viewing it from above. Relating this to the concept of the virtual self, this can be expressed in terms of the viewer having created a virtual self that is regarding the cube from above even though the real self might be regarding the image from a different angle. This shows the role of experience in determining point of view. It is more difficult to see the house from below than above because houses are generally seen from above or straight on, but never from below. If the Necker Cube is regarded long enough, however, a second image will start appearing, namely that of the house viewed from below. Those two conflicting points of view will be perceived alternately. Although normal images do not alternate as unambiguously as the house in **Figure 10**, they do have the same additional complexity required to generate that conflicting point of view. The effect this processing difficulty has on learning is often overlooked and it is one that can be greatly reduced with a well-designed virtual environment. (Psootka, 1995, p. 411)

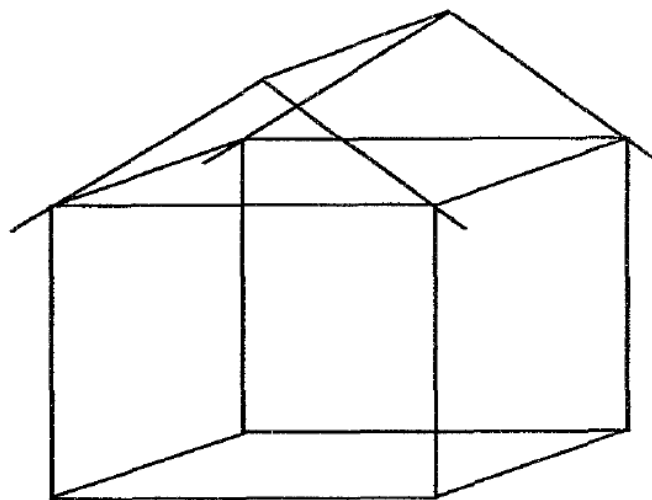


Figure 10. Modified Necker Cube resembling a house. It is much more difficult to see the house from below than from above, as in normal experiences houses are viewed from above or from straight on. (Psootka, 1995, p. 411)

2.1.4. Ongoing research

The previous section introduced immersion and factors that influence its perception and in some cases affect the overall VR experience. Here, some of these factors which still pose a challenge to current VR technologies will be outlined and discussed.

Perhaps the first issue that comes to mind when the future of VR is being discussed, is hardware comfort and convenience. The advances made in this area over the past three decades cannot be understated. When one considers the very first HMDs such as The Sword of Damocles or VCASS and compares them to current HMDs such as the Gear VR this becomes immediately apparent. Not only has the form factor of these devices been reduced, but also their weight. However, these devices still have a ways to go before they are as seamless as experiencing the real world. This ties-in with the issue of some VR headsets being tethered. An example of this is the HTC Vive.

A focus on user interface takes this concept of reaching a VR system that mimics reality as closely as possible one step further. Every kind of human-computer-human interaction should be so natural and intuitive that neither learning nor adaptation should be necessary. However, today we are yet far from this vision. Today's interfaces require much getting used to, complicated calibration steps and non-intuitive interaction paradigms. Hence they are not easy to operate by the unskilled user. Since in the real world every action performed on an object causes a reaction that is felt from this object, this means that every input device must simultaneously be an output device that supports haptic feedback. In the real world, with the help of our sense of touch, this action-reaction relationship allows us to perform tasks without necessarily having to see what we are doing. Gloves with feedback and exoskeletal manipulators are the first steps in improving haptic interfaces. An extension of them may be force feedback suits delivering haptic sensations to the whole body. (Mazuryk & Gervautz, 1996, p. 54)

Finally, the issue of computer performance is one that is not specific to VR. More powerful processors and graphics cards along with more efficient image rendering algorithms would allow for virtual environments that are more realistic, but at the same time do not come at a performance cost. Not only this, but such advances in these technologies would allow for more immersive and diverse worlds that, when viewed in the scope of gaming, would close the gap between traditional video games and VR.

2.1.5. VR in Game Engines

There are many game engines out on the market today that offer various tools and features for game development. Some of these game engines have also incorporated VR support to allow for the development of virtual reality games that can be played on many of the currently available VR headsets. As mentioned earlier, Nation Feathers VR has been developed using Unity. Unity is by far the most widely used third-party game engine due in part to its user friendliness and familiarity. This, coupled with a very active user community makes this game engine very appealing especially to new developers and teams looking to create products in a short period of time. Unity is one of the game engines that offer VR support. Currently supported VR devices include Oculus, Samsung Gear VR, PlayStation VR, Google Cardboard™ and Daydream™, and HTC Vive.³

Other game engines with VR support are Unreal Engine™⁴, CryEngine™⁵, Lumberyard™⁶, Fuzor™⁷, Stingray™⁸, and Magic Leap™⁹. CryEngine is developed by German video game and software developer Crytek™. Although not as prevalent as Unreal or Unity, it has nonetheless been used to produce big names in the video game industry such as the *Far Cry* series. Another contender is Amazon's Lumberyard, which is based on CryEngine's architecture. A major feature of Lumberyard is that it allows developers to host their games on Amazon's servers. Furthermore, the engine's source code is made fully available, allowing for it to be altered to suit one's needs if necessary. The downside to both game engines is that their range of supported platforms is relatively more limited than either Unity or Unreal.

Unreal is without a doubt among the most notable game engines available today due to its popularity, power, and comprehensiveness. Again, this game engine's source code is available to be tweaked by developers as they see fit, adding to its flexibility. Built in, deeply integrated multiplayer networking is another one of the engine's strong points. Where Unreal Engine falls short, however, is a limited community presence. This is due, in some part, to the fact that the engine is utilized mainly by bigger game

³ <https://unity3d.com/public-relations>

⁴ www.unrealengine.com

⁵ www.cryengine.com

⁶ aws.amazon.com/lumberyard/

⁷ www.kalloctech.com

⁸ <https://www.autodesk.com/products/stingray/overview>

⁹ www.magicleap.com

studios that aren't always very eager to share their experience. This often leads developers to have to come up with their own solutions to issues that may be shared by many others. Finally, with Unreal Engine's flexibility and extent comes a certain level of complexity that gives it quite a steep learning curve. This is, in many cases, unsuitable for many projects and teams and is oftentimes a big reason why Unity is preferred.

2.2. Tools for VR development in Unity

Following the brief overview of game engines in the previous section, this section focuses on Unity and offers a glimpse at some of the tools it provides for development and VR development specifically. Many of the points introduced are elucidated in later chapters.

Depending on the target platform, different software packages, libraries, and development kits are offered to aid in development in Unity. For instance, Google Cardboard and Daydream applications make use of the Google VR SDK, Microsoft's HoloLens uses Windows Holographic, and the HTC Vive requires OpenVR. Unity development for the Gear VR on the other hand, requires no additional software to run a VR app than that required to run Android apps. To aid with development and to familiarize new developers with the environment, Unity provides free sample VR projects that can be downloaded from the Asset Store. Along with the sample projects, other useful items such as scripts, 3D and UI assets, and various other items are also available for download.

Depending on the complexity of the VR project being developed, Unity's standard utilities library may not be sufficient. This is where the Oculus utilities for Unity come in handy. This optional package includes prefabs, C# scripts, sample scenes, and more to assist with development. For the purposes of Nation Feathers VR, this package was essential for providing more comprehensive access to the handheld controller that was not available with Unity's standard library. For a more concrete example, Unity's standard utilities library allows for swipe, touch, and tap detection on the controller's touchpad. However, the nature of the movement implemented in the game required accurate touch point recognition to detect the position of the player's finger on the touchpad. This was provided by the Oculus utilities. Another instance where Oculus' wider range of functionality was helpful was when there needed to be a distinction between touchpad and trigger clicks. Unity's utilities do not offer this distinction as

they regard both actions as equal. Oculus, however, regards each click as a different action depending on the button.

Other used functionalities provided by Oculus include an FPS-Controller. This asset gives the game a first person perspective. Among other things, it also adds footstep sounds and bobbing movement for walking. This package was not used for Nation Feathers VR as its functionality went beyond what was needed for the scope of the game. Furthermore, some features such as the bobbing movement would be too risky for the game as it may cause dizziness for some players.

Another useful feature that came with Unity's standard utilities was the reticle. This pointing device designed to move with the player's head movement formed the basis for user interaction in Nation Feathers VR. The functionality had to be altered, however, to fit the game's needs. This is touched on more in section 4.2.3. A key to increasing the speed of game creation is to create custom editors. A custom editor or inspector replaces the default editor shown when a component is selected. Section 4.7 talks about this more.

Finally, the profiler is an essential tool to keep track of game performance. This is especially important in mobile VR development where resources are extremely precious and must be used frugally. The profiler helps with this task as it shows in great detail and in real-time things like CPU and GPU usage, memory, audio, and rendering.



Figure 11. The profiler window. The window's upper section shows the resources (CPU, Rendering, etc.) being used by the game on a timeline containing a graph which shows data usage. The data at the bottom of the window correspond to a specific frame. The frame can be changed by moving the white vertical line across the timeline.

2.3. Spectrograms

A spectrogram is a visual way of representing the signal strength, or “loudness”, of a signal over time at various frequencies present in a particular waveform. Not only can one see whether there is more or less energy at, for example, 2 Hz vs 10 Hz, but one can also see how energy levels vary over time. In some sciences, spectrograms can be used as visual representations of sound waves produced by humans, animals, machinery, etc. (Pacific Northwest Seismic Network)

Spectrograms have applications in various fields, such as in natural language processing and speech recognition (Greenberg & Kingsbury, 1997, p. 1647). They can be utilized in the area of automatic speaker recognition (ASR) where the goal is to extract, characterize, and recognize the information in a speech signal in order to convey speaker identity. ASR systems can be divided into two branches: speaker identification and speaker verification. Speaker verification attempts to decide if a speaker is who he or she claims to be. This requires a 1:1 match where one speaker’s voice is matched to one template. Speaker identification, on the other hand, attempts to decide if the speaker is a specific person or is among a group of people. This is a 1:N match where a voice is compared against N templates. (Kekre, Kulkarni, Gaikar, & Gupta, 2012, p. 27)

Furthermore, spectrogram variations, such as the modulation spectrogram, can be used to provide clarity and stability in speech representation in the presence of speaker variability and distortions such as spectral shaping, background noise, and reverberation that typically have little or no influence on the acoustic intelligibility of speech, but can drastically alter conventional speech representations like the sound spectrogram. (Greenberg & Kingsbury, 1997, p. 1647)

A spectrogram can be regarded as a three dimensional graph; two of those dimensions are plotted on the x and y axes while the third is represented by color. As shown in **Figure 12**, the change in time is shown on the x-axis while the y-axis represents the change in pitch (i.e. the perceived frequency). Brightness signifies loudness, in other words the brighter the spectrogram at a certain point, the louder the sound.

How spectrograms were created for Nation Feathers VR is explained in section 4.5.

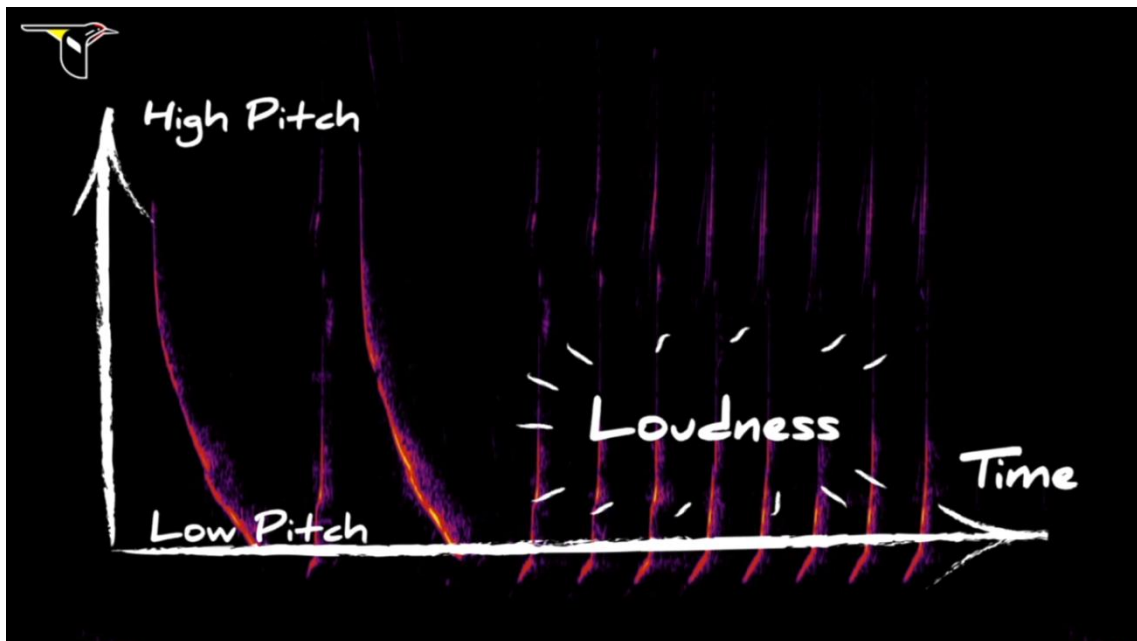


Figure 12. Labeled image of a northern cardinal's spectrogram showing how spectrograms are to be interpreted. (Bird Academy)

2.4. Visualization and Sound

This section visits, from a technical perspective, the inner workings of what is heard and perceived in the game. This includes how 3D models are formed, what constitutes their structure, how they are affected by light, and how materials and textures affect their appearance. It also goes over Unity's 3D sound, which is a vital aspect of the game. Furthermore, how the sound is perceived by different players and what factors affect the variance in perception will also be discussed.

Sight and hearing are in the forefront when it comes to how much the five senses contribute to the perception of the virtual world (see **Figure 13**). It is natural then that most research will be focused on advancing these two aspects. As shown by **Figure 13**, sight plays the most important role of all when it comes to the consumption of virtual reality content. It provides the most information to our brain and grabs most of our attention. This is followed by sound, which in the case of Nation Feathers VR is of special importance. Touch is normally not an essential feature for immersion, but increases in importance in tasks where precise manipulation is required (see **Motivation**). Due to taste and smell's difficulty to implement and marginal role they are usually not included. (Mazuryk & Gervautz, 1996, pp. 15-16)

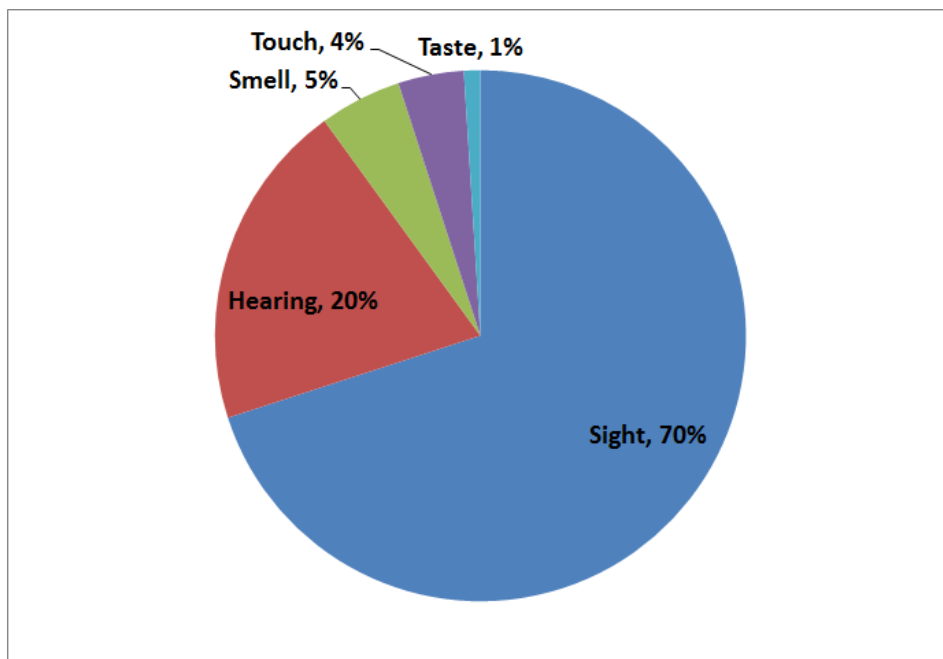


Figure 13. Significance of each sense for experiencing VR. (Mazuryk & Gervautz, 1996, p. 15)

2.4.1. 3D Models

In this section, the building blocks and components of 3D models are discussed. It aims to give an introduction to the concept of 3D models in a general sense without going into detail about the nuances of specific implementations, e.g. games, animated films, etc. Concrete examples of how 3D models were implemented in the game are discussed in section 4.3.

The field of modern computer graphics is in large part based on and influenced by a concept developed in the 1400s called linear perspective (Figure 14). Linear perspective is the observation that a realistic representation of a 3D environment in 2D could be calculated based on rules that govern how human eyes see the world around them. It stemmed from the observation that parallel lines seem to converge as they move farther away from the eyes. Those lines are not actually coming together, but rather this effect is due to the eye's nearly spherical shape, which allows for information reception from all sides. Because the rules of linear perspective worked and could be preserved and put down on paper, many artists were able to replicate the results and linear perspective became a standard tool for most artists from the time of its discovery all the way up to the present day. As can be expected, with the invention of the computer this knowledge was turned into software and with that the modern era of 3D computer graphics was born. (Paquette, 2008, p. xi)

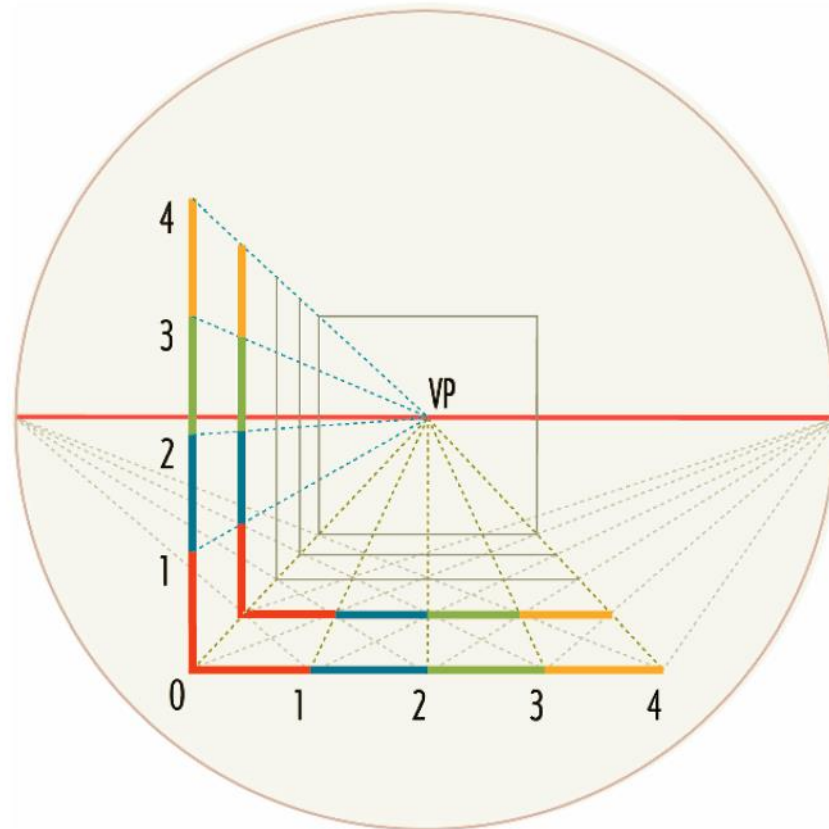


Figure 14. A visualization of linear perspective. The colored bars are for measurement, while the red line represents the horizon line (i.e. the viewer's eye level). VP denotes the vanishing point (i.e. the point on the horizon line where parallel lines seem to converge). The circle represents the field of view (FOV). The points where the lines meet the FOV are diagonal vanishing points; their effect can be perceived, for example, when the viewer is facing the corner of a building and its sides seem to get smaller or more elongated as they recede towards the vanishing points. (Paquette, 2008, p. xiii)

Modeling is usually the first of three steps when it comes to generating computer graphics content; the other two are animation and rendering. *Modeling* is achieved by designing the models or objects that reside in the simulated world. By assigning motion to these models, they become *animated*. And they are *rendered* by defining them to have desired surface properties. Since models are represented graphically using mathematical formulae on a computer, it follows that objects that are represented with simpler mathematical formulae, such as spheres or cubes, are easier to model. These objects are called primitives. In some cases, more complex shapes can be created by combining these primitives. However, more advanced modeling techniques may be required for creating other models. (Govil-Pai & Pai, 1998, p. 11)

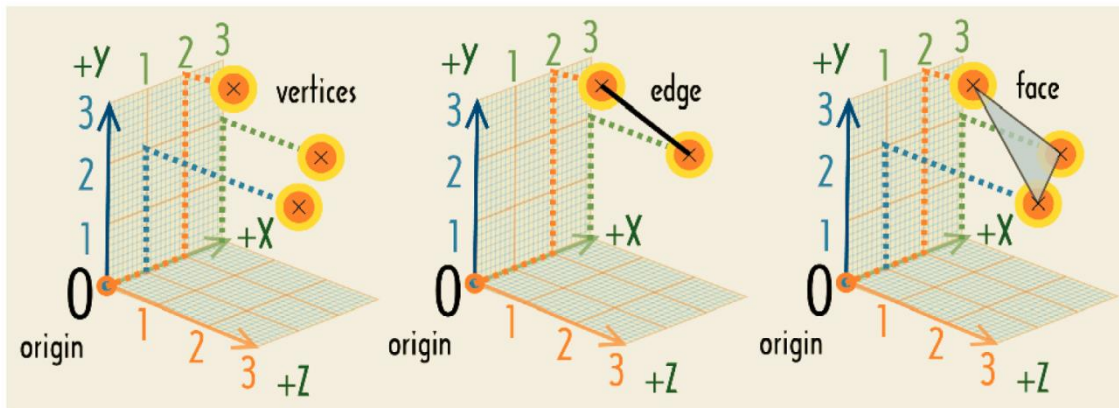


Figure 15. Vertices, edges, and the resulting triangular face represented in 3D space. (Paquette, 2008, p. 7)

In a computer graphics program such as Blender™¹⁰, a 3D model can be broken down into several components (**Figure 15**) that represent its building blocks. Together, these components form what is called a mesh. This mesh can be thought of as a representation of the 3D object within the program. (Flavell, 2010, p. 37)

The first of these components are *vertices*. On their own, vertices are points floating in 3D space with x, y, z coordinates measured from an origin point. In order to define a solid-looking object, vertices are connected point to point by *edges*. The resulting structure is what is referred to as a wireframe and it helps by making the object much easier to see and make out. The structure which has been created so far still does not contain any surface to which to add colors or textures. To do this, *faces* are created by filling in the edges of the wireframe. 3 Edges result in a triangle (commonly referred to as *tris*), while 4 edges result in squares. When the mesh is made up of faces, it finally looks solid. (Flavell, 2010, pp. 37-39)

2.4.2. Materials, Shaders and Textures

An object that has structure is only half of the story when it comes to how it is perceived. The other half is determined by things like *textures*, *materials* and *shaders* and how they are affected by and behave in response to light. This section serves as an overview of these concepts. Implementation details are discussed in **Environment**.

Perhaps the best place to start is with materials, since they act as the base or reference point for textures and shaders and are a common concept to all 3D applications (Goldstone, 2011, p. 14). Whereas 3D models are in a sense “tangible”, that is they can

¹⁰ www.blender.org

be used independently of other game assets, materials on their own do not achieve much. This alone can make materials a more difficult concept to grasp, however it is exacerbated by their relative lack of real-world analogs. For example, 3D models and 2D images can be very easily likened to sculptures and paintings respectively. Materials have no such straightforward real-world counterpart. Rather, materials are abstract packets of information that layer onto 3D models. They define the surface properties of any 3D object they are attached to. These can include color, shininess, and even subtle roughness. For the sake of concretizing this concept, if 3D models are to be thought of as sculptures, then a material is the medium from which this sculpture is made. (Hocking, 2015, pp. 69-71)

On their own, materials will lend a functional, if bland appearance to a game level (**Figure 16**). The next step is to add a texture to the material. A texture is any 2D image being used to enhance 3D graphics. In addition to the noun usage of the word, it is also routinely used as a verb to describe the general action of using 2D images in 3D graphics. Of the different uses of textures in 3D graphics, perhaps the most straightforward is when they are used to be displayed on the surface of 3D models. Other uses include masks to cut out shapes and normal maps to add an illusion of bumpiness to a surface. (Hocking, 2015, pp. 75-76)

The final surface component to be discussed is shaders. Shaders are small programs or scripts that outline instructions for how to draw a surface including whether to use textures. These instructions are used by the computer to calculate the pixels when rendering the image. Although shaders can be used for many different kinds of visual effects, the most common one takes the color of the material and darkens it according to the light. Since every material has a shader that controls it, a material can be thought of as an instance of a shader. By default in Unity, new materials are set to the standard shader. This shader displays the color of the material, taking into consideration any textures that have been applied to it, while also applying basic light and dark shades across the surface. (Hocking, 2015, p. 81)

As mentioned above, shaders can help determine how a surface reacts when light is shined on it. This puts a great emphasis on the type of lighting used in any given scene. Section 4.8 goes into greater detail about light from a Unity perspective.

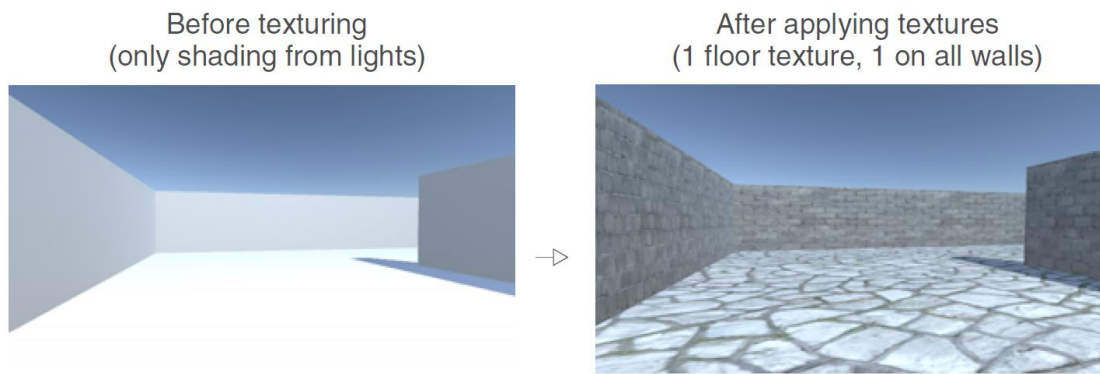


Figure 16. A scene with 3D ground and wall objects. The left image shows the scene before texturing with the default material applied, while the right image shows what the scene looks like after texturing. (Hocking, 2015, p. 75)

2.4.3. 3D Sounds

For the reasons of visual sensory dominance in humans stated in section 2.4, the tools for auditory manipulation in computing have not received the same amount of focus by developers over the years as their visual counterparts (Begault, 2000, p. x). Yet, Nation Feathers VR relies heavily on hearing as a means for acquiring information in the game. The nature of the game itself calls for accurate localization by sound in order for the game to function as intended. However, despite the aforementioned advantages visual tools have had over years, sound has not been left entirely without its own set of tools in the world of digital media. Of special importance for the use case of this game, are 3D sounds.

In 3D sound (also known as virtual acoustics, binaural audio, and spatialized sound), a distinction needs to be made between *natural spatial hearing* (NSH) and *virtual spatial hearing* (VSH). NSH refers to how sounds are heard spatially in everyday hearing, with the ears uncovered, the head moving, and in interaction with other sensory input. VSH, which is a special case of binaural hearing¹¹, refers to the formation of synthetic spatial acoustic imagery using a 3D sound system and stereo headphones. The transmission path from the operator of a 3D sound environment to the listener can be described by many different models all of which share a common root: *source*, *medium*, and *receiver* (**Table 2** and **Figure 17**).

¹¹ Binaural hearing is hearing that involves the two ears.

Table 2. A verbal illustration of the differences between NSH and VSH in the source-medium-receiver model. (Begault, 2000, pp. 4-5)

	Natural spatial hearing	Virtual spatial hearing
Source ¹²	Very rarely is a single source involved in emitting the sound.	A number of sources are individually positioned in the environment.
Medium	The source arrives at the listener by means of reverberations and the effects of physical objects on the propagation of the sound.	Sound reaches the listener by means of sound reproduction nonlinearities, signal processing, and headphones.
Receiver	The listener’s physiology. This involves the whole hearing system, from the ears to the final stages of perception in the brain.	

Other than to achieve this game’s specific goals, in general the reason that localizing objects from stereo sound is important is that it adds to the sense of presence. However, accurate localization will be affected by the shape of each individual user’s outer ear or pinna. (Psozka, 1995, p. 408) To see how this can play a role in the user’s perception of the sound, a good understanding of how sound is processed needs to be established. This is illustrated in **Figure 18**.

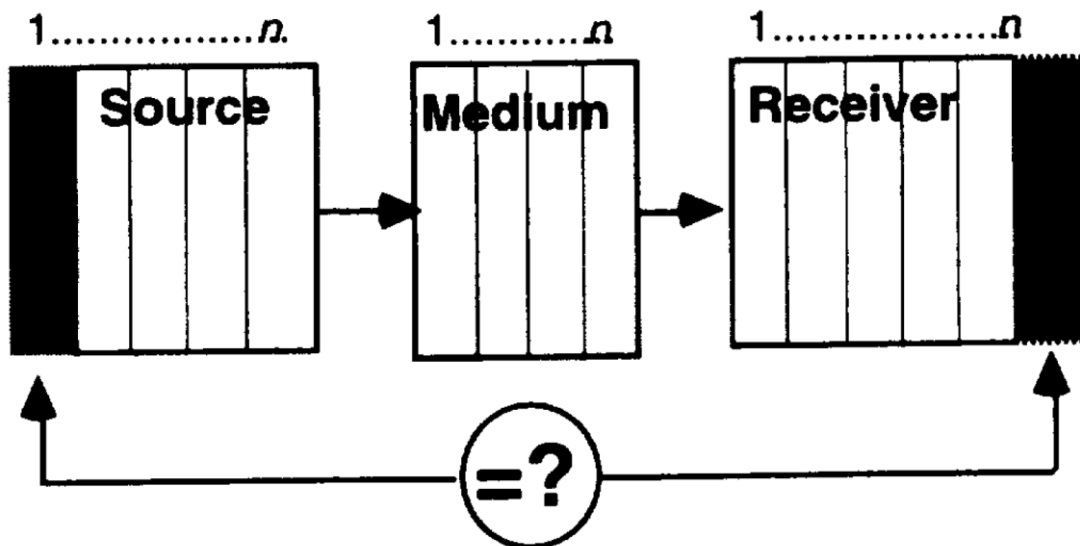


Figure 17. A source-medium-receiver model. The dotted line sections numbered 1 through n represent the physical, neurological, or perceptual transformations that each element contains in the communication of the spatial location of a sound source. For a 3D sound system, the question is to what degree the 1st element of the source equals the nth element of the receiver (Begault, 2000, p. 5)

¹² While one or more vibratory sources can be involved, it is convenient to describe a sound system in terms of a single source in isolation. (Begault, 2000, p. 4)

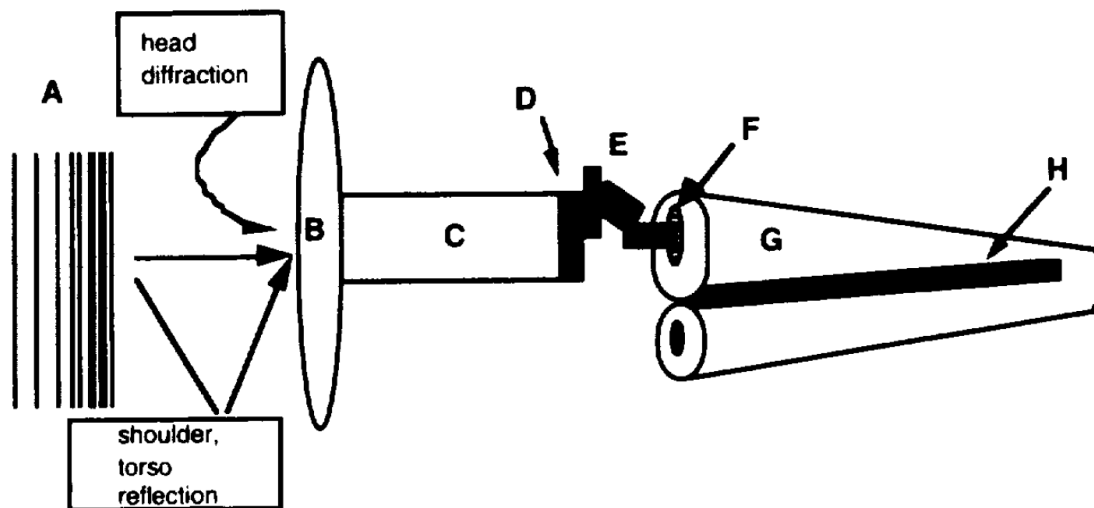


Figure 18. Highly simplified schematic overview of the auditory system. (A) Sound. (B) Pinna. (C) Meatus. (D) Eardrum. (E) Ossicles. (F) Oval window. (G) Cochlea. (H) Basilar membrane. (Begault, 2000, p. 7)

On the listener's side, natural and synthetic hearing experiences consist of both physical and perceptual transformations of an incoming sound field (Begault, 2000, p. 7). What follows is a verbal description of the different parts in **Figure 18**, which represents the listener's auditory system. The figure's description shows what each label corresponds to. Sound is first transformed by the pinna (i.e. the visible portion of the outer ear) and proximate parts of the body such as the shoulder and head. Following this are the effects of the *meatus* (i.e. the ear canal), which leads to the middle ear. The middle ear consists of the eardrum and *ossicles*, which are the small bones popularly termed the hammer-anvil-stirrup. In the middle ear, sound is transformed from acoustical energy to mechanical energy. The ossicles convert the mechanical energy fluid pressure within the inner ear or *cochlea*¹³ via motion at the *oval window*. The fluid pressure causes frequency dependent vibration patterns of the *basilar membrane* within the inner ear. This causes numerous fibers protruding from auditory hair cells to bend.

Now that the path that sound takes before it is processed by the brain has been introduced, variances in user perception of 3D sound can be discussed. As could be seen in **Figure 18**, spectral filtering of a sound source caused primarily by the outer ear before the sound reaches the eardrum occurs when a sound source is picked up by the ear. This phenomenon is referred to as the *head-related transfer function* (HRTF).

¹³ In **Figure 18** the cochlea, denoted by the letter G is "unrolled" from its usual snail shell-like shape.

HRTF, or binaural HRTF when referring to both left and right ear HRTFs, can be thought of as a frequency-dependent amplitude and time-delay differences that result primarily from the complex shaping of the pinnae. (Begault, 2000, p. 41)

The folds of the pinnae cause minute time delays within a range of 0 – 300 μ sec that cause the spectral content at the eardrum to differ significantly from that of the sound source. The asymmetrical, complex construction of the outer ears causes a unique set of microtime¹⁴ delays, resonances, and diffractions that collectively translate into a unique HRTF for each sound source position. (Begault, 2000, p. 41)

HRTFs vary from person to person due to the fact that pinnae vary in overall shape and size between individuals. In a perfect world, custom HRTFs would be used for each person that would match their own body and ear geometry precisely, but this is not practical. Luckily, individual HRTFs are not that dissimilar and usually a generic reference set is adequate (Oculus)¹⁵. If the game's usage of 3D sound falls under a category where generic reference sets are not sufficient, this may perhaps explain discrepancies in audio source localization success among players discussed later in this document. However, further investigation is required to substantiate this claim.

2.5. Conclusion

This chapter discussed much of what shapes the current state of VR technology. From its early history and the advances it has made over time all the way to old and new challenges facing it today still. Along the way, concepts relating to the development and research of VR technology and content such as human perception and possibilities for practical VR implementation were discussed. Following this, audio and visual representations of media in VR as well as other digital platforms were explored. The goal of this chapter has been to provide a basic, yet broad and clear understanding of virtual reality as it can be seen and experienced today. This understanding should serve as a reference point for topics that are to be discussed later on in this work.

After having observed the world of VR and other related topics from a distance, the following chapters offer a closer look at how some of the principles discussed can and have been implemented in practice; more precisely, how those principles were

¹⁴ Very short interval of time. Measured as 0.01 millionth of a second. (Merriam Webster)

¹⁵ 3D Audio Spatialization:

<https://developer.oculus.com/documentation/audiosdk/latest/concepts/audio-intro-spatialization/#audio-intro-spatialization>

implemented in and affected the development of Nation Feathers VR. The first thing to be discussed is Nation Feathers VR's game concept. This will establish a good basis on which to later explore the specifics of the developmental thought process and implementation details.

3. Game Concept

This chapter explores what exactly Nation Feathers VR is and what it aims to achieve. Furthermore, it discusses the general game flow as well as the specific scenes and how they differ. The overall system structure is also discussed along with how its modules and individual components interact. Finally, the bird species chosen for the game, the criteria upon which they were chosen and the attributes used to describe them are all discussed later on in this chapter.

3.1. Scenes

In total, the game consists of three play modes and a menu. Together, they make up the game's four scenes. Two of those scenes serve the game's main purpose which is to educate and entertain. The remaining two scenes are auxiliary scenes that act as facilitators and familiarizing tools for the main scenes.

3.1.1. Menu Scene

The menu scene is the first scene players are met with and it is where they go after each level to choose which of the three play modes to play. It's made up of three images each labeled with a name and showing the scene it represents. As the player gazes at each of the images, they are enlarged to give feedback to the player. Using the trigger button, players can choose the scene they want and are then teleported to their chosen scene. Above the images sits the game's logo accompanied by the Cornell Lab's logo. The movement of the players is limited to head movement as there is little sense in moving around a menu screen. The back button can be used at any time to take players back to the menu screen. This can be seen in **Figure 19**.

3.1.2. Intro Scene

The Intro Scene is designed to familiarize players with game mechanics and controls. It takes a sequential or step by step approach to explaining the main aspects of the game such as player movement and interaction with the environment. As such, it follows a stricter narrative compared to the other scenes. This narrative mimics other typical video game tutorials where the player is required to finish the current task before moving on to the next one. To give the player instructions, a combination of 3D-objects, world space text and HUD text is used. Given the relatively limited 101° field of view offered by the Gear VR headset, screen space becomes a precious commodity to be used sparingly for those things that are most essential to the player. Thus, 3D-objects and world space text are preferred over HUD text whenever circumstances

allow their use, with 3D objects being best suited as they do not seem out of place and add to the immersion effect. The different types of instructions are shown in **Figure 20**.

After players walk a certain distance to a predetermined checkpoint, their movement is constrained as they are introduced to Unity's 3D sound by way of an animated bird (see **Animation**) flying around them in a circle. To calculate the bird's circular flightpath, the time taken to execute the last frame is multiplied by a constant speed value. The sine and cosine of the result is then calculated and multiplied by height and width values respectively to determine the size and shape of the spherical trajectory. This, essentially, determines the bird's x and z positions in the next frame. In other words, its next position on the flightpath. Finally, to have the bird revolve around a specific object or point, that object's position is added to that of the bird to give the bird's final position. To keep track of the bird's movement, 2 colliders are used; one that is attached to the bird and another that is static and acts as a wall. Each time the bird passes through the collider it is registered as an extra lap. After the bird has completed a certain number of laps, it picks the tree closest to it to "land" on. In other words, that tree becomes the new sound emitting object and players are tasked with finding it. Completing this, they are given their first look at interaction with objects in the game before going back to the menu screen to pick the next level to play.



Figure 19. Screenshot from main menu

3.1.3. Learn Scene

The first of the game's two main scenes, the Learn Scene is where players are given a chance to listen and see the birds before being tasked with identifying them in the Challenge Scene. Currently, there are eight birds that appear one after the other so that at any given time only one bird is vocalizing. Where, when and in what order they appear is not predetermined, but rather calculated in real-time depending on multiple factors.

Initially, there is a single bird assigned to a tree. This tree, which acts as the current sound emitting object, was chosen based on its relative location to the player. In more detail, whenever it is time for a new sound to be generated into the scene every tree's location is checked one by one and the first tree which is within a certain distance from the player is chosen. This distance is derived from the hearing range of the audio source and it helps make sure that the tree is neither so far away that the player can't hear it or too close that it would be too easy to find.

Once a tree is found, the next tree is similarly chosen to host the next sound. Each sound is given a difficulty level (see **Database**) which determines when it will be called. Lower difficulty birds are called to the scene first and when they have all been used up, then the more difficult ones start being used. Once the player finds the correct bird, the tree can be clicked to display its name, image, and a spectrogram of the vocalization. Finally, when all birds have been found, the player is prompted to click the back button to be taken back to the menu.



Figure 20. From left to right: 3D object text, world space text, screen space text

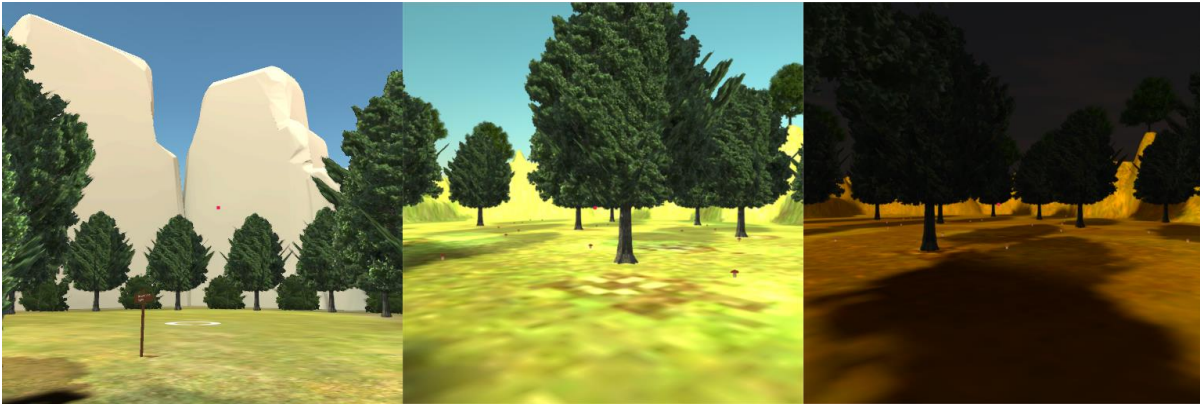


Figure 21. From left to right: Screenshots of Intro, Learn and Challenge Scenes

3.1.4. Challenge Scene

Similar to the Learn Scene, the Challenge Scene asks the player to acoustically locate the birds. The major difference, however, is that instead of the information being displayed right away once the bird is found, the player first needs to correctly identify the bird that is vocalizing. In order to achieve this, a signpost is dropped in front of the player for them to choose the bird (see **Signpost**). There is no limit to the number of tries the player gets and they can always come back to the same tree to try again at a later time. To not have to search for the trees all over again after finding them the first time, the trees are marked visually with the silhouette of a generic bird unrelated to the bird in question. This is paralleled in the Learn Scene by an actual image of each bird that remains on the tree once the bird is found to guide players back to its location if they ever wish to listen to it again. Once all birds have been found, regardless of whether they have been identified or not, the game prompts the player to press the back button again to return to the menu. At this point, if the player has already played through all levels they may choose to either replay a certain one or end the game. The different scenes are shown in **Figure 21**.

3.2. System Structure

Just as the game is divided into 4 scenes in total, the system is designed to have scripts¹⁶ with different levels of specialization. This means that while some scripts are assigned to one or a limited number of scenes, others are more general and are assigned to more or even all scenes. These core scripts dictate the progression of each

¹⁶ The scripts being referred to here are “core” system scripts, in that they influence major system components. Numerous other, smaller scripts are also required to make the game run, but they will not be mentioned in this section.

scene in accordance with its goals. On one end of the spectrum, extremely specialized scripts are those that are only found in a specific scene and are thus tailored to its requirements. On the other end, extremely general scripts are common to all scenes and are thus identical in their source code, if not necessarily in their functionality. As an example, the *GameController* script is an extremely general script found in all scenes which among other things, regulates the game by calling game state altering functions and influences system variables. However, even though it is shared by all scenes, its behavior is not the same for all of them. Within the *GameController* script, certain behaviors are triggered only in specific scenes while others are indeed common to all scenes. Not all scripts are equally specialized or general. They vary in the amount of scenes that share them. Whereas some scripts are exclusive to just one scene, others, such as the *GameController*, are common to all. **Figure 22** shows these relationships.

In order for scripts to be useable during the game, each script must be attached to a game object in the scene where it is to be used. This means that for every script that will be used in any given scene, a corresponding game object needs to be created to hold it. This game object need not have either form, shape or any geometry. In fact, it does not need to be visible at all. All that is required of this game object is that it is available in the scenes object hierarchy in order for its script to run and be accessed by other scripts.

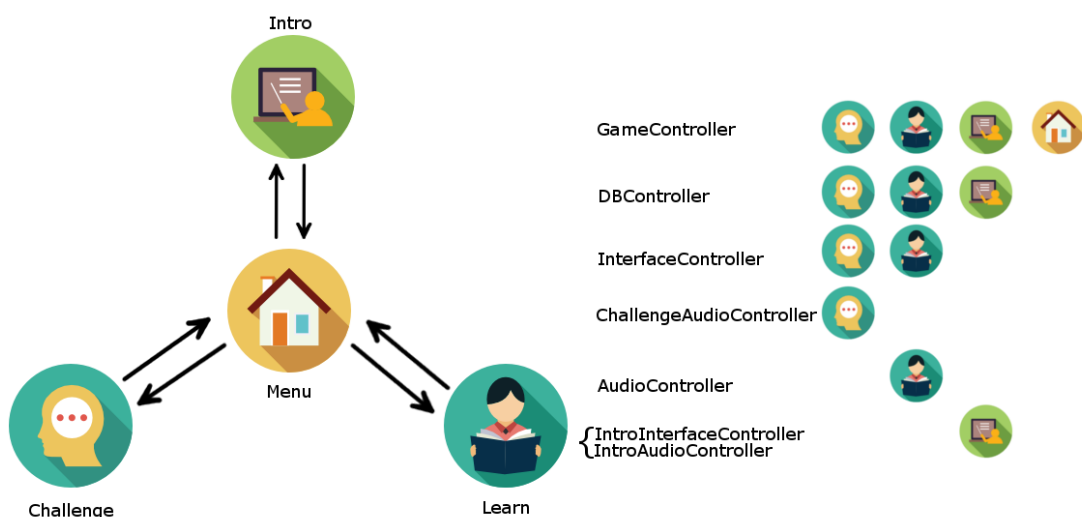


Figure 22. Overview of the game's scene structure and its scripts. The arrows show how the player can move between scenes in the game. On the right is shown how the scripts are assigned to the specific scenes. *GameController* for example, the most general script, is assigned to all scenes. *AudioController* on the other hand is very specialized and is assigned only to the Learn scene.

As mentioned earlier, *GameController* may be regarded as the game's main managing script. It regulates the flow of each scene and contains functions for moving back and forth between them. *DBController*, which is discussed in Database, contains all the necessary variables and functions for interacting with the Sqlite database. *InterfaceController* and *IntroInterfaceController* are responsible for managing their respective scenes' UI. The same applies for *AudioController* and *ChallengeAudioController* which are responsible for managing their assigned scenes' audio.

Other notable scripts are discussed later in **Implementation Details**. Some of these control more technical aspects of the game flow and are thus discussed in more technical detail and their functionalities explained. Others such as editor scripts (section 4.7) are only introduced and their influence outlined.

3.3. Birds

The birds featured in Nation Feathers VR are an important aspect of the game. Careful thought had to be put into deciding which birds made an appearance and which didn't. The first thought was the target audience. Although the game is meant to be played by both experienced and unexperienced birders and bird enthusiasts, certain considerations had to be taken in order to accommodate those unexperienced birders, since they are the ones who would benefit most from the game's learning phase. This meant that the majority of the birds picked to be included in the game are relatively easy to identify for an experienced birder. Experienced birders can still benefit and enjoy the game, the downside however is that the choice of birds may be too simple for them.

With Macaulay Library sitting right across the hall from BRP, it may seem like a trivial matter to accommodate both target groups by simply acquiring a larger set of birds which includes both easy to identify and challenging birds and simply letting the player choose the difficulty level at the start of the game. To explain why this is more difficult than it may seem at first glance, it must be made clear what exactly it means to include a bird in the game.

A bird component consists of a database entry, a spectrogram, a 2D image, a sound file, and a 2D animation. Each of these items must be created for and added to every bird species that is to appear in the game. So although countless audio files of bird calls

and songs may be available through Macaulay and other sources, each of these songs must be accompanied by the previously mentioned items. This takes time and is the main limiting factor when it comes to the inclusion of more bird species in the game.

Currently the game has a total of nine birds, eight of which appear in the Learn and Challenge scenes and one which appears in the Intro scene. **Figure 24** shows these birds and the attributes¹⁷ assigned to them. The images shown are not to scale.

Worth noting is that the *Time of Day* attribute shown for each bird does not relate to migration patterns, but rather to the time of day when these birds are active under normal circumstances. This distinction is necessary since some diurnal birds will migrate at night. For one, this behavior allows them to avoid daytime predators such as hawks and falcons. Another reason is that nocturnal migration typically offers cooler and calmer atmospheric conditions. (Alerstam, 2009)

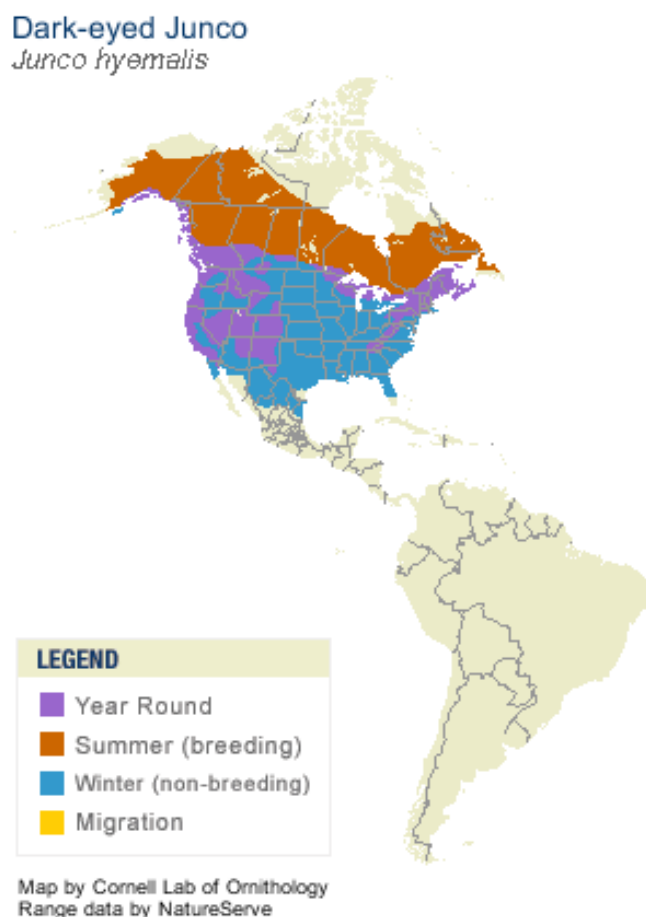
The *Time of Year* attribute, which indicates when the bird species can be found, relates exclusively to the species' presence in the Northeastern United States. For example, as **Figure 23** shows, the Dark-eyed Junco can be found year-round in the Northeastern United States. However, in Canada it can only be found during summer when it is breeding. In the Midwestern United States on the other hand, the species can only be found in the winter during non-breeding periods. This means that **Figure 24** will show 'year-round' for the dark-eyed junco's *Time of Year* attribute.

When choosing which birds to add to the game, one the main thoughts was recognizability. As mentioned before, the game focuses on North American bird species. While some birds are not seen that often and can cause quite a stir when spotted for the first time in a season, others are so ubiquitous year-round that they can be familiar to people who are not necessarily interested in birding. The latter are the main target group for the bird set used in the game. The nine birds used in the game, which can be seen in **Figure 24**, are among the top 25 most common birds in North America (Crossley, 2011).

Another important thing to consider was the kind of sound the birds produced. All these different bird species each have their own unique calls and songs. What is interesting however is that they are not equally identifiable by all their songs or calls.

¹⁷ The attributes shown in **Figure 24** are idealized, accurate attributes. For reasons discussed in section 4.1, the actual attributes used are more general and meant to apply to all birds.


































So as a general rule, only a bird's most distinctive sound would be used while other more ambiguous ones would not. Furthermore, some birds such as some species of woodpecker may be more associated with non-vocal sounds. An example of this is drumming. Drumming is when a woodpecker rapidly pecks on a resonant object to create a pattern of sound. Both male and female Hairy Woodpeckers, for instance, use their drum as a form of communication, which is why they may even drum on metal objects. That is not to say that woodpeckers cannot or do not vocalize. They can and do. However, most people associate a specific drumming pattern more than a specific call with a specific species of woodpecker. This is one the reasons why no woodpeckers make an appearance in the game, since the acoustic direction goes more towards bird songs and calls.



18

Figure 23. Range map of the Dark-eyed Junco showing year-round, winter, and summer presence of the species. (All About Birds)

¹⁸ https://www.allaboutbirds.org/guide/Dark-eyed_Junco/id

Name	Image	Habitat	T.o.Y.	T.o.D.
Northern Cardinal			365	
Red-winged Blackbird			365	
Bluejay			365	
Tufted Titmouse			365	
Pine Siskin				
House Finch			365	
American Goldfinch			365	
Dark-eyed Junco				
White-breasted Nuthatch			365	
White-crowned Sparrow				









Coniferous forests		Deciduous forests		No strict preference		Year-round		365
Winter		Shrubs		Marshes		Diurnal		

Figure 24. Birds featured in the game and their real attributes. Column T.o.Y. shows the time of year when the bird species can be found in the Northeastern US. T.o.D. shows whether the bird is diurnal or nocturnal (disregarding migration patterns). Habitat is a strict indication of the bird's preferred habitat; in reality it is possible that the bird is present in other environments. (All About Birds)

3.4. Conclusion

The design of the system discussed in this chapter serves as a good overview to bridge the gap between the theoretical and the practical. It talked about what exactly the game is and how its narrative progresses throughout. The role and narrative of each individual scene within the larger scheme of the game was explained. Furthermore, how those scenes relate to each other, their differences and commonalities, and why they are required by the game are important topics that were discussed. Finally, the last section of the chapter went over the thought process behind the selection of bird species for the game.

It is necessary to have a solid perception of how the different components work and fit together before their individual inner-workings are observed. The following chapter will take all that has been discussed here and inspect it through a different lens, namely that of the developer.

4. Implementation Details

Chapter 4 is the main chapter on development. It goes over much of how the functionalities and processes in Nation Feathers VR work. It starts off with a discussion on the database used in the game and how the data is organized and accessed. This is followed by a thorough inspection of the UI. Different kinds of UI, their purpose, and other reflections are examined. Following this, visual details such as lighting, animation, and different methods of creating the 3D objects for the game's environment are explained.

4.1. Database

Designing the database, there were a couple of things that needed to be considered. First, how can the data be structured so that in the future if the game's environment were to be expanded or become more varied that only birds which are suitable to that environment can be retrieved? For example, a barred owl may not be generated in a day scene and an upland sandpiper shouldn't be generated very close to a water body. Secondly, where is the actual data going to be stored and how can the desired amount of information be stored in the database tables with the least level of complexity?

Table 3 and **Table 4** show the attributes that make up a bird in the game. *Habitat*, *TimeOfDay* and *TimeOfYear* are the constructs necessary to generate birds catered to the environment. The current scenes don't make much use of this specificity as the lack of scenery diversity would make adhering to that rule very limiting in terms of the number of birds that can appear. So to accommodate as many different types of birds as possible, all birds being used currently are given attributes that allow them to exist in an arbitrary environment even if those birds may really only inhabit a specific niche.

An expansion or specialization of the environment (see **Figure 25**), however, would encourage the use of more personalized attributes as that would be truer to the species' natural behavior while at the same time allow the appearance of a larger number of diversified birds. How this may function is by having the game scout the area around the player similar to how it finds trees that are at an appropriate distance to host the next bird. Then, depending on where the player is and what type of environment surrounds them, the game will use the attributes given to each bird in the database to summon only those birds that fit that environment.

Table 3. Birds database table, red denotes primary key

Column	Type	Description
Name	CHAR(50)	Name and scientific name of bird.
Comment	TEXT	Brief information about bird.
Habitat	TEXT	Keywords that describe where the bird mostly vocalizes; e.g. tree, brush, etc.
TimeOfDay	TEXT	Keywords that describe when bird is active; e.g. dawn, dusk, etc.
TimeOfYear	TEXT	Seasons when bird is active.
Difficulty	INTEGER	General level of difficulty to identify bird under normal conditions.
ImageFile	TEXT	Name of image file associated with bird.

Table 4. Sounds database table, red denotes primary key, blue denotes foreign key

Column	Type	Description
ClipName	CHAR(100)	Sound file name of specific call or song for a certain bird.
Spectrogram ¹⁹	CHAR(100)	Video file name containing spectrogram of sound file.
BirdName	CHAR(50)	Name and scientific name of bird.
Difficulty	INTEGER	Difficulty to identify bird given this specific call or song.

As can be seen from **Table 3** and **Table 4**, the datatypes used are either text or numbers and do not hold the actual data files used. The data files are stored in Unity inside of a game object called *DBController* which, as the name suggests, controls all traffic to and from the database by providing the necessary functions for interacting with it.

¹⁹ This column was initially intended to be used to ease video file retrieval. However, video files are now being retrieved by matching the index of the sound file in the sound file list to the index of the corresponding video in the video file list. It has, thus, become unnecessary and, therefore, contains no data.



Figure 25. Depiction of a more diverse landscape that could make full use of the database’s bird attributes. Player shown as purple capsule.

4.2. User Interface

Designing UI for VR can be a challenging subject. The need for maintaining immersion is so great that every chance must be seized to somehow embed the interface into the environment in the form of 3D objects so that it blends in with its surroundings. The best example of this in *Nation Feathers VR* would be the signpost (see **Signpost**). To be able to use such game objects, however, there must be enough space to place them and they must also be placed in the player’s field of view as they should not be moved after their initial placement since they are supposed to be a static part of the environment. In cases where 3D object UI is not an option, standard UI utilizing images and text must be used. A good example of this is the *Infobox*.

4.2.1. Infobox

The Infobox is the main provider of information in the game. It consists of a text box containing the name of a bird, an image component containing the bird’s picture and a video player that plays a spectrogram of the bird’s vocalization (see **Figure 26**). In the Learn Scene, the Infobox appears in front of each tree containing a bird when the player interacts with it. A billboard script attached to the Infobox rotates it to make

sure it is always facing the player. The video player component is really an image that transitions between three textures.

The first is the video itself, which plays when the player clicks it. The other two are video icons, one being an idle icon which is shown when neither the video is playing nor the player is gazing at the video player and the other being a highlighted video icon that appears when the player gazes at the video player and the video isn't playing. Since the aspect ratios of the icons and the video player differ, the canvas, where all UI elements are drawn, needs to assume the dimensions of the item that is currently being shown. Otherwise, the image would appear either stretched or squeezed. The Infobox differs a bit in the Challenge scene where the player shouldn't be given all the information at first interaction. Instead, when the tree is first clicked the image of a silhouetted generic bird is displayed, which acts as a placeholder for the actual image and as a tree marker in case the player decides to return to the tree to answer the question at a later time. Only when the bird has been correctly identified is the normal Infobox displayed.

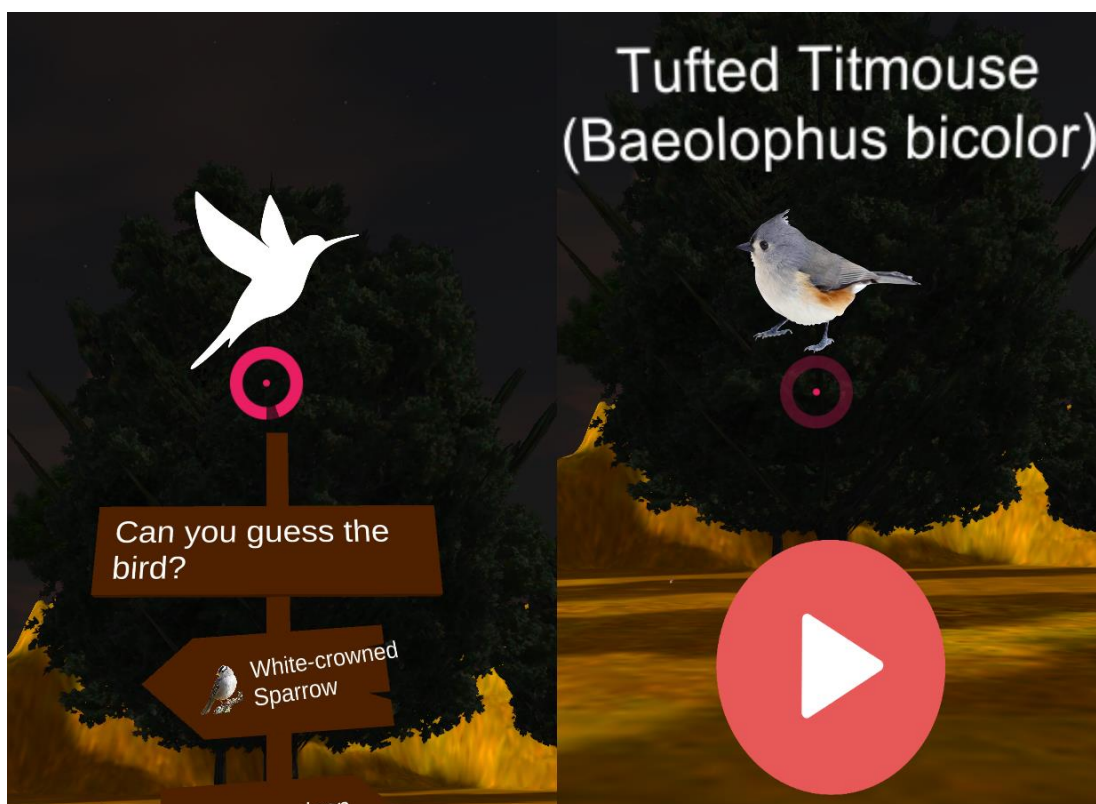


Figure 26. Screenshots from Challenge Scene. From left to right: Image of silhouetted, generic bird before correct identification and full Infobox featuring a tufted titmouse. Images also show animated reticle and partially visible signpost.

4.2.2. Signpost

The signpost is where the player attempts to identify the bird. Once the player has found the target tree and clicked it, the signpost falls from above at a certain distance away from the tree and in the player's line of sight. There is no restriction on the number of attempts allowed to the players, so they can technically brute force their way to the answer if they wished.

The signpost consists of five sections the top one simply holding text that prompts the player to answer. The other four sections are interactive and each contains the image of a bird and its name (see **Figure 27**). A false answer is met by a dull thud while a correct answer is met by a joyous ring and the bird swooping around the tree before the Infobox finally appears and the signpost disappears. If the player chooses not to answer the question, clicking the tree or moving away from it activates the next sound and deactivates the current tree, effectively removing the signpost till the tree is activated again.

To mark the tree as not yet answered, the silhouetted image of a generic bird is left behind on the tree so that players can find their way to it again. The text on the signpost becomes bold and underlined when the player gazes at it in order to give the player some feedback. The signpost is a great example of UI that blends in with its surroundings further enhancing the player's experience. An earlier version of the signpost was a regular UI with options similar to a menu that would float in front of the tree somewhat similar to the Infobox. However, since earlier tests showed that the signpost was both more visually pleasing as well as made more sense in the game's overall narrative it was chosen over a regular UI.

4.2.3. Reticle and Player Gaze

There is more than one way to point and click when it comes to the Gear VR. Many games use the controller as a pointing device where players move their hands around in order to point at menu items and other ingame objects. A thin beam of light is also extended from the controller, as it is represented inside the headset, to better represent where the controller is pointing. Several pointers are shown in **Figure 28**.

The way this is done in Nation Feathers VR is that the headset itself is used as the pointing device while the controller is used only to move the player and click. A reticle placed in the center of the screen acts as a visualization for a ray that is cast from the

camera that detects when it collides with other objects. When the ray collides with another object it looks for an interaction script that every interactive object needs to have attached. The interaction script then decides what happens in the case of a click, double click, simple gaze, looking away or theoretically a variety of other actions. The reticle as well as any other UI item drawn in screen space is rendered at the very end of the ray. Since the ray is constantly increasing and decreasing in length this means that any object rendered at the end of the ray will be increasing and decreasing in apparent size possibly causing discomfort or disorientation. Therefore, the length of the ray needs to be taken into consideration and the scale of screen space items adjusted accordingly to maintain that same apparent size. Another advantage to having the ray as a reference to the position of all items drawn in screen space is that there will be no variance in depth when two objects are drawn at the same time in screen space. This variance would cause the eyes discomfort as the eyes' lenses attempt to focus on one item or the other.



Figure 27. Screenshot of signpost from Challenge Scene.

To give players feedback when they are currently looking at a tree with a bird, a particle effect used to be used, which would appear around the tree. Although very aesthetic, this approach was somewhat computationally expensive and not worth the performance overhead associated with it. It has since been replaced by a less demanding revolving reticle effect and bird animations. The basis for the raycast system comes from Unity's standard assets. It must, however, be built upon and tweaked to suit the game's needs.

As the raycast is being used now, its length only changes depending on where it hits another object, all the while having a preset limit that it may not exceed. As different trees were being tested (see **Trees**) some of their shapes differed so that they became narrower as they got closer to their peaks. This had the effect that if the player was looking towards the top of such a peaked tree, the raycast would sometimes max out in length in its failed attempt to reach the tree's surface. To remedy this, the simple, yet limited, solution would be to increase the maximum length of the ray. Although this would solve the case where the ray cannot reach the tree's surface near the peak, it would mean that players can interact with the trees from a longer distance than was desired for the game. Another solution is to increase the length of the ray in relation to its angle so that as the player looks upwards or downwards the length of the ray increases. The change in the ray's length as its angle changes thus forms a semicircle. Finally, a more precise solution would use simple trigonometry to change the length of the ray as the angle changes. As the visual style of the game changed to feature trees whose surface was not as sharply curved near the top, the need for changing the length of the ray with the player's head movement became less important and the length of the ray returned to being constant.



Figure 28. Different pointers used in the Gear VR. Left to right: Normal reticle used in the game, animated reticle used in the game, particles used in game's first iteration, pointer used in Oculus Store.

4.2.4. Navigation Hint

Although Unity's 3D sound is very accurate and many user tests have proven its usability, others have brought to light that it is not perceived by all players the same way with many voicing difficulty in finding the audio source when given only acoustic hints to rely on. Therefore, a visual hint was incorporated to aid those players for whom the sound directions were too vague. This can be seen in **Figure 30**.

When a player double clicks the trigger button, a number of footprints appear in front of the player on the ground leading in the sound emitting tree's direction. The footprints appear one after the other and remain visible for a short while before disappearing again. Two images are used to create this effect, one for the left footprint and one for the right. In total, there can be no more than seven footprints at any given time. The number of footprints depends on the distance between the player and the target tree; the closer the player, the fewer the footprints. More precisely, the ratio between the distance and the number of footprints is 15 meters to 4 footprints rounded to the nearest integer.

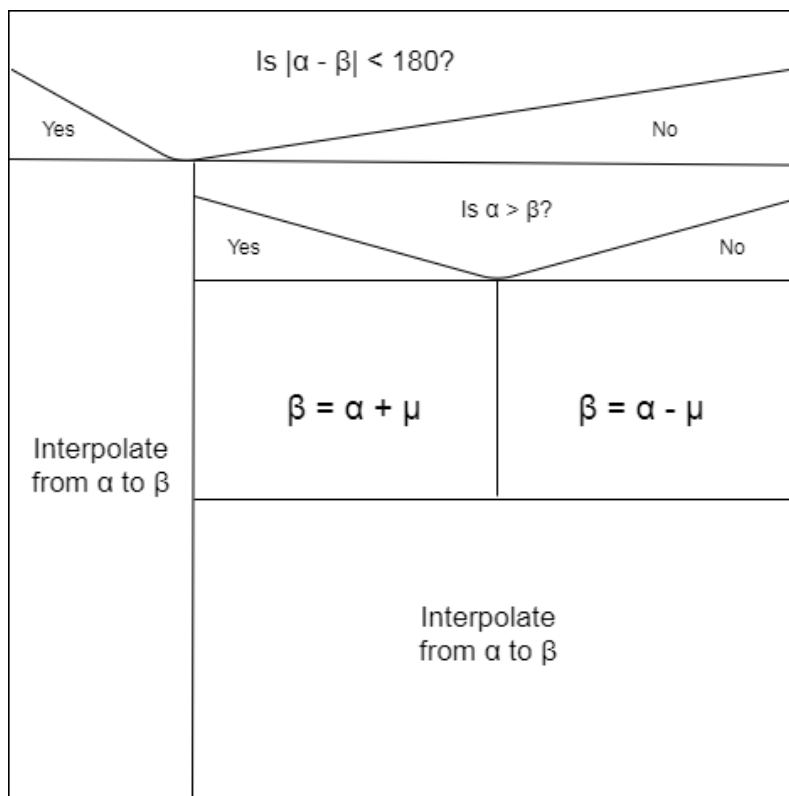


Figure 29. Algorithm to determine shortest path for footprints. α : initial direction, β : final direction, μ : smallest difference between α and β

Developing the footprints, a few main things had to be considered. The first footprint needs to appear in front of the player at a distance that would allow it to be seen clearly, assuming a natural front facing head posture of the player. The angle each footprint assumes must gradually point in the tree's direction so that the footprints as a whole form a curve starting in front of the player that smoothly turns towards the tree as accurately as the number of footprints used allows. This is only in cases where the tree does not lie directly in front of the player. If the tree does lie in front of the player, then the footprints will, of course, need to form a straight line. Finally, the footsteps must take the shortest path towards the tree and not force the player to make a left turn, for example, when a right turn would be faster.

To achieve all this, first, both the current direction the player is facing (initial direction) and the direction the player would be facing if looking directly at the tree (final direction) are found. Then, by using the 15:4 distance to footprint ratio, the number of footprints to be used is determined. The algorithm then interpolates between the initial direction and the final direction using the number of footprints used so far divided by the total number of footprints to be used as an interpolation value, which is a value between 0 and 1. The structure chart in **Figure 29** shows the method by which the shortest path is determined.

If the difference between the initial direction and the final direction is less than 180° , then the shortest path is a direct interpolation from the initial to the final direction. However, if the difference between them is larger than 180° , then a direct interpolation from the initial to the final direction would be the longer way around, thus requiring further manipulation of the angles in order to take the shorter path. In the example shown on the right in **Figure 31**, a direct interpolation from red to blue, i.e. from 0° to $\sim 195^\circ$, would take the red path which is longer than necessary. Subtracting 165° (smallest difference between 0° and 195°) from 0° gives -165° , which becomes the new final value, thus ensuring an interpolation in the opposite, shorter direction. Unity understands -165° to be equivalent to 195° as can be seen in **Figure 32**. Had the situation been flipped and the initial direction been at 195° and the final direction at 0° , then again the direct interpolation from 195° to 0° would have been the longer path. Adding 165° to 195° gives the new final direction of 360° , thus ensuring an interpolation over the shorter path.

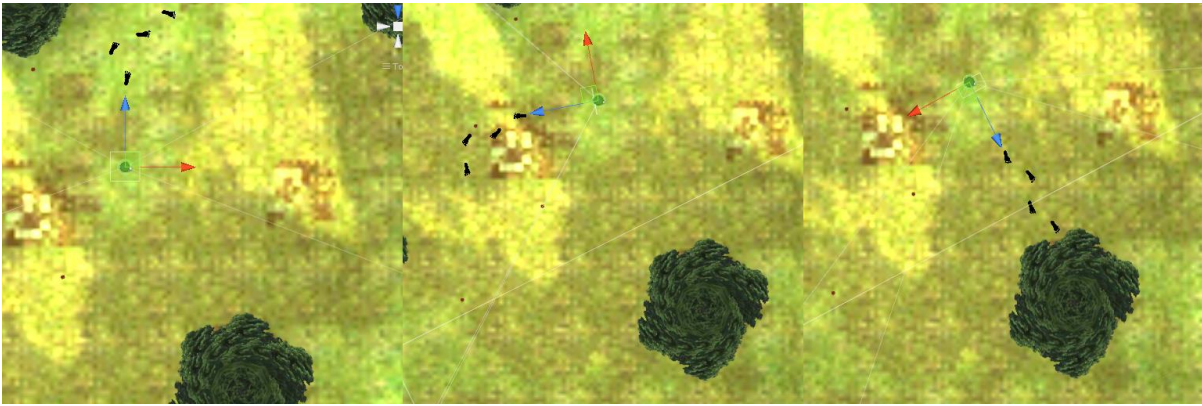


Figure 30. Screenshots from the editor view that demonstrate how footprints take the shortest path. The blue arrow shows the direction where the player is facing. The tree shown in the bottom right corner is the target tree.

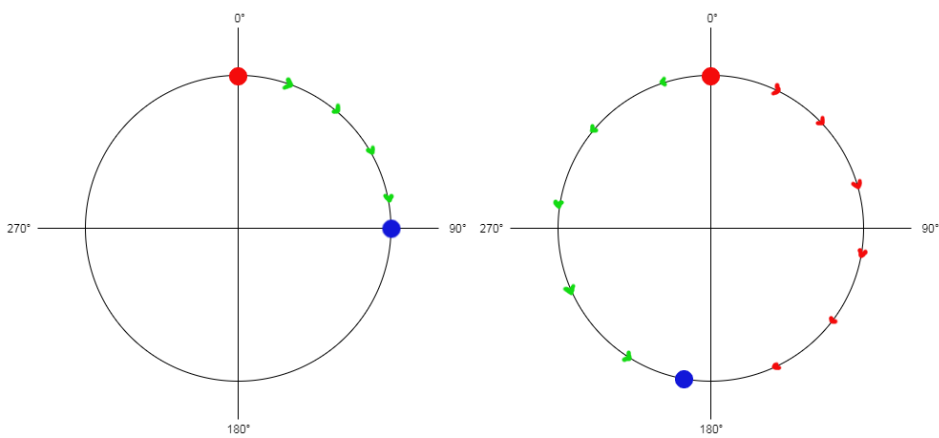


Figure 31. Left: interpolation path if $|\alpha - \beta| < 180^\circ$, right: interpolation paths if $|\alpha - \beta| > 180^\circ$ (Green is correct, red is wrong). Red and blue dots denote α and β respectively. α : initial direction, β : final direction.

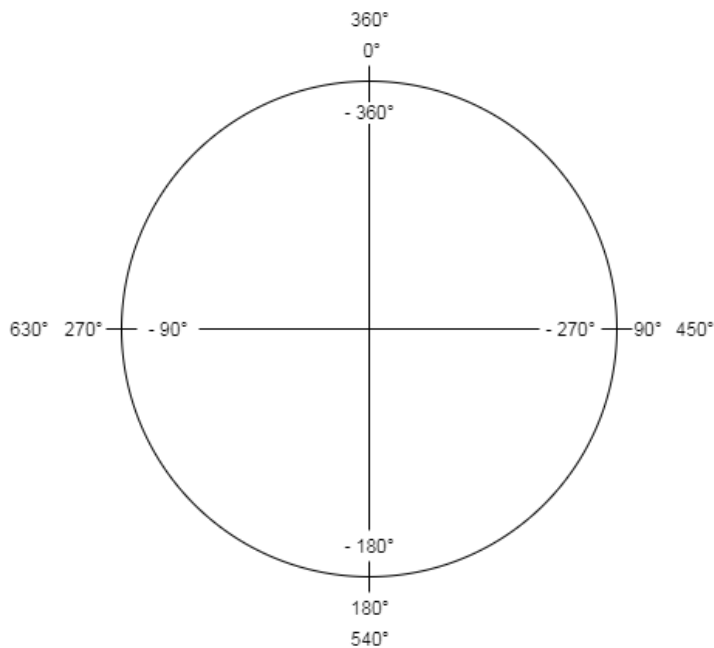


Figure 32. Angles as they are interpreted by Unity in the algorithm's use case.

4.3. Environment

The assets used in the game form a decisive contributor when it comes not only to visual appeal, but also to the game's overall performance. Nation Feathers VR has gone through more than one visual iteration each attempting to find a more suitable balance between aesthetics and performance. There are many factors that affect how an object may hinder or boost performance. The polycount in an object's mesh as well as the total polycount of all objects in a scene are two such factors. As mentioned in **3D Models**, the polycount is simply the number of faces, e.g. triangles or vertices, in a 3D object's mesh. The lower the polycount the less detailed it will seem, but at the same time more expensive to render. The opposite is true for 3D objects with a higher polycount. An even stronger factor is transparency and overdraw.

When a mesh is opaque the GPU has several techniques it can use to ignore surfaces that can't be seen in the final image. They are still not free, but can significantly reduce the cost of occluded surfaces. The basic technique here is depth sorting and early z rejection. When each opaque object is rendered it first tests to see if it is the closest object that's been rendered for that pixel, and if something else already rendered closer, the GPU does not render it. Transparent objects cannot do this since pixels cannot as easily be rejected just from seeing if something has rendered closer. This has to do with the nature of transparent objects as you can see more than one at a time if they are overlapping. Instead, the individual meshes are roughly sorted furthest to closest and rendered one after another.

The total number of objects used also takes a toll on memory and if it starts getting too much the game will become noticeably slower even if all the other factors are kept in check.

4.3.1. Grass

Grass has seen many changes till it reached its current form. The first thing to be tried was 3D grass blades, shown in **Figure 33**, each having 94 vertices and 42 triangles. Although completely opaque and thus avoiding any issues with transparency, the polycount quickly mounted up as the blades were scattered across the terrain. The next approach was to use single transparent planes, 8 vertices and 4 triangles each, with grass textures embedded on them. These planes would have billboard scripts on them so that they always faced the player and not seem to disappear when viewed from the side.

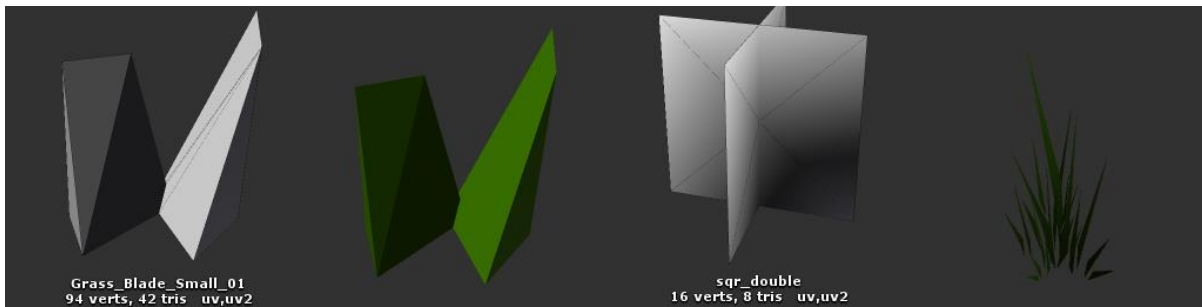


Figure 33. Left two images: Grass blade; mesh shown left. Right two images: Double plane grass; mesh shown left

As the number of planes grew, however, they started to take a toll on memory, thus prompting the use of the Mesh Combiner to combine their meshes into one single mesh placed on a single object. However, the fact that each plane needed to rotate individually to face the player meant that combining them into a single object would remove this ability as this single object, if treated as a billboard, would cause all planes to rotate synchronously as a group. Therefore, to eliminate the need for billboarding, double transparent planes instead of single ones were used so that as the player moves around such an object, instead of the plane disappearing when the player looks at it from the sides the second plane, which is placed perpendicular to the first one remains visible.

The low polycount of this grass type and combinability of its meshes made it, insofar as these two factors were concerned, very optimal. The issue that remained was that of overdraw. Although this phenomenon was not problematic when the grass was scattered across a terrain lacking trees, the addition of trees made the grass's presence unsuitable (see **Trees**). The trees being of higher priority than grass warranted yet another shift in the grass' implementation strategy. It was decided to simply paint the terrain using different textures of grass, mud and dirt and add opaque foliage such as mushrooms to create the forest floor.

4.3.2. Trees

Similar to the grass, trees were also tested profusely to achieve the best balance. The very first approach was to use low poly, stylistic models. These were used interchangeably for a period of time to see which ones struck the best quality-performance-balance. A desire to change the visual style of the game from a stylistic approach to a more real one, ushered a search for a different kind of 3D model; one

that was able to provide the level of visual fidelity expected from a realistic environment while at the same time not hinder the game. The first model to be tested was very similar to that of the transparent grass planes, but instead of the textures embedded being those of grass, the ones to be used here were to be tree textures and scaled to a much larger size.

Although providing very high quality trees for a fraction of the polycount, this method suffered from the same flaw with transparency and overdraw, but this time at a larger scale since the trees were much larger than the grass and were able to fill a very large portion of the screen if the player was close enough to them. So to reduce the amount of transparency in the mesh, an opaque “blob” was added to the center of the crown. Taking this one step further saw the transparent planes being trimmed to fit more closely to the visible parts of the tree’s texture. Even after all these optimizations, the game still slowed down when the player looked too closely at a tree. To improve this effect, the collider surrounding each tree was made bigger to prevent players from getting that close to trees. **Figure 34** shows important stages in the trees’ transformation.

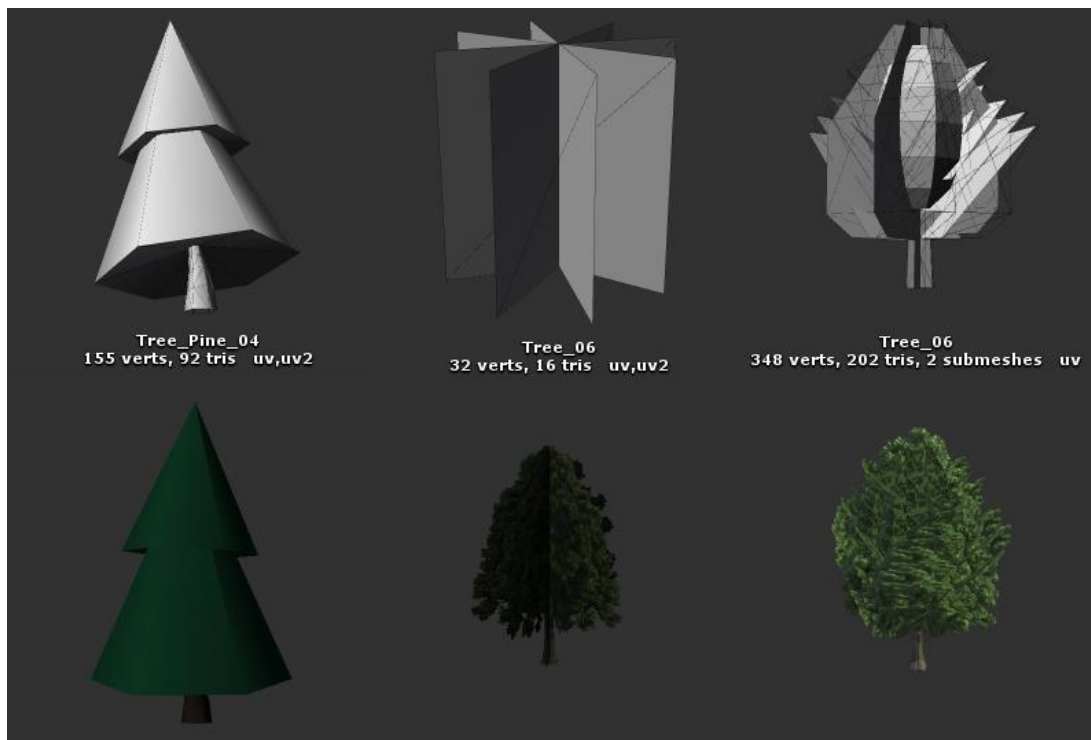


Figure 34. Demonstration of the main stages in the trees’ progression. Meshes shown in upper row. From left to right: Stylistic opaque tree, tree made up of transparent planes, semi-transparent tree with cutout edges and opaque blob (currently in use).

4.3.3. Boundaries

Games implement setting boundaries to their world very differently. Vast water bodies, insurmountable cliffs, finite natural resources or even spherical worlds are all ways with which this can be accomplished. Barriers can have more than one purpose.

In Nation Feathers VR, they are used to set a boundary to where the player can go as well as to prevent the player from seeing the edges of the terrain. Nation Feathers VR also tinkered with more than one approach when it came to boundaries, with the Intro Scene being the one exception. From the very beginning, the Intro Scene used large cliffs surrounding the terrain to act as physical and visual boundaries. Although their size meant that they took a lot of space in the lightmaps, the small size of the Intro Scene's map meant that a small number of cliffs can be used to encompass the whole map. An image of these cliffs can be seen in **Figure 35**.

Due to the size of the Learn and Challenge Scenes, they couldn't be surrounded by cliffs without the size of the scenes' lightmaps becoming unreasonably large. With natural-looking water being too demanding, the alternative was to use a wooden fence to surround the scene that acted as a physical barrier if not a visual one given its smaller height and partially see-through topology. So to prevent players from seeing the terrain's sudden drop off at the edges, the terrain was made longer so that all that was visible was a longer stretch of land before the terrain blended seamlessly with the sky. However, the fence didn't fit the game's forest narrative as it made the area seem more like an orchard or a park than a forest.

The fence was, therefore, removed in favor of a more natural looking barrier. This barrier came in the form of a ridge surrounding the map created by raising the terrain itself at certain points. This created a sort of depression in the terrain where the game takes place. At certain points along the ridge, single plane billboard trees are placed to lend a more scenic view to the barrier. **Figure 35** shows the different types of barriers used throughout the game's development.

To make sure curious players can't climb any of the natural barriers, invisible planes are placed in series close to where the natural barriers are.

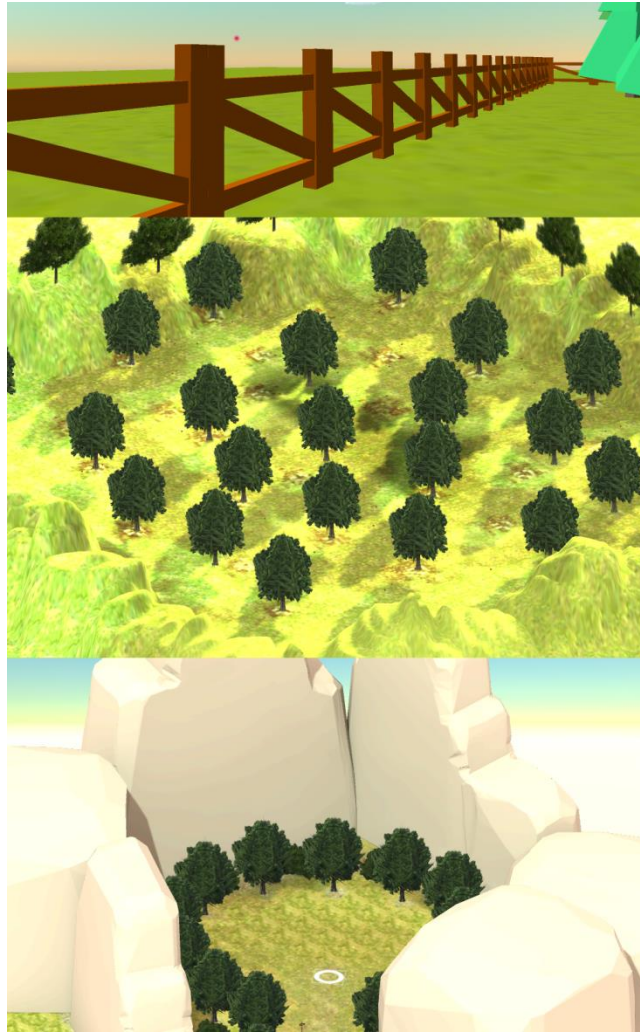


Figure 35. Barriers as they are used in different scenes. Top to bottom: fence used in first iteration's Learn Scene, terrain barrier currently used in Learn and Challenge Scenes, cliffs used in Intro Scene.

4.4. Animation

2D animated birds are used in all the game's scenes aside from the Menu Scene. They are shown circling other objects in response to different actions by the player. Although 3D animated birds would be more fitting to the game's style, 2D birds offer some advantages that make them a viable option as well. First, the availability of accurate 3D models of birds is scarce. For every bird in the game, there would need to exist a detailed 3D model that accurately captures that bird's identifying features. Furthermore, this model would need to be rigged to allow for joint movement so that the model can be animated.



Figure 36. Part of the image sequence of a flying blue jay.

Assuming that such models are available, their meshes would still need to be simplified to function on mobile VR and thus lose detail and identifiability. On the other hand, 2D images of birds are readily available and can be easily edited to form the different motions of a bird's flight sequence. Flipping through these images at a certain speed allows the transition to be perceived smoothly by the human eye. The same concept is used for GIFs or videos.

4.5. Creating the Spectrograms

The spectrograms used in Nation Feathers VR were created in Raven Pro 1.5 using default spectrogram parameters. These parameters are as follows:

- **Window function:** Hann
- **Window size:** 512 samples
- **Window overlap:** 50%
- **Discrete Fourier transform (DFT) size:** 512 samples

Raven provides a total of six different window functions (five of which are shown in **Table 5**). In order to understand what a window function is, *sidelobes* must first be introduced. Without going into too much detail, a *bandpass filter*²⁰ does not completely block the passage of all frequencies outside of its nominal *passband*²¹. For each filter there is an infinite series of diminishing sidelobes in the filter's response to frequencies above and below the passband. (Charif, Waack, & Stickman, 2010, p. 342)

²⁰ Allows frequencies within a certain range to pass and rejects frequencies outside that range.

²¹ The range of frequencies or wavelengths that can pass through a filter.

The magnitude of the sidelobes in a spectrum or spectrogram can be reduced by multiplying the record by a window function that tapers the waveform as shown in **Figure 37**. Each window function reduces the height of the highest sidelobe to some particular proportion of the height of the central peak; this reduction in sidelobe magnitude is termed the sidelobe rejection, and is expressed in decibels. Given a particular record length, the choice of window function thus determines the sidelobe rejection, and also the width of the center lobe. (Charif, Waack, & Stickman, 2010, p. 343)

The Window Size parameter controls the length of each data record that is analyzed to create each of the individual spectra that together constitute the spectrogram. The default unit for this measurement is number of samples. (Charif, Waack, & Stickman, 2010, p. 38)

Window overlap is usually expressed as percent of window size. For example, an overlap of 50% means that each window begins halfway through the preceding window. An overlap of -100% means that one window of data is skipped between successive windows that are analyzed; -300% skips three frames, and so on. (Charif, Waack, & Stickman, 2010, p. 121)

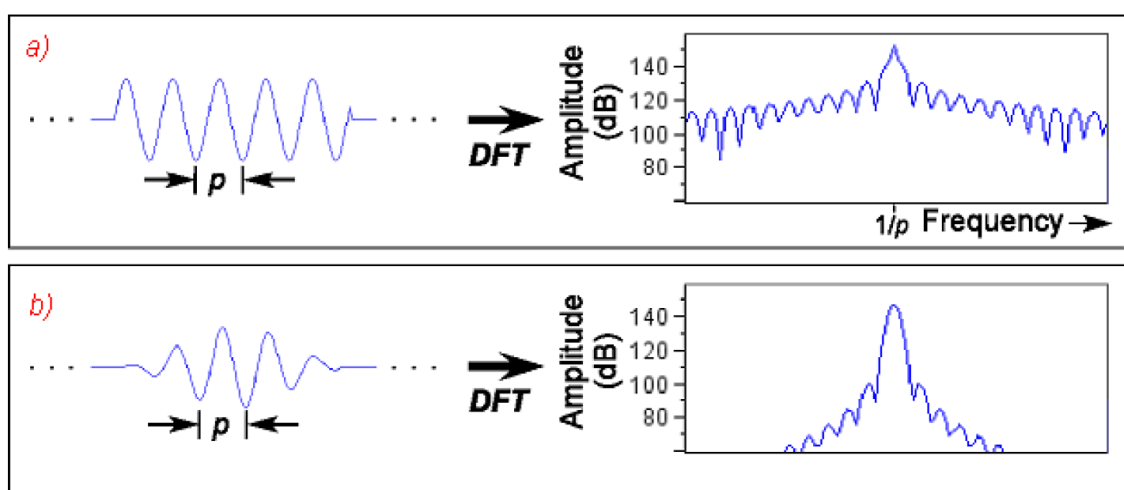


Figure 37. The effects of discrete Fourier transform (DFT) and the window function. Each panel shows a signal on the left and its spectrum on the right. (a) A single record of an untapered sinusoidal signal flanked by sidelobes. (b) A single record of a sinusoidal signal multiplied by a “taper” or window function has smaller sidelobes. (Charif, Waack, & Stickman, 2010, p. 343)

Table 5. Sidelobe rejection for Raven’s five window types. The sidelobe rejection for each window type is expressed as the height of the highest sidelobe relative to the peak of the main lobe. (Charif, Waack, & Stickman, 2010, pp. 343-344)

Window type	Sidelobe rejection (dB)
Blackman	-57
Hamming	-41
Hann	-31
Rectangular	-13
Triangular	-25

Finally, the DFT size helps determine the frequency grid spacing. The frequency grid spacing (FGS) can be understood as the height of the individual boxes in an unsmoothed spectrogram, which is simply a more accurate representation of the actual data in the spectrogram (see **Figure 38**). Another value, the *sampling frequency*, which is a fixed value for a given digitized signal, is divided by the DFT size to finally yield the FGS. (Charif, Waack, & Stickman, 2010, pp. 122-123). The formula is as follows:

$$\text{FGS} = (\text{sampling frequency}) / \text{DFT size}$$

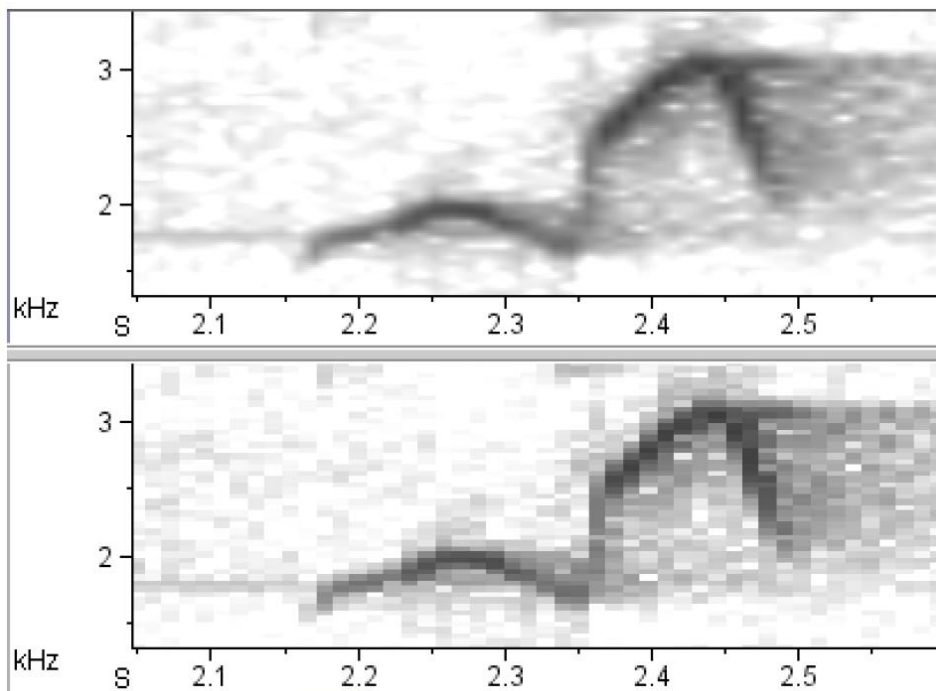


Figure 38. Close-up of portion of two spectrogram views of a signal digitized at a sample rate of 44.1 kHz. The upper view is smoothed while the lower view is unsmoothed. For both views, window type = Hann, window size = 512 samples, and overlap = 50%. (Charif, Waack, & Stickman, 2010, p. 135)

Raven is only used to create the spectrograms. At this stage they are not yet playable outside of Raven as videos. To create useable videos from Raven's spectrogram output, Camtasia™²² was used to take live captures of Raven playing back the spectrograms along with the sound. These live captures are generated as videos with specific parameters (see section 4.6) that are then used in the game. For every bird song or call used in the game, a spectrogram needed to be created for display. The process of playing video in Unity is discussed in much more detail in section 4.6.

4.6. Playing the Spectrograms

To get the spectrograms to play as videos in the game was no easy task and required lots of trial and error till a method was found that worked sufficiently and consistently well. The main issue was an unpredictable and inconsistent delay in the audio when the spectrogram was played. Furthermore, the problem seemed to worsen when the video was set to loop forcing interface design to have the player manually reactivate the video every time playback finished, which was not the original intention. Unity refers to Android's documentation when it comes to playing videos on mobile. So the logical thing to do was to create the videos with the same format that was recommended by Android.²³

When this wasn't enough to solve the audio offset issue, different formats were used. While some of these formats worked better than others, none of them offered a reliable standard to be replicated and used with confidence in the game. Further research suggested that perhaps the solution did not lie within the video format, but rather with how Unity played the video itself. Lines of code were thus added to regulate playing and stopping the videos. Essentially, what they do is prepare all resources and preload some or all of the content to be played before the video is actually played.

So far, this has yielded the strongest results and is what is currently used to power video playback in Nation Feathers VR. The fact that the videos are no longer dependent on format has provided the freedom to use formats that seem most suitable for the game. Currently, 720p, mp4 videos playing at 30fps are being used. A bug was later discovered that would cause the game to crash if a video was stopped by the player before it was done playing. So far no solution has been found to this

²² <https://www.techsmith.com/video-editor.html>

²³ <https://developer.android.com/guide/topics/media/media-formats.html#recommendations>

problem so the workaround for now is to prevent video playback from being interrupted.

4.7. Custom Inspector

The Inspector window in Unity displays detailed information about the currently selected game object, including all attached components and their properties, and allows for modifying the functionality of game objects in a Scene (Unity Technologies)²⁴. Although Unity's default inspector offers many options for controlling various aspects during development, it doesn't fulfill all the specific needs of every single developer out there. That's why Unity also offers the ability to create one's own custom inspector windows that can cater to the specific needs of the project and the developer. This has also been utilized during the development of Nation Feathers VR whether in the form of custom inspectors acquired from the Asset Store or ones written specifically for the needs of this project. A major advantage of custom inspectors is that they can give scripts the ability to execute in *edit mode*, i.e. when the game is not running. **Figure 39** shows a custom inspector.

Examples of custom inspectors used include:

- **Aspect Ratio Adjuster:** Maintains the original aspect ratio of an image while setting it to a custom height.
- **Asset Usage Detector:** Shows a list of all game objects that are using a specific asset.
- **Give Positions:** Given a group of objects, creates a new group of objects and gives them the same positions as the first group. Differs from duplication in that the new group may contain different objects than the first group.
- **Select Specific:** Given a string and a parent object, can choose all child objects whose name either matches or contains the string.
- **Selection Mover:** Given a selection of objects and a target parent object, assigns the selection to the target parent.
- **Select Lightmap Static:** Selects all objects in the scene that are marked lightmap static.

²⁴ <https://docs.unity3d.com/Manual/UsingTheInspector.html>



Figure 39. Part of the inspector window of an object with the Mesh Combiner editor script attached to it. If the script were left unaltered, this part of the inspector window would contain nothing but the name of the script. However, for the purpose of combining game object meshes in edit mode, the editor script was altered to add two functional buttons which, when clicked, combine the meshes of all the game object's children into a single mesh that is added to the game object.

- **Mesh Combiner:** Combines the meshes of many objects into a single mesh.
- **Generate Terrain:** Given an object or a group of objects, initializes them and scatters them randomly across the terrain. Takes into consideration the needed distance between each object and the distance to the edge of the terrain.

4.8. Lighting

Lighting a scene is a computationally expensive task. This is especially true for mobile VR where optimizations need to be made whenever possible. Left to its own devices, Unity will make all light calculations in real-time. This means that shadows, direct and indirect light and other lighting effects will be generated frame by frame as the game runs. This causes performance drops that would render the game unplayable due in no small part to lags. Lags, being a universally unwanted side effect in games, are doubly unwelcome in VR as the consequences, depending on the player, may include visible discomfort and possibly even nausea.

In the case of lag, mismatched motion will usually be the culprit. As an example, this can be seen when a player moves their head, but the image fails to follow in real-time and only does so in a jittery motion, in one sudden motion or not at all. So to avoid having real-time illumination, so called light maps can be used. Light maps are created when the lights in a scene are “baked” before the game is run. Baking is simply a term referring to the precomputation of light in a scene. These precomputations are saved as textures called light maps that are placed on the objects whose light has been baked. For light to be baked for an object, it needs to be marked as static meaning that it will not be moved once the game has started.

If a static game object does end up moving in the game, its pre-calculated light map will not be updated to reflect this movement, resulting in an unnatural discoloration of the moved object and the area surrounding it. This means that light maps can be created for all objects that are guaranteed to not move in the game such as trees, rocks, cliffs, etc. Dynamic objects, however, that are meant to move in the game and are not marked as static need to be illuminated differently. Options include using unlit shaders with these objects.

Shaders, which were visited in section 2.4.2, are small scripts that contain the mathematical calculations and algorithms for calculating the color of each pixel rendered, based on the lighting input and the Material configuration (Unity Technologies)²⁵. Unlit shaders in particular are not affected by illumination so they can be used without risking the extra load of calculating light in real-time. The downside of unlit shaders is that they don't respond to the lighting in the scene. For example, a scene may be too dark or too bright or anywhere in between and the object using the unlit shader will not show any change depending on its surroundings. A workaround here is to use custom textures and materials that match the lighting settings in the scene so that even without calculating light for the object it will still fit in with its surroundings right out of the box.

Lacking this, however, developers may resort to a mixed lighting mode where static objects can be baked as usual while dynamic objects have their light computed in real-time. It must be noted, however, that at any given time there may only be a limited number of dynamic objects in the scene, as an excess would lead to the original problem that was circumvented by baking. Luckily for Nation Feathers VR, the game makes good enough use of unlit shaders for the bird animations (see **Animation**) that almost no real-time lighting is required.

²⁵ <https://docs.unity3d.com/Manual/Shaders.html>

5. Evaluation

In order for any product to be deemed successful, it needs to be measured by how well it achieved the goals it set out to achieve. Nation Feathers VR set out to help its players learn about identifying bird vocalizations while offering a pleasant VR experience. Thus, the game's success must be measured by how well it achieved those goals.

This chapter attempts to analyze, evaluate, and assess to what degree the properties mentioned above were realized. This will require a review of the results achieved by the work, how it was received by the people who tried it, and how efficient it really is at performing its task.

5.1. Training Effectiveness and Enjoyment

This section discusses how well the game achieved its main goals: effective learning and entertainment. These two points can be said to be independent of each other; a player can have an enjoyable experience while learning nothing or learn without necessarily having enjoyed the game.

This infers that the two events are also not mutually exclusive, which is essential for the game's goals. As a result of this, the measures of entertainment and training effectiveness are to be inspected separately from each other. Also related to these two aspects is ease of use and usability.

The usability of the product can be a great determinant of both enjoyment and learning. If the product is too difficult to use, players will have to spend some time before actually being able to use it intuitively if they even reach that point at all. This will have a negative effect on the general enjoyment of the experience.

Furthermore, if players are too busy getting used to the UI and controls then much of their concentration will be wasted on that rather than the actual content. On the other hand, an intuitive game would allow its players to get right into the game and spend most of their time on actually solving the task at hand.

The participants or players whose experiences form the basis of this evaluation cover a wide age range from very young participants aged 6 to 10, to older ones up to age 60, and many ages in between. The participants also varied in their levels of experience

with VR and video games in general as well as their knowledge of birds and identification.

As mentioned above, the game's overall success will be measured based on 3 pieces of information provided by each participant. The first one is *training effectiveness*. Training effectiveness for each player is a percentage rounded to the nearest 10 that shows how many correct identifications were achieved by the player out of the total number of identifications attempted. As a prerequisite for a reliable answer, the player must have also played the learning phase of the game at some point in time prior to providing his or her identifications in the challenge phase. This is to make sure that the knowledge the player is utilizing in the challenge phase was actually provided by the game's learning phase.

Of course, this assumption can only really be made for players who are to some degree new to or unfamiliar with birding and bird vocalizations. With more experienced birders who are participating, the effectiveness of the game's training process cannot truly be gauged since it is unknown whether the player already possessed this knowledge before attempting to make the identification or whether he or she acquired it while playing through the learning phase. Furthermore, since the number of tries available in order to identify a bird is unlimited, a threshold has to be determined to differentiate a correct answer from a random or lucky guess.

For the purposes of the evaluation 2 tries were chosen. This means that a correct identification that took place after 2 or more false guesses does not count as correct, but rather as a lucky guess. In order then for an identification to be viewed as correct, the player must have correctly identified the bird in one or two tries. This 2-try threshold was determined based on the fact that there are 4 choices in total available when a player is choosing the correct answer. Furthermore, in order to make the percentage values more reliable, only participants who have attempted at least 3 questions will be included in the analysis.

The remaining two properties, *ease of use* and *enjoyment*, are each provided by the player on a scale of 1 to 10, 10 being the most positive result in both cases. Acquiring the values for *ease of use* and *enjoyment* happened by collecting verbal feedback from the participant after playing the game. *Training effectiveness* could be determined by

monitoring the player's actions through an image sharing program called *MirrorOP*²⁶. MirrorOP consists of two components, a sender app and a receiver app, that work together to display the image from a source device onto a target device.

To make this work, the source device (the phone where Nation Feathers VR is running) must run the sender app and the receiver app would run on a computer. Since MirrorOP transfers the image to the target device just as it appears on the screen of the source device, this means that the final image appearing on the computer is a stereo image containing the two feeds meant for each eye that the VR headset corrects so that it is seen properly by the user.

The participants seen in **Table 6** had a chance to play the game at different times during late development. In other words, many of these participants played the game before development officially ended. This means, it may be the case that some minor features were added after, which those players did not get to see. However, due to the fact that all evaluation values were taken during late development, the core of the game was established and the version which was played looked, felt, and played very similarly to the final version.

Participants are not mentioned by name and are categorized into three age groups: 6-12, 13-30, and 31-60. These age groups were chosen on the basis that their members are more likely to have similar responses to VR and similar experiences, or levels of competence, with video games in general.

Due to VR's relatively lower dissemination compared to regular video games, a participant's level of VR experience is based on whether he or she has already previously tried VR at least once. For regular video games on the other hand, a participant's level of experience is determined by whether they currently spend or have previously spent a sufficient portion of time playing video games that they possess a good level of familiarity with them. The kind of video game does not play a role here.

²⁶ <http://www.mirrorop.com/>

Table 6. Player feedback and performance. Each row represents a player. Main attributes of success are given by *training effectiveness*, *entertainment*, and *ease of use*. *Training effectiveness* is the percentage of correct answers from the total questions attempted rounded to the nearest 10. *Enjoyment* and *ease of use* are values on a scale of 1-10 (10 being most positive) provided by the players.

Age Group	VR Exp.	Gaming Exp.	Birding Exp.	Training Effectiveness	Enjoyment	Ease of use
6-12	No	Yes	No	40%	9	3
6-12	No	Yes	Yes	100%	8	8
6-12	No	Yes	Yes	80%	9	6
6-12	Yes	Yes	Yes	90%	6	6
6-12	No	Yes	Yes	70%	8	8
6-12	No	Yes	No	10%	8	2
6-12	No	Yes	No	30%	9	7
6-12	No	Yes	No	50%	9	7
6-12	No	Yes	No	30%	9	5
13-30	No	No	No	20%	6	5
13-30	No	Yes	No	50%	8	7
13-30	No	Yes	Yes	100%	10	10
13-30	No	No	No	60%	8	9
13-30	Yes	Yes	No	60%	7	10
31-60	No	Yes	No	40%	8	7
31-60	No	Yes	Yes	100%	3	7
31-60	No	No	No	50%	6	7
31-60	Yes	Yes	Yes	90%	7	10
31-60	No	Yes	Yes	100%	5	4
31-60	Yes	Yes	No	60%	7	10
31-60	Yes	Yes	Yes	100%	9	10
31-60	Yes	Yes	No	80%	9	10
31-60	No	No	No	20%	4	4
31-60	No	No	No	30%	6	6
31-60	No	No	No	40%	5	4

Looking at the 25 entries in **Table 6**, some general trends start to appear. One of the most obvious ones, and perhaps no surprise, is the correlation between birding experience and learning effectiveness. All participants who stated having birding experience achieved higher than 50% in learning effectiveness, with the lowest value even being 70%. Perhaps another value that was to be expected is generally higher levels of enjoyment among younger participants. This seems to hold true regardless of the participant's perceived ease of use, which is the property that has the most potential to negatively affect enjoyment

One other less obvious trend that can be deduced from age and ease of use is that the lowest ease of use values appear in the age group 6-12. This suggests that participants above the 12 year old threshold had an easier time coping with the controls and UI and following instructions. Aside from the obvious intellectual advantage that older participants have, some of the young participants closer to the lower end of the 6-12 range had smaller hands such that they had to grip the controller with both hands in order to reach all the necessary buttons. This natural physical disadvantage may have hindered their ability to interact with the game the same way as older players.

Two other interesting trends are how gaming experience affected ease of use and how ease of use affected enjoyment. In the 6-12 age group there seemed to be no correlation between the latter two properties, i.e. ease of use and enjoyment. **Figure 40** shows graph representations of all the relationships and trends mentioned above. For the former however, it would be difficult to deduce any reliable information since the number of participants with no gaming experience is much lower than that of participants with gaming experience.

5.2. Success Verdict

As mentioned above, if the game's overall success is to be determined by the properties ease of use, enjoyment, and training effectiveness, then averages will have to be calculated for each of these properties. First, measuring overall training effectiveness. To be able to determine a fair training effectiveness average, it makes sense to include only participants who do not have any significant birding experience (the red bars in graph 1 shown in **Figure 40**). This would ensure that the training effectiveness achieved was directly influenced by the players' experiences in the learning phase.

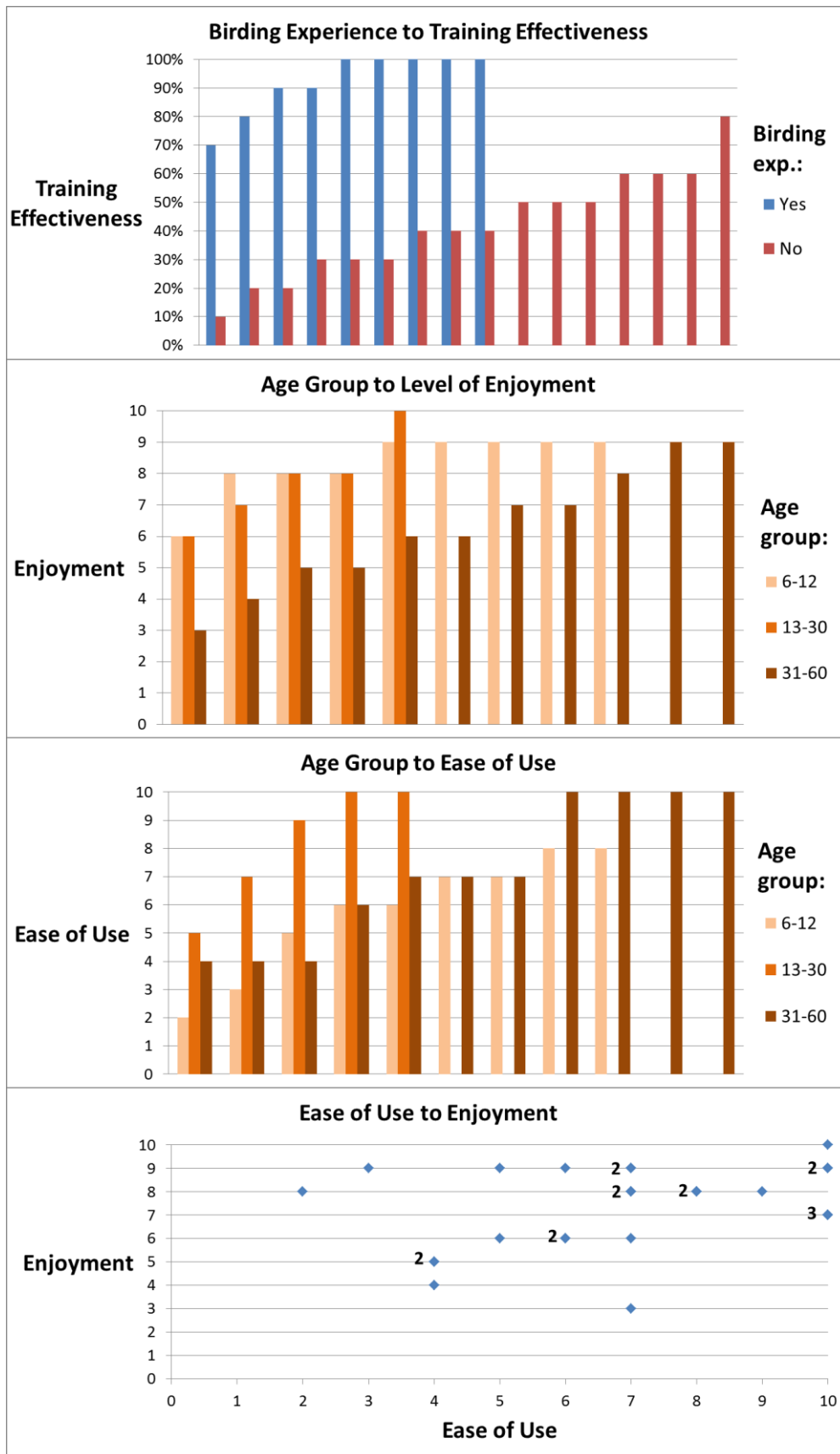


Figure 40. Values from **Table 6** represented as graphs showing potential correlations. Each bar represents a participant. Top to bottom: 1) How birding experience affects training effectiveness. 2) How enjoyment varies across age groups. 3) How ease of use varies across age groups. 4) How ease of use affects enjoyment. Number labels indicate duplicate values. In the bar graphs, group values are arranged horizontally in ascending order.

Furthermore, the typically higher training effectiveness values of participants with birding experience may yield a higher average leading to the conclusion that the learning phase provides a very effective training method, even when this may not be the case.

Using these models, calculating the average values for the three properties would yield approximately:

- **Training effectiveness average (no birding experience): 42%**
- **Enjoyment average: 7**
- **Ease of use average: 7**

Analyzing these values, it can be concluded that UI, level design, instructions, and controls managed to perform their task well to provide the player with an intuitive experience allowing for more time and energy to be invested in the actual task at hand. The same can be said for the enjoyment average. The game was able to trigger positive responses from most participants and it can be said that in general the game offered a pleasant experience for most.

However, the results for training effectiveness were not as positive. According to these results, when it came to training effectiveness the game's performance as a learning tool was less than optimal. On average after a single learning session, players were able to correctly answer approximately 40% of all questions attempted. It is unknown whether these results would have been better had the learning phase been played more than once. Of course, to be able to deliver more conclusive answers regarding training effectiveness a larger, more diverse group of participants would have to take part in the evaluation.

Regardless however, there are some straightforward alternatives that may improve training effectiveness without drastically changing the game mechanics. First and foremost is the time taken between learning and testing. As players go through the learning phase and are introduced to the eight bird species featured there, they have a chance to spend as much time as they would like listening to the vocalizations and hearing them all multiple times. Not all players necessarily make use of this

opportunity. It may be the case that constant learning without any mental stimulation to offer a change of pace causes players to want to get to the challenge phase as quickly as they can.

A possible solution to this is to offer players the chance to test their newly acquired skills while they are still in the learning phase. This could be done by means of “pop quizzes” that would appear a little while after certain birds have been heard to test the player on the birds’ vocalizations. This way, players can better familiarize themselves with a vocalization by having to access from memory that newly acquired piece of information soon after acquiring it. Furthermore, it lowers the risk of the game flow becoming monotonous by constantly switching the task from learning to challenge.

To conclusively improve the game’s training effectiveness would require a much more comprehensive study of how players felt about the learning phase and what they think may have made the experience more beneficial to them. This can be done through surveys, for example. Moreover, further variations of the game narrative in the learning phase can be explored and their performance thoroughly examined in order to determine their effectiveness.

6. Conclusion

Developing for mobile VR certainly placed some constraints as to what may be included in the game and how ideas needed to be implemented. From the art style all the way to how data was saved and accessed in the game, careful planning and execution was required in order to deliver an acceptable, stable experience. And even with all those thoughts in mind the game still has its fair share of performance drawbacks. Occasional spikes in resource usage triggered by certain ingame actions are often noticeable, which is an undesirable feature in any product. Two main reasons for these performance drops are the artistic direction and the design decision to generate game assets in real time.

Notwithstanding the painstaking measures taken to model efficient 3D objects, which were described in **Environment**, the decision to give the game a more realistic look was a risky one. A stylistic environment, although not providing the same visual appeal many players come to expect nowadays, would not have been as computationally demanding as a realistic one.

Generating objects in real time, as opposed to having the location and the order of the birds be predetermined, became part of the game design very early on during development. It aimed to capture that sense of the unknown experienced during birdwatching when it isn't known where or what the next bird to sing will be. The other reason why this decision was made, as mentioned in **Game Concept**, was so that it can be ensured that the player will always be in hearing range of the next vocalizing bird regardless of where the player is on the map. This is done by finding the location of the player and going through all of the game's trees to find one which is at an appropriate distance from the player to which to add the bird. This, of course, has the disadvantage of increasing the processing demand while the game is looking for an appropriate tree. And this is precisely what happens when a player clicks a tree for the first time, which is the moment when the game starts looking for the next tree. It can be expressed simply as the game trying to do many things at once; from rendering the infobox with all of its contents to finding a tree, as well as a bird, and assigning them to each other.

Regardless of these shortcomings, being conscious of resources and occasionally resorting to minimalistic designs should by no means be thought of as a holdback. Even on high-end systems, optimization becomes imperative as datasets become

larger and instructions become increasingly complex. Learning to develop with that in mind may save all parties involved a lot of time and energy. Nation Feathers VR tries to incorporate as many of these ideas as possible, in order to create as seamless an experience as can be achieved. As it stands, an educational VR game for identifying bird vocalizations has been created and although the feedback from tests and playthroughs has generally been positive, it remains to be seen how the game can be made into a more effective learning method

Looking down the line, there remains much that can be improved and expanded upon in Nation Feathers VR. Currently, there are only nine bird species that can be seen in the game and this list can easily be augmented to include more diverse species. As discussed in **Implementation Details**, this should be coupled with an expansion of the environment to include more varying landscapes, seasons and terrains. Doing this will allow these birds to each have more specific attributes that more accurately reflect their habits and habitats. This way, players will not only be treated to a more visually appealing environment, but they can also have the choice of learning about or identifying birds from specific regions or birds that exhibit specific behaviors.

As for performance, ways to optimize the game's two main bottlenecks described earlier, namely the realistic art style and the real time generation of assets, must be found. An overhaul of the game design may be necessary that would change the way assets are accessed during gameplay so that the system is not burdened with too many tasks at the same time. Furthermore, more efficient 3D objects can be researched and modeled that offer a better balance between render demand and visual fidelity. Another alternative would be to resort to the original plans of having a more stylistic approach to the game's design.

When it comes to gameplay, the game does not have to be limited to just two game modes, i.e. learning and challenge. Rather, more game modes should be offered to give the player more variety and freedom of choice. For example, game modes involving soundscapes can be added where a certain bird song needs to be made out or where the player needs to determine how many bird species can be heard in a given soundscape. Another mode could test players on their ability to determine what part of the world a certain species comes from. Of course, this would require that birds from other parts of the world, not just North America, be added to the game. Moreover, since the same bird species may produce different vocalizations depending

on different behaviors or actions, it may be an interesting, if a bit advanced, game mode if players were to guess what behavior a specific call or song signifies.

Moving forward, one thing that must be kept in mind is user feedback; regardless of what the developer does, it all comes down to how the user, in this case the player, perceives it all. No matter how justifiable or logical a feature or aspect of the game seems in the eyes of the developer, at the end of the day if it does not resonate with the player, then something needs to be changed. It is therefore imperative that frequent user tests be carried out among a large, diverse group of participants and the feedback noted and incorporated accordingly. Being involved with a project for some time can and will blind developers to flaws or ambiguities in the product. Part of the reason for this is that they have grown so accustomed to it that it all becomes intuitive to them, which often may not be the case for normal users. So letting players in on the development process from the very beginning will no doubt help prevent this from happening.

In closing, while Nation Feathers VR contains much room for expansion and improvement, it offers a solid basis for future work and a different approach to both gaming and birding in its use of sound as a game mechanic. Hopefully, such educational game models used in combination with VR will draw in both birders and non-birders alike, especially younger audiences. This can help raise awareness of wildlife conservation efforts worldwide not only for birds, but also for many other species, such as elephants and marine life, if the game is adapted to focus on them.

Bibliography

(n.d.). Retrieved September 11, 2017, from Arch Daily:

<http://www.archdaily.com/19263/cornell-ornithology-laboratory-rmjm>

(n.d.). Retrieved December 09, 2017, from Merriam Webster: <https://www.merriam-webster.com/dictionary/microtime>

(n.d.). Retrieved from All About Birds: <https://www.allaboutbirds.org>

Alerstam, T. (2009). Flight by night or day? Optimal daily timing of bird migration.

Journal of Theoretical Biology, 530-536.

Arvanitis, T. N. (2006). Virtual Reality in Medicine. In A. A. Lazakidou, *Handbook of Research on Informatics in Healthcare and Biomedicine* (pp. 59-62).

Begault, D. R. (2000). *3-D Sound for Virtual Reality and Multimedia*.

Bioacoustics Research Program. (n.d.). Retrieved September 11, 2017, from Cornell Lab of Ornithology: <http://www.birds.cornell.edu/brp/>

Bird Academy. (n.d.). *Visualizations Can Help*. Retrieved September 11, 2017, from All About Birds: <https://academy.allaboutbirds.org/topic/visualizations-can-help/>

Charif, R. A., & Pitzrick, M. (2008). *Automated detection of Cerulean Warbler songs using XBAT data template detector software*. Cornell Laboratory of Ornithology. Ithaca, NY: Bioacoustics Research Program Cornell Laboratory of Ornithology.

Charif, R., Waack, A., & Stickman, L. (2010). *Raven Pro 1.4 User's Manual*. Ithaca, NY: Cornell Lab of Ornithology.

Cornell Lab of Ornithology. (n.d.). *About Us*. Retrieved September 11, 2017, from Cornell Lab of Ornithology: <http://www.birds.cornell.edu/page.aspx?pid=1609>

Crossley, R. (2011). *The Crossley ID Guide*. Princeton University Press.

Ellis, S. R. (1994, January). *What Are Virtual Environments?*, 2.

Flavell, L. (2010). *Beginning Blender*.

Goldstone, W. (2011). *Unity 3.x Game Development Essentials*.

- Govil-Pai, S., & Pai, R. (1998). *Learning Computer Graphics*.
- Greenberg, S., & Kingsbury, B. E. (1997). The Modulation Spectrogram: In Pursuit of an Invariant Representation of Speech. *ICASSP*, 3, pp. 1647-1650. Munich.
- Hocking, J. (2015). *Unity in Action*.
- Kekre, H. B., Kulkarni, V., Gaikar, P., & Gupta, N. (2012). Speaker Identification using Spectrograms of Varying Frame Sizes. *International Journal of Computer Applications*, 50(20), 27-33.
- Mazuryk, T., & Gervautz, M. (1996). Virtual Reality: History, Applications, Technology and Future. 19-20. Vienna, Austria.
- Oculus. (n.d.). Documentation.
- Pacific Northwest Seismic Network. (n.d.). Retrieved from <https://pnsn.org/spectrograms/what-is-a-spectrogram>
- Paquette, A. (2008). *Computer Graphics for Artists*.
- Pruett, C. (2015, May 12). *Squeezing Performance out of your Unity Gear VR Game*. Retrieved November 2017, from Oculus: <https://developer.oculus.com/blog/squeezing-performance-out-of-your-unity-gear-vr-game/>
- Pspotka, J. (1995). Immersive training systems: Virtual reality and education and training. *Instructional Science*, 405-431.
- Rubin, J. (2016, March 30). The quest to solve VR's biggest problem: walking around. (W. Fenlon, Interviewer)
- Satava, R. M. (1993, May). Virtual Reality Surgical Simulator. *Surgical Endoscopy*, 7, 203-205.
- Satava, R. M., & Jones, S. B. (1998). Current and Future Applications of Virtual Reality for Medicine. *PROCEEDINGS OF THE IEEE*, VOL. 86, NO. 3, pp. 484-489.
- Sharaki, O. (10. July 2017). Nation Feathers VR. *Praxismodul schriftliche Arbeit*. Mittweida, Saxony, Germany: Mittweida University of Applied Sciences.

Székely, G., & Satava, R. M. (1999). Virtual reality in medicine. *The BMJ*.

Unity Technologies. (n.d.). *Documentation*.

Waldby, C. (1997). The body and the digital archive: the Visible Human Project and the computerization of medicine. *Vol 1(2)*, 227–243.

Installation Instructions

The fact that Nation Feathers VR is an unpublished game makes installing it a little more challenging than simply downloading it from the Oculus Store. The steps for installing the game on your target mobile device are described below. These steps are intended for windows users and are valid as of the date of this writing. For the most up to date information and for more details please check the [Oculus documentation](#)²⁷. Before you proceed, make sure to consult [Samsung's website](#)²⁸ for information on compatible devices.

1. Download the [Android SDK](#)²⁹.
2. A signature must be generated for every mobile device that is to use the app. Information on how to do so can be found [here](#)³⁰.
3. After generating a signature, unzip the project and copy the signature file to *Assets/Plugins/Android/assets/* in the project's folder.
4. Make sure the mobile device is attached to the computer then open the project in Unity and go to *File > Build and run* to transfer and install the application on the device.

Once installed, start the application. You will then be prompted to insert the device into the headset.

²⁷ <https://developer.oculus.com/documentation/unity/latest/concepts/unity-mobileprep/>

²⁸ <http://www.samsung.com/global/galaxy/gear-vr/>

²⁹ <https://developer.oculus.com/documentation/mobilesdk/latest/concepts/mobile-studio-setup-android-win/#mobile-studio-install-standalone-win>

³⁰ <https://dashboard.oculus.com/tools/osig-generator/>

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

I am aware that the thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future theses submitted. Further rights of reproduction and usage, however, are not granted here.

This paper was not previously presented to another examination board and has not been published.

Mittweida, December 21, 2017

Omar Sharaki