Technological Educational Institute of Crete

Department of Informatics Engineering

Bachelor Thesis

Edutainment: Ανάπτυξη παιχνιδιού με έμμεσο σκοπό την εκπαίδευση.
Edutainment: Development of a video with an indirect goal of education.

Vogiatzakis Sokratis (A.M. 4412)

Supervising Professor: Dr. Vidakis Nikolaos

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Vidakis Nikolaos, for his guidance, help, and patience during the conception and creation of my thesis.

Secondly, I would like to express my gratitude towards the people of NiLE Lab who aided me in both learning about and how to use some of the technologies used and helping me by giving me ideas about the game.

Lastly, I would like to thank all my friends (and my mother) who helped me test the game and provided me with very helpful insights on how to improve it.

Abstract

In the past two decades, video games have been a major source of entertainment for people of all ages and have been especially attractive to children. Trying to capitalize on the engagement the children have with this rather new medium, people have attempted to add some educational value to them, in varying amounts, thus creating the term **Edutainment**.

This thesis is an implementation of this concept, wherein the goal is to make a video game that appeals to young children and attempts to present them with information about a subject (geology) while also including some brain puzzles.

The game consists of a world that's split into different areas, each with its own unique activity that the player must complete, and through these activities either exercise their thinking skills or learn something new. All of the models, activities, and mechanics in the game have been chosen and adjusted for young children (colorful, easy, slow-paced).

Σύνοψη

Τις τελευταίες δύο δεκαετίες, τα βιντεοπαιχνίδια έχουν γίνει ένα μεγάλο μέρος της διασκέδασης για ανθρώπους κάθε ηλικίας, και συγκεκριμένα είναι ιδιαίτερα ελκυστικά στα παιδιά. Στην προσπάθεια να αξιοποιήσουν την αφοσίωση αυτή που δείχνουν τα παιδιά σε αυτό το σχετικά νέο μέσο, ορισμένοι άνθρωποι προσπάθησαν να εμπλουτίσουν τα παιχνίδια με εκπαιδευτικό χαρακτήρα, σε ποικίλες ποσότητες, και έτσι δημιουργήθηκε ο όρος "**Edutainment".**

Αυτή η πτυχιακή έχει ως σκοπό να υλοποιήσει αυτήν την ιδέα του Edutainment, στην οποία ο στόχος είναι η δημιουργία ενός βιντεοπαιχνιδιού το οποίο θα αρέσει στα παιδιά μικρής ηλικίας και θα προσπαθήσει να τους παρουσιάσει πληροφορίες γύρω από ένα θέμα (γεωλογία), ενώ παράλληλα θα υπάρχουν και νοητικοί γρίφοι και δοκιμασίες που θα πρέπει να περάσουν.

Το παιχνίδι συμπεριλαμβάνει έναν κόσμο, χωρισμένο σε διαφορετικές ζώνες, κάθε μια από την οποία θα έχει μια μοναδική δραστηριότητα την οποία ο παίκτης πρέπει να ολοκληρώσει, και μέσω αυτών των δραστηριοτήτων τα παιδιά είτε θα ακονίζουν τις νοητικές τους ικανότητες είτε θα μαθαίνουν κάτι νέο. Όλα τα μοντέλα, δραστηριότητες και μηχανισμοί του παιχνιδιού έχουν επιλεγεί και προσαρμοστεί για να είναι προσβάσιμα στα μικρά παιδιά.

Table of Contents

List of Figures

List of Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
FPS	Frames per Second
NPC	Non-Player Character
FOV	Field of View
CPU	Central Processing Unit
LOD	Level of Detail
HUD	Heads Up Display
JSON	JavaScript Object Notation
WASD	Keyboard buttons, W, A, S, D
V-Sync	Vertical Synchronization

List of Tables

1 Introduction

It's no secret that today's children are drawn to video games and spending a good portion of their free time playing them. Even though this form of entertainment seems fairly recent, in reality the first video game was created back in 1950 [1] but the purpose of it was not what it is today. Even as the years went by, and the popularity and power of computers grew, video games were still a niche form of entertainment for very few people, namely computer experts. It was 20 years later when games started becoming a more mainstream activity with the introduction of consoles, and since then, their popularity has been steadily increasing. This increase of the so called "gamers" didn't stay with adults though, as consoles evolved into handheld consoles and later with the introduction of 3rd-party applications on mobile phones, more and more children were able to access, play and enjoy video games. As of today, children are the second largest group of people (by age) who play video games (in the US) [2].

Seeing the potential of the video game attraction, some developers dating back to 1983 [3] started developing games that would help people learn and get education through them. This genre of video games was called **Edutainment**. Edutainment is a derived word that states a mixture of entertainment and education or marriage of education with entertainment [4]. Edutainment is applied in order to teach learners how they should use their own knowledge, analyzing things that they learn, combining things that they perceive or evaluating things that they learn [5]. Later down the path of this genre of video games, there have been some types of games that are specifically designed for the sole purpose of education. These are called **Serious Games**.

Since children are mostly interested in having fun and playing video games for them is all about that, the aim of this thesis is to create a simple and fun video game that is amusing to play as a child, and to include some minor educational elements as goals for them to achieve in the game in order to progress through it and complete it. These educational elements include both the standard "read through information, get asked a question on said information", and some alternative methods of education, like mini brain-puzzle games, through which the player exercises both their cognitive skills and their knowledge. As a bonus to that goal, and since there is no defined or de facto way of adding educational value to any game, there are going to be variations of the game (easily adjustable through the game's settings), which will allow the inclusion of such educational elements be toned down in order to survey and draw easier conclusions as to at which point children are "getting tired" of the educational aspect of the game.

1.1 Main Goal

The main goal of this thesis is to create an interactive, fun, attractive and easy to play video game, designed for children of small age. The game will include activities to help train some of the players' cognitive skills, as well as implement other mini-games nested in the main game, which should allow the player to gain new knowledge. There will also be the standard ways of presenting information and the standardized multiple answer question to test their gained knowledge, but these will be used only in the highest "difficulty" of the game.

The main tool used for this is the Unity3D engine, because it's a free to use and easy graphics engine while also allowing access to their Asset Store [6] which includes all sorts of free and paid assets, some of which were crucial in the creation of the game. Most of the game models used were imported through the Asset Store, the models and animations from Mixamo [7] and all the functionality (scripting) of the game done by me.

Ultimately, this game should be like an implementation of an idea that can be expanded upon to create more, bigger, and richer games, each with unique educational elements but under the same concepts of game design where a user can increase/decrease the level of educational elements present in the game, like a common difficulty slider.

1.2 Motivation

Children spend a huge portion of their free time playing video games. These video games don't have to be solely for pure entertainment, they could be a means to teach them and allow their skills to improve without them feeling like they are being "forced" to go to school and study.

Video game development used to be a very difficult task which required knowledge of many fields in order to create something enjoyable, but technology has made vast improvements to the ease of development, creation of helpful tools and access to other people's work, so much so, that a person with the bare minimum of programming knowledge can create a video game. So, with this thesis, a personal goal was to see how easy it was to build a game. The end result is a video game that has simple controls, easy to understand in-game goals and looks appealing to children.

1.3 Background Work

Edutainment is a mixture of two distinct and rarely combined activities for humans, education and entertainment. Although there have been several research directions, but most of them can be split into two categories: 1. Using elements of entertainment for education, and 2. Integrating educational elements into entertainment. The main goal of combining these two different activities is to "improve the educational process by incorporating elements of entertainment into it" [8].

Even though most people would think that the idea of Edutainment is recent, they would only be correct about the term. The term edutainment was first used by Robert Heyman in 1973 [9] and has since been referred to with this or similar names. Contrariwise, the concept of edutainment, or more generally, the process of learning while playing, has been mentioned many centuries ago, dating back to Plato's works in which he "philosophized that reinforcing certain behaviors exhibited in play would reinforce those behaviors as an adult" [10], which is a philosophy that is consistent with modern psychologists, such as Jean Piaget, who suggests "play is the work of children" [11].

If we take into consideration that the idea of edutainment is not an immediate result of video games, and thus think of all the other mediums that offer a similar experience, we can conclude that there is a part of edutainment that is adaptive. As mentioned previously, edutainment can be regarded as two distinct activities that have joined forces, the entertainment and the education. The education part is almost constant, with which we mean the main methods of presenting or giving education has not changed much in the history of humanity, but entertainment on the hand has had major differentiations with most of them being recent.

These recent differentiations come in the form of media, with the first big change coming with the television, and the most recent one with video games.

1.4 State of the Art

There have been lots of attempts at creating video games with educational traits. The level of educational goals in these games range from scarcely present to being the sole purpose of the game (serious games). In this paragraph, some of the most notable games will be mentioned, given some general description about their educational traits and what makes them unique, and finally compare them to our game.

Animal Jam: Animal Jam is a free to play, online, 2D video game. It allows the player to learn about zoology by "...playing games, win trivia challenges, ..., and learn about science and nature." [12].

The game is mostly a single player experience with the player only being able to play these challenges which are tied to the educational part. In addition to this, a lot of other random educational elements appear such as random trivia facts when completing collections, as well as a famous herpetologist, Brady Barr, who is included as an in-game character who teaches and entertains players [13].

Big Brain Academy: Big Brain Academy, also known as Big Brain Academy: Wii Degree, is a 2D, single player, educational video game. Its gameplay is centered around playing mini-games from a variety of categories, such as Visualization, Computation, Memory. It features a multiplayer mode as well, in which teams of up to seven players can compete in a few special ways, but using the same mini-game style of gameplay [14].

The Cluefinders: The cluefinders is a video game series with the goal of educating children. It's a 2D, single player game, with mini-games as the main gameplay element. The game is design for children above a certain age, and thus all the mini-games include more advanced material like algebra, grammar and spelling. One of the ways the developers made sure the player had the chance to learn something while playing, is by presenting the learning material right before the puzzles they had to solve (L.A. Times, 2001).

Immune Attack: Immune attack is a free to play, single player, 3D video game. The purpose of the game is to teach high school or college students about immunology. The gameplay consists of the player assuming control of a nanobot that assists the cells in fighting bacterial and viral infections, and through that, educating the player. According to a research conducted by Dr. Melanie Stegman, the students who played this specific game performed better than their peers who did not [15].

I.M. Meen: I.M. Meen is a single player, educational video game, that's designed to teach grammar to children. Although it is an older video game, made in the DOS era, it is an excellent example of a video game that tried to incorporate the educational mini-game style activity in a completely foreign and more widely-used as a pure entertainment gameplay genre, the first-person-shooter. The game also featured a difficulty slider that allowed parents to alter both the content of the shooter gameplay, and the mini-games [16].

Quest Atlantis: Quest Atlantis was a multiplayer, 3D, educational video game that aimed to combine strong points of other commercially successful video games with educational goals, such as being able to personalize and control your virtual character in a 3D world, in

which you could navigate to certain areas to complete certain activities. The game was successful at a high enough level to gain the attention and funding of the *Bill and Melinda Gates Foundation,* which then was relaunched as Atlantis Remixed with many improvements [17].

Meister Cody: Meister Cody is online, single player, 2D video game. Its uniqueness stems from two facts. The first fact is that even though it is a simple mini-game oriented gameplay, it includes a diagnostic system which helps parents, teachers, and therapists to detect dyscalculia and help treat it. The second fact is that the graphics of the game are quite appealing to young audiences, in contrast to most other games of its type who try to aim for realistic or just acceptable graphics, thus making the game much more approachable and enjoyable by children [18] [19].

WolfQuest: Wolf quest is a 3D, single and multi-player, open world, simulation video game. The goal of the game is to teach the player about its subject's natural life and habitat, the Gray Wolf, by allowing them to play as one such wolf and try to survive. This sub-genre of video games is quite rare by itself, but its success [20] shows that there is potential in designing games to teach through the gameplay itself, in such situations, which is quite different than the mini-game or standard text-with-information model [21].

Global Conflicts: Palestine: Global Conflicts is a 3D, open world, serious video game. The content of the game is quite mature, but the concept of the game shows that educational video games don't have to be restricted to school-level education, but rather can be expanded upon to more difficult to grasp concepts such as journalism. Through its systems, it promotes critical thinking and the need to search before drawing conclusions, a very important, yet often neglected skill.

Our game has used and expanded upon some of the unique features each of these games offers, with the addition of one different feature. First of all, the game is designed as a 3D open world map with a customizable player character, just like Quest Atlantis. It helps reinforce the personal connection the player feels with their avatar. The major change here, is the attempt to use much more children-likeable graphics/animations, based on the Meister Cody game approach. Moreover, the 3D world allows for more gameplay-oriented tasks to be used as a means for education.

Secondly, an amalgamation of features combined, gives us the main differentiation of our game. Based on the common use of mini-games as an educational tool, like Animal Jam, but also wanting to combine it with more specific education subjects like Immune Attack, in our case Geology, the concept of creating different kinds of "educational levels" in a single game was created. Taking it a step further, the game is designed to not only include these levels of educational activities, but to also function as a normal game without them, effectively switching genres.

Needless to say, a lot of design choices were inspired by all these games, such as presenting the information right before the question (Cluefinders), or the use of mini-games at the end of a level (I.M. Meen), or in our case, activity.

Below is a table which shows and helps with the comparison of some of the most famous and notable games in the same genre, including this project in the final row.

[***Table 2 - Comparison of Notable Educational Games***]

	3D	2D	Mini-Games	Navigable World	Multiplayer	Variable Level of Education
Animal Jam	✗	✓	✓	✗	✓	✗
Big Brain Academy	✗	✓	✓	✗	✓	✗
The Cluefinders	✗	✓	✓	✗	✗	✗
Immune Attack	✓	✗	✗	✓	✗	✗
I.M. MEEN	✗	✓	✗	✓	✗	✗
Meister Cody	✗	✓	✓	✗	✗	✗
Quest Atlantis	✓	✗	✓	✓	✓	✗
WolfQuest	✓	✗	✗	✓	✓	✗
Global Conflicts: palestine	✓	✗	✗	✓	✗	✗
Crystal World	✓	✗	✓	✓	✗	✓

1.5 Outline

The current thesis is divided into 5 main chapters and special sections for the bibliography.

Chapter 1: Introduction

This chapter includes the introduction to the thesis, the main goal with a brief summary of what was achieved, as well as the motivation behind the choice to make it.

Chapter 2: Technologies, Tools and Assets

The technologies, tools and assets used for the creation of the video game. For each one of these aspects of the game there will be a brief explanation of what it is, why it was chosen over other possible choices as well as some possible negatives that might come from it.

Chapter 3: Combining Everything Together

How each of the aforementioned parts of the game were used to create the world and build the systems of the video game. There's also going to be an explanation of how the game's theme was created and helped with the design of the game, as well as general design philosophies that were used.

Chapter 4: The Crystal World

This chapter will showcase the final product, how everything ties together to create the video game and some in-game screenshots. There's also going to be an analysis for the parameters used in each component of the game and the reasoning behind it.

Chapter 5: Final Notes and Future Goals

The final chapter will include some notes on the game's design that were not major enough to be included previously, some potentially subsequent goals for the project and some hopes for the future of the Edutainment genre in video games.

2. Technologies, Tools and Assets

This chapter will include the main building blocks used for the creation of the video game. There will be a brief explanation of each chosen technology, tool, or asset, a reason for choosing it over other similar alternatives wherever these exist, and the reasoning behind picking certain assets over others. Wherever needed, a table will be provided with clear comparisons of the options presented.

[***Table 1 - Game Engines***] [22]

	Unity3D	Unreal Engine 4	CryEngine	Torque 3D
2D Support	✔	✘	✘	✘
3D Support	✔	✔	✔	✔
Programming Language	C#	C++	C++/Lua	C++/TorqueScript
Visual Programming	✔	✔	✘	✘
Cross Platform Support	27 platforms	16 platforms	4 platforms	4 platforms
Synchronous cross-platform development	✘	✘	✔	✘
Usage Cost	Free	Free	Subscription	Free
Publishing Cost	Free (conditional)	5% of revenue	Free	Free

2.1 Game Engine: Unity

Unity [23] is a cross-platform real-time engine developed by Unity Technologies. Unity gives users the ability to create games and interactive experiences in both 2D and 3D. The choice here was very simple. The alternatives were not many, but Unity is the most popular and for good reasons. Its user-friendly UI, paired with a huge community that helps with answering questions and providing tutorials, as well as an enormous amount of assets in its own Asset Store, all shape it up to be the best choice for beginners and a strong contender for advanced users.

The main drawback of using Unity, is that it's designed to be as generic as possible in order to appeal and be accessible to as many people as possible, and the cost that comes with it is the optimization. Many of the default settings are not performant at all, and quite a lot of digging is required to make the best use out of it for any specific scenario. This drawback only starts showing itself as the scope of the project gets bigger, which makes it much harder to figure it out later.

2.1.1 Terrain

Terrain is a tool included with unity. It offers a fast way of creating the landscape of your game. It allows you to edit the size and height of it, add, paint, and blend textures on it, quickly or even automatically place trees, grass or other detailed meshes such as rocks. And all of that without leaving the Unity editor. The alternative to this, is to create the exact shape of the landscape using another software, such as Blender, and import it as a normal object.

The choice here was the built-in terrain tool due to the simplicity, ease of use, and time it required to reach the outcome. Even though the selected solution seems the best from all aspects, the negative comes from the performance cost of using it. Unity has been heavily criticized over this feature ever since it was created, with many attempts at alleviating the problem, but it still is a major issue to consider. For this project's scope, the performance had not reached unacceptable levels, so we omitted to not replace it, but for a future goal as mentioned in the last chapter, it might be required.

2.1.2 Programming language

The programming language used for the project is C#. C# is a general-purpose, multi-paradigm programming language encompassing strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines [24]. It was developed around 2000 by Microsoft within its .NET initiative. Mono, an open-source platform based on .NET [25], has been the main scripting language used within Unity along with Boo and UnityScript. The choice here was dictated by Unity, since the support for Boo and UnityScript was terminated recently, but, nevertheless, the choice would have been C#. The reason being, C# is a widely used programming language which is an asset to know by itself. There is, and was, no noteworthy negative to this choice.

2.1.2.1 Scripting

Scripting is the process of creating a *script*, which is Unity's way of allowing the programming language to be used to add extra functionality to your project. Scripts are components, each of which can contain a piece of code which is just like creating a class in the classical object-oriented paradigm.

Components are Unity's way of implementing composition (as opposed to inheritance), by allowing said components to be *attached* to GameObjects (fundamental objects in Unity that serve as containers for components) thus allowing the creation of custom and complex Objects. Examples of these components are the Rigidbody, a component responsible for adding physics to any object, or a **Collider** which is a component that allows the collision detection and enables physics collision of objects within the game

Scripting is essential to almost any kind of game created, because it's what allows, and controls many aspects of the game, such as the user's input, or more abstract concepts like the score system a game might have, or even basic computer functionality like reading/writing computer files.

2.1.2.2 Data Storing: JSON

JavaScript Object Notation (JSON) is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). There are many alternatives to JSON, namely all the ways someone can store data in a file. The reason JSON was chosen is personal preference since the benefits are familiarity.

The use of JSON in this project is to store the available Questions/Answers the game supports as a mini-test of knowledge for the player.

2.2 Audio

Audio in a video game consists of all sounds and music that exist and can be heard at some point in it. It is an artistic tool mostly, with rare exceptions of some games that incorporate it as a gameplay tool, and it is used to make the game feel less monotonous to the player. Most common use of audio in a game is to match certain actions or events with a specific sound, for example footsteps or gunfire, but it can also be used for other purposes like an ambient sound for a forest, or some constant music that plays in the menus, or to give feedback to the player that something has happened from simple things like clicking a button, to more complex like completing an in-game task.

Audio can be stored in many formats and Unity supports a good amount of it. In addition to the format, the audio can also be further compressed using some other methods, and there is also an option on how to load/store the audio while playing the game. Most of this comes down to performance optimization, but explained simply, depending on the length and frequency of your sound one should choose specific combinations of formats, compression algorithms and load methods.

2.3 Camera

Camera is the tool through which the player "sees" the world. More specifically, it's the tool responsible for telling the game engine what to render in the player's screen. In Unity's case, it's a component that can be attached to a GameObject and it's responsible for a few things, including the distance after or at, which the rendering of the world starts or stops, respectively, the player's FOV, as well as some other technical aspects regarding post processing and lighting.

2.3.1 Cinemachine

As in all software development, one of the most important ways to develop faster is to reuse pre-existing, tested, and fully functional code in your current work, so you don't have to spend time rewriting the same thing. Such is the case with Game Development, and one of the best examples of this is the built-in **Cinemachine** tool Unity offers.

Cinemachine is a camera system made by Unity that can provide a ready solution for most common camera requirements in games, as well as allowing a developer to create custom cameras with it. These custom cameras include the ability to use the *Timeline* feature of Unity to create in-game *Cutscenes* or *Cinematics*. Between this and the writing the custom code from scratch, the choice was obvious. The benefits are much less time spent on creating and debugging the camera controls, as well as allowing the use of a tool that's going to get future updates by someone other than the developer, in this case me.

2.4 Third Party Assets

Assets in Unity are any type of file that comes from an outside source. Whether it's something like a simple 3D model, or something more complicated like an entire framework of creating huge outdoor areas, these assets are an essential part of creating a video game. Unity itself provides an *Asset Store* where people can upload their assets for free or for a price.

This store is an extremely important part for independent developers who can't afford or don't know how to create some artistic assets, for example, so they are given the chance to get them and import them easily to their projects. Such is the case for this project, with most, if not all, of the assets used, come from an external source, mainly Unity's own Asset Store, but also some other places.

2.4.1 Models and Animation

The models in Unity are simple objects that are consisted of two main components, the mesh filter and the mesh renderer. The filter is where the mesh of the object is stored, while the mesh renderer is what makes it visible in the 3D world. In addition to that, there's also the skinned mesh renderer component, which is used instead of the normal mesh renderer when the imported mesh has animation which can bend the mesh's shape.

Each mesh consists of vertices that define its shape. The number of these vertices in a mesh usually define its level of detail, but the higher that number is, the higher the computational cost for rendering it gets as well. Even though modern graphics cards are capable

of rendering millions of vertices with ease, it's always preferred if the number stays as low as possible.

Animations in unity can be considered either the simple transform (position, rotation, scale) changing animations, or complicated *bone* animations. Technically, both can be done within the unity editor, but only the simple ones are a feasible choice. The more complex ones are usually done in other software and then imported as an asset.

When implementing character/animal animations, there are two ways of doing it. One is using the custom bone animation to dictate how the mesh is animated and then applying physics or non-physics transformational movement to that object using code, while the second one, called root motion, is allowing the movement of the object to be dictated by the animation without any transform altering code.

The first option enables a far more editable and "free" movement, for example the running speed of the player can be adjusted by changing a variable value. The downside to this, is the usually unrealistic combination of animation and movement because they rarely sync correctly. The second option allows the animator to *bake* the movement each animation does into the animation, allowing for a far more realistic and lifelike movement. The drawback, of course, the very difficult/limited customization of the movement parameters.
For this project, since the movement of the character is not supposed to be fast, the player's movement is implemented using root motion, and any other movement is implemented with physics forces applied through code.

2.4.1.1 Mixamo

The website were the character models and animations were obtained from was *Mixamo* [7], a website by *Adobe*. Mixamo is a website that hosts a wide range of models and an even wider range of cutting edge, motion captured animation that anyone can choose from and download.

All of the animation works with all the models available and a preview can be seen with the selected model, while also having some tweakable parameters, like choosing the animation speed, the animation frames or even the frame rate of the animations. In addition to that, a custom model can be uploaded and rigged through their website, in order to make it compatible with their animations.

2.4.1.2 Other in-game models

All of the other models in the game, like the trees, the buildings, the boat, etc have all been imported through Unity's Asset Store.

Since most of these were made by different artists, they all have a different style, color palette, but efforts have been made to make them all look as homogenous as possible through post processing effects, such as color grading.

Furthermore, the choice of models was made with the end user and optimization in mind, so realism, for example, was placed further down the stack of priorities and in contrast things like cartoonish looks and bright colors were placed at the top.

2.4.2 Audio Assets

The audio assets that have been used in this project come from external sources. The footstep sounds come from Unity's Asset Store, while the menu/UI along with the ambient sounds come from *Sonniss* [26] with license-free assets.

3. Game Implementation

This chapter focuses on how the game was implemented. Section 3.1 describes the scenario upon which our game has been based. Section 3.2 describes the important parts of the game's implementation.

3.1 Game Design: The Game's Use Case Scenario

When the game starts, the player chooses the educational level of the game and presses Play. The educational level dictates the frequency of mini-games/questions in the game. After the game starts, the player finds themselves in a village with an NPC character calling them (using a waving animation and having an exclamation mark above their head). When the player interacts with the NPC, a dialogue appears through which the player is informed about the objective of the game, which is to collect the crystals. Looking around, the player will notice some crystal fragments. From this moment on, the player can make a choice of where to go. They can either try and find all the crystals in the village or go to the ship. The ship allows them to transport to other areas, namely the Maze and the Forest. Going to the maze, the player has to navigate it twice, once to collect the crystal and once to return to the ship. Going to the Forest, they see the crystal in the middle of the forest with an animal (Rhino) walking around it. If the player tries to go near the crystal, the Rhino attacks them and pushes them back, denying them access to the crystal. In order to bypass the Rhino, the player must collect the crystals in the village, talk to the NPC again, and then pick up the food that's near the NPC in order to use it as bait to distract the animal and then collect the crystal. After the player has collected all the crystals, the camera moves to the village area of the game, where the player is congratulated by the NPC and can see their character along with the NPC's model to dance to a famous song.

3.2 Game Development

This section describes how each aforementioned technology, tool, and asset was used to construct the game. For every facet of the game there will be an explanation of how and why it was designed that way.

3.2.1 Creating the player character

Creating the player character in a game means making a choice in both how the character looks, as well as how the player controls them. In Unity terms, this means choosing a model (mesh) to depict your character, attaching an animator controller, and finally creating a script to translate player input to character movement. The way objects are setup in Unity is by creating a *Game Object* and then attaching all the required components to make it function as you want. In the player's case, the required components are a Rigidbody, for applying physics to the 3D object, the Collider which dictates the space the objects occupies in the world in order to calculate collisions, an Audio Source for all the various sound effects that are going to come

out of the player's model, and finally all the custom scripts like the movement, climbing, footstep sound and others.

3.2.1.1 Choosing the camera

The first choice in regard to the player character is where the camera is going to be placed. There are two main choices with this, the first person and the third person camera. The first-person camera is when the virtual camera is placed right in front of the player model, if there is one, and it's supposed to present the game through the eyes of the character. The third person camera is a camera that's floating usually above shoulder height of the player's model and is capable rotating freely in all directions around the it.

Even though each style is mostly a preference, for this project we decided to go with a third person camera because we think it's important that the player sees and connects with their avatar, and in order to do that through a first-person perspective, a lot of animation and sound work must be done as a collaboration by artists/programmers. Plus, having a third person perspective allows for a future implementation of custom character creation which makes it an even more personal experience.

3.2.1.2 Using the model

A generally good practice in video game development is to separate the visual part of an object from the functional part of it. This allows the developer to later replace the graphics of it, or the colliders, without having to recreate the entire object with its components and values in the new object.

The model is the main visual part of the character. The technique used here is to create a hierarchy of objects, with the parent object being where the player's functionality (components/scripts containing logic), and as a child to this object, another object that contains the *mesh renderer/mesh filter* and any other possible visual aspects of the model.

3.2.1.3 Adding the animations

To add animations to a model in Unity, you need to attach an Animator Component to it, and then use an Animator Controller with the desired animations created into it. To create the Animator Controller in Unity, you drag and drop the animations in the special window it has, create the transitions between the animations, and finally create special parameters (floats, bools, triggers) and match add them to the animation transitions as transition conditions.

The animations that the characters would use had to be as natural as possible since most modern games tend to have life-like characters with very human animations and most children who have already played video games would be accustomed to such quality. Also, since the decision to use root motion was taken, the animation itself had to include it as well as facilitate all the range of movement the game had to support.

In the current state of the game, the movement range is not very diverse, meaning only basic running in all directions is essential to playing the game, but there are some flavor animations included that give the player some more gameplay options to help increase the engagement while playing. These animations include jumping, falling from a ledge, waving

when interacting with an NPC, and climbing. Climbing, in particular, is just a fun mechanic to make the navigation more fun and interesting.

3.2.1.4 Scripting the movement

Scripting the movement means creating a set of controls for the character and, namely the keyboard/mouse or any other possible input for the game, and then mapping them to specific actions in the game.

To give an example, in order to move the player forward we need a keyboard input, *W*. We write in script to check for the keyboard button *W* and when it's pressed, it calls a function that does something to move the character. In this project, since the motion is created by the animation itself, we have mapped the forward running animation to a float parameter so when the *W* button is pressed, the parameter becomes one (1). Using this technique, we have mapped every single possible input of the player to the corresponding change in character movement, namely the WASD buttons as movement and the *Space* button as the jumping animation.

Since the game currently supports two modes of movement (walking, climbing) and in the future it could support more (swimming, flying), we have split the movement logic in different scripts in order to make it as modular as possible and decrease the complexity when it comes to logic about choosing which animations to use. In the current iteration, there is a script for Movement and a script for Climbing, each of which is activated when certain, well defined conditions are met, such as touching a custom *Climbing object*.

3.2.2 Creating the world

Designing and building the game world is a very complex process. As far as game development goes, it's a whole different category of its own. It includes creating the place on which the player walks, the objects that exist in the world like sea, trees, buildings, etc., as well as more subtle but meaningful details like texture blending, to make the whole scenery more believable.

3.2.2.1 Building the terrain

By using Unity's terrain system, it's very easy to get started. After creating the terrain object in Unity but before doing anything else, since we have acquired heightmap data from an external tool, terrain.party [27], we need to make sure we know the dimensions of the heightmap data. In our case, we've picked the island of Santorini, which was roughly 17 km² in the website. In order to depict it with a 1-1 scale, we can use the same resolution in the corresponding settings of the terrain, but that will be an enormous amount of data most of which won't be used and will be consuming computer resources, so we have scaled it down to something less than half the actual size. Then, using a script that maps the data of a heightmap to a number and uses it to automatically increase the terrain's height.

The next part is texturing the terrain. In the terrain's settings, you can add the textures (images) you want to use in the terrain and then use some built-in or custom tools/brushes to paint the terrain. Since this is a very long and tedious work, one very common technique is to create a script that automatically paints the terrain using the textures you've used, based on specific parameters. Such a script has been used in this project, with the parameters being the height of the terrain, e.g. 0 → 0.33: use sand texture, 0.34 → 0.66: use grass texture and

0.67 → 1: use rocky/mountainous texture, and the second parameter being the slope, e.g. wherever the slope is > 0.7 use another rocky texture.

3.2.2.2 Creating the playable zones

After creating the terrain, the next step is to design and populate the playable areas of the game. The game is not designed to be an open world, meaning the player cannot freely roam around the whole map. Instead, there are zones that the player can visit, and these zones are surrounded by invisible walls to block the player from walking away from them.

In each zone there are trees/grass/bushes that have been added using one of tools included in terrain. This tool allows the developer to quickly "paint" these, as well as other detailed meshes (e.g. rocks) with a few random variations, should they want to. The nice thing about this tool is that it allows the objects to use their LODs to adjust their performance cost on specific distances from the camera, while also being affected by the *wind zone* component that Unity provides.

Apart from that, each zone has its own unique models, like rocks, buildings and the maze, that are part of the zone. These objects must be placed by hand by being dragged and dropped on specific locations in the world.

3.2.3 Adding the sounds

There are four types of audio used in the game, the audio that's playing constantly as a background ambient sound, the UI button hover/click/feedback sounds, the sounds that come out of the player character, and the NPC's voice.

The background audio has been implemented by adding an *Audio Source* component on the *GameManager* object in the world. There are two kinds of audio. The first one is the ambient sound of the world. The sound of this audio is 2D, meaning it is not coming from a specific place or object in the world. The selected background audio has been placed in this component, its volume has been lowered so as to not be distracting and it's been marked as looping, which means it will repeat itself when it ends. The second kind of audio is the sound of the sea. This component is an object which moves as the player moves but only stays up to the shore, and is also a 3D sound, meaning the closer the player is, the louder the sound appears. This implementation simulates real-life and is used for more realism, thus immersion for the player.

The audio of the UI is being controlled by a custom script. This script is responsible for checking the OnMouseOver events of whichever object it is attached to and plays a specific sound that lets the player know they have hovered over an interactable object. The other feedback sounds that come from the interaction with the UI, for example when the player clicks on a letter of the virtual keyboard in *The Hangman* mini-game, are controlled by their corresponding scripts that have their logic. When the function that determines if the input was correct runs, it references the *Audio Component* of the *Game Manager* and plays the sound.

The in-game audio that comes from the player's actions, like character sounds when jumping, footsteps of running, picking up collectibles (Crystals), etc., are all bound in their specific interactions. The pickup of an object, for example, is played when a physics collision

event between the player and the crystal is detected, while the character sounds are bound to the animations themselves through Unity.

The NPC's voice is a custom recording of a person reading the text. It's useful as a way for children who can't read quickly before the text disappears, which is especially important during the first fly-over overview the player gets when they first talk to the NPC as they might miss something by trying to focus on reading. It also adds another layer of immersion to the game.

3.2.4 Making the user interface

The user interface in video games is considered any (usually) 2D image or text that's displayed over the 3D world. It's used as a means to help player keep track of all sorts of things, like the objectives they have completed, their score count, their remaining lives, or access menus and settings, depending on the game. In our game, most UI components have a special custom script attached to them, that handles the Mouseover events to increase the scale, add shadow and play a sound in order to give feedback to the player about the currently "active" object.

The following paragraphs describe the "normal" UI, meaning everything that is not a part of the educational side of the game, and the "mini-game" UI that includes the educational parts.

3.2.4.1 Normal user interface

There are a few distinct types of normal UI in the game, the HUD, the pop-up menus and the normal menus. The HUD is all the UI elements that are constantly present in the player's screen, like which crystals are collect, or the controls of the game. The pop-up menus are the dialogues that appear in certain parts of the game to give player some extra information or guidance. The menus include the main menu of the game, as well as the ESC menu with the settings menu.

The HUD in this game is very minimal for two reasons, first one is there are not many things going on that require constant tracking by the player, second one is the audience for the game is children, who mostly have a harder time grasping the concept of HUD and reading the information it provides, thus ignoring it. As such, the HUD only includes an icon of the crystals that have been obtained by the player and is currently placed at the top left quadrant of the screen.

The pop-up menus are used as a way for the player to communicate with the NPC of the village, who is designed to give the players information, like the objective of the game, or assistance when needed, like when they attempt something that didn't work and offer a suggestion. It is essentially a way for developers to communicate with the player without breaking the immersion of the game.

The main menu of the game consists of the four interactable main elements, in addition to the credits/references at the bottom. The Play, Exit, Change Model buttons and the educational slider are the main elements, which concern the game itself, while the bottom part of the screen contains images/texts with links to corresponding websites. The main elements are what Unity calls *World Canvas,* which means they are positioned inside the 3D world, as a 2D element. The rest are normal 2D UI.

An in-game menu is the commonly used menu to allow the player to pause the game and it usually contains the settings that allow them to adjust some technical aspects of the game. To bring it up the player must use the commonly used ESC button. Then, they are presented with three options, the Resume, Settings, Main Menu. Resume does the same as pressing the ESC key again, resuming the game. The settings open up another menu where the adjustment toggles/sliders are nested. These include a FOV, View Distance Pixel Error and Mouse Sensitivity sliders, as well as toggles for Motion Blur and V-Sync. The Main Menu button returns the player to the main menu.

3.2.4.2 Mini game user interface

Each mini-game's user interface is different in terms of design, but they all share a couple of settings, namely the change of colors (mainly to allow for more preferences) and the toggle to mute the feedback sound of their UI. They also share the design philosophy of the game that its audience is children of small age.

The Pair Matching mini game consists of a grid which requires the cell number to always be a multiple of two (2), and for the sake of aesthetics the decision to make it a grid of even dimensions was made. Every time the game is started the grid is randomized created and randomized by code. Whenever a player clicks on a cell, the cell stays open by revealing the image that is hiding and waits for a second cell click. If the second image is the same as the first one (**figure 10**), both cells get a green background and a correct sound.

The Short Memory mini game is a grid as well, but without the restriction of having a multiple of a number in cells. Even though the game can be infinitely scaled in difficulty by increasing the number of cells required to be remembered, in the current iteration the number of cells stays constant as a 4x4 grid, with the number of required cells required are only four (4).

The Hangman mini game is completely different from the rest. It requires a mini-database of words, which is currently stored in the script, and whenever it starts a random word is chosen and its length used to create the corresponding amount of underscore (_) characters. These characters, as well as a virtual keyboard are presented to the player. Some custom behavior was required for this mini-game. The first one is the creation of a virtual keyboard, currently using the English alphabet but can be easily changed, which keys have been binded to the physical keyboard's input for a better accessibility. The second is the auto-scaling of the UI based on the length of the word, since all words are available, and their lengths are varied. As far as playing the game is concerned, it functions as a normal hangman game. Whenever the player clicks or presses a letter it is marked with a red or green background, accompanied by the corresponding sound, to give the player the feedback of their actions. The game is won when the whole word is found before the player runs out of available tries.

All of these mini-games were created from scratch and can work as standalone or be implemented in any other game.

3.2.5 Implementing the game mechanics

The game mechanics are what the player does in the game, the main logic of the game's flow. For this game, it's collecting the crystals. Of course, a lot of other intermediate steps have to

be made in order to reach that goal, and these steps also have to be programmed or created in some way. The steps include the creation of areas and activities in them as well as the appearance of the educational part.

3.2.5.1 Creating the main progression of the player

A main design philosophy for the game was to create the world in such a way that it promotes some simple exploration, but also respects the fact that its main audience will be children who might have difficulties progressing through the game. As such, the progression was designed to be linear enough to help with guidance, but not a one-way road to discourage exploration.

This was achieved in a couple of ways. First of all, there is no clear "*Do Next"* indicator or UI the shows objectives in the game, there is only a progress tracker which shows which crystals are already collected and, once at the start of the game when the player first interacts with the npc, they are shown a fly-over overview of each in-game activity area with its crystal highlighted. Secondly, the level design is created without hints as to where the next objective is. This is implemented with the navigation system between the different activity areas of the game, the ship, which is available to the player from the first moment and allows for traveling to any area.

3.2.5.2 Creating each zone's challenge

When creating the zones, the were two things that had to be considered. The first was how to create a visually distinct area that fits with the rest of the game world. The second was how to implement a challenge, or activity, in this area that was easy for the children to understand and complete.

The game currently consists of three main areas, each with their boundaries (*invisible walls*). The first one is *The Village* and it's where the player starts the game. The village consists of a few buildings, a campfire, and a couple of NPCs. It's meant to be a familiar experience for any player as an area. The activity here is to help the old man find and collect the broken pieces of a crystal around the village. This serves as the easiest challenge in the game and one that will be used in the other crystals as well if the higher *Educational Setting* is selected.

Another area is the maze. As soon as the player arrives they are greeted with an uphill road and big stone walls, indicating something is behind them, only to be revealed that the walls are a maze. The activity here, of course, is the navigation of a maze. At the end of the maze, there's a crystal, and after they collect it (or the pieces) up, they have to go through the maze, but in reverse this time.

The last area the player can visit is *The Wild Animal* area. It's a mini-forest with thick bushes that obstruct the vision from the distance, in order for the player to not know what's hiding behind them. When the player goes through them, they can see the crystal as well as an animal (a magical Rhino). The rhino proceeds with attacking them and sending them back to the ship. This challenges the player to think of some way to bypass the rhino. The challenge here is that after the player completes the first trial, the campfire now has some meat on it that the player can pick up and set as a gift to the rhino. When they do, they are free to pick up the crystal.

3.2.5.3 Deciding when and how to present the educational mini games

One of the goals in this thesis was to find a way to implement educational traits to an otherwise normal video game, with an adjustable way. The solution to this came with the form of an "**edutainment**" slider, a setting in the game's main menu, which will dictate how much the educational part will be "interfering" with the gameplay.

3.2.5.3.1 Presenting the Questions

The questions are going to appear in the second level of "**educational value**" setting. They are essentially the standard mini-quiz a student can complete in order to be tested on gained knowledge.

Essentially, if this level is chosen, the crystals will appear broken in pieces, in order to let the player know they have to pick them up. Whenever a piece is collected, a text of information regarding a specific question will be shown to the player, and after all the pieces have been collected, the player will have to answer the multiple-choice question in order to combine the pieces. Otherwise the pieces will break, and the process is repeated.

3.2.5.3.2 Presenting the Mini Games

Each mini-game will be chosen randomly to appear after each crystal is collected on the second level of the "educational value" setting. These will be short, with a timer of thirty seconds to one minute, and with a scoring system that will determine if the player performed well. If they did, the crystal will be collected, if not, the player will have to replay the mini-game.

4. The “Crystal World” Game

This chapter contains screenshots of in-game areas, models, systems and menus. For each one of these there will be a short description of what it depicts, and what their purpose is.

4.1 User Interface

This part showcases the different types of user interface currently in the game. 4.1.1 shows the main menu, 4.1.2 shows the settings menu, 4.1.3 shows the pop-up dialogue UI.

4.1.1 Main Menu

The main menu is the first screen the player sees. The available options are Play, the Educational Slider, Change Model, Exit.
Play changes the scene and allows the player to play the game.
Educational Slider allows the player to adjust the educational settings of the game. Currently there are three levels, 0/1/2, as can be seen by the window next to the menu.
Change model allows the player to preview and choose their preferred character.
Exit closes the application.

Figure 1 - Main Menu

4.1.2 Settings Menu

The settings menu consists of the Pixel Error, View Distance, Field of View and Sensitivity sliders, as well as the toggles for Motion Blur and V-Sync.
Pixel error slightly changes the terrain, which helps improve performance.
View Distance changes how far it sees, or *renders*, objects. The lower objects, the better the performance.

Field of View adjusts the camera's field of view. It's essentially like moving the camera closer of further from the character.
Sensitivity adjusts how sensitive is the mouse input in the game.
V-Sync locks the Frames-Per-Second to a standard 60 FPS. Otherwise it's uncapped (unlimited).
Motion blur changes how blurry the game looks when the player turns their camera.

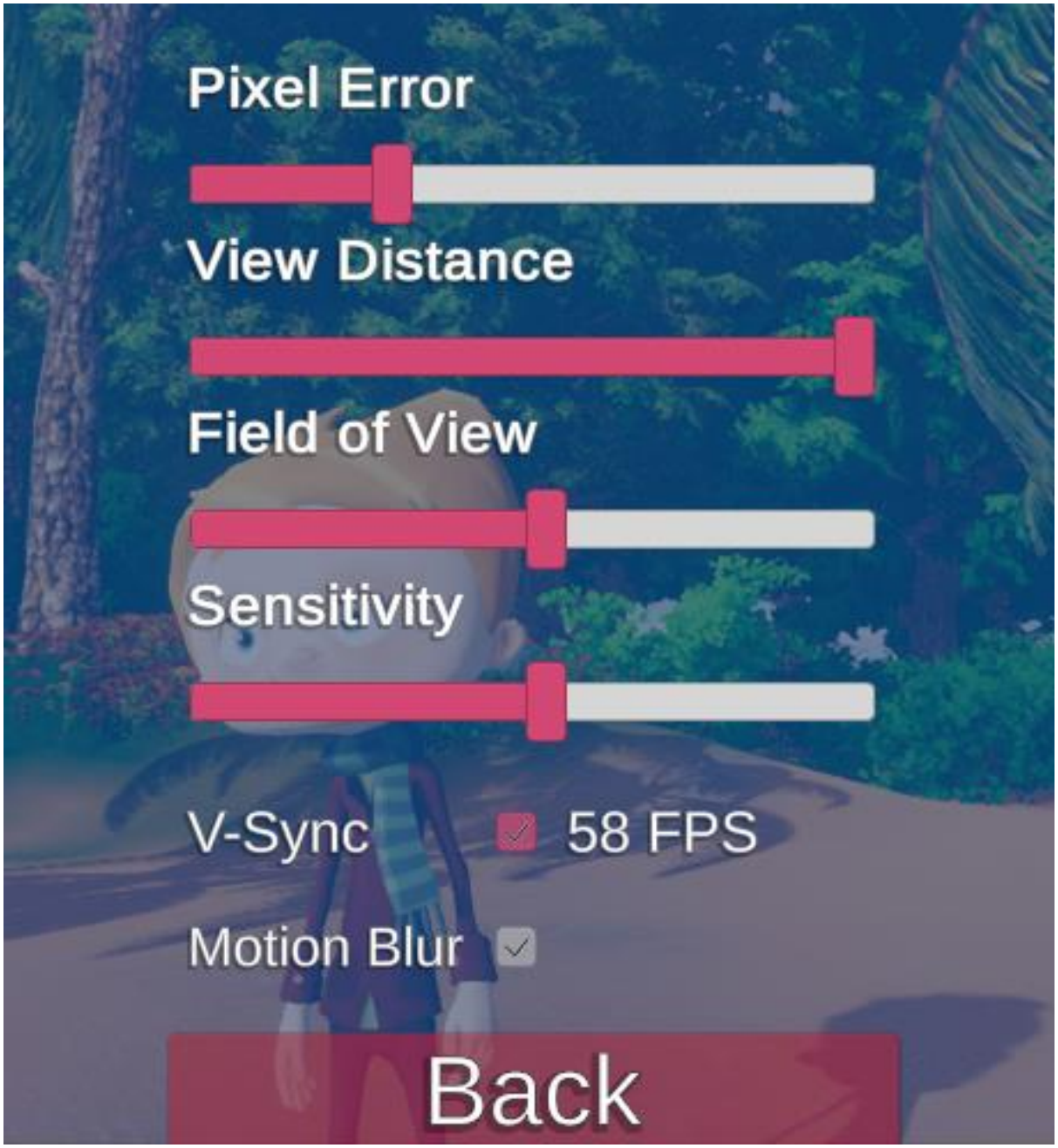

Figure 2 - Settings Menu

4.1.3 Pop-up Menu

Two examples of pop-up menus. The **3a** shows the NPC guiding the player when they first talk, while the **3b** is a tip that appears when the player encounters a problem while they are playing to help them overcome it.

	
3a	**3b**

Figure 3 - Pop-up Menus

4.2 In-game activity areas

This part will showcase all the available areas the player can visit while playing the game.

4.2.1 Travelling between the areas

The ship is what the player will use to travel between areas. In each area, there's going to be a small bridge that leads to the ship. When the player approaches it, the menu appears and allows them to choose which area they want to visit. It also excludes the one they are in at the moment.

Figure 4 - Travel UI

4.2.2 The Village

The first area of the game, where the player starts. This is where the pieces of the Red Crystal are located.

Figure 5 - The Village Area

4.2.2.1 The Old Man NPC

The NPC of the village. Responsible for guiding the player. Can be interacted with by getting in close range. When first interacted with, a mini tour of the island using the camera and a text explaining the objectives, with a voice over.

Figure 6 - The Old Man NPC

4.2.2.2 The Crystal Pieces

Some of the scattered pieces of the village crystal. There are currently seven pieces, most of them hidden, which helps promote exploration to the player.

		
7a	**7b**	**7c**

Figure 7 - Crystal Pieces

4.2.3 The Wild Animal

The area of *The Wild Animal* Activity This is where the Pink Crystal is located. The area is dense with vegetation in order to promote a sense of mystery and danger, and to surprise the player with the animal that's waiting for them near the crystal.

Figure 8 - Wild Animal Area

4.2.3.1 The Animal (Rhino)

The animal next to the Pink Crystal. The rhino roams around this area and attacks the player on sight. The player gets pushed backed before they can reach the crystal. Feeding the animal puts it to sleep and allows the player to collect the crystal.

Figure 9 - The Wild Animal

4.2.4 The Maze

A view of the maze area from above. This is where the Blue Crystal is located. The player must navigate at least once the maze, which depends on whether this is the last crystal they choose to pursue.

Figure 10 - The Maze

4.3 The Questions

The questions are a method to test the player's knowledge of information. It functions just like a multiple question quiz would, meaning the player is presented with a question and some possible answers below it. The "Source" is a button which takes you to the question's source in the internet.

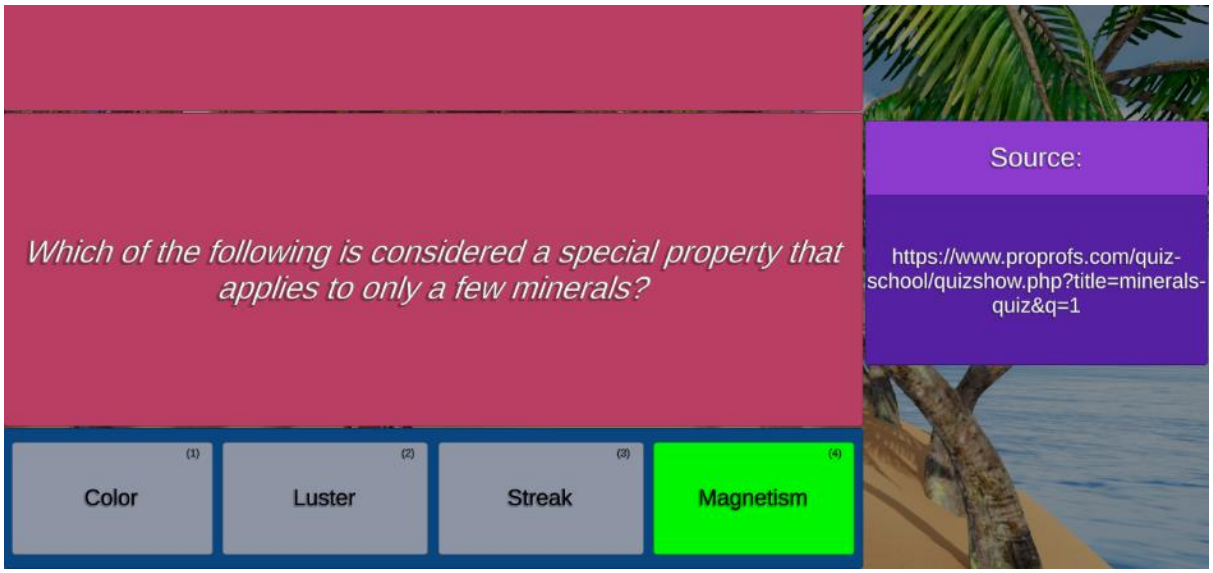

Figure 11 - The Questions

4.4 The Mini Games

These mini-games are what the player will face on the first level of extra "educational" value someone can add to the game.

4.4.1 The Pair Matching

This is a memory game. The player can click on any square with the "?" on it to reveal the image that's hidden below it. Then, when they click a second image, if the images are the same they get marked as found, else both images flip back to closed. The game ends when all the images have been matched correctly.

Figure 12 - Pair Matching

4.4.2 The Short Memory

This is different kind of memory game. The player is presented with a grid of squares and a timer counting down. When the timer ends, some of the squares turn around to reveal an image, and then flip back. The player is then tasked to remember which squares had the image and click on them. If they click on all the correct images, they win, otherwise if at any point a wrong one is chosen the game is lost.

Figure 13 - Short Memory

4.4.3 The Hangman

The hangman tests the player's vocabulary and thinking skills. The player is presented with a word but with its letters removed and buttons for the alphabet. Whenever they click on a letter, if the word contains it, it's revealed and written, otherwise it registers as a wrong letter. After a set amount of tries, or if the player reveals the whole word, the game is over.

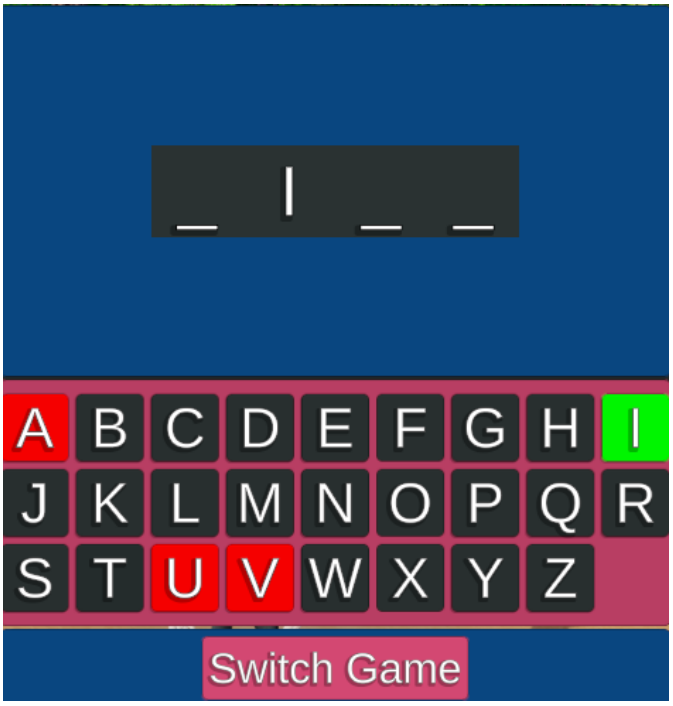

Figure 14 - The Hangman

4.4.4 Quick Maths

This mini-game is designed to test the player's quick calculation skills. The player is presented with an equation which they must solve as quickly as possible. The answer as well as a few other wrong ones are presented as buttons to be clicked.

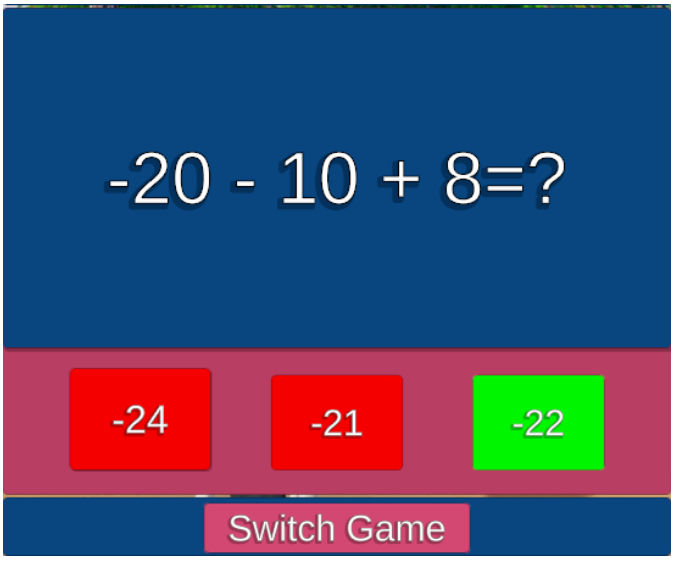

Figure 15 - Quick Maths

5. Conclusions and Future Work

5.1 Optimization

A big part of each video game is (or should be) the optimization process. The optimization process is the process through which you make the game require less computational resources from the computer to run. This is directly tied to FPS in games and is one of the most important things when designing them. A few examples of optimization which are used in, but are not specific to video games would be:

- Changing parts of the code so that it can be executed faster by the CPU.
- Using certain compression techniques for Audio/Images so that they require less memory.
- And more...

And some video game specific ones would be:

- Caching object references once instead of searching for them every time you want to use them.
- Combining meshes and materials of objects, as well as batching or static marking certain objects in order to reduce batches (CPU commands).
- Light baking certain lights, which increases memory requirements but drastically reduces CPU usage.
- Using LODs on objects to reduce vertices count when viewing them from a distance.
- *Many*, *many* more.

All of the above techniques, plus a few more, have been used in the creation of this project.

5.1.1 Further Optimization for Mobile/Web Platforms

The power of modern computers/gaming consoles is highly capable. Technology in both hardware and software has evolved rapidly, thus allowing developers to create ever increasingly complex video games, both in the visual and in the AI departments.

These platforms, however, due to their physical size/architectures/available resources are able to utilize the latest and greatest the technology has to offer, in contrast to the two emerging and steadily increasing in popularity platforms, The Web and Mobile devices. These two platforms are not ideal for developers to develop games on but are ideal for the consumers because everyone already has them/access to them, which in turn, makes it a big target for developers to pursue.

The main problem when it comes to these platforms, is the processing/graphics power they are able to use, which is extremely limited in comparison to the previously mentioned platforms. This is where the optimization part of game development comes into full play. Developing for web/mobile is in and of itself a very different platform to develop due to different control schemes/paradigms, expectations, physical limitations etc., but generally, if you want to port a video game to these platforms, a heavy amount of optimization is required. This optimization may even include some serious altering of visuals (e.g. unsupported shaders) and/or gameplay elements.

This project was not designed to be played in a web/mobile environment but has been modified in the process in order for it to facilitate the process, should it be wanted.

5.2 The game as a template

This project has been through a lot of variations in regard to who the audience is. Most of these variations came from the fact that there had to be some sort of educational value added to the game, but there has always been a difficulty in combining the normal-type gameplay with the educational parts of it. In one variation of the game, each of the "challenges", either normal or educational, were separate with each having their own place in the world, while another variation had them always be combined which created a forced dual challenge that didn't feel natural to encounter as a player.

The solution to these problems came as the middle ground of these two extremes. There had to be a way for the challenges to be incorporated in the game activities, without them being be active all the time, and since there had been two distinct levels of educational activities (one being the simple mini-games and the other the quiz-like questions) the idea of separating these into *educational levels* as a setting was created.

As the development of the game with this idea of having separated but easily combined activities in mind continued, and even more possible challenges and activities came up, a more abstract and overarching design philosophy started becoming more appealing to me and more fitting for this genre of game. This philosophy is about designing worlds, preferably small and strongly themed, in which there are activities for the player to do, but with each world having a specific difficulty of activities both gameplay-wise and educational-wise.

This helps the developer to come up with interesting standalone activities for both types of gameplay, each of which is designed around the theme of the corresponding world, which in turn can be easily combined into the whole educational value setting without having them feel forced or out of place. An example of this in this project is the activity of having the player looking for and picking up broken parts of a crystal around an area, and when the educational setting has been turned on, every crystal piece found by the player is also accompanied by a piece of geological information about crystals, and in the end of the activity the player is asked a multiple choice question about the given information.

5.2.1 Expanding the "difficulty settings".

Using this template of creating new worlds with variable difficulties is easy to grasp from a developer standpoint, but it does not guarantee the system will work with further additions to the game. Further studies must be made, with children involved, in order to find better ways to make this setting more focused and broader at the same time, in order for it to be able to facilitate a larger audience of children. Whether it's a wider range of ages, diverse ethnicities, children with special needs, or even children with different "video game skills" but alike in any other way, the difficulty can and should be accordingly adapted.

5.2.2 Allowing non-developers to adjust the game to their preferences.

One of the future goals for this project would be the ability for teachers, parents, etc, to be able to customize the game world to their preferences. This is currently available in a small scale (in the questions/answers JSON file), but with further development it could made into choosing which world "map" to use, choose the 3D objects they want to use and their location in the world. In addition to that, if more mini-games or forms of examination are developed for the game, they could be available as an *add-on* to add wherever they see fit.

5.3 The Future of Edutainment

It's no secret that video games as a medium for education has not been utilized as much as it could and should be. Either in school or in their free time, children are naturally attracted to playing (video) games, and such it's a perfect opportunity for developers and educators to combine forces and create something to help with teaching.

What we propose as a start to this is, either through public or private funding, create a team of developers and teachers to come up with a concept and the prototype of a game, designed for a very specific audience (specific in order to make the process faster and results easier to evaluate), which they can run for a limited time in one or a few schools. An even better scenario would be if the game is designed to be played from home, given in the form of homework or even as just a normal video game and be encouraged by the parents.

After the pilot program is over, and the evaluation starts happening, the results will, of course, be an increase, a decrease or no significant difference of performance from children. In the case of increase, the program could be expanded to include more activities, tasks, etc for more children to be able to play it, and further study the results. In the other two cases, the program could either be withdrawn or brought back to the table for improvements and adjustments, and then be reintroduced to children.

References

[1] M. Simmons, "Bertie the Brain programmer heads science council," *Ottawa Citizen*, 1975. [Online]. Available: https://news.google.com/newspapers?id=rKYyAAAAIBAJ&sjid=pe0FAAAAIBAJ&pg=916,3790974&dq=josef-kates&hl=en. [Accessed: 17-Apr-2019].

[2] J. Desjardins, "The history and evolution of the video games market - Business Insider," *Business Insider*, 2017. [Online]. Available: http://www.businessinsider.com/the-history-and-evolution-of-the-video-games-market-2017-1. [Accessed: 17-Apr-2019].

[3] P. W. Mitchel, "A summer CES-Report," *The Boston Phoenix*, 1983. [Online]. Available: https://news.google.com/newspapers?id=gn0hAAAAIBAJ&sjid=tYoFAAAAIBAJ&pg=5584%2C3561802. [Accessed: 17-Apr-2019].

[4] F. Colace, M. De Santo, A. Pietrosanto, and A. Troiano, "Work in progress: Bayesian networks for edutainment," in *Proceedings - Frontiers in Education Conference, FIE*, 2006.

[5] D. Charsky, "From edutainment to serious games: A change in the use of game characteristics," *Games and Culture*. 2010.

[6] Unity Technologies, "Asset Store," *assetstore.unity.com*, 2019. [Online]. Available: https://assetstore.unity.com/. [Accessed: 17-Apr-2019].

[7] Adobe, "Mixamo," *mixamo.com*, 2019. [Online]. Available: https://www.mixamo.com/. [Accessed: 17-Apr-2019].

[8] M. Banek Zorica, *EDUTAINMENT AT THE HIGHER EDUCATION AS AN ELEMENT FOR THE LEARNING SUCCESS*. 2014.

[9] P. Brusilovsky, O. Stock, and C. Strapparava, Eds., *Adaptive Hypermedia and Adaptive Web-Based Systems*, vol. 1892. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

[10] P. Wilkinson, "A Brief History of Serious Games," Springer, Cham, 2016, pp. 17–41.

[11] D. Cohen, *The development of play*. Routledge, 2006.

[12] B. Goodman, "Young Explorers Experience Ultimate Online Playground With National Geographic Animal Jam," *National Geographic*, 2010. [Online]. Available: https://web.archive.org/web/20130701065844/http://press.nationalgeographic.com/2010/08/03/young-explorers-online-playground-animal-jam/. [Accessed: 17-Apr-2019].

[13] D. Takahashi, "National Geographic virtual world Animal Jam hits a million kids | VentureBeat," *Venturebeat*, 2011. [Online]. Available: https://venturebeat.com/2011/05/16/national-geographic-virtual-world-animal-jam-hits-a-million-kids/. [Accessed: 17-Apr-2019].

[14] J. Gerstmann, "Big Brain Academy: Wii Degree," *GameSpot*, 2007. [Online]. Available: https://web.archive.org/web/20070629083034/http://www.gamespot.com/wii/puzzle/gentlebrainexercises/review.html?om_act=convert&om_clk=tabs&tag=tabs%3Breviews. [Accessed: 17-Apr-2019].

[15] M. Stegman, "Immune Attack players perform better on a test of cellular immunology and self confidence than their classmates who play a control video game," *Faraday Discuss.*, vol. 169, pp. 403–423, 2014.

[16] B. Cook, "I.M. Meen," *allgame*, 2013. [Online]. Available: https://web.archive.org/web/20130306231653/http://www.allgame.com/game.php?id=15837&tab=review. [Accessed: 17-Apr-2019].

[17] "Transformational Play Engine," *Atlantis Remixed*, 2019. [Online]. Available: http://atlantisremixed.org/. [Accessed: 17-Apr-2019].

[18] "CODY training," *uni-muenster.de*, 2017. [Online]. Available: https://www.uni-muenster.de/CODY/training.html. [Accessed: 17-Apr-2019].

[19] "Meister Cody | Meister Cody Talasia und Meister Cody Namagi," *meistercody.com*, 2018. [Online]. Available: https://www.meistercody.com/?lang=en. [Accessed: 17-Apr-2019].

[20] W. T. Grove, "Gamasutra - Unite 2008: Highlights And Award Winners," *Gamasutra*, 2008. [Online]. Available: http://www.gamasutra.com/php-bin/news_index.php?story=20813. [Accessed: 17-Apr-2019].

[21] "WolfQuest," *wolfquest.org*, 2017. [Online]. Available: http://www.wolfquest.org/. [Accessed: 17-Apr-2019].

[22] N. Vidakis, K. A. Barianos, G. Xanthopoulos, and A. Stamatakis, "Cultural Inheritance Educational Environment: The Ancient Theatre Game ThimelEdu," in *European Conference on Games Based Learning*, 2018.

[23] Unity Technologies, "Unity," *Unity Technologies*. Unity Technologies, 2019.

[24] Microsoft, "Introduction - C# language specification | Microsoft Docs," *docs.microsoft.com*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction. [Accessed: 17-Apr-2019].

[25] Mono project, “About Mono | Mono,” *mono-project.com*, 2015. [Online]. Available: https://www.mono-project.com/docs/about-mono/. [Accessed: 17-Apr-2019].

[26] Sonniss, “Sound Effects Libraries Categories,” *Sonniss.com*, 2019. [Online]. Available: https://sonniss.com/sound-effects. [Accessed: 17-Apr-2019].

[27] © OpenStreetMap contributors, “terrain.party,” *terrain.party*, 2019. [Online]. Available: https://terrain.party/. [Accessed: 17-Apr-2019].

Appendix

Available in-game models: