Text

Description automatically generated

Computer Games Development SE607

Project Thesis

Year IV

Dialogue Tale – A Dialogue Editor and Dialogue System for Unity with a VR example

**[Dawid Jerdonek]**

**[C00247646]**

**[24/04/2023]**

Submitted to the Department of Computing

in partial fulfillment of the requirements for the degree of

Bachelor of Science in Games Development

at the

South East Technological University

May 2023

© South East Technological University 2023. All rights reserved.

# Text Description automatically generated

**Contents**

[Acknowledgements 3](#_Toc133352802)

[Project Abstract 3](#_Toc133352803)

[Project Introduction and/or Research Question 3](#_Toc133352804)

[Literature Review 4](#_Toc133352805)

[Evaluation and Discussion 7](#_Toc133352806)

[Project Milestones 13](#_Toc133352807)

[Major Technical Achievements 16](#_Toc133352808)

[Project Review 18](#_Toc133352809)

[Conclusions 20](#_Toc133352810)

[Future Work 21](#_Toc133352811)

[References 22](#_Toc133352812)

[Appendices 23](#_Toc133352813)

# Acknowledgements

I would like to thank the following people who assisted in completing this project including;

My Project Supervisor Noel O’Hara.

All the 4th Year Games Development Lecturers at South-East Technological University Carlow which have provided crucial feedback, remarks, and criticism.

# Project Abstract

The problem which I have chosen to research is the complexity of dialogue creation in games and other forms of media stemming from this. In my project I aimed to research this issue and take steps to eliminate the problems at hand while making the project a more accessible and intuitive alternative to the traditional way of writing dialogue. The most obvious issue which looms over dialogue creation is simply put the inexperience of writers in coding or programming. Very often writers or individuals who are tasked with writing a story or dialogue to a game are unequipped to write it in a way which is easily accessible by a game. These writers have their own area of expertise which is writing not coding or in this specific case the syntax of dialogue. With this project I aimed to eliminate this issue and minimise the errors and further work that writers would have to deal with due to their lack of knowledge in programming and how dialogues are implemented into games.

The solution in my mind was simple, to create a Dialogue Editor which would not only easily allow writers to write dialogue for a game but on top of this represent the dialogue in a visual way which clearly indicates to the writer the different paths of dialogue and how the flow of said dialogue will run. These things are not obvious at all when simply writing a text file, especially when having to add syntax on top of it. As an additional piece of this puzzle, I have made the decision to create a Dialogue System which would process all of the text created by writers in the Dialogue Editor and then have the capability to apply it to a game scene or simply save or load the dialogue to a place on the hard drive. With the Dialogue System and the Dialogue Editor intertwined and working together the problem at hand would not only be solved but also the new way of creating dialogue would be a new easy to understand and attractive way of creating dialogue for Unity games for veterans and beginners alike.

# Project Introduction and/or Research Question

This project, Dialogue Tale has been inspired by a few key factors. To begin, I have a truly great passion for video games. I wished to base my project on something which would allow me to have a direct connection to video games. It could be a game or something that could be used in games or in the process of creating games. After some in-depth research I have settled on the latter. Alongside my passion for games, I have a lot of love for writing and storytelling. I immensely enjoy well crafted universes and characters in games that are captivating and relatable in different ways. I love the feeling of being transported to a different place through playing video games. The true power of dialogues in media such as Games can should never be underestimated and more so undermined. *"The most fundamental communication mechanism for interaction is dialogues involving speech, gesture, semantic and pragmatic knowledge."* (Merdivan, E. et al., 2019). So that is what I focused on. Through my research I found how dialogue for games can be tricky and more complex than it should be. So, with the problem identified I had a proposal for a project which involved streamlining the creation of dialogue to the point where anyone without some knowledge for computers could easily hop into a Dialogue Editor which I would create and easily edit and create new dialogues for a game.

This project has truly a lot of potential as it alters the idea of something complex and confusing to the average writer into something that could be used by anyone who wishes to simply write dialogue. The way it is formatted the project can easily be adapted into a Unity package which would allow users all over the world to utilise these new systems rather than having to work through writing files manually. It not only simplifies the writing process for games, but it also speeds it up and makes it reliable as no human error can be made when writing syntax as simply there is no syntax to write thanks to this project.

In context of research my Research Question is:

How can Dialogues be written quickly and efficiently, allowing for quick insertion of the Dialogue into a Unity game?

Main contributions of this project are:

* A custom Dialogue System and Parser allowing for loading, saving and utilisation of dialogue files.
* A customisable Dialogue Editor allowing for easy and quick visual editing of dialogue trees.
* A Template Scene incorporating all of these components to showcase the example use of the Dialogue System and how it could be utilised in a game scene.
* A Maze Generator which randomly generates a maze with the use of a backtracking algorithm on every start resulting in a unique maze every time.

# Literature Review

To create a successful iteration of a Dialogue System and Dialogue Editor I needed to research these topics in the context of games and mainly Unity to fully grasp and understand what came before and what I can possibly create to improve upon this and to allow my project to have a place in the area. Grinding down a branching dialogue system to its basics it *“…is different ‘menus’ of choices where each choice that the player makes triggers a response from the NPC, followed by a new menu of choices.” (Mikkelson, 2020).* With this in mind, I could understand and note down that for my system to work I needed choice nodes where the user specifies one or more choices and then subsequent responses depending on this choice. But a Dialogue System needs more nuance not just a set number or responses and dialogues for every conversation. *“Some dialogue trees have no outcome. They might strictly be informational, with a little sass sprinkled into the player’s options for sheer flavour and role-playing value. But, more complicated dialogue trees will have two or more different outcomes that will happen based on the player’s choices” (Mikkelson, 2020).* The user ability to add nodes at will and delete them just as easily was a must-have to fully embrace the idea of this dynamic Dialogue System and Editor.

A means of storing Dialogue was a key component of the project as a whole. There was an obvious requirement to store the Dialogue in a file that could be accessed, loaded, edited and saved without causing any interruptions or hassle to the user. I have researched the possible file formats, syntax and file parsing which resulted in various different possibilities and to play into the project idea a little, “paths” which I could choose to follow. *“Dialogue management can be formulized as a Markov Decision Process which is defined as a tuple” (Merdivan, et al. 2019).* I had the task of choosing what file type to use and how I would format these dialogue files. XML became the frontrunner and soon enough I had to learn *“...how to read the data from XML using C# and how to assign that data to a List so that it can be used elsewhere in your program. This is, essentially, the very basics of reading an XML file in Unity.”* (*Philipp, 2015).* Throughout this blog article, it is very clearly outlined how to format a basic XML file to have it be then read by Unity. I could get my dialogues written in XML with the correct use of elements, *“So, my <name> tag is technically an element. Whatever that element contains is the data that is in between the <name> and </name> tags.”* (*Philipp, 2015).* By understanding this key concept of XML files, I could access them and using these elements I could separate the file into not only dialogue and replies but also, I could give a number to each dialogue stage and even store multiple characters in a single XML file*.* Of course, I would have to create my own Parser and decide my own way in which I would syntax my dialogue, but this article was a great steppingstone and it thought me a lot about how XML files worked and how Unity could utilise them*. “Once you understand the basics of XML reading in Unity, you can further refine your code and create an incredibly dynamic system.”* (*Philipp, 2015).* ADynamic system was exactly what I required.

As part of my “Template” scene, I needed to create a Maze Generator, that had the capability to procedurally generate a maze every time I started the project to showcase procedural generation and dialogue dynamics. A great resource for learning and researching this topic was “AI for games, Third edition” by Ian Millington. This book goes in depth on the uses of AI in video games and it has a whole chapter dedicated to Procedural Content Generation. That is the field which I needed to explore and research for this project most, with major focus on its maze generation notes. With so many different ways of creating a maze I needed a guide, *“Procedurally generated dungeons go right back to Rogue, and have featured prominently in the recent revival of Rogue-lite games. There are many different algorithms and variations, enough to be able to fill a book on their own.”* (*Millington, 2019).* One way of procedurally generating mazes which this book outlines is with the use of an algorithm known as “Recursive Backtracking”. The book explains the process which the algorithm undertakes every time it is run, *“To create a maze, we can use a simple backtracking algorithm. The level is split into a grid of cells; all of them initially unused. Initially the entrance cell is excavated and this becomes the current cell. Then the algorithm proceeds iteratively. At each iteration a random unused neighbor of the current cell is chosen. The current cell is connected to that neighbor, and the neighbor becomes the new current cell. If there is no unused neighbor, then we return to considering the previous current cell. When we are all the way back to the starting cell, and it also has no more unused neighbors, then the algorithm is complete”* (*Millington, 2019).* This clearly explains to me what the code is supposed to do, with the guidance of this book I went off to create my own version of such an algorithm.

Another way of creating a Maze which I have researched is with the use of Recursive Subdivision. With the use of “Recursive Backtracking” the algorithm produces *“results that can display a characteristic kind of randomness: elements are placed into the environment in ways that don’t always make sense as a whole. As an alternative, we can progress top-down, beginning with the whole space, and refining it into smaller and smaller areas”* (*Millington, 2019).* “Recursive Subdivision” is a solution to this problem. However, the randomness that comes with “Recursive Backtracking” was not an issue at all in my project. This specific algorithm *“is used to layout rooms, which are then later connected by corridors”* (*Millington, 2019).* Meanwhile all I needed was a simple maze without the creatin of rooms per say. Even more so I wanted to highlight the randomness of the maze, hence I chose to stick with “Recursive Backtracking” and write an algorithm based on that.

**Existing Work**

Many of my research attempts led to multiple forums or online databases where questions about writing branching dialogue were often asked. The one outcome which has come up more often than any other was an application known as “*ChatMapper*” which offers a sample for users to try out for themselves, “*This sample makes use of Responsive Designed HTML export, Multi-Language Dialogue Nodes, Recorded Audio for English, Spanish and French Text-to-Speech, Actor Expression States, Scripting and Variables” ChatMapper* (2010). This is a powerful tool with a plethora of simple and complex features allowing its users to create branching dialogue and much more. It has different pricing plans depending on a user’s needs and spending specifications. This tool even has its own integration with a Dialogue System that is used in Unity from a website known as “*Pixel Crushers*”. It is clear that *“ChatMapper”* is equipped with a suite of features to assist user in the creation of dialogues for various purposes. The *“Dialogue System for Unity (PixelCrusher) makes it easy to add interactive dialogue and quests to your game. It's a complete, robust solution including a visual node-based editor, dialogue UIs, cutscenes, quest logs, save/load, and more. The core is a lean, efficient conversation system. A large collection of included, optional add-ons make it quick and easy to drop conversations into your project and integrate them with other products. No scripting is required, and complete C# source is included.” ChatMapper* (2010). However, my project amongst many had one glaring difference, it was to be integrated into Unity without a steep price point, this would allow me to fully focus on making the functionality as customised as possible towards the use of Unity and only Unity rather than focusing on many different aspects of dialogue creation and different engines.

The previously mentioned *“Pixel Crushers”* is also a website with lots of viable information for a project like mine. Best said in the website writer’s own words, *“Pixel Crushers creates cross-platform Unity 3D middleware products that provide elegant, practical solutions to the tough problems in narrative AI.” (PixelCrushers, 2018).* The system is extensive, in-depth and offers many useful tools for all users in need on a Dialogue System in Unity. The system is so impressive and functional that it was even used by critically acclaimed studios *“Studio ZA/UM”* and *“Torpor Games”.* I was able to take inspiration and feedback from this Dialogue System and apply all important information to my own project. *“Built-in support for Unity UI, TextMesh Pro, PlayMaker, Bolt, Adventure Creator, ORK Framework, Corgi, TopDown Engine, and many more third party assets” (PixelCrushers, 2018).* This asset proved to be the most similar product that I have found to what I wished to create. I simply wanted a more transparent and customisable version of this which adjustments at various points of the system. I wanted to make my system look and work in a unique way, taking inspiration only from the functionality of the system not code or visuals. A major difference between my project and this asset once again would be the accessibility,the *“Dialogue System for Unity”* created by *“Pixel Crushers”*, can be found on the Unity Asset Store and purchased for €84,43.

Unity Asset Store also has another version of a Dialogue Editor uploaded on it. *“A Unity Tool for creating and adding conversations into your games.” (Grasshop Dev, 2020)*. Through my research I have investigated its functionality and capabilities, how it works and the user feedback. The tool simply named *“Dialogue Editor”* is a free to download unity package allowing users to create their own dialogue. Exactly what my project ideal is. As outlined by the developer of this asset the features of this Editor include, *“Editor window for creating and editing conversations.”, “Pre-made UI prefab, so no UI programming is required.”* and *“Simple DataStructure, available if you want to write your own custom UI”. (Grasshop Dev, 2020)*. This proved to be a valuable resource as it had its own take on Dialogue editing and a ton of user feedback. I kept this user feedback in mind while creating my own version of a Dialogue Editor, ensuring that the most glaring criticisms and issues would not plague my project.

# Evaluation and Discussion

Upon initial running of grid spawning the maze plane would be covered by spawning of prefabs creating a grid. This grid initially had to entrance and exit which was needed for a maze. Hence, I removed two walls at opposite diagonal ends of the maze (Figure 1). Henceforth the maze would always have a start and end in the same place, yet the maze would still be random. (Figure 2, Figure 3, Figure 4, Figure 5)

Initial grid created before backtracking algorithm is ran: Figure 1

Generated maze once backtracking algorithm is ran: Figure 2, Figure 3, Figure 4, Figure 5

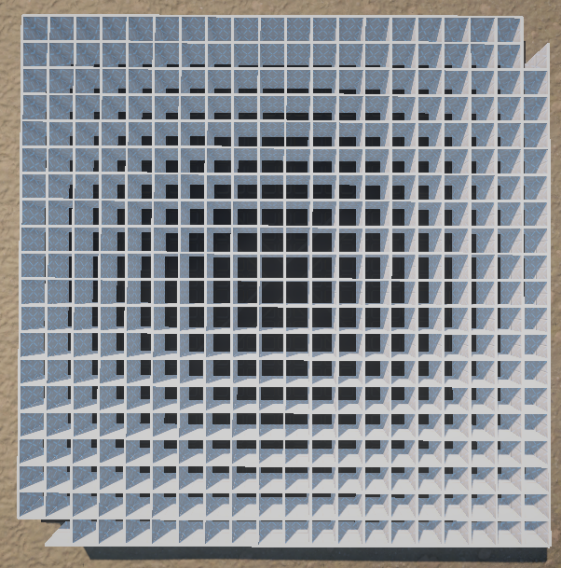


Figure 1

A picture containing text, outdoor object, manhole cover, grate

Description automatically generatedA picture containing text, manhole cover

Description automatically generated

Figure 2 Figure 3

A picture containing text, outdoor object, manhole cover, grate

Description automatically generatedA picture containing text, outdoor object, manhole cover, grate

Description automatically generated

Figure 4 Figure 5

I wanted to end up with a simple looking and easy to use Dialogue Editor. I needed to hide all complexity and display the absolute minimum at all times to avoid discombobulating the user.(Figure 7) The experience needed to be sleek and without issues. As simple as the user pressing a button and getting exactly what they expected no unnecessary surprises. User friendliness was one of the many areas of focus for this project. Through some of my user testing I was able to see the errors users would make and with that data I was able to try nullify the risk of those errors occuring by eradicating the source of the issue from the project. Regardless if the issue was bad UI design or a functional error, everything was easy to spot when a sample user tested the program.

Old HUD design visible at all times allowing users to toggle buttons and text: Figure 6

New HUD only showing up when Saving or loading files no extra text or buttons at top: Figure 7

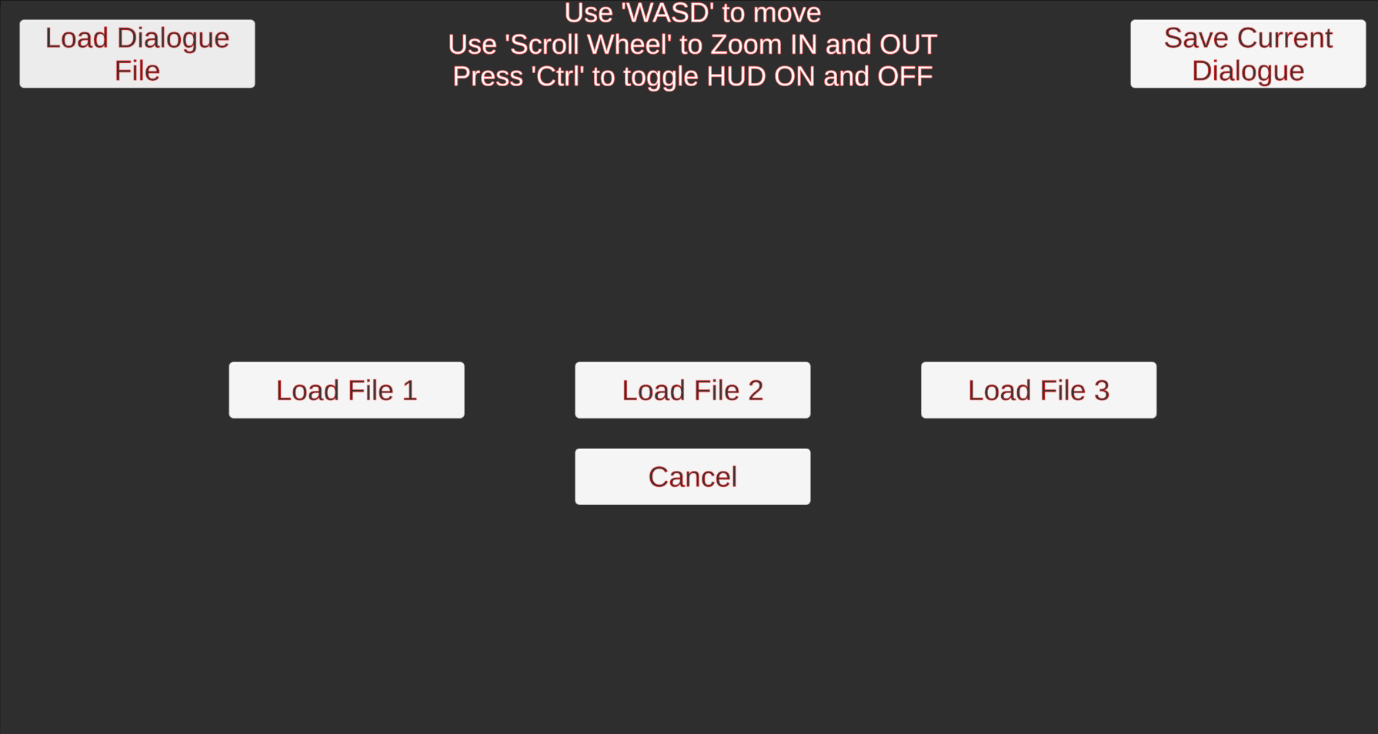


Figure 6

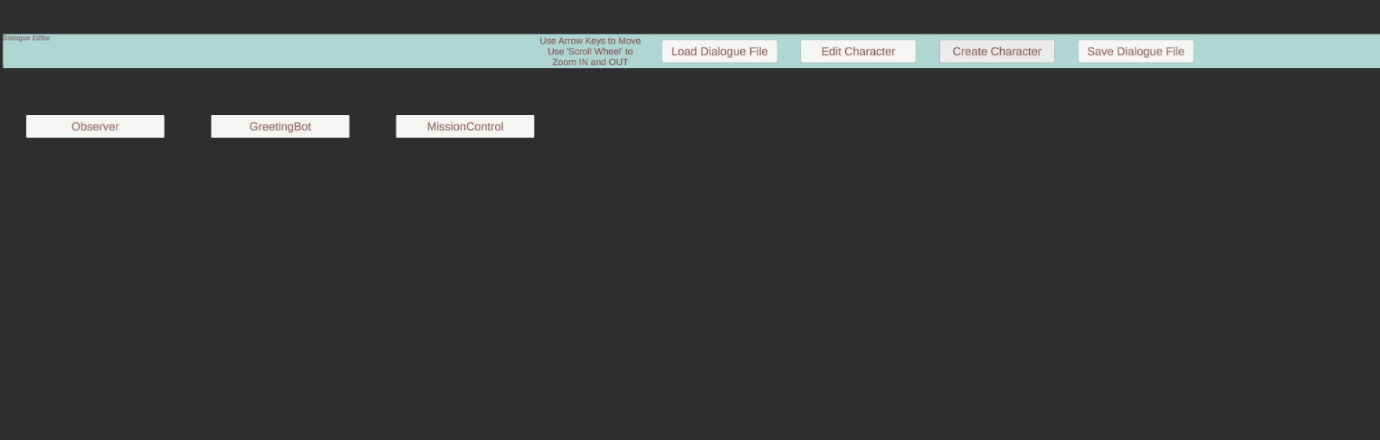


Figure 7

The project required a parser that could parse XML into a dialogue (Figure 8) and then once a user would choose to save their changes it would parse the dialogue into XML (Figure 9). This proved tricky but attainable. Once I have created my first XML as a base for all dialogues, I was able to begin work on my parser. Upon creating a Dialogue struct all that had to be done is correctly get rid of syntax from the XML to populate the dialogue lists in Unity.

Parser function which is responsible for removing syntax from loaded XML file and assigning it to correct dialogue variables: Figure 8

Parser function responsible for processing in engine dialogues and outputting them into an XML file format with the correct syntax: Figure 9

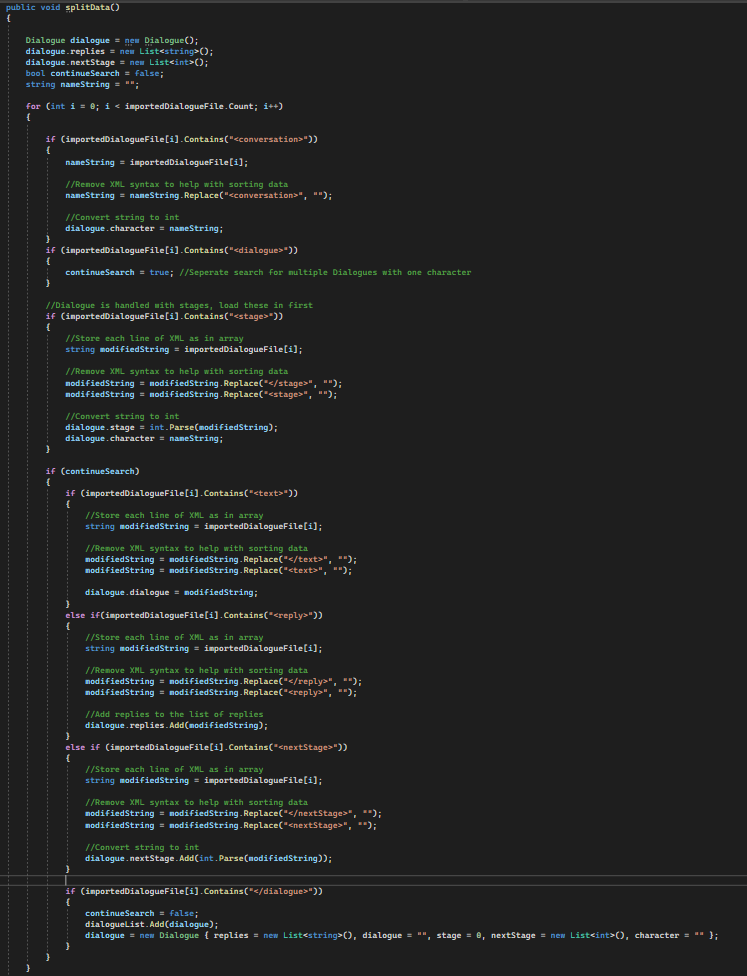


Figure 8

Text

Description automatically generated

Figure 9

The creation of a HUD for the template scene was a must. It was necessary to display all the edited dialogues and character names once the player would encounter the characters. A major part of this project is the Dialogue System after all and the way it dynamically works with multiple different characters is a highlight. It allows for characters to be easily distinguished between each other. (Figure 10 & Figure 11)

HUD template: Figure 10

HUD while “Template” scene is running to showcase its functionality: Figure 11

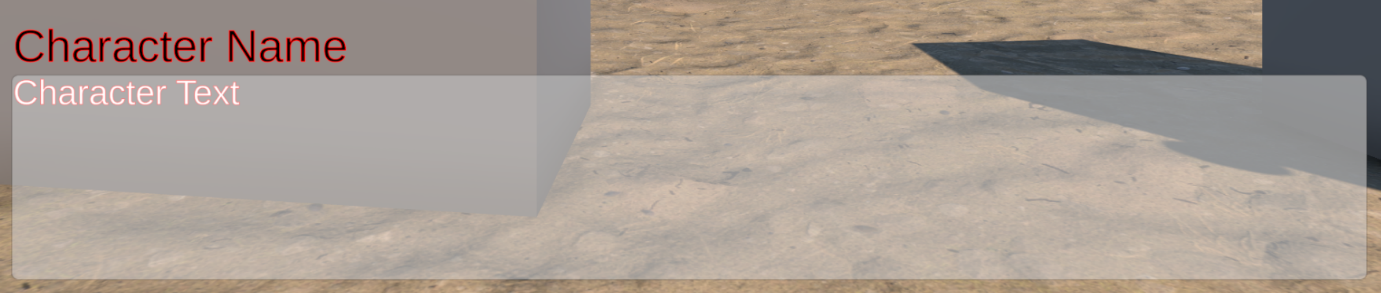


Figure 10

Text, letter

Description automatically generated

Figure 11

# Project Milestones

**14th of December 2022:** Fully functional Maze Generator.

This was the first major project milestone which I have achieved. 14th of December was the day when I completed the Maze Generator. I used the backtracking algorithm to procedurally generate a random maze every time the scene would be run. I had finalised this feature a day before a live demonstration of progress to supervisors. I was very happy to complete this as up until this point I had only learnt about maze generation and tried some testing and small examples. This was the first major feature I had completed and integrated into my project. I was seemingly on schedule however it became apparent to me that I needed to make a clearer journey to the final submission day with more solid deliverables planned.

**15th of December 2022:** Code and progress demonstration, feedback, and discussion of future of project.

The Project Supervisors have gathered for a demonstration, the main goal being an inspection of current project direction and assistance on making any adjustments and improvements which would allow for a better project to be delivered.

I delivered a reasonably satisfactory amount of progress at this time however the final direction of the project has not been fully recognised and set in stone. This meeting allowed for a fleshing out of the project goal and setting a clearer direction with the help of supervisor feedback. Incredibly important information and feedback was shared in this meeting which acted as a foundation for the rest of my project. After this meeting I had obtained a clear and vivid image of what I want to achieve and where I want this project to be on submission day.

I was on schedule with my work however my direction was not clear enough prior to this point and I needed to readjust myself to adhere to a more suitable schedule for myself. I needed to focus more on the importance of the Dialogue editor rather than having so much focus on the other aspects of this project, the Template Scene, and my Maze Generator. Having those two in a very elaborate state was great, however I have not given enough attention to the Dialogue Editor at this point in time.

**18th and 19th of January:** Created a basic early version of the Dialogue System and begun work on its editing capabilities.

This is the day that I completed the creation of the very first basic version of my Dialogue System. It was not much more than a simple blueprint which would later guide me to create a more functional system. This was the first time where I began working through and noting what I would like to have my future Dialogue Editor do. I have created my first version of the Dialogue Editor also, although both of these were extremely limited in functionality and did not add much it was a crucial step in the development of my project. An interesting detail of note is that this initial Dialogue Editor was a version which was part of the in-game world. Meaning dialogue would be able to be edited in real-time while a game runs. However, this was not the goal of my project, and as I have gone through more and more research it became even more obvious that I should focus on a separate scene for the editor as it would direct focus to what this project is about. It also became clear that Virtual Reality is not a good option when it comes to performing anything which involves editing text.

**10th of February 2023:** The Upload of my Dialogue Parser.

How storing of to store a file to then be accessed in Unity efficiently? How can dialogues be stored? What is the best file format to use for storing dialogue data for Unity? How parsers work? How can dialogue be stored and accessed using syntax?

Those and many more were the questions which I have asked and undertaken extensive research about. For the weeks prior to this Milestone date, I have gone through many different ideas and “prototypes” in search for an easy and efficient way to save, store, load, edit and access all my dialogue files. After all my research and tests, I have finally decided to settle on the use of XML for my dialogue storing. It was easy to syntax and very malleable to whatever I needed. After I settled my way of storing the dialogue, I needed a way to load it into Unity and apply it to NPC’s so that the characters were able to use all the stored dialogue. This is where a parser needed to be made. So, I have undergone research, and through much trial and error I finally came out on top with a well refined and useable version of my own custom written Parser. I was incredibly happy when I got it to work. This day marks a huge Milestone of my project as the parser is now one of the major components of my project. With the parser created I was able to move to the implementation and improvement of my Dialogue System and further evaluating its editing capabilities.

**17th of February 2023:** Completed the first functional iteration of my Dialogue System.

This day marks the upload of my first fully working Dialogue System, it had many flaws and was very “hard-coded” in a sense that there was not much room for change or customisation. It was only working with my set of characters. However, it was a great foundation for my future modular rework of the system. A lot of the code was good and would prove to be useable in a modular system I would just have to evaluate and rework the way in which I stored and processed dialogue and characters in the future.

**15th of March 2023:** Major Progress Demonstration to Supervisors.

The Project Supervisors have gathered again for in depth demonstrations on project progress and discussions about the final deliverable as well as positive and negative feedback alongside, constructive criticism. Most importantly it was a check on progress allowing for feedback and important advice.

After the last major meeting in December, I re-evaluated my progress so far and thought through the future of my project. Thanks to all the feedback I was given back then, I slowly but surely got back on track towards my final goal. For this demonstration I had created an initial setup for my Dialogue Editor, at this stage I was only able to edit my Dialogue in its raw form as an XML file, that is no good for the final submission as it is exactly what I am trying to get rid of with this project. However, this was a good basis and example to show that I am capable of creating a Dialogue Editor and with enough work put into the project I would be able to achieve my final goal. In retrospect the Dialogue Editor was nothing of note on this day, but my main progress thus far was my Dialogue System which allowed me to save, load, preview and use dialogue files in my “Template Scene” which I had created. I was able to showcase the ideal where I have a conversation with an NPC, but I am capable of editing this dialogue with my very early basic version of the Dialogue Editor, then saving this to my chosen save file. Once saved I was able to load in this custom dialogue and show the change in what the same NPC says depending on how I edited his dialogue.

**12th of April 2023:** Dialogue System Rework to move to a more modular approach.

In this Milestone I have successfully achieved the removal of all my “hard-coded” components from the Dialogue System. I have created a new class called “DialogueSystemNew”. This new class was an expansion and improvement of my previous Dialogue System. It was a more refined and a much cleaner version which also was capable of handling dialogue in a modular way. This meaning that I no longer needed to name characters in any special way and give them their own names in code. This modular approach has allowed me to create any character of any name and correctly parse all components of their dialogue and store them correctly to allow for use in my Template scene to showcase how the dialogue works exactly.

This Milestone was originally completed around one to two weeks prior of April 12th. However, I have had internet issues at the time of achieving this milestone. Shortly after our Easter break began and I chose to focus my time on researching other Dialogue systems as well as ways of displaying these dialogue “Trees” effectively. As a result, I was unable to upload this to GitHub until April 12th but expanded some more and made some improvements during this time without uploading.

**18th and 19th of April 2023:** Important progress milestone prior to the final Submission Day.

These days were an important milestone for my progress on this project as this day marks the completion of all basic functionalities of the project. On the 18th of April I have created a system for displaying a Dialogue tree, in a set of nodes in a correct and easy to understand shape and manner. On the 19th I finished up the basics of the editor. The Editor was capable of editing each node of a character’s dialogue as well as each answer. It was also capable of creating new characters from scratch on top of that. From this point on I was able to fully direct my focus on more extensive editing as well as all extra functionalities. Another priority from this day was creating a more user-friendly (and customisable) experience alongside adding some more substance to the visuals of the project to create a more pleasing image to the eye while using the Dialogue Editor. On this day I was on track to completing this project on schedule.

**24th of April 2023**: Final Submission Day

Today the final submission day. Today is the day where I must submit all the work I have gathered throughout the year. All the project code, documentation, and other deliverables.

After being able to direct all my focus to this project for the recent few days I am incredibly happy with where my project is. I am very proud of my achievements and how the final product turned out. I was able to keep on schedule for this final submission and I have everything ready. My final delivery is on-time.

# Major Technical Achievements

**Modular Dialogue System**

First major technical achievement of this Final Year Project is my Dialogue System. I am extremely proud and happy to have created a fully functional Dialogue System that is fully dynamic in the context of multiple characters in a game scene and further editing of the dialogue stored by the Dialogue System.

My original Dialogue System needed to separate my dialogue file into multiple dialogues each for a different character and these characters had to be hardcoded. Hence this was no good as the whole idea of my Final Year Project was to simplify dialogue creation and editing which would in turn allow for easy implementation into a game. So, the Dialogue System needed to be reworked. The current Dialogue System is the result of that rework. I tend to call in the Modular Dialogue System or the Dynamic Dialogue System as it refers to what it is at its core. It stores dialogues in a dynamic way so that characters do not need to be hardcoded. All that the characters require is simply a tag which contains their name, the same one that is used by the Dialogue Editor. For example, the user creates a character named Bob and he is given dialogue by the user. To add dialogue to this character I simply give him the tag named “Bob” and the New Dialogue System does all the work for the user and assigns all the dialogue that was created to this character. To do this easily I created a struct called Dialogue. In this struct I separated the dialogue into the key components I used when writing some sample dialogues in the XML file. The variables I used in this struct are as follows. Character for a character’s name. Dialogue for each piece of dialogue a character says. Replies for storing all the possible replies that the player can give to a character’s specific piece of dialogue. Stage which would store the current index of the dialogue indicating how far into the conversation the player is.

I have kept this final relevant detail about the Dialogue System till last as it coincides with my next point and that is the Parser. The Parser takes an XML file stored on the computer and parses it in a way to remove all syntax and allow the Dialogue System to format it in a way that is both easy to access and understand. The Dialogue System has its very own Dialogue Struct which populates the new Parsed Dialogue file in a relevant and formatted manner allowing the Dialogue System to access any given piece of Dialogue with ease.

**XML Parser and Formatter to format Dialogue back into XML**

As mentioned above the Parser is the second major achievement I write about. After settling on the XML extension for my dialogue files I needed a way to load the XML file into Unity and then convert it into useable Dialogue without any of the syntax that XML contained to differentiate between different data types or in this case different components of a Dialogue. As a result, I needed a Parser and after doing research I have decided that I wish to create my own Parser rather than using a premade one or basing it off something that came before. I wanted to research necessary topics, but I still wanted this project to keep as much identity as possible. So, after much trial and error, many failed attempts I got progressively closer and closer to a result which could be used for the Dialogue System and Dialogue Editor.

As mentioned prior, I created my own Dialogue struct in the Dialogue System class. This allowed me to write the XML parser in a way that took advantage of the struct. All that was needed was for me to correctly loop through the XML file in search for syntax that corresponded to a variable in the struct. Once found, the appropriate line is processed by the Parser to remove the syntax. Once the line is processed the line is then assigned to the appropriate variable in the dialogue struct. The most efficient and intuitive way of storing all these variables was to make them into lists which could be accessed by the correct line index. Or by searching for a character name and then specifying the stage number depending on previous progress through a dialogue tree.

**A Fully working** **Dialogue Editor**

The Dialogue Editor was a crucial part of the Project and an integral component to allow for full dialogue tree customisation, which is what this project is about. Initially the first version of the Dialogue Editor which I worked on was part of the “Game” now “Template” scene. This way the user would have to enter the gameplay scene every time they wished to edit dialogue. On top of this I have my “Template” scene created to work for VR mainly which is not that useful in terms of writing dialogue as it is quite tedious and takes significantly longer than simply typing the dialogue on a keyboard. Hence soon enough after gathering some supervisor feedback and thinking through this issue I have decided to go in the direction of a completely separate scene for the Dialogue Editor. This would not only make the project more intuitive and easier to use but it also gave the project a genuine purpose as it could even be made into a Unity package fully manifesting the idea of creating a user-friendly Dialogue Creator/Editor for anyone to download and use.

After this imperative change it became clear that the project really came into final gear and my idea was becoming more and more realised. Now the Dialogue Editor is its own entity in a separate fully dedicated to loading, analysing, editing, and creating dialogues for characters. Its functionalities include creating a new character from scratch, adding new dialogue nodes, adding reply nodes, deleting dialogue or reply nodes, editing existing characters and their dialogue and player reply nodes, adding new nodes to these existing characters, saving edited dialogues, loading edited or example dialogues, customising the user-experience of the dialogue editor and more. All without having to worry about any syntax as the editor takes care of all the confusing and complex things instead of the user. These edited dialogues can then be loaded into a game scene with the use of the Dialogue System. All that is needed is an object in the scene with a tag corresponding to a character name and that is all, the user is then able to load any dialogue he wishes into the game. The Dialogue Editor is without a doubt one of my great technical achievements of this project and together with the Dialogue System they play an integral part in this project.

To make the experience more user-friendly and personal. Each user has the ability to drag the dialogue nodes around the Dialogue Editor. This is a feature of the Dialogue Editor which creates additional freedom for each user but also it is a great means of counteracting the issues that come with a large dialogue tree. The larger the dialogue tree the more confusing the visual representation gets. The nodes get messy, begin to overlap and can overall get unnecessarily convoluted. With the dragging of nodes in the project, this issue is no longer as prominent as any user can now move every single node to any place in the Dialogue Editor. The lines connecting nodes will always stay attached to easily signify node connections, so it does not matter how far the user chooses to space out the nodes. Overall, this feature greatly improves visibility and the feeling of personalisation in the Dialogue Editor.

**Maze Generator**

Lastly, I would like to give a mention to my Maze Generator as it deserves to be part of my major technical achievements. Through the use of a backtracking algorithm, I create a procedurally generated maze every time I load into the “Template” scene. To do this I have a Maze Plane which acts as the dimensions for this Maze. I then populate the plane with wall prefabs, this created a grid which the algorithm will traverse. I choose an end and start node in this grid. After that the algorithm begins traversing the grid to the end point destroying each wall it passes through. Once it reaches the end node it begins its backtracking visiting each unvisited node and destroying each wall it passes through once again. The algorithm stops once all nodes in the grid have been visited. A random maze is created as a result.

Its functionality is great and works just as intended, the only issue which I would have wanted to address given more time is its overall impact on the project. It does not have as big an impact as some of the other technical achievements as I did not have enough time to expand on my “Template” scene to showcase how Dialogue can be changed in real time depending on environment. The main goal was to have characters guide the player through the Maze, the neat twist is that the Maze is always randomly generated, this combined with my Dialogue System would’ve altered the maze characters dialogue each time as the maze would be different each time.

# Project Review

**What went right?**

Firstly, and most importantly, the project as a whole went right. It went very well and despite some hiccups and issues along the way the project ended up in a satisfying state. I was able to deal with those issues in a smooth and professional manner allowing me to have a final submission which reflects well all the work which I have put into the project.

The Dialogue Editor had its own fair share of gimmicks and specifics which had to be adjusted and addressed to ensure the correct functionality of the Editor. Now the Dialogue Editor not only works perfectly fine but is also incredibly satisfying to use and perfectly equipped to do its job. On top of this the Dialogue Editor also has customisation options allowing users to fit the Dialogue Editors visuals and some of its functionality to their own needs and desires.

The Parser was a great weight off my shoulders once I have finalised it. It went perfectly right and does exactly what I want it to do. It is capable of taking my XML files and parsing them in a way to be understandable by both the user and my Dialogue System in Unity. Then once a user wants to save the Parser is capable of reformatting the Dialogue back into XML with syntax to ensure all data is stored correctly in an orderly fashion.

The improved new Dialogue System that I had to work on after inspecting the limitations of my previous Dialogue System ended up taking some portion of my time, yet in the end it was worth it as now the project had a fully dynamic Dialogue System which did not have any pre-set character data hardcoded. It is incredibly useful, and I can see it fit in very easily into many games where a user uses the tools provided to add dialogue to their game.

**What went wrong?**

The original working Dialogue System which I created was not what I needed to make. I wasn’t fully focused and in understanding of exactly what I needed at the time of making the first Dialogue System hence when it came to the point where I finalised some of my Dialogue Editor capabilities it became apparent that the initial version of the Dialogue System will not work. It was simply too specific. I needed a system that could be dynamic and modular. One that didn’t need any specific characters or didn’t expect any specific names. I needed a Dialogue System that could take in Dialogue data and assign it correctly to any character in a scene only based on the properties of the specific character object. Due to this I had to return to the Dialogue System and re-evaluate some of the ways in which I stored information and how my Dialogue System would process data.

**What is still outstanding/missing ?**

The one outstanding piece of content in my project would have to be a more complex implementation of the Maze Generator. I have written an algorithm which allows me to create a procedurally generated maze. I have created a plane and I populate it by instantiating walls to create a grid on the plane. The algorithm I wrote for this project then runs through the grid and destroys the wall that is passes through when visiting a neighbouring node.

**If starting again, how would you approach this project differently?**

Instead of instantly focusing on the Maze Generation I would first focus on the creation of a Basic Dialogue System and proceeding from there I would delve deep into the creation of a complex Dialogue Editor which gives a user all that they would desire all put together into a neat, pretty package that is easy to use and not complicated for the end-user but incredibly detailed on the inside. That is what I aimed for, and I have gotten to a satisfactory point yet, I would shift much more of my focus onto the editor itself if I could have started again as I believe that simply the Editor deserves much more focus than I ended up giving to it., simply to even make it more personal for each user and more customisable. However I am still incredibly satisfied with the Editor I have in place, I just wish I could continue my work on it to perfect it even more.

**What advice would you have for someone attempting a similar project in the future?**

My advice would be to definitely give a significant percentage of attention to user testing. Simply give a friend or a family member 5 minutes or so to go “have a spin” of your Dialogue Editor. It might not sound like much, but the feedback gotten from someone using your project is invaluable. It allows you to put yourself in a user’s shoes and see the project from an outside perspective. How do they go about performing simple tasks? Is the UI intuitive and the user feel comfortable or is it not obvious enough? Those and many more important questions can be answered within minutes with the help of user testing.

Ensure that you understand how you are storing and parsing information about the different dialogues. It is of highest importance that this is understood by the developer of a project like this one as these things are constantly coming up and the project simply needs to be handled in an adept way in regard to this. This is a prime example of where issues could appear if the creator does not understand the way their own parser works or how the XML or file of choice stores information. Issues and necessary code additions will show up all over the project as the dialogue have to be stored, loaded and parsed so that it can display correctly in Unity, overwritten, and saved and reformatted back into XML so that when loaded in again the same loader and parser can be used.

Another thing I could recommend to someone is a greater focus on the overall user experience. Make the system more approachable and visually pleasing. In projects of this sort, code and functionality is of course above all else, most important. However not a single person will choose to use the system if it is unintuitive, convoluted, confusing and much too complex to understand or simply if it doesn’t look nice. First impressions matter!

**Were your technology choices the right or wrong ones?**

My technology choices were the correct choices. I wanted to create a Dialogue Editor that could be used for a game engine. Choosing Unity was a great choice as it is an engine that is very often used by smaller and indie studios as well as students and enthusiasts like myself. Unity also does not have its own built in Dialogue Editor which has given me the perfect opportunity to truly create something with meaning and it gave my project a place where it could be used.

# Conclusions

In conclusion my project has fulfilled and answered my research question accordingly.

How can Dialogues be written quickly and efficiently, allowing for quick insertion of the Dialogue into a Unity game?

I believe that my project is a prime example of how this can be achieved. The Project “Dialogue Tale” tackles each layer of this question and presents a simple yet effective way in which the question is answered. The project has the tools which are presented to a user, allowing said user to write dialogue quickly and efficiently in an easy to use and understand manner. The project is transparent and hides as little as possible from the user, allowing a user to understand and learn how the Dialogue Editor and System both work. Once the dialogue is written by the user, the dialogue can be loaded into the “Template” Scene to showcase the functionality of the Dialogue System. The research question presented by me asks for a quick insertion of the dialogue. The project does exactly that, all that the user has to do is to add a component to the target character object and then add a tag which coincides with the name given to the character in the Editor. That is all, the Dialogue System then proceeds to do all the complex work of loading, parsing, reformatting, and assigning the dialogues to correct characters for the user.

# Future Work

Since games and writing are both an incredibly large part of my life and a significant portion of my passion goes directly to them, I intend to align my future as closely as possible with those two.

The finalisation of my FYP and its subsequent submission do not mean the end of my work on this project. I have put a significant amount of work, effort, research and, genuine interest into this project and its research. As a result, I do not intend to finish the project yet. My FYP and what I have planned for it will be finished and submitted, yet the project has much more potential than that. Hence, I plan to extend work on this project for a further few months, perhaps over the Summer. I will dedicate my own free time to perfecting this project and to transforming it into something more than just my FYP.

My intentions are very simple, I wish to further refine and develop this System and once ready I would intend to release the project as a package onto the Unity Asset Store. I very much understand that the chances of having thousands of downloads is very low, however it is something I would want to achieve. I believe this would be a great achievement and asset to have and showcase not only on my CV but as part of my career. I genuinely enjoy working on my Dialogue Editor and System regardless of some of the issues and great difficulties I have faced. I really believe that this project can be something great and an amazing opportunity to not only create something big but also to put it out there into the world and gain potential feedback and further improve the package as time goes on.

The next important steps on the journey to creating a fully functional releasable package from my FYP would be focusing more on the User-Experience. The idea of this project was to simplify the creation of dialogue for game creators. It is to remove any potential syntax errors and simply focus on the writing of dialogue, in an easy to navigate and intuitive environment. And I do believe that I have achieved that, however I would focus on this even more as my first step. I would get friends and family to test this for me and make notes how they use the system. How easy it is for them to use and what are the main errors and/or complaints they have. I would then ask them if they feel like there is enough customisation and editability in the system and if they have any suggestions or wishes in regards to what they would like to see added or what action they would like to perform. I would focus on this user-testing in particular because I have done a slight amount of user testing for this FYP and Research. All those user-testing sessions proved to be incredibly valuable towards my progress, further understanding and overall direction. Once the feedback is received, I would simply use the notes from that feedback and try to find sensible ways to implements these features or improvements.

The last very clear idea in my mind in regard to further work on this project would be to fully incorporate the Maze Generator into my “Template” scene. My “Template” scene is where I showcase and test all of my dialogue. It places the Dialogue System into a game setting with different characters and it assigns each dialogue to each corresponding character. I have a Maze Generator in this scene which randomly creates a maze using a Backtracking Algorithm. I would like to fully incorporate this Maze into my Dialogue System showcase. I would do this to show how dialogue can be changed during the game also as it runs. Simply put I would have a set of characters inside of the maze and each one of them would tell the player the next few directions they must take. Since the Maze is randomly generated this would mean that the dialogue for those particular characters needs to change during runtime of the scene. My current Dialogue System allows for this already, but I simply would not have enough time to fully implement this example now and I feel like it is imperative to include this feature as part of future work.

# References

**Books**

Millington, I. (2019) “Chapter 8: Procedural Content Generation,” in AI for games, Third edition. Boca Raton, Florida: CRC Press, pp. 705–726.

**Articles**

Mikkelson, N. (2020) *Game Writing: Structuring Branching Dialogue*, Indie Writing. Available at: https://indiegamewriting.com/writing-1-structuring-branching-dialogue/ (Accessed: January 10, 2023).

Merdivan, E. *et al.* (2019) *Dialogue Systems for Intelligent Human Computer Interactions*, *Science Direct*. Available at: https://www.sciencedirect.com/science/article/pii/S1571066119300106?via%3Dihub (Accessed: January 11, 2023).

Liemhetcharat, A. (2021) *Unity: Reading external XML files*, *GitConnected*. Level Up Coding. Available at: https://levelup.gitconnected.com/unity-reading-external-xml-files-ed199df66288 (Accessed: January 25, 2023).

Philipp, M. (2015) “How To Read An XML File In Unity,” Studica, 15 December. Available at: https://blog.studica.com/read-xml-file-in-unity (Accessed: January 30, 2023).

**Videos**

Board To Bits Games. (2016) Saving Data in Unity: XML Files, [video online]. Available at: https://www.youtube.com/watch?v=6vl1IYMpwVQ&ab\_channel=BoardToBitsGames (Accessed: January 27, 2023).

**Websites**

Grasshop Dev (2020) *Dialogue Editor*, *Unity Asset Store*. Available at: https://assetstore.unity.com/packages/tools/utilities/dialogue-editor-168329 (Accessed: February 10, 2023).

*ChatMapper* (2010) *Chat Mapper AI*. Available at: https://www.chatmapper.com/ (Accessed: February 12, 2023).

*PixelCrushers* (2018) *Dialogue system for unity*, *Pixel Crushers*. Available at: https://www.pixelcrushers.com/dialogue-system/ (Accessed: February 12, 2023).

# Appendices