

Computer Games Development SE607

Technical Design Document

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

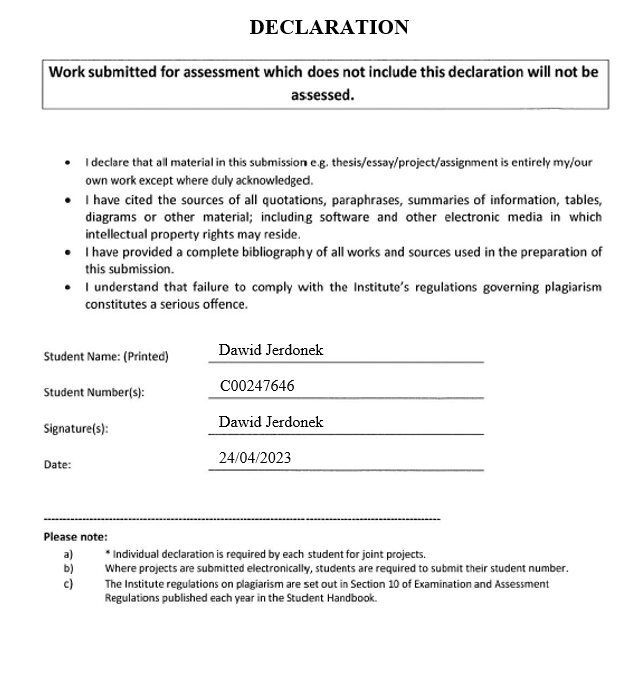
Bachelor of Science in Games Development

at the

South East Technological University

May 2023

© South East Technological University 2023. All rights reserved.



Contents

[Acknowledgements 3](#_Toc133432769)

[Technical Design 3](#_Toc133432770)

[References 15](#_Toc133432771)

# 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.

# Technical Design

“Dialogue Tale” is a unity project containing features allowing users to create, write, edit, delete, load, and save dialogues for characters.

The project utilises lists and parsing to store and process XML files and then assign them correctly. The Dialogue System stores these parsed lists and assigns the correct dialogues to corresponding characters. The Dialogue Editor is a simple environment in which users can access these dialogues in a visual manner and edit them to their liking. They can then save the edited dialogues to their disk. Once a user has edited dialogue, they can test the capabilities of the Dialogue System with the “Template” scene which loads dialogue and assigns it to one of the three sample characters. The player can then test the dialogue tree by pressing ‘P’ or ‘L’ to choose a reply to what the character is saying, if applicable.

**Technologies**

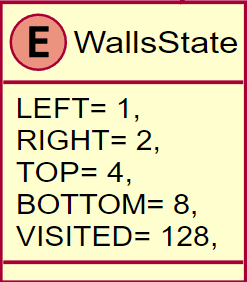
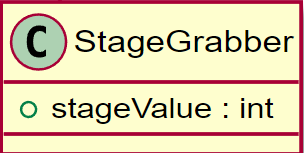
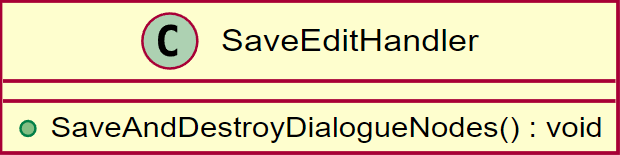
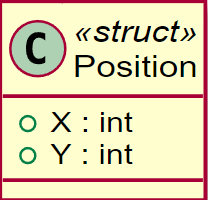
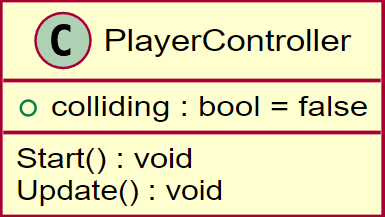
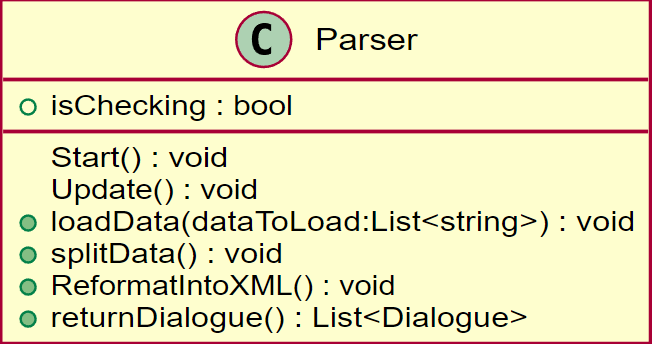
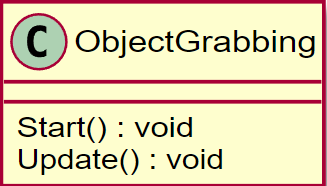
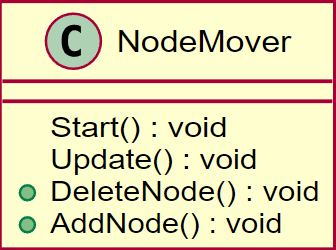
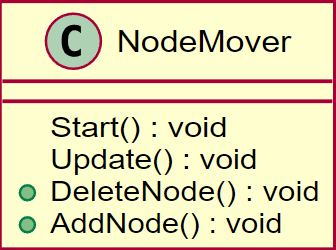
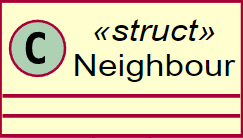
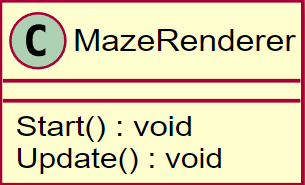
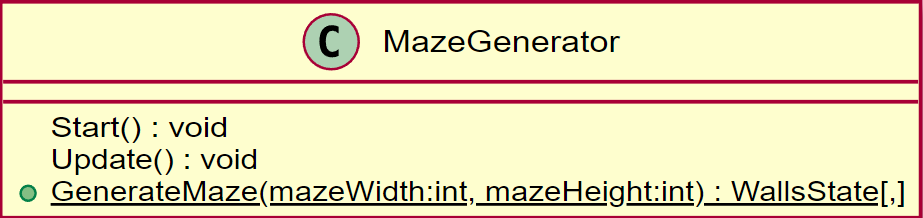
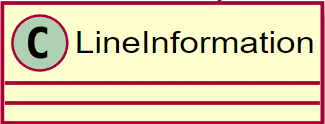
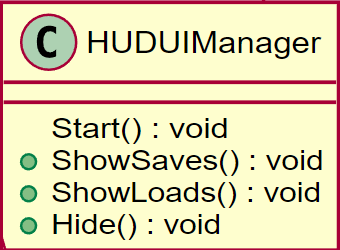
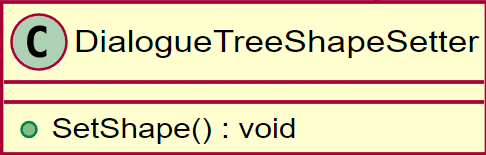
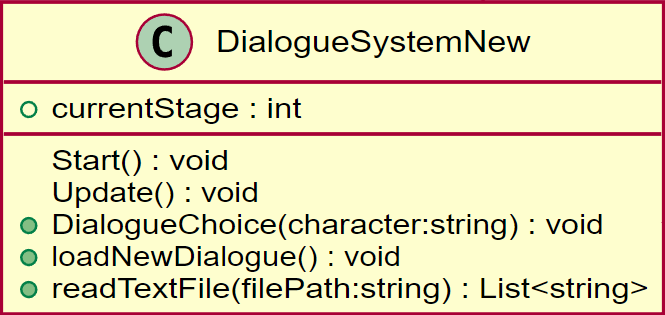
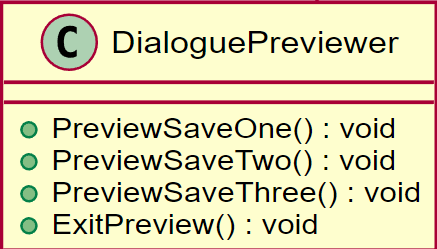
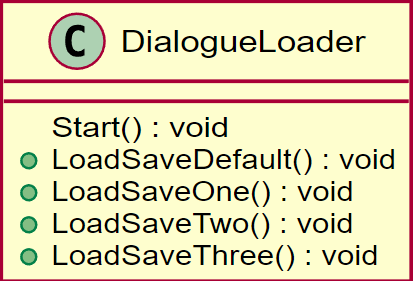
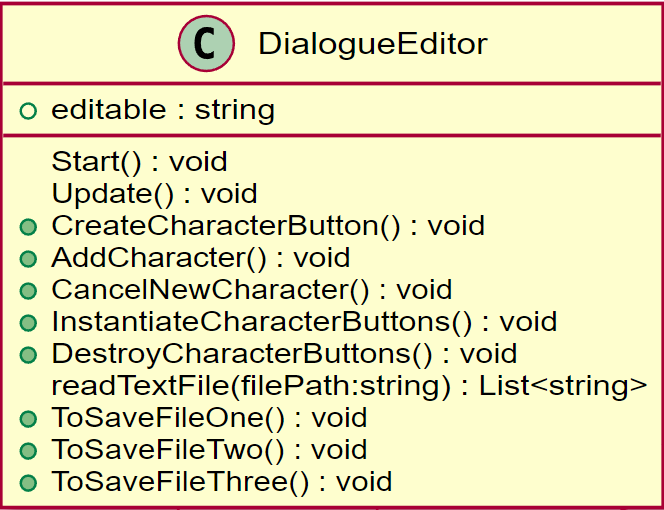
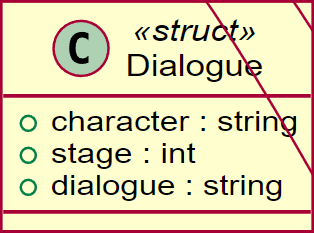
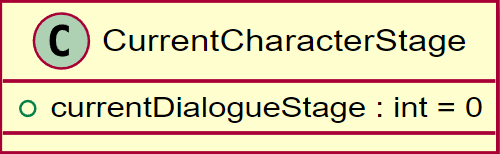
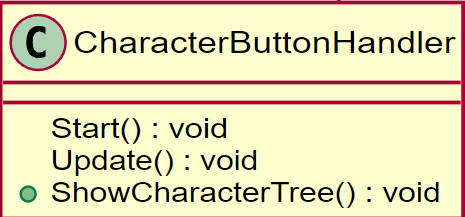
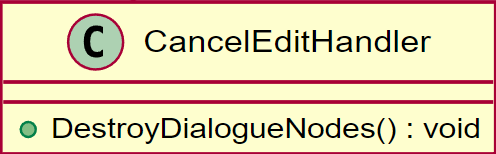
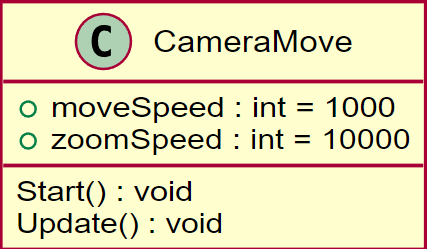
* **C#**
* **Unity Engine**
* **Oculus Quest 2**
* **XML**

The project is written in C# and is used and ran within the Unity Engine in its built-in editor window. This is the case as the idea of the project was to create a Dialogue Editor which simplifies the creation of dialogue for Unity. Oculus Quest 2 is the Virtual Reality headset I used for the setup and playability of the “Template” scene for testing however everything can still be tested from within the Unity Editor. The form of dialogue storage on disk used in this project is XML file. An XML file has great syntax and can be formatted easily for the uses and needs of a dialogue system. With the use of a custom Parser the project removes any syntax when loading the XML file into Unity to create dialogues.

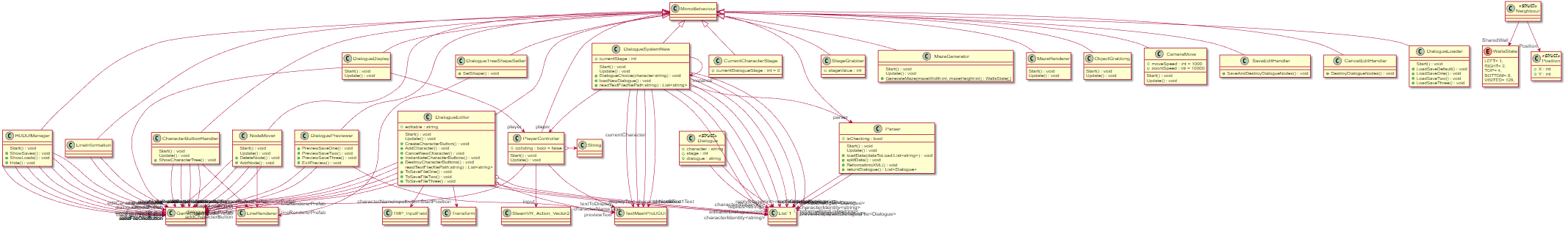
**Features**

* **Dialogue Editor**
* **Dialogue System**
* **Dialogue Parser**
* **Maze Generation**
* **Virtual Reality Integration**

**Class Diagrams**



**UML Diagram**



**CRC Cards**

|  |  |
| --- | --- |
| **CameraMove** | |
| **Responsibilities** | **Collaborators** |
| * Move camera in four directions (UP, DOWN, LEFT, RIGHT) * Zoom camera in and out in Dialogue Editor * Have boundaries set up so camera does not go too far from Editor |  |

|  |  |
| --- | --- |
| **CancelEditHandler** | |
| **Responsibilities** | **Collaborators** |
| * Cancel the changes made to a dialogue tree. * Destroy all node of the dialoge tree. * Destroy all related buttons to dialogue tree. | * DialogueEditor * SaveEditHandler |

|  |  |
| --- | --- |
| **CharacterButtonHandler** | |
| **Responsibilities** | **Collaborators** |
| * Spawn in all nodes related to a character when a character button is pressed. * Setup lines from reply nodes to dialogue nodes. * Assign all relevant data to each node in the dialogue tree. | * DialogueSystemNew * DialogueEditor * StageGrabber * DialogueTreeShapeSetter * LineInformation |

|  |  |
| --- | --- |
| **CurrentCharacterStage** | |
| **Responsibilities** | **Collaborators** |
| * Stores the current stage of the character dialogue |  |

|  |  |
| --- | --- |
| **DialogueDisplay** | |
| **Responsibilities** | **Collaborators** |
| * Sets the correct character name on HUD for dialogue | * PlayerController |

|  |  |
| --- | --- |
| **DialogueEditor** | |
| **Responsibilities** | **Collaborators** |
| * Create new characters for dialogue tree * Cancel the creation of a new character * Instantiate buttons for choice of which character to edit * Destroy the buttons for choice of which character to edit * Read in the XML text file and store it * Save parsed dialogue to different XML files | * DialogueSystemNew * DialogueLoader * Dialogue |

|  |  |
| --- | --- |
| **DialogueLoader** | |
| **Responsibilities** | **Collaborators** |
| * Load dialogue from custom saves or from default save | * DialogueSystemNew |

|  |  |
| --- | --- |
| **DialoguePreviewer** | |
| **Responsibilities** | **Collaborators** |
| * Load a dialogue XML file into a preview to show dialogue file before it is loaded and used * Enter the preview screen * Exit the preview screen | * DialogueSystemNew |

|  |  |
| --- | --- |
| **DialogueSystemNew** | |
| **Responsibilities** | **Collaborators** |
| * Load XML file from disk and pass it into parser * Read from the XML file * Setup a list of all character identities based on characters in the dialogue file. * Display the correct current character dialogue in HUD * Set up and handle dialogue choices of player | * PlayerController * Parser * Dialogue |

|  |  |
| --- | --- |
| **DialogueTreeShapeSetter** | |
| **Responsibilities** | **Collaborators** |
| * Finalise a dialogue tree shape and node positions. * Draw remaining lines from reply nodes to dialogue nodes. | * DialogueSystemNew * LineInformation * StageGrabber |

|  |  |
| --- | --- |
| **HUDUIManager** | |
| **Responsibilities** | **Collaborators** |
| * Display save, load, and cancel buttons * Disable save, load, and cancel buttons |  |

|  |  |
| --- | --- |
| **LineInformation** | |
| **Responsibilities** | **Collaborators** |
| * Store start and end position of line. * Update line positions if nodes are moved |  |

|  |  |
| --- | --- |
| **MazeGenerator** | |
| **Responsibilities** | **Collaborators** |
| * Stores Positions, Walls states and neighbours * Run Recursive Backtracking algorithm to create procedural maze. |  |

|  |  |
| --- | --- |
| **MazeRenderer** | |
| **Responsibilities** | **Collaborators** |
| * Populate maze plane by creating a grid from wall prefabs * Specify maze size | * MazeGeneretor |

|  |  |
| --- | --- |
| **NodeMover** | |
| **Responsibilities** | **Collaborators** |
| * Drag nodes around in Dialogue Editor * Delete nodes from dialogue tree * Add nodes to dialogue tree * Update parsed data with new nodes | * StageGrabber * Dialogue * DialogueEditor * DialogueSystemNew * LineInformation |

|  |  |
| --- | --- |
| **ObjectGrabbing** | |
| **Responsibilities** | **Collaborators** |
| * Apply Physics to objects grabbed in VR |  |

|  |  |
| --- | --- |
| **Parser** | |
| **Responsibilities** | **Collaborators** |
| * Split data into dialogues by getting rid of syntax from XML * Create list of parsed dialogues * Reformat parsed dialogues back into XML format with correct syntax | * Dialogue |

|  |  |
| --- | --- |
| **PlayerController** | |
| **Responsibilities** | **Collaborators** |
| * Display and hide HUD for dialogues * Move player * Check for trigger collisions |  |

|  |  |
| --- | --- |
| **SaveEditHandler** | |
| **Responsibilities** | **Collaborators** |
| * Save edited dialogue trees to parsed data * Destroy the dialogue tree on display | * Dialogue * DialogueEditor * DialogueSystemNew * CancelEditHandler |

|  |  |
| --- | --- |
| **StageGrabber** | |
| **Responsibilities** | **Collaborators** |
| * Store current node stage value |  |

# 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).