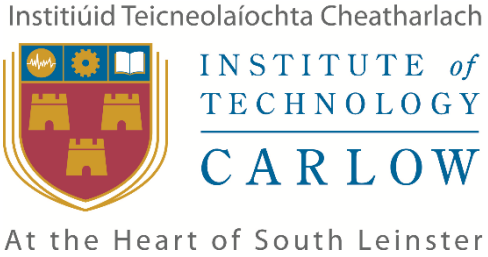

Computer Games Development Project Report Year IV

Patrick Donnelly
C00236160

10/11/2022

DECLARATION

Work submitted for assessment which does not include this declaration will not be assessed.

- I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.
- I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.
- I have provided a complete bibliography of all works and sources used in the preparation of this submission.
- I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: (Printed) Patrick Donnelly

Student Number(s): C00236160

Signature(s): *Patrick Donnelly*

Date: 24/04/2023

Please note:

a) * Individual declaration is required by each student for joint projects.
b) Where projects are submitted electronically, students are required to submit their student number.
c) The Institute regulations on plagiarism are set out in Section 10 of Examination and Assessment Regulations published each year in the Student Handbook.

Contents

Acknowledgements

I would like to thank my supervisor Martin Harrigan for assisting me in structuring my workflow and managing my project milestones. I would also like to thank him for the advice and direction he provided throughout my project.

I would like to thank my lecturers from SETU for providing me with the materials necessary to develop my skills over my 4 years at SETU.

Project Abstract

(Removed needs Rewording)

Project Introduction and/or Research Question

(Removed needs Rewording)

Literature Review

(To Do: Need analyse my notes from my research and put in only appropriate research)

Replace this text with an appropriate Literature Review.

The literature review places your research in context. You aren't the first person to investigate or research a particular topic. Present a short literature review with the following goals:

- Give the reader a good overview of the key concepts;
- Describe the most relevant work (in your own words) that other people have done in this area;
- Use proper academic writing with references.
- Show how the existing work influenced your project.

Evaluation and Discussion

(To Do: Check if enough data was collected from play tests, analyse and review, ask supervisor if it's appropriate to put here as well as what else is expected from this section)

Replace this text with Results and Discussion.

Describe the results using diagrams such as graphs etc. as appropriate, and discuss what the results mean.

Example: Results indicate that once the threshold gets over a certain point it significantly reduces player performance and player experience

Project Milestones

Other project commitments are the reason for the discrepancies in my sprint scheduling, particularly in February where all my time was dedicated to a month-long group project.

Sprint 1:

Date: 25/11/2022 TO 01 /12/2022

Objectives:

- Create Player.
- Start Check Collision.
- Start Check Room Validity.
- Create a grid that can be resized.

Sprint 2:

Date: 02/12/2022 TO 09 /12/2022

Objectives:

- Create Player movement.
- Create and assign Colliders to game objects.
- Create a method to validate rooms.
- Create buttons to use as UI in the game.
- Create a method to clear the grid of all objects.
- Create a method to assign colliders to sets of walls instead of individual wall tiles.
- Create a class that handles collision between game objects (AABB collision).
- Handle collision between player and walls.
- Handle collision between player and obstacles.
- Animate the player.
- Create a method to place floor tiles automatically in a valid room.
- Display to the user which wall tiles are invalid when they try to generate an invalid room.
- Create a class for obstacles and allow the user to place them on the grid.
- Give the player an attack.
- Check collision between the players weapon and obstacles and destroy obstacles on contact.

Sprint 3:

Date: 13/01/2023 TO 19 /01/2023

Objectives:

- Tag system.
- Level loading and saving. Investigate file formats, and third-party libraries (YAML, csv, JSON etc.)
- Add buttons to the UI for changing grid size, items, terrain, decorations etc.
- Update Player Collision with other objects.
- Add the ability to delete placed objects.
- Add Game States.
- Allow cleared cells to have objects placed in them again.
- Add a build phase and test phase.

Sprint 4:

Date: 20/01/2023 TO 27 /01/2023

Objectives:

- Expand on palette of entities you can place.
- Reset object positions when going from the test phase back to the build phase.
- Allow grid and colliders to be toggled on and off. Create UI buttons to handle these functions.
- Create Ui Buttons for each object type and each object that can be placed.
- Manage how placed objects are stored.
- Automate the UI for placeable objects so sprites placed in folders in the project repo can be read in automatically and have buttons assigned automatically as part of the UI.
- Create Categories of objects and create tab states to keep track of which category is selected for object placement.
- Create rows of selectable objects and create UI buttons to navigate through rows of objects.

Sprint 5:

Date: 27/01/2023 TO 03 /02/2023

Objectives:

- Create a dialogue box.
- Allow text to be read in from the user and displayed on inside the dialogue box.
- Allow text to be deleted and removed from the dialogue box at run time.

Sprint 6:
Date: 03/03/2023 TO 09 /03/2023

Objectives:

- Create a text editor GUI for the input and output of text.
- Wrap text so that it fits the dialogue box regardless of length.

Sprint 7:

Date: 10/03/2023 TO 16/03/2023

Objectives:

- Create an inspector to attach pieces of text to triggers. Decide on what the GUI for the inspector will look like.
- Finish GUI for the text editor.
- Create an input field class to handle the input text for the body of the dialogue and the name (title) of the file.
- Allow the text to be saved to a text file with the text from the body being the dialogue to display and the text from the header being the name of the saved file.
- Allow saved text files to be loaded in.
- Allow loaded in text files to be edited.
- Display all saved text files in a list to the side of the editor.
- Update the list every time a text file is saved.
- Create pop ups to make the user aware if a file exists if they try to overwrite it.
- Create a button that will clear the text in the input fields.

Sprint 8:

Date: 17/03/2023 TO 23 /03/2023

Objectives:

- Allow text files to be completely deleted from inside the editor.
- Create a checkbox class. In this instance use a checkbox to preview any input text from the text editor in a dialogue box. Toggle on and off.
- Add multiline input fields.
- Create the GUI for an inspector.

Sprint 9:

Date: 24/03/2023 TO 30/03/2023

Objectives:

- Split placeable objects into their own classes.
- Create a texture manager to manage the textures being loaded in, to prevent duplicate textures in memory.
- Create a font manager.
- Allow an object to be selected by double clicking.
- Display the inspector of a selected object.
- Allow objects to be relocated once selected.
- Create GUI that will allow dialogue to be attached in the inspector.

Sprint 10:

Date: 31/03/2023 TO 06 /04/2023

Objectives:

- Create a drop-down menu for the inspector.
- Create an in-range condition that can be set in the inspector that will trigger dialogue if the player is within range of an NPC that has this enabled.
- Create an interacted condition that can be set in the inspector that if enabled will trigger dialogue any time the player interacts with an NPC.
- Allow dialogue to be loaded in and saved to a selected object.
- Add a main menu.
- Install YAML.
- Save Data to a YAML File
- Load data from a YAML File
- Add the ability to scroll the screen based on the mouse position.

Sprint 11:

Date: 07/04/2023 TO 13 /04/2023

Objectives:

- Increase Grid Size.
- Update YAML to save data for the increased grid size if necessary.
- Create a separate view for the game scene and User Interface.
- Add zoom in and zoom out functionality.
- Anchor UI regardless of game world position.
- Add functionality to highlight an area and fill that highlighted area with the currently selected object (multi object placement).
- Create a save game GUI where the user can enter their game name.
- Create folders at runtime to store all game data.
- Create pop ups for when the user is about to overwrite existing game data, when a game is saved successfully and when they try to save a game with no name entered.
- Add a colour picker so the user can customise their game names colour for display on a game menu at a later time.
- Add text formatting like italics and underlining for display on a game menu at a later time.

Sprint 12:

Date: 14/04/2023 TO 20 /04/2023

Objectives:

- Update data being saved in YAML files.
- Update Game list at runtime every time a new game is saved.
- Make the view follow the player's movement.
- Allow colliders to be placed independently of objects.
- Allow terrain to be deleted.
- Allow multi object deletion.
- Adjust UI.
- Various bug fixes.
- Develop a website to showcase the project.
- Create and edit a Video to showcase the projects features.

Major Technical Achievements

(Not sure on this – ask supervisor)

So far, I have managed to create a no code game editor that allows the user to build, save and load their games without writing a single line of code. This is achieved through a friendly user interface and pre-programmed game components and entities that the user can click on in the UI View and place onto a grid in the Game View.

Game Building:

Users can create a game world from the ground up. Various types of objects such as terrain, walls, enemies, items and more can be placed, to bring the users vision to life.

Game Saving:

Users can save their game as they build, resuming the build at any time they wish.
Data is stored in directories containing unique YAML files that are created at runtime.

Game Loading:

Users can load their creations to continue editing or to play through their creations at the click of a button.

Built in text editor:

Users can create dialogue scripts within the built in Text Editor. These scripts can be attached to NPCS and Enemies through the inspector to create narrative experiences in their games.

Dynamic Loading:

User created dialogue and games are instantly assigned UI buttons so that the player can access their files as soon as they are saved without having to restart the software.

Sprites placed in the Project Directory are automatically loaded in and given Buttons in the GUI so that instances of those sprites can be placed on the grid as objects.

QoL Features:

Features such as multi-object placement and deletion exist to make the user experience in building games as quick as possible. Users can place or delete multiple objects by highlighting an area in the Game View. Pop ups will display in areas where the user might risk losing or overwriting data like on the Save Game screen.

Project Review

Although I am happy with the progress I have made to date on the No-Code Game editor I freely acknowledge that there is much more work to be done to make it live up to the experience I envisioned.

The editor itself in terms of design and user interface has come together well.

I however acknowledge that it lacks essential features that I had planned to implement but due to time constraints fell beyond the scope of the project.

In hindsight I would have liked to use an external library such as “ImGui” to implement the UI, and I would have pushed harder for permission to do so. Not only would this have perfectly suited my vision for simple intuitive UI design, but it also would have allowed me to spend time developing gameplay features that would have enhanced the creative experience of my editor and greatly increased the quality of the products users could design within my editor. As things stand it would be fair to say that approximately a third (possibly more) of my project time was spent on creating a simple User Interface that functioned as expected from scratch. Time, that I think, could have been better spent focusing on gameplay elements that the user could incorporate into their games.

I also think I should have increased my initial research phase before I began implementing code. I believe if I had done so I would have realised an ECS design would have perfectly fitted my project. I think this would have allowed my editor to be more flexible and efficient. As well as this I think implementing an ECS design would have cut development time and made the structure of my written code easier to navigate and behaviour more readily modifiable at runtime.

In retrospect I should have spent more time researching basic game engines as I believe a no code game editor shares similar methodologies in terms of its aesthetic and how certain functionality could be implemented. Examples of this would be UUIDS and Serialisation.

I would have liked to have given my research and design phase greater focus as I may have been able to better foresee some of the delays and many complications I encountered.

I adopted a can-do attitude throughout the process that I think worked to my detriment in the long run. I allowed the scope of my project to go beyond the realities of what could be achieved within the allocated timeframe, possibly not fully understanding the measure of time I would have to invest in projects that ran in tandem with this one. I focused on features that I now see were less important to the overall project but took a lot of time to develop like the text editor. Although this is a neat feature it should not have taken priority over gameplay mechanics especially due to the fact that it took a month to develop without altering the overall experience of the software much. Gameplay mechanics like enemy behaviour, weapons and unique items would have made more of an impact with a shorter turnover time in terms of development.

Conclusions

(Removed needs further reflection)

Future Work

(Work In Progress)

I have a lot of ideas I would like to build on in relation to my ***No-Code Game Editor***, some of which are implemented to a certain degree already, albeit still a work in progress. I think with further development this No-Code Game Editor could be a robust piece of software that could be used to showcase the flexibility of C++ and SFML.

Dynamic Behaviour:

I would have liked to have implemented the ability to add unique behaviour to Entities. The ability to determine an enemy's behaviour at run time would have allowed users to assign different types of AI behaviour to them such as the ability to pursue the player, find a path to the player or even flee.

Multiple Grid Layers:

Allowing the user to add multiple layers to their Game would allow them to create bigger game worlds. They could separate interiors from an overworld layer to manage their game world more efficiently.

Individual Menu Customisation:

Giving the player the ability to customise a menu for their game would add to the game feel and the quality of the game’s users create.

Dynamic Audio:

Allowing the user to customise audio on various aspects of the games they create such as background tracks and overworld sounds would help with immersion and the uniqueness of a game. It would also serve as a nice introduction to sound design, a key element in games.

Gameplay Mechanics:

Implementing more weapon types or interactive objects like keys, power ups and switches would allow for creative scenarios unique to each game.

Tutorial System:

A tutorial system would really benefit an editor like this, as the goal is to introduce a non-technical user to game design without introducing skill barriers, one which explains components through pop ups and prompts as the player navigates the software would be the most user friendly.

Entity Component System:

Serialisation:

References

(Do not forget insert references for the literature review)

Appendices

Replace this text with Appendices.

This might include ethics application and other relevant material e.g. copy of any questionnaires used.

(If playtesting data used insert google docs)