SETAP

December 16, 2022

Bradley Edmondson - UP2015336, Imran Mulindwa - UP2071998, Hassan Alghamdi - UP2092245, Fahad Al Rakan - UP2092234, Jude Whiting - UP2065429,

Contents

1 Introduction

We are making a top down 2d survival game.

2 Problem Specification

2.1 Eliciting requirements

To elicit requirements we have used interviews and questionnaires. Our target age group is that of people 15 years to 30 years old. This is because this reflects the demographic of people who play games within this genre. Of course while people outside of this group can play the game. The likelyhood of them doing so is lesser

Our interviews were a discussion with 9 people. these ranged in age from 15 to 21. People had the abiliy to discuss within the answers and expand on their views even if it was not fully related. our questions were

1. What kind of games do you play?
2. What are your thoughts on open-world games?
3. What platform do you play games on?
4. If you play games for an end goal, what kind of end goal are you looking for?
5. Are you more into singleplayer or multiplayer games?
6. What features do you like in the games you play?

2.2 Interviews

In this section I include the data we got from interviews.

Age and Gender

- P1: 20, male
- P2: 19, female
- P3: 19, male
- P4: 20, male
- P5: 19, female
- P6: 20, Male
- P7: 19, Male
- P8: 20, Male

What platforms do you play on?

- P6: PlayStation, PC
- P7: PlayStation
- P8: PlayStation

What kind of games do you play?

- P1: Fifa, Call of Duty
- P2: Life simulation, Roleplay, Building eg Sims, Minecraft, Stardew Valley
- P3: Single player action adventure games, eg God of War, Resident Evil
- P4: Party games, Mario Kart, Overcooked, Roguelikes
- P5: Minecraft, Sims, Dead by Daylight, story based indie games
- P6: Single player, Open-world e.g. God of War, Grand Theft Auto, Darksiders
- P7: Sandbox, Action and Adventure
- P8: Sports

What features do you like in the games you currently play?

- P1: Receiving strong rewards for doing well or getting lucky
- P2: Building, character customisation
- P3: Good story, world design, rpg system e.g. armour and stat customisation
- P4: It's just lots of fun when you're in the same room as someone playing a game together, with lots of excitement and emotions.
- P5: The story, character customisation
- P6: Hack 'n' slash, variety of enemies, quicktime events, range of difficulty
- P7: Graphics design
- P8: User able to make decisions regarding the game

What are your thoughts on open-world games?

- P1: Played them when I was younger,I played minecraft and terraria
- P2: Like them, think they're fun
- P3: Can be pretty good, sometimes lack detail

How difficult do you like your games to be?

- P1: Hard, because you get a stronger sense of accomplishment when you overcome the challenge
- P2: On the easier side
- P3: A good balance (fairly challenging)
- P4: A bit hard - hard
- P5: Challenging
- P6: Very hard

Would you like it if a game heavily punished you for making a mistake?

- P1: Completely restarting when you die is too much, keeping some items and abilities is fine
- P2: No, roguelike style still too much
- P3: As long as you don't get killed for a BS reason, then losing everything and restarting is fine.
- P4: Keeping some items and abilities when you die seems fair
- P6: Yes, it makes you work harder and develop a mental strategy for each fight
- P7: No because all of your achievements playing the game will be lost
- P8: No because you will have to replay the game all of over again

2.3 Requirements

These requirements are garnered from our interviews 2.2. Our main line is our user requirements with each of them broken down into a set of system requirements.

These are ordered in the priority of these tasks. The first ones being the most important, the latter being the least. This is both for the user requirements and the system requirements within them.

Users should be able to play the game on multiple operating systems

- Users with different operating systems like windows, Linux will gain access to the features that are provided by the application
- The system should be easily installable on platforms that it is compatible from different operating systems e.g., windows, Linux

- When the system is in use, it shouldn't take more than 3 minutes to load data and should respond to gameplay commands such as movement.
- Users who have different platforms other than Windows or Linux, the system may not be accessible.
- The game should not crash, instead the error should be handled and an error message shown to the user.
- Users should find the system intuitive to use. Using long standard practices from the game industry such as movement commands and so on.
- The system should not take more than 10 seconds to launch the game.

Users should be able to explore a procedurally generated map.

- The map size will be 100 x 100 and include 10,000 tiles in total.
- Each tile should be able to carry 1 item and character (user or NPC).
- The player will move by moving from tile to tile.
- Users will spawn in the middle of the map at point (50, 50).
- The map will be modelled as an object.

Users should be able to interact with NPCs.

- Interactions will be both hostile and passive depending on the type of NPC.
- Each NPC will be modelled as an object that interacts with the user.
- NPCs will drop items when killed, which the user should be able to collect.
- There will be passive NPCs that can sell you stuff using a bartering system.

Users should be able to save their progress into a save file.

- Will produce a save file for storing, maintaining and accessing data
- The file should store the world map as a 2D array.
- The file should store the user's current inventory and that of any chests they have placed.
- The file should store the user's position and the position of any spawned NPCs or items.
- The file should store the user's current points acquired and stats (health, speed etc.).
- A save file for the current game will be saved up to every 10 minutes.

Users should be able to collect resources.

- Resources will be represented as different assets on the games stage.
- Users will have to manually collect items, by pressing the letter e key.
- Each resource will add to the point system.
- They will be modelled internally as objects.
- Rarity will be given as an attribute to give the object its points value.

Users should be able to generate points.

- Points will feed into a point system which will generate highscores.
- Users will gather points from collecting resources, killing NPCs, crafting items etc.
- Points will improve a user's stats (health, speed etc.)
- Each point collected must be shown to the user within 2 seconds.
- The system should provide users of the total points collected on each session.
- The system shall measure the time of each session and show it at the end of the session.

Users should be able to use weapons.

- Weapons will be found or crafted by the user.
- Weapons will deal a set amount of damage.
- Different weapons should deal different amounts of damage.

Users should be able to use a simple crafting system.

- Users will craft new items using items from their inventory that they have collected.
- Crafting recipes will show up depending on what's in a user's inventory.
- An item's rarity should change when item's of different rarities are crafted together.

Users should be able to play together using local multiplayer.

- A game should be able to be set as joinable or not when it is started.
- Up to 4 users should be able to connect to the same game.
- The system should be scalable enough to support multiple users, on the same network, at the same time while maintaining optimal performance outside of constraints we cannot control such as users hardware and networks.
- Users should identify during multiplayer games, with a name they provide before a connection is created.
- The game should not crash under the pressure of many users playing.
- TODO: mention protocol

Users should be able to use a chest system.

- Chests should be able to be crafted and placed into the world by users.
- A user should be able to view all the items that are in a chest.
- Items will be able to move between the user's inventory and the chest.

Users should be able to select difficulty

- Enemies will be stronger the harder the difficulty is
- Resources may be harder to find
- Crafting may require more resources
- Difficulty should be selected when a new game is created.

The System should be accessible

- The application will have features that match the geographical location of the users i.e. languages, time zones. This will give the users opportunity to choose which geographical location is the best to use
- The game must be engaging, with users responding back telling us how they enjoyed what they can and can't do.
- The game must support different colour spectrums to support colour blind people
- This will be activated in a small options menu

3 Design

3.1 Use cases

3.1.1 Use Case: Enter Username

- Actor: User
- Description: When playing on a multiplayer game a user attempts to enter a username to go by on the server.
- Pre-Condition: User has attempted to create or join a multiplayer game.
- Post-Condition: The user joins the game with their entered username.
- Data: Input: User's username / Output: Game loads/Error message
- Valid Case: User loads an input text box and a message asking them to input a username. The user inputs a username and presses enter. The user is then sent to the game loading screen.
- Error Case: Username not valid: If the user name is invalid an error message will be shown to the user and they will be sent back to the enter username screen.

3.1.2 Use Case: New Game

- Actor: User
- Description: A user attempts to create a new game.
- Pre-Condition: User has loaded up the game and has selected 'New Game'.
- Post-Condition: The game has been created and the user is loading into the game.
- Data: Input: Game name / Output: Game loads/Error message
- Valid Case: User loads an input box to enter a name for that game to be saved under. The user enters a name and presses enter. The user is then sent to the game loading screen.
- Error Case: Name is invalid: If the name is invalid an error message will be shown to the user and they will be sent back to the enter name screen.

3.1.3 Use Case: Load Game

- Actor: User
- Description: A user attempts to load a previously created and saved game.
- Pre-Condition: User has loaded up the game and has selected 'Load Game'.

- Post-Condition: The user is loaded into the game at the point of where they last saved that game.
- Data: Input: Selecting a game to load / Output: Game Loads/Error Message
- Valid Case: User has been presented with a list of saved games and chooses the correct one to load. The user is then sent to the game loading screen.
- Error Case: File corruption: A save file could corrupt and then an error message would be sent to the user and they would be sent back to the menu screen.

3.1.4 Use Case: Save Game

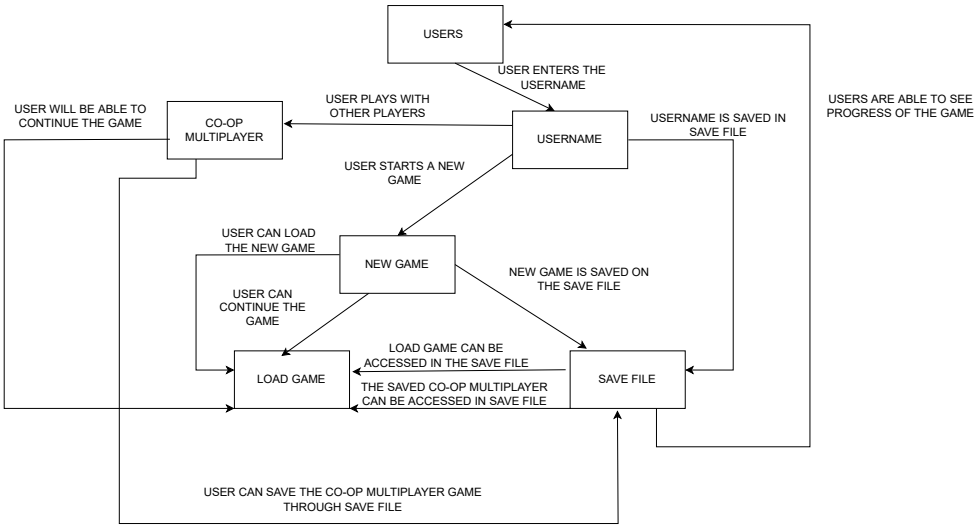
- Actor: User
- Description: A user attempts to save their current progress in their game.
- Pre-Condition: User has clicked the button on the screen titled 'Save Game'.
- Post-Condition: All the data about the current game is stored into a save file, overwriting any previous save data.
- Data: Input: Pressing the save button / Output: Success/Error Message
- Valid Case: The game freezes while the game saves and then the user is given a success message to say the game has saved. Then the game resumes as usual.
- Error Case: Saving was interrupted, the user is given an error message and told to retry. This then resumes the game and the user can retry saving it.

3.1.5 Use Case: Join Game

- Actor: User
- Description: A user attempts to join a multiplayer game that another user created.
- Pre-Condition: User has loaded up the game and selected 'Multiplayer'.
- Post-Condition: The user is told to enter a username.
- Data: Input: Selecting a game to join / Output: Joins game/Error message
- Valid Case: The user is taken to the enter username page and once that is completed, successfully loads into the game along with the other players.

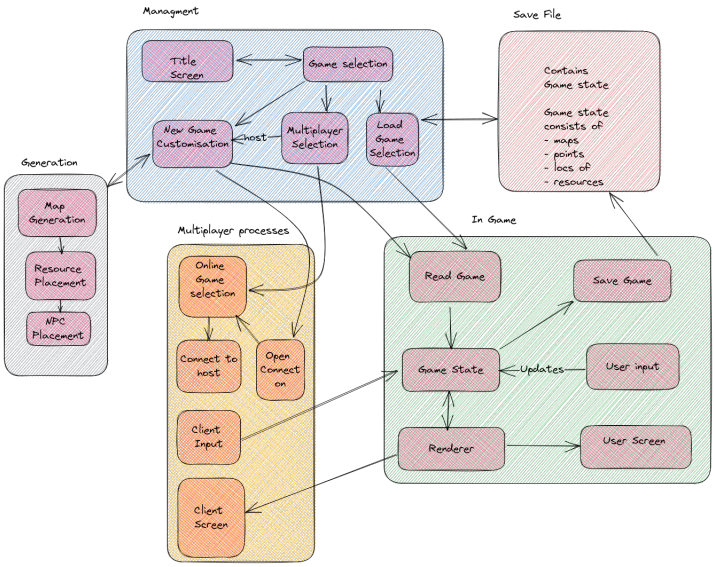
- Error Case: Network connection error, the user has received a network error and is presented with an error message and told to retry. This then places at the user on the main menu/

3.1.6 Overall Use cases



Users: This use case is described as people who will be involved in playing the game. username : This is the procedure that allows users to gain access to the system. When users have entered their username, they will be able to acquire the features of the system like new game, load game etc. in the login, users won't have to re-enter their usernames since they are saved in the save file New Game: This use case allows users to start the game from the beginning therefore anyone who plays the 2D game from the beginning will be able to understand how the game works quickly. Users will be able to continue the game through load game and the game will always be saved in the save file Load Game: This use case will allow users to continue the game where they have previously left off. So, users will have the opportunity to not start the game again. The load game will be got from the save and accessed Save File: This use case allows the user to save the whole content of data through gameplay of the user. It also shows where the user stopped playing within the game. The save file will also contain the co-op multiplayer Co-op multiplayer: users will be able to play with other users and always continue where they have left off from the load game when they save their game in the save file. Once, they continue the game they will not lose their previous achievements except when they didn't save the game

3.2 Architecture



Our architecture consists of 5 main parts. These include

- The managment of the game: section which manages the loading of games. It contains the selction menus and guides the users towards creating or loading a new game
- Game generation: This creates the datastructures that will be read by the game state which then runs the game
- The game itself which will read the game save into the game state, as well as update the game state based on user input this gets passed into the renderer which renders it to the user screen
- the save file which stores the data, this will perodically be re saved.
- The multiplayer processing, this encapsulates all of the multiplayer game processing.

4 Implementation

Our implementation will be completed in the Lua language and the love game engine. We chose these tools as:

- Lua is a small and simple language to learn. Being similar to python while having many differing uses from python
- Love is a simple game engine, focusing in on 2d game development and having a small and simple interface while still being able to express larger and more complex games

We will be using git and github for our code management.

- Issues will be created to discuss features in implementation
- Pull requests are used to implement and test features

5 Testing

Testing will mainly consist of unit testing. This allows us to test all of our functions in isolation and check that we do not break compatibility within revisions to each prototype. We will test against our non-functional and functional requirements after every prototype iteration.

The requirements will be individually marked as either completed or missing or will have a description of the level of success if the requirement is only partially met.

We will also test the efficiency of our non-functional requirements where we can, and give a mean average of the time taken to run a function for every prototype to help track efficiency improvements throughout the project cycle.

This will be followed by integration testing and then eventually user testing, where we will ask fellow students to play each iteration of the prototype, and give us feedback on what areas they think need improving and what features they think are missing from the game. This will be important as it will provide us with an unbiased opinion on our project.

6 Critical Analysis

6.1 Leadership

Our methodology is Scrum. This will involve us assiging tasks into week long sprints. These tasks will be tracked on a Kanban board hosted on github

We will have the roles of A scrum master and then scrum workers. The former will delegate the tasks and monitor the progress of these tasks, with how these tasks progressing being decided at the end of the sprint during our next meeting.

The Scrum master will be voted in during our weekly meetings. Everyone will have a chance too but people will submit themselves. If no one does then we will select the next person to not be scrum leader.

6.2 Monitoring

All tasks will be out on a monitoring board with a SCRUM Master monitoring progress throughout the week Tasks are assigned at the beginning of each week. As tasks are finished they will be move into a different section of the monitoring board If a task is not finished during a sprint, then either more people will be assigned to it, it will get continued over to the next week or it will be reassigned to someone else

6.3 Conflict resolution

If a conflict of opinion comes into being. Then we will discuss this first in our online communication channels. If it does not resolve there then we will move this to a meeting, after discussion with the entire group we will vote on it. A majority vote will be taken. If that fails to resolve the issue we will seek mediation from a third party

6.3.1 Case study

When deciding for this project we had mulitple projects we wanted to do. This was not completed during our first session and we had a split between an image sharing app and this 2d game. We discussed this over our discord server and decided this would be better to discuss in person. During this in person meeting we wrote up the pros and cons of each, listing out the technologies and the interesting sections of the problems. and finally came to a vote. This process of multiple discussions was simple yet effective

7 Contributions

UP Number	contribution
UP2015336	equal
UP2071998	equal
UP2092245	equal
UP2092234	equal
UP2065429	equal
UP928651	no contribution