SETAP

Names go here

December 13, 2022

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 Requirements

These requirements are garnered from our interviews. 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 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.

- 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.).

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.)

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.

Users should be able to use a chest system.

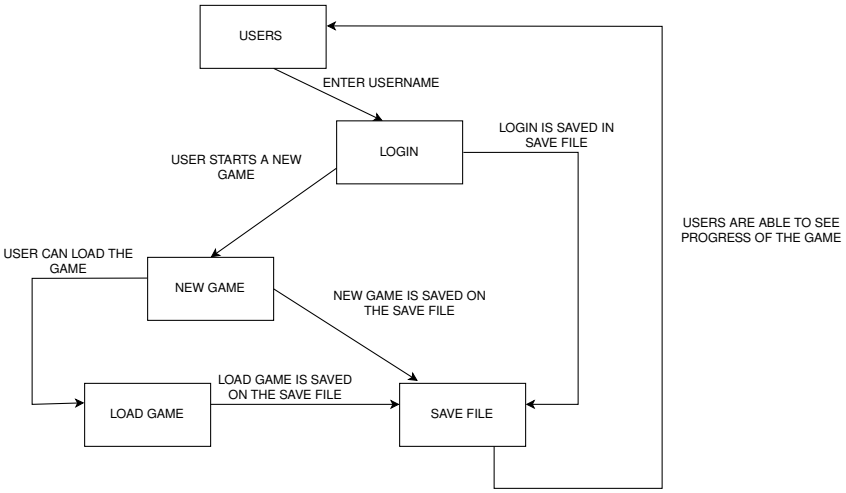
- Chests should be able to be crafted and placed into the world by users.
- A user should be able to view all the items that are in a chest.
- Items will be able to move between the user's inventory and the chest.

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
- Users with different operating systems like windows, Linux will gain access to the features that are provided by the application
- 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.
- The system should be easily installable on platforms that it is compatible from different operating systems e.g., windows, Linux
- 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.
- Will produce a save file for storing, maintaining and accessing data
- Users should find the system intuitive to use. Using long standard practices from the game industry such as movement commands and so on.

- 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 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.
- It will have different colour systems to cater to colourblind people.
- The game should be on different platforms e.g. (Laptop, pc, and smartphones).
- Users should identify during multiplayer games, with a name they provide before a connection is created.
- Visual Artefacts should not impede gameplay.
- Each point collected must be shown to the user within 2 seconds.
- The map will be saved as a 2D array.
- A save file for the current game will be saved up to every 10 minutes.
- The game should not crash under the pressure of many users playing.
- The game must be engaging, with users responding back telling us how they enjoyed what they can and can't do.
- The system should provide users of the total points collected on each session.
- The game must be accessible for all user segments.
- The system shall measure the time of each session and show it at the end of the session.
- The system should not take more than 10 seconds to launch the game.

3 Design


USERS
ENTER USERNAME
LOGIN
LOGIN IS SAVED IN SAVE FILE
USER STARTS A NEW GAME
USERS ARE ABLE TO SEE PROGRESS OF THE GAME
USER CAN LOAD THE GAME
NEW GAME
NEW GAME IS SAVED ON THE SAVE FILE
LOAD GAME IS SAVED ON THE SAVE FILE
LOAD GAME
SAVE FILE

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

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 compatability within revisions.

This will be followed by integration testing and then eventually user testing

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