|  |
| --- |
| CPSC 2720 FALL 2019 |
| Project Report for  Text-Based Adventure Game: Psylum  /var/folders/z0/pgf9tdys0lvdq17761z986zc0000gn/T/com.microsoft.Word/WebArchiveCopyPasteTempFiles/66qjS7aJYeVAAAAAElFTkSuQmCC |
| /var/folders/z0/pgf9tdys0lvdq17761z986zc0000gn/T/com.microsoft.Word/WebArchiveCopyPasteTempFiles/wGVYWVbrafDswAAAABJRU5ErkJggg==  Team: Castor |
|  |
| **Troy Hatchard, Lambee Mangal, Austen Oviatt and Sarah Ren** |
| **October 20th, 2019** |

Table of contents

[1. Introduction 3](#_Toc22492410)

[2. Project Management 3](#_Toc22492411)

[2.1 Team Organization 3](#_Toc22492412)

[2.2 Risk Management 3](#_Toc22492413)

[3. Devevlopment Process 4](#_Toc22492414)

[3.1 Code Review Process 4](#_Toc22492415)

[3.2 Communication Tools 5](#_Toc22492416)

[3.3 Change Management 5](#_Toc22492417)

[4. Software Design 5](#_Toc22492418)

[4.1 Design 5](#_Toc22492419)

[4.2 Design Rationale 5](#_Toc22492420)

[5. Appendices 6](#_Toc22492421)

[5.1 Appendix A: Figures and Tables 6](#_Toc22492422)

# Introduction

Pyslum is a mind-bending text-based adventure game. The game is set in a mental hospital/asylum and the mission is to escape! The game starts off without the player knowing where they are and why they’re there and only gets weirder as the game goes on. Every puzzle/challenge, item and interactions help you advance the game and get closer to escaping.

This project report reviews the project management, development process and software design of the game. The project management section will review team organization and risk management. The development process section will cover the communications and procedural aspect of the project and the software design will describe the game design and rationale behind it.

# Project Management

## Team Organization

* Team Lead – Troy Hatchard
* Design Lead – Sarah Ren
* Quality Assurance Lead – Lambee Mangal
* Documentation Lead – Austen Oviatt

## Risk Management

### Requirements/Design/Estimation

1. The team planned a project that is too large

Design the game in a way such that if the planned project ends up being too large, cutting out parts of the game wouldn’t impact the rest of the project. The team agrees and understands what parts could potentially be removed if need to be; ensure that the minimal viable product is delivered, while the rest is additional

1. The team underestimated how long parts of the project would take

In addition to ensuring that at least the minimal viable product is to be delivered on time and backup codes are available, create a project Gantt Chart for major deliverables and project milestones and dedicate one full day of the week to ensure these meet estimated deadlines. All group members are aware and agree to keep other team members informed if a part of the project is falling behind.

1. Major changes to design are needed during implementation

First and foremost, follow good design procedures to ensure that a major makeover of the design is not required if a change is made. All changes must be discussed as team to ensure everyone has a clear understanding of the impact of the changes. After changes are discussed and agreed upon, review and implement new design.

### People

1. Addition or loss of team member

In addition to consistent communication throughout the project, team members understand each role and responsibility to the extent that if the team loses a member, other team members are able to overlap duties to ensure that it has a minimal to no impact on the project. If a new person joins the team, the team will provide sufficient training and checking in with the new member to ensure a smooth transition and clear understanding of the project.

1. Unproductive team member(s)

Team Leader is to have a one-on-one discussion with the team member to better understand why he/she is being unproductive and ensure he/she is aware of their responsibilities as a member of the team. If the individual continues to be unproductive, a team discussion is required to determine what action is to be taken going forward.

1. Team member(s) lacking the expected technical background

Team is encouraged to ask questions and share knowledge of C++. Although the team is there for support and mentoring, members with a lack of experience with C++ are also expected to invest additional time on their own.

1. Major life events

Team members understand each role and responsibility to the extent that if a team member is unable to be there due to unforeseeable circumstances, other team members are able to overlap duties to ensure that it has a minimal to no impact on the project.

### Learning & Tools

1. Inexperience with new tools

Utilize the team members that have more knowledge on specific tools than other team members. If no team members know, then the team will invest some time for brief tutorials.

1. Learning curve for tools steeper than expected

All members of the team have agreed to put in any required time learning and polishing skills for the necessary tools. If the learning curve for a tool is too high, we will analyze whether the tool brings as much to the project as it is requiring from us, and will potentially choose an alternate tool to accomplish the desired task.

1. Tools don’t work together in an integrated way

Team will discuss the impact of the specific tools in question. After an analysis of the possible time lost due to a change of tools, the team will decide whether to continue with the current tools, or change tools and make up the time.

# Devevlopment Process

## Code Review Process

All code writing will be done on a branch, not on the master. Any time a branch needs to be merged with the master, a pull request will be created. One member will then pull the specified code to their local machine and ensure that it runs properly before approving the pull request. Any member can approve a pull request for any other member, but no member will be allowed to approve their own pull request.

## Communication Tools

* Trello - for the design, planning phase, and to track task assignments
* Facebook Messenger/Text - group chat for continuous communication
* GitLab – for code and issue tracking

## Change Management

All bugs must be reported. Each team member is responsible for dealing with bugs found in the code that they are working on. Team lead will triage reports with the support of the quality assurance lead. The team as a whole will discuss how the bug will be dealt with if necessary.

# Software Design

## Design

1. Starting for each of the games

To start each game, when the user opens the game they will be prompted to either start a new game or load an existing one. On a new game, the game will output the description of the starting environment and then move right into the game loop. (See Figure 2, Appendix A)

1. A turn for each of the games (including alternatives)

A turn will begin once the game has started. Each turn consists of a player input and a game output. The player must choose one of the valid options (aliases will be implemented) and will be given a help function to show the valid options available. The player will input an option and the game will return an output based on the player’s action. (See Figure 3, Appendix A)

1. Saving the game

A game save will be handled by the saveGame() function. This function will pull player data from the player object and will pull game state data from the GameState object, then write this to a file. (See Figure 3, Appendix A)

1. Loading the game

A game load will be handled by the loadGame() function. This function will pull player data and game state data from a file and create a GameState and Player with this data. (See Figure 3, Appendix A)

## Design Rationale

We intended the game to be as simple as we could make it without compromising on OOD principles. Some notable inclusions are the save and load functions. We originally implemented the save and load functions only within the Player class as they were a part of the actions a player could perform during their turn. We moved the player actions into an interface to abstract out the methods. We then realized that the user could not load a game at program start without the program creating a player object before the game even started. For this reason, we moved this function implementation into the Game class. This will likely be moulded a little more in later stages of the project.

The Tile interface was created to allow for some expansion in the way the map works. Currently the game has hallways, which were originally intended to be different than a room. However, the program will treat them the same as a room. But we left the Tile interface to allow for expansion.

Our player has a direction property. In many text-based games, the player moves by entering a compass direction. Since our game takes place indoors and the player wakes up disoriented, the user will only input left, right, forward, backward. This means that for the same hallway tile, the player’s options will change based on the player’s direction; a player’s left and right will be different when they come into the same hallway from the elevator versus the from patient’s room. For this reason, the player will have a direction property that will change based on where they came from on the previous move.

# 

# Appendices

## Appendix A:

Fig. 1: Class Diagram

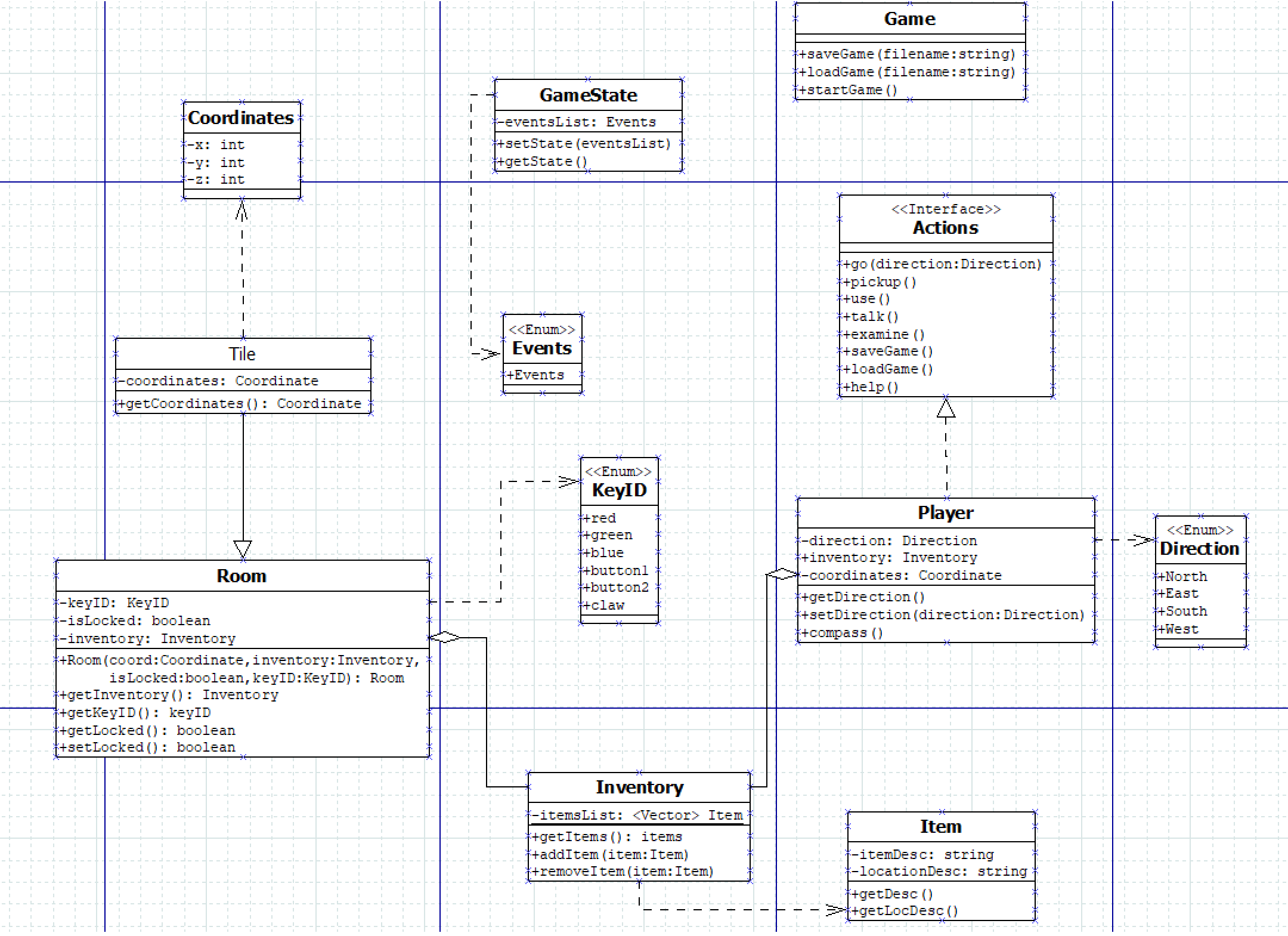


Fig. 2: Game Start Sequence Diagram

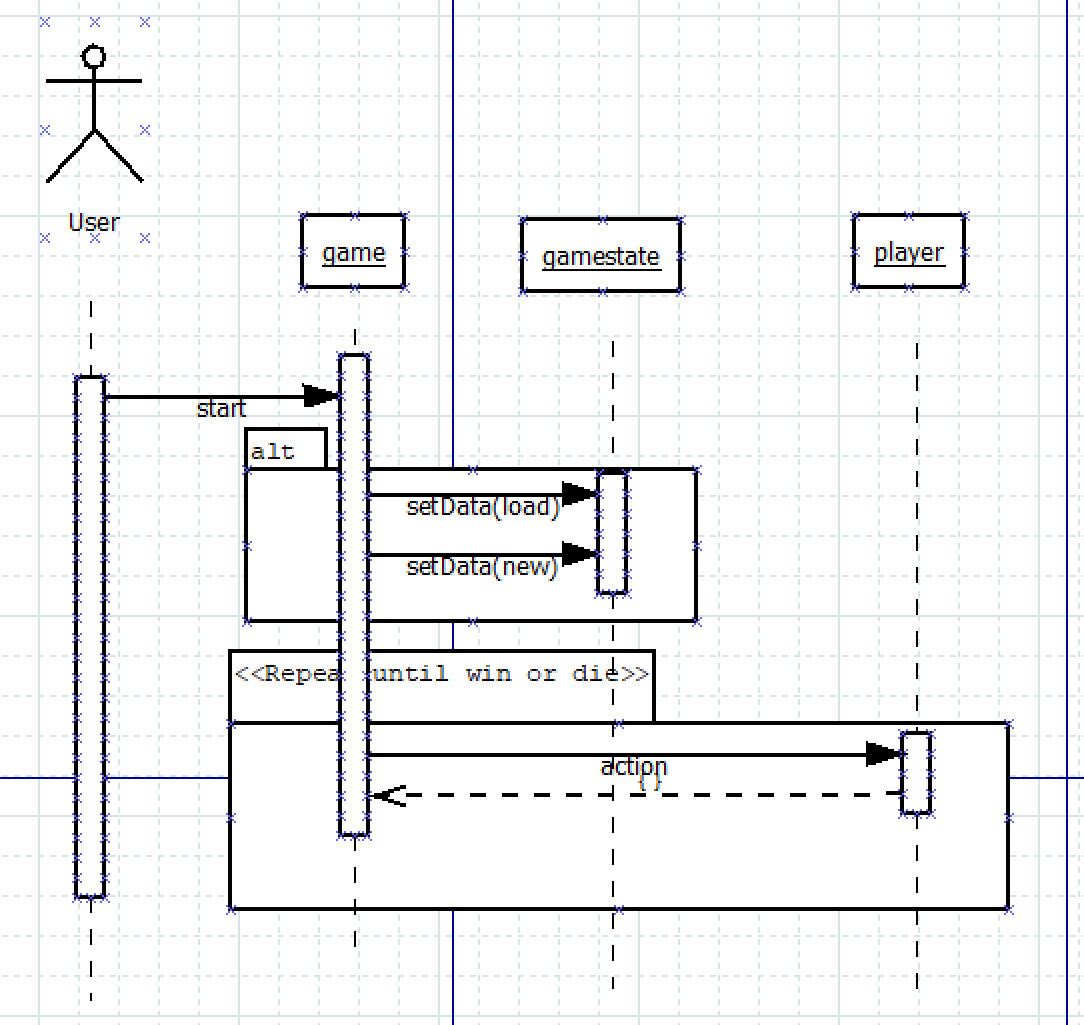


Fig. 3: Turn Sequence Diagram

