Software Engineering

Blue Team Final Report

CM50109 Coursework 2: A multi-player on-line game

Ben Boreham
Ruowen Cheng
Rikki Hodder
Jiaping Yu
Lucy Nelson
Yalin Shi

Table of Contents

Introduction

This report aims to provide a detailed history of the design and creation of our team's "Dungeon Explorer" game. Each step of the process has been documented and compiled in a chronological manner. During the six-week process of design and implementation, weekly versions of documentation were developed to capture the course of action taken by the team during that week. These weekly reports make up the backbone of this document and with the addition of auxiliary commentary, we have aimed to clearly communicate the narrative of the project from start to finish.

For this project, we employed a scrum management style. Using the pre-scheduled customer meetings as start points, each weekly sprint ran from a Wednesday to the following Wednesday. During each sprint, the team identified new tasks to add to the backlog of work to be done. This working list was organised into priority order during scrum meetings and was used to help assign weekly workload. Documentation was recorded to encapsulate the process and progress made. The order of weekly documentation within this report will adhere to the format seen below in Figure 1. To finish each sprint the team conducted a retrospective to confirm "done work", acknowledge successes and exceptions, and to prepare questions for the upcoming customer meeting. Repeating this process helped us keep a good handle on the project and maintain pace and steady progress. We hope this report is a clear and informative insight into the creation of our game.

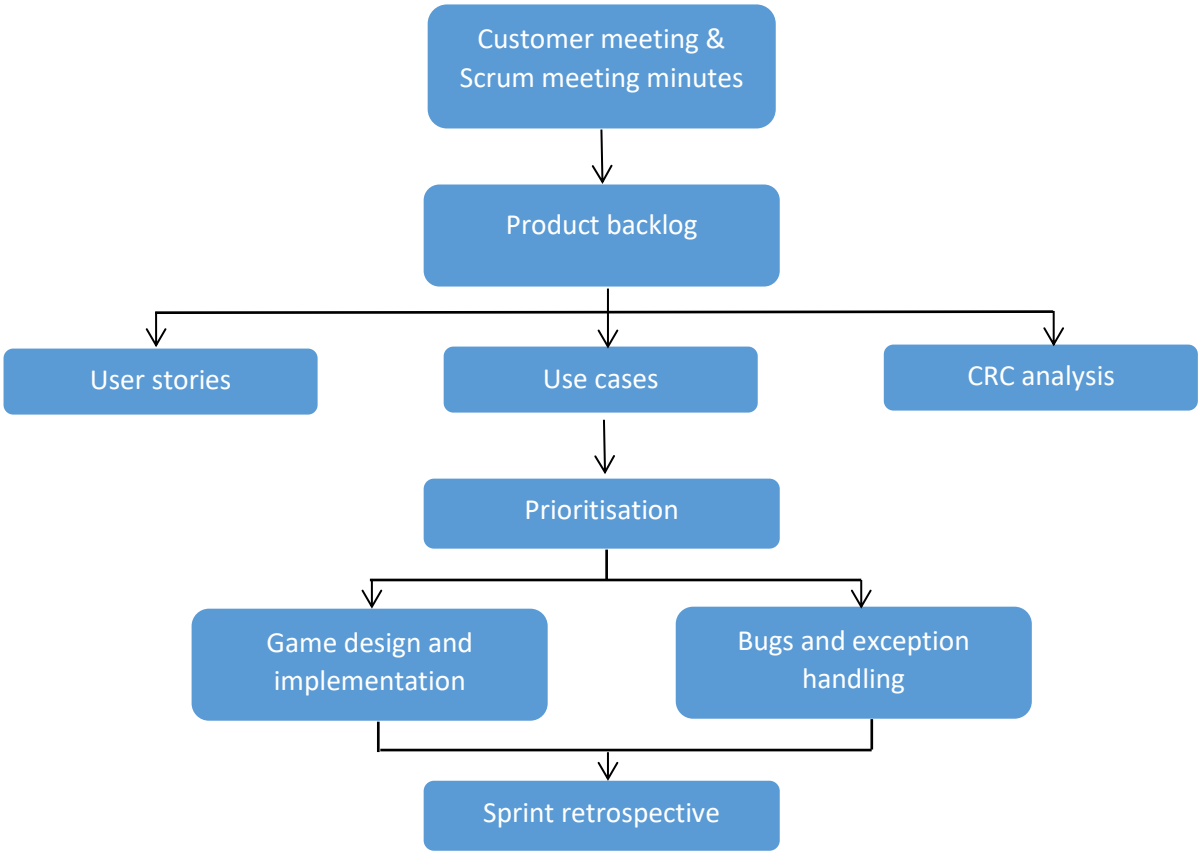


Figure 1: Flow diagram of weekly report content structure.

Week One

Overview

The initial week was a slight struggle for our team as each of the partners were still finalising their report from the first coursework. This dented the progress we could have made and perhaps left us in a worse position than we had hoped to be in. However, our first meeting was short and productive. The main outcome was to each go away and look at the basics of developing a two-dimensional game in Java. Upon our next meeting, each with a marginally greater amount of game development knowledge, we decided to think about the most important aspects to get us started. Obviously, we would need to have some sort of window to play this game in. It would certainly not be feasible to use the command line or console each of us are used to, either within the command prompt or some IDE, respectively. We therefore designed our first user story, which is at index 1 in our user story section. The remaining user stories were chosen by the importance we delegated to the set of game requirements on the coursework sheet. We felt as though the inclusion of multiple rooms or the implementation of a scoreboard were not as important as actually having something the user can control. We therefore made it a priority to have these user story requirements functioning as soon as possible.

Classes

- Game
- Window

Review

- The Java AWT (Abstract Window Toolkit) allows a java developer to use a set of classes that create GUI's such as a pop-up window. There is also the swing library in java, which allows the use of JFrame.

- We created a Window class that will specify what the window will do once created. Within this class we created a new instance of the JFrame and used some of its methods to specify the size and location of the window once the game has been started.

Customer Meeting Minutes

Group Name: Blue

Project Name: A multi-player online game

Date of Meeting: 1/11/2017
Start and End Time: 14:45-14:57

Objective
Customer vison of the game.

Attendees
Lucy Nelson, Rikki Hodder, Ben Boreham, Yu Jiaping, Ruowen Cheng, Yalin Shi

Agenda
User interface
Multi-players interaction
Multiple levels
Dungeon
Testing
Regular meetings

Decisions
User Interface
The user interface could either be player eye view, bird's eye view or restricted eye view. For this project the team has agreed bird's eye view.

Multi-Player Interaction
Human player will be moving around and picking up the resources. The 'bot' player could either be competing for resources, chasing the human player if it is in the vicinity. The question arises of how intelligent the 'bot' player is as software can make decisions faster than human play. Need to try to make the competition between the software and human player equitable.

Multiple Levels
Should levels increase with difficulty?
Dungeons
No requirement which is beyond capability. The function is entirely up to the team.

Testing
Testing will come later but once there is something to test (whether it fails or passes), customer can be notified of progress in the next meeting.

Regular Meetings
One customer meeting every week. Team meetings must also be frequent.

Action
User stories, user cases and CRC analysis needs to be conducted for following week.

Next Customer Meeting
Next meeting Wednesday 8th November 2017

Scrum Meeting Minutes

Group Name: Blue

Project Name: Dungeon Explorer- A multi-player online game

Week 1: 1/11/2017 - 7/11/2017

Objective
Basis of developing a two-dimensional game in java

Attendees
Ben Boreham, Lucy Nelson, Rikki Hodder, Ruowen Cheng, Yalin Shi, Yu Jiaping

Agenda
Research game design
Minutes of meeting
Create user stories
Communication

Decisions
Research game design
Everyone had a varying skill of coding. It was however important for all team members to be at the same standard of coding. This will allow us to carry out pair programming in the future. It was suggested that everyone watches tutorials online.

Minutes of meeting
Assign team member to write up Scrum and customer meeting minutes every week.

Create user stories
User stories such as developing a window for the game to be played, wanting the character to move around the window and collect treasure were discussed in this meeting. The highest priority user story was developing a window.

Communication
Whatsapp group and Slack were both set up for communication between team members. Google Drive, Trello and Github were set up so that potential code, documentation and tasks could be shared.

Tasks were allocated evenly amongst the team.
Action
First versioning on user stories and user cases.
Meeting minutes
Develop a window for the game to be played on for next week

Product Backlog

Using the feedback from the first customer meeting, we identified starting points for the various areas of the project. Using the software platform Trello, we separated elements of the workload into a colour coordinated scheme:

- **Orange** – Communication & decision-making documentation
- **Blue** – Product content e.g. user stories, use cases, CRC analysis
- **Green** – Game & user interface design
- **Purple** – Code implementation
- **Red** – Project management tasks
- **Yellow** – Product documentation

At this stage in the project, the backlog contained fairly generalised tasks. But, as we progress, the context should become clearer as it was easier to identify more specific goals for completion.

Figure 2 shows an excerpt from our week one Trello board. The backlog section is a working record of all the tasks that have been identified as necessary to the project. Prioritisation of these tasks then give way to a sprint backlog for weekly completion.

The aim of the first week's sprint goals are centred around outlining the scope of project and identifying the fundamentals of a game design and narrative. This should provide the team with a good foundation to build from and inspiration for more detailed questions for the next customer meeting.

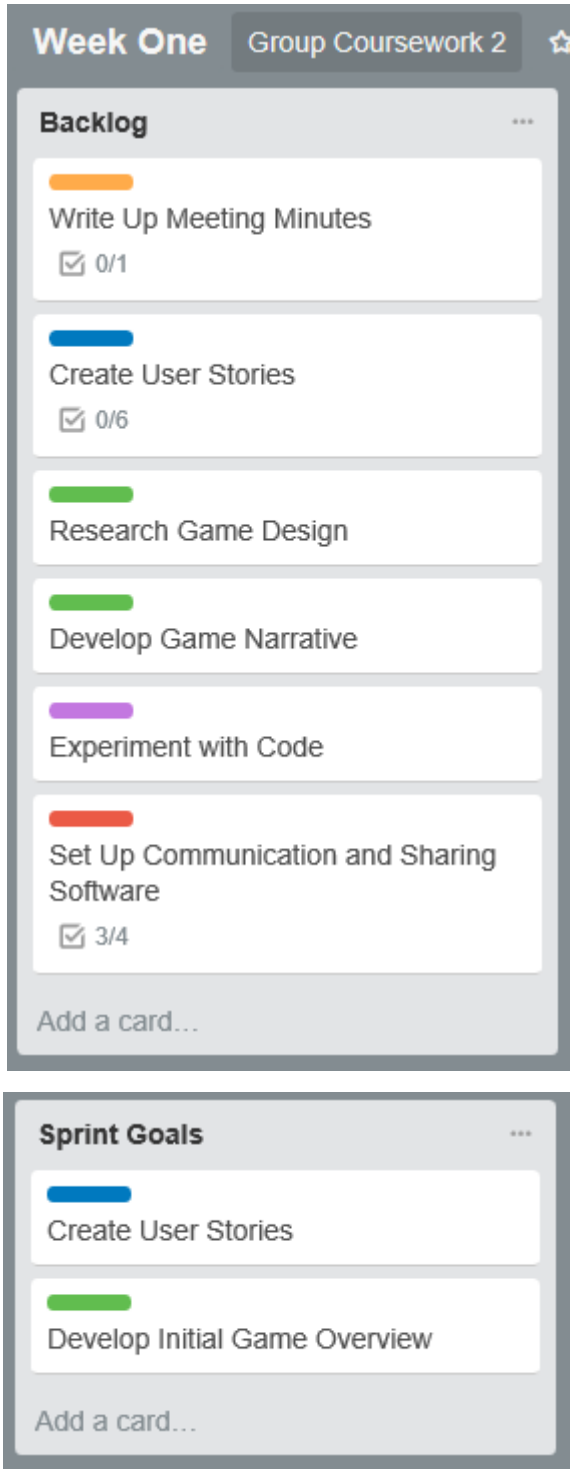


Figure 2: Trello board - Product backlog & Sprint goals

Prioritisation

As we will see later in the report, once established, many of the documents required weekly updating. These tasks became permanent fixtures in the sprint backlog and were generally processed first. However, the order in which user stories and use cases were converted into defined classes and added to the game were a matter of debate.

To identify the priority order for implementation of new features, the group gathered at for a meeting. User stories and use cases were drawn up on cards and each team member was given three tokens with values of 1, 2 or 3 written on them. Each person then placed their notes on the three use case cards they identified to be the most important, with the token of value three being placed on the card of highest importance. The total scores for each card were counted up and their priority arranged by highest first. The first week's process can be seen below in Figure 3. After the process the group discussed the decisions to allow anyone to raise issues that may have been overlooked by the rest of the group and when everyone was satisfied the priority order was set.

Once agreed upon, a table was constructed to help visualise the order, as seen in Table 1. This priority matrix also helped indicate the scope and difficulty of the task at hand, giving the group a chance to assign appropriate resources as well as predict and mitigate for potential risk. This strategy, combined with the use of code version control on GitHub, would later help maintain a working prototype whilst introducing and refactoring features.

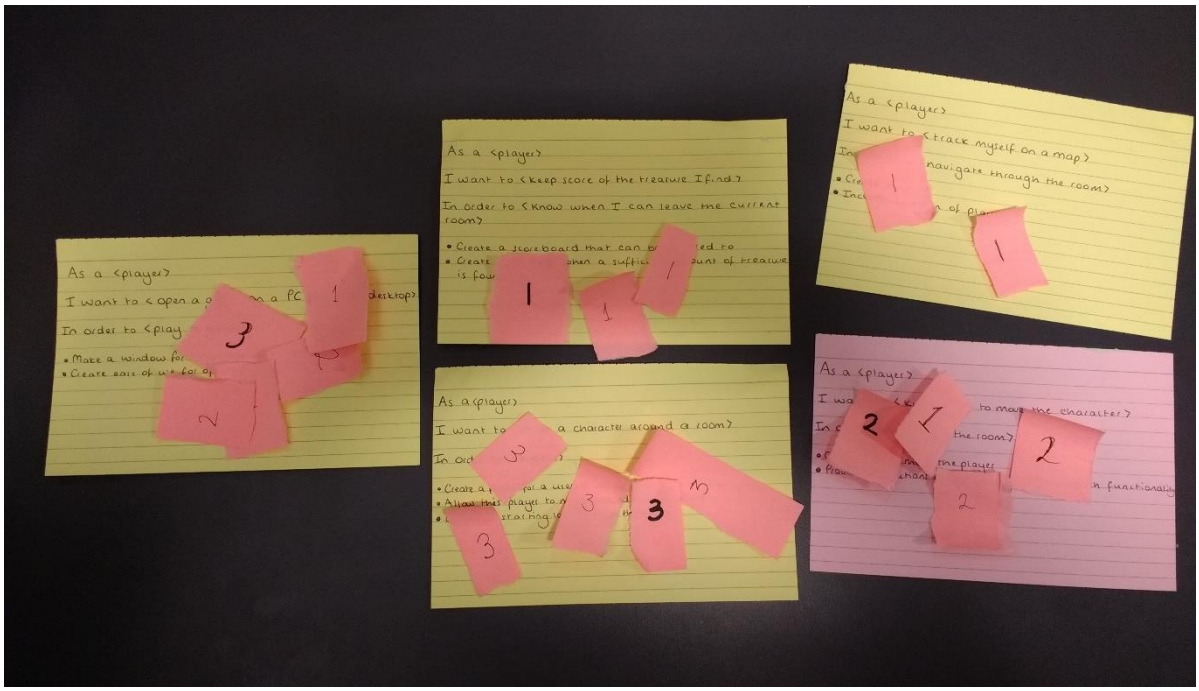


Figure 3: Example of Use Case Prioritisation Process

Table 1: Week One Priority Matrix

User Story ID	User Story Name	Primary Actor	Complexity[a]	Priority[b]	Completed[c]
US1	Open game	Player	Medium	1	Yes
US2	Movement	Player	High	1	No
US3	Exploration	Player	Low	2	No
US4	Score	Player	Medium	5	No
US5	Map	Player	High	5	No

[a] high, medium or low

[b] 1 (high) - 5 (low)

[c] yes, no or partially

Justification

US1: Having a window to play the game in will be the most important initial task, as this will help us visualise any further implementations.

US2: The inclusion of a movable player is also of utmost importance. We will need something for the user to interact with, so we have set the priority as a 1. We also envision this being a hard task, so the complexity has been set to high.

US3: Once we have implemented the movement functions for the character, this story should be an easy progression.

US4: Since we are collecting treasure, a scoreboard will be a good edition to keep track of the amount of treasure the user has collected. It may be difficult to have a pop up window within a pop up window to display the treasure, however. So we have set this to a medium complexity.

US5: Having a map mirror each of the actions within the game window seems to be a hard task, without knowing how to implement this functionality yet, we have presumed this will be hard.

User Stories – Version 1

Our first user story, user story 1 (US1), is the first task that the team decided was the most important. Without a window, there will be no way for us to represent a two-dimensional game.

The next appropriate task is for the user to be able to interact with this window. As they are to control a player character, we made it a priority to be able to move this character around the room with a given set of keys on the keypad. This is the case for US2. In addition to this, our requirements specify that the user should be able to explore a dungeon, we therefore appended US2 with a new user story in itself. This is to be able to explore the room within the window. This is still vague for now as we have had limited time to collaborate with our customer, so we have decided that collecting treasure will give a sense of exploration.

It seems natural that when one obtains collectibles within a game, such as treasure, they will want to keep count of the amount they obtain within their play through, as this give a sense of achievement. We have also added the criterion of creating a message when a sufficient amount of treasure has been obtained. This can be used in the future to signal the end of the game, or to move on to further rooms, which is a further requirement specified by the customer. US4 represents this task.

Finally, we have added the inclusion of a map that the user can refer to. As of now, we envision this to mirror the movements of the player within the room. Our final user story, US5, represents this.

1. Open Game

As a <player>, I want <to open a game on a PC from the desktop> in order to <play a game>.

- Make a window for a game to be played

2. Movement

As a <player>, I want to <move a character around a room> in order to <carry out the tasks within the game>.

- Create a player for a user to interact with
- Allow this player to move around a the window

3. Exploration

As a <player>, I want to <explore a room> in order to <obtain treasure>.

- Create obtainable treasure
- Provide a location for the treasure in the window

4. Score

As a <player>, I want to <keep score of the treasure I find> in order to <know when I can leave the current room>

- Create a scoreboard that can be referred to
- Create a message when a sufficient amount of treasure is found

5. Map

As a <player>, I want to <track myself on a map> in order to <navigate through the room>

- Create map
- Include location of player

CRC Analysis – Version 1

V1: Start to create a Window of the game, tries to create a player as well.

Classes

- Game
- Window
- Player

Game (Main class)	
• Creation of objects	Everything

Window	
• Creates the window for the game	Game

Player	
• Object Location • Object Speed • Object ID	Game Object

Class Diagram

In the first week, we have three classes: Game, Window and Player.

First, if we want to play this game, we should start the game and then will appear a suitable game window, the window can load all the items of the game. The player is in the game window, the player have a location when start the game and he can move according to a certain speed in the room.

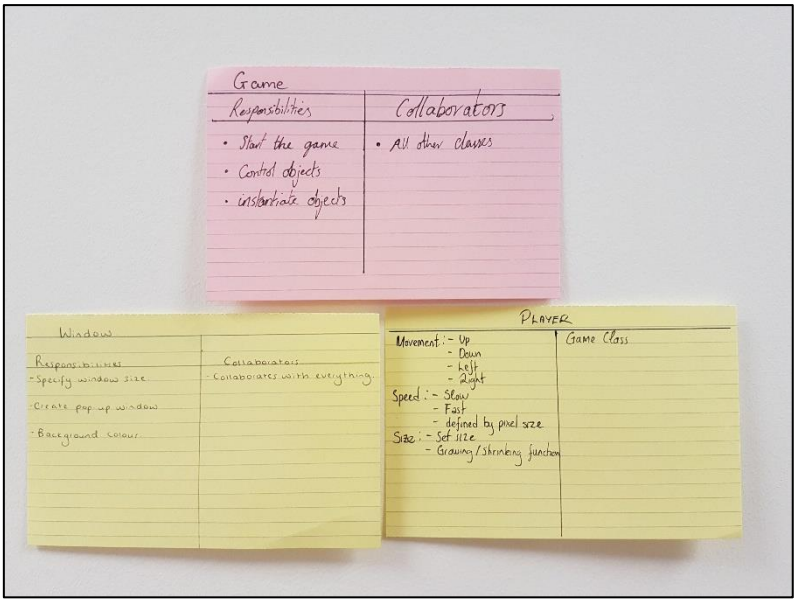


Figure 4: Week One - Class Card Association

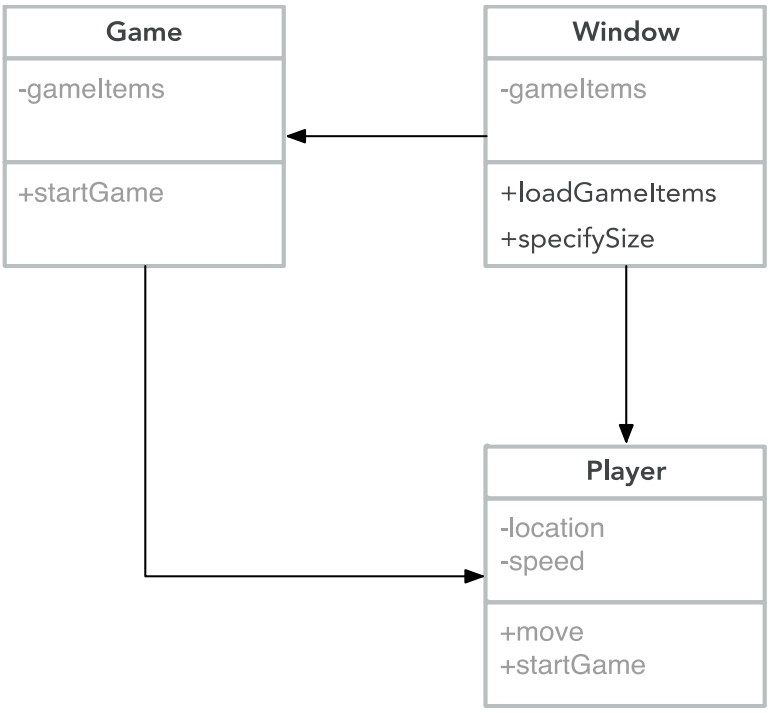


Figure 5: Week One - Class Diagram

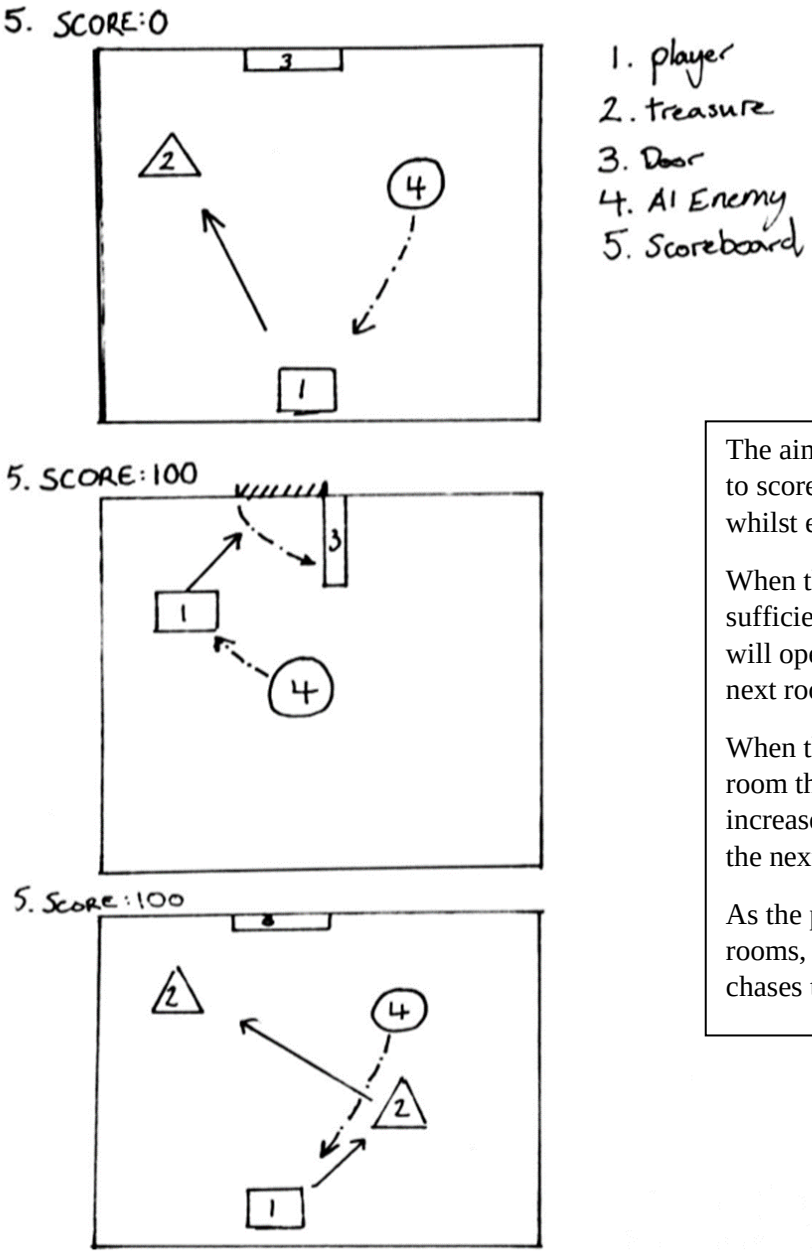
Game Design & Implementation

The starting point for our game design came from the combination of key points in the project brief and information attained during the first customer review. Between these two sources, fundamental requirements were extracted and a basic mental image of the game could be formulated.

It was clear that several elements were vital to the narrative; the game must contain:

- A playable character
- An enemy character
- Access to a second room
- Some treasure to collect

From this, in the most basic form, a game model was devised.



The aim of the game is for the player to score points by collecting treasure whilst evading an enemy.

When the player has collected a sufficient amount of treasure, a door will open and they can advance to the next room.

When the player advances to a new room they are required to collect an increased amount of treasure before the next door will open.

As the player progresses through the rooms, the speed at which the enemy chases the player increases.

Figure 6: Game Design Concept

This version is a draft frame includes some essential elements in the game. According to the requirement of the customer, the initial game design contains:

- A suitable size WINDOW.
- A PLAYER can move up, down, left and right, collect treasure.
- An ENEMY chases the player.
- A door enables the player to access to further rooms.
- Some treasure can be collected by the player.

User interface design

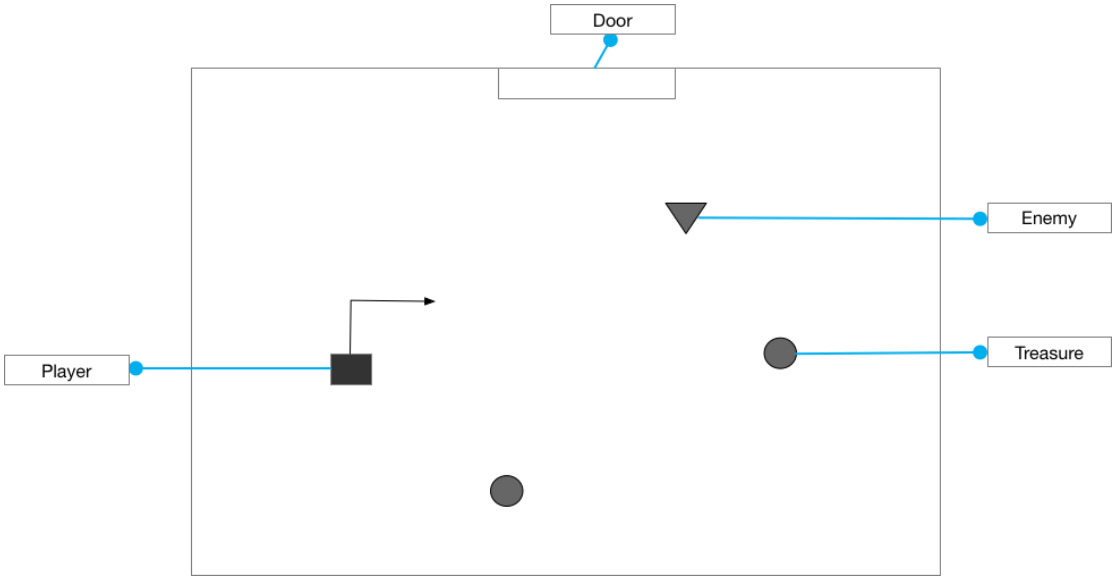


Figure 7: Game User Interface Design - Week One

Implementation view

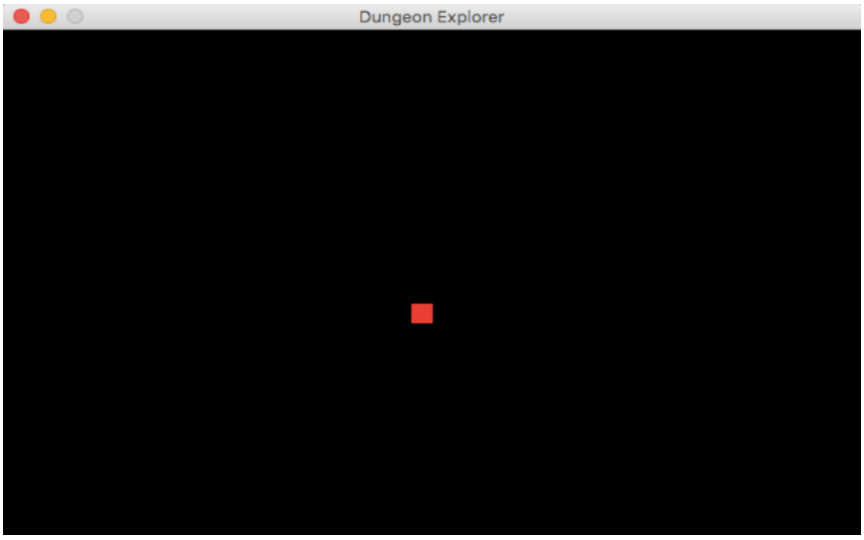


Figure 8: Implemented Game View - Week One
A controllable player character in a window

Week Two

Overview

The amount of progress made this week was the kind of progress we were expecting to make from the onset. As of the start of the week, we had a window that would open when the code was run from eclipse. We now had to think about the inclusion of distinct objects within this window, such as a controllable character and collectible treasure. It was very important to think about how we would need to structure this from the start, so we made a design of where we would like to see our game within the next few weeks. An example can be seen in Figure 12 on page 33.

Now, as we can see, there are going to be multiple objects within this game, so we decided that it might be a good idea to have some sort of parent class that contains a set of shared properties that these distinct objects will hold. It is clear that each object within the game will need a location, or starting location. There will also be at least two objects, the player and the enemy, which will both need some sort of movement function. So we defined these two sets of variables within our parent class, which we named Game Object.

We could also see that we are going to need a way to detect things such as collision between two objects. For example, we will need to know when the player reaches the boundaries of the window. We do not want them to keep moving in one direction, out of the defined window, indefinitely. We also need to know when the player character has interacted with the gold, or the enemy bot has collided with the player. After some research, we found that it would be a good idea to include an enum type that we could also include as an object in our Game Object class that would provide each subclass with its own identity. We then created an accessor method that would return the ID of the object.

Reviewing the user stories from the previous week, we also needed to create a player that could traverse the pop-up window. It is here that we realised whilst it may be simple to provide a dot on the screen, using the “Graphics” class, it will take a wider set of methods to implement the actual movement of the player. From the research online, we could see that we would need a method to process the logic of the game by using a game loop, along with a tick and render method, as they are often called. Each object will have their own tick and render method that will define their updated movement and rendering, respectively. The rendering of each object will be done using the Graphics class that we have used for the player class, as this has a good set of methods for displaying objects in a window.

The final problem we had was the way we would implement the usage of the keyboard to move the player character around. The Key Event and Key Adapter classes were the obvious routes to go down. Once we had implemented these, we were able to have the player character respond to a set of commands given by the user.

In terms of the set of requirements for the game, we decided that the inclusion of an enemy would be a natural progression. This was due to the current design of the game. With the parent ‘Game Object’ class that aids in creating new classes that have specified location and movement, it would be easier to add an automatically moving enemy than to create a new room, for example. We therefore designed a new set of user stories that will satisfy the requirements of including an enemy bot within the game.

Classes

- Game
- Window
- **Game Object**
- **ID**
- **Player**
- **Key Input**

Exceptions

The usage of a game loop seemed to be beyond the scope of any member of our team.

Minutes of Customer Meeting

Group Name: Blue

Project Name: A multi-player online game

Date of Meeting: 10/11/2017
Start and End Time: 10:37-10:53

Objective
Adaption on documentation and implementation of the function of the game.

Attendees
Lucy Nelson
Rikki Hodder
Ben Boreham
Ruowen Cheng
Yalin Shi

Absent
Josie sent her apologies as she was in Durham.

Agenda
User Stories
User Cases
CRC Analysis
Tools to Support Collaborative Working
Code Design

Decisions
User Stories
Only 6 user stories were presented to the customer in this meeting. There needs to be about 9 to 12 user stories. Not all user stories may be used. Example of user story "As a player I want to collect treasure so that we can score points". Feedback from customer concluded that player might not pick up treasure but may be on the same square as treasure. Acceptance criteria for this user story "Player collects treasure which leads to the score being updated.

Use cases must be versioned- old ones carried forward or dropped and new user stories added.

User Cases
After creating user stories, use cases must be looked at.

CRC Analysis
User stories are developed into CRCs. Left hand side is what we want the class to be able to do. Right hand side the other classes that it needs to work with in order to fulfil the behaviour.

Versioning
All documentation must be updated and versioned. As the project moves further on, user cases might be created that were not noted when first making user stories.

Tools to Support Collaborative Working
Screen shots of communication, photos of artefacts (e.g. user cases and versioning). WhatsApp group had already been set up the following week for team member to communicate and arrange meetings. GoogleDrive also step up tools such as GitHub, Trello and Slack.

Minutes of meeting need to be provided as it a natural outcome of holding a meeting. An update on the backlog must be produced. Team members being assigned tasks and keeping an update on assignments for scrum.

Code Design
Must embark on any design and code work as soon as possible- even if it is a program saying that test had failed. Look into model view controller. What are the object in the system etc.?

Action
Set up GitHub, Trello and Slack.
Add more user stories (versioning) and prioritising which will be implemented first.
More user cases and CRC analysis.
Start designing code. Can use open source code.

Next Customer Meeting
Next meeting Wednesday 15th November 2017.

Scrum Meeting Minutes

Group Name: Blue

Project Name: Dungeon Explorer- A multi-player online game

Week 2: 8/11/2017 – 14/11/2017

Objective
Review of customer meeting and allocation of tasks

Attendees
Ben Boreham
Lucy Nelson
Rikki Hodder
Ruowen Cheng
Yalin Shi
Yu Jiaping

Agenda
Game Overview
Documentation

Decisions
Game Overview
From last week a window had been created. We agreed that the next priority was to create a character that could move in four different directions. The issue with the moving character was that it could move outside the boundary of the game. For following week boundary should be implemented.

Documentation
More user stories and cases were developed. In the game the team decided that they wanted extra rooms for finding sufficient treasure. Also a means of keeping track of score and opponent player was discussed. Version 1 of CRC analysed was created in scrum meetings. Weekly minutes written up.

Action
Versioning of CRC, user cases and user stories
Research game design
Implement game framework
Meeting minutes
Add a game boundary

Product Backlog

Week two

During the latest customer meeting we discussed the need for a greater number of user stories to better encapsulate what would be required of the program. To accomplish this, a more complete vision of a game narrative would be needed.

Other essential product documentation such as use cases and CRC's have also been prioritised for development. These will be necessary to help convert the concepts of game narrative and design into a more concrete framework for implementation. It was also noted that some form of code implementation would be useful as soon as possible to establish a context relating code to documentation.

From a team management perspective a few ideas were highlighted as useful software solutions to issues concerning version control and remote communication.

The key components from this week's backlog were agreed by the team as expanding on user stories and beginning the development of a first version for all other product documentation. Refinement of the current game overview would be a good starting point.

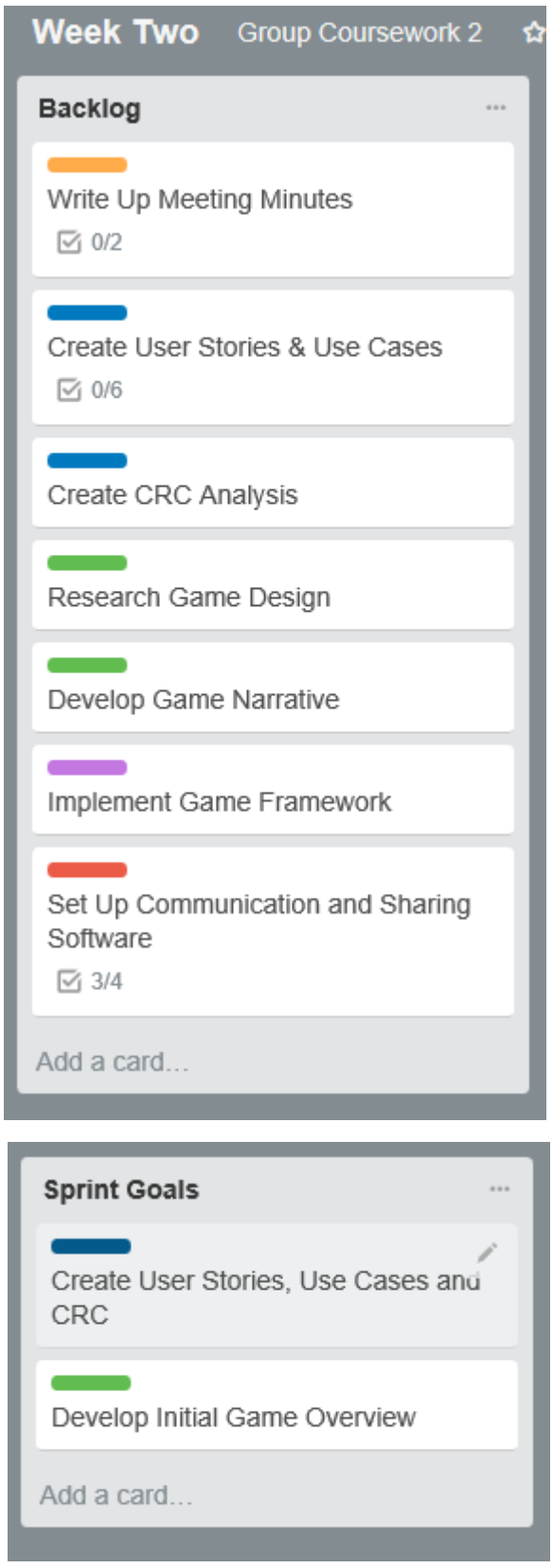


Figure 9: Trello Board Week 2 - Product Backlog & Sprint Goals

User Stories – Version 2

Amendments

Since we have already created a window that will open when the code is run from within eclipse, we decided that we will need to provide a better way to open the game from within a computer. It is not feasible for a user to download eclipse and all source code before even being able to play a game. We therefore included a new assessment criterion for the first user story.

We also realised that we need to have the player start in a particular location, which we have currently set to be the bottom left hand corner of the screen, otherwise the player will not be included in the window at all with the current code implemented.

Additions

We have added three user stories this week. US6, US7, and US8. US6 specifies a set amount of treasure to leave the room. US7 and US8 relate to the implementation of an enemy bot within the game

User Stories

1. Open Game

As a <player>, I want <to open a game on a PC from the desktop> in order to <play a game>.

- Make a window for a game to be played
- Create ease of use for opening game

2. Movement

As a <player>, I want to <move a character around a room> in order to <carry out the tasks within the game>.

- Create a player for a user to interact with
- Allow this player to move around the window
- Create a starting location for this player

3. Exploration

As a <player>, I want to <explore a room> in order to <obtain treasure>.

- Create obtainable and collectible treasure
- Provide locations of this treasure in the window

4. Score

As a <player>, I want to <keep score of the treasure I find> in order to <know when I can leave the current room>

- Create a scoreboard that can be referred to
- Create a message when a sufficient amount of treasure is found

5. Map

As a <player>, I want to <track myself on a map> in order to <navigate through the **room**>

- Create a map that mirrors the action within the window
- Include location of player

6. Win Condition

As a <player>, I want to <find a sufficient amount of treasure> in order to <escape the dungeon>

- Include multiple pieces of treasure
- Provide a way to keep count of the treasure collected
- Provide an exit to the room

7. AI Bot

As a <player>, I want <there to be an AI opponent> in order to <provide competition in the game>

- Create a bot
- Provide automatic movement for this bot
- Create a way for the bot to seek the player

8. Collision

As an <AI opponent>, I want to <catch the player> in order to <end the game>

- Create a way to detect collision between the player and enemy

Use Cases – Version 1

1. Open Game

As a <player>, I want <to open a game on a PC from the desktop> in order to <play a game>.

Use case	Open the games user interface
Summary	We want to create a window for the game to be played in
Actor	Player
Trigger	The player will try to open the game window
Primary Scenario	1. The user opens the game and the game window pops up on the screen in an appropriate position as specified by the developers.
Exception Scenario	1. The player cannot find the file 2. The game does not open
Pre-conditions	Storage location availability
Post-condition	The game window is now open

2. Movement

As a <player>, I want to <move a character around a room> in order to <carry out the tasks within the game>.

Use case	Move the character
Summary	This use case allows the player to move a character
Actor	Player
Trigger	The player presses a predefined button to move
Primary Scenario	1. The player character is instantiated within the game window and is visible to the user. 2. The player decides in which direction they would like the character to move 3. The player then presses the appropriate button to move the character. 4. The player will move in that particular direction, until 5. The player removes their finger from the button that determined movement. 6. The character comes to a halt and the action is terminated.
Exception Scenario	**Buttons not defined** 1. The buttons for movement are not defined or have not been instructed by the user 2. The character does not move.
Pre-conditions	The player character is instantiated within the window with a set of buttons that determine their movement.
Post-condition	The player will move in the direction specified by the player.

3. Exploration

As a <player>, I want to <explore a room> in order to <obtain treasure>.

Use case	To explore a room and collect treasure
Summary	We would like to give the user a sense of exploration by including collectibles such as treasure
Actor	Player
Trigger	The player is instantiated within a room and is able to move towards collectibles.
Primary Scenario	1. The player will be placed at a starting location within the game window. 2. The player will be able to traverse the room as instructed by the user playing the game. 3. There will be distinct parts of the room, distinguished by obstacles, such as walls within the room, which will include collectibles, like treasure, that can be interacted with. 4. Once interacted with, this will then disappear from the screen and will be the property of the player character
Exception Scenario	**Buttons not defined** 1. The buttons for movement are not defined or have not been instructed by the user 2. The instantiation of the walls within the room overlap with the treasure, making this unobtainable by the user.
Pre-conditions	The player character is instantiated within the window with a set of buttons that determine their movement.
Post-condition	The player will interact with a collectible object and this will then disappear from the window.

4. Score

As a <player>, I want to <keep score of the treasure I find> in order to <know when I can leave the current room>

Use case	To keep track of treasure collected
Summary	This use case will allow the user to keep track of the treasure they have collected. This will allow them to know when they can progress within the game
Actor	Player
Trigger	The player collects one unit of treasure
Primary Scenario	1. The character is adjacent to a piece of treasure in a room 2. The player directs the character to move in the direction of the treasure [Alternative: The player does not direct the character to move in the direction of the treasure] 3. The character moves in that direction thus interacting with the treasure 4. Once the two objects meet the treasure will disappear. 5. A counter will increment by one unit indicating that this has been stored within the player's inventory.
Exception Scenario	The player does not direct the character in the direction of the treasure 1. The player will move in a direction towards empty space, and, the character will: 2. Either move to an empty space in the room, or not move, as specified in the use case alternative for use case 2. 3. The treasure is still present in its location.
Pre-conditions	The player has started to navigate the room by moving the character
Post-condition	The player will obtain a piece of treasure

5. Map

As a <player>, I want to <track myself on a map> in order to <navigate through the room>

Use case	Display player location on an HUD at a specified location in the window.
Summary	This use case will create an HUD that will be visible at all times the game is in play. The player will be able to see their location in the current room.
Actor	System
Trigger	Whenever the player location is rendered in a new position, the map should mirror this and also render the new location of the player on the map
Primary Scenario	1. The user instructs the player to move and this is subsequently amended in the HUD. 2. Once the player has progressed to a successive room, the HUD will update itself and render a new image to match the layout of the current room.
Exception Scenario	1. The map does not reference the current room 2. The map does not update when the player locates to a different room
Pre-conditions	The game has started
Post-condition	The map is updated.

6. Win Condition

As a <player>, I want to <find a sufficient amount of treasure> in order to <escape the dungeon>

Use case	To create a lower limit for the amount of treasure a player can collect to escape the dungeon
Summary	This use case will provide a means for a player to finish after collecting a certain amount of treasure.
Actor	Player
Trigger	The player collects the set amount of treasure. We could set this to 75% of the total treasure in the dungeon.
Primary Scenario	1. The player obtains the piece of treasure that will exceed the lower limit needed to progress. 2. A door will appear at a set location in the room that will allow the player to leave the dungeon. 3. The player will move toward this door and interact with it. 4. Once it has been interacted with, the player will receive a completion message.
Exception Scenario	The player may be 'hit' by an enemy before the door image appears. In this case, the player will receive a game over message.
Pre-conditions	The player is able to obtain a set amount of gold by navigating the room and avoiding the enemy
Post-condition	The player will obtain enough treasure to open up a doorway to a dungeon exit.

7. AI Bot

As a <player>, I want <there to be an AI opponent> in order to <provide competition in the game>

Use case	Inclusion of an AI opponent
Summary	We would like to include an AI opponent that also traverses the map within a specified movement function
Actor	AI opponent & Player
Trigger	Starting the game
Primary Scenario	1. The AI enemy constantly follows the player at a speed 75% that of the playable character 2. The enemy catches the player which results in a game over message.
Exception Scenario	The AI enemy becomes trapped in a part of the room and is unable to move. For example, if the player is directly in front of the enemy, but there is a wall in the way, the enemy will continue forward indefinitely whilst the player is in this position as they have been programmed to only move in the direction of the player.
Pre-conditions	The game is open and the player has started the game
Post-condition	A player-seeking enemy is in the room A random walk object is included in the room.

8. Collision

As an <AI opponent>, I want to <catch the player> in order to <end the game>

Use case	To include a bot in the room that
Summary	We would like to have the AI opponent track the player throughout each room and try to 'catch' them in order to end the game. This relates to US7, but implements a collision function to detect this.
Actor	AI opponent
Trigger	Collision between the player character and the enemy
Primary Scenario	The AI opponent will attempt to catch the player, either: 1. Reaching the same position as the player at the same point in time, in which point the player will receive a message saying 'game over'. 2. The player will then be removed from the game window and will have to restart their attempt.
Exception Scenario	1. The player will avoid the AI opponent and the game will carry on as normal. That is, the AI opponent does not share any of the same pixel allocations as the player.
Pre-conditions	The player and AI enemy are both in the same room
Post-condition	The game ends and the player receives a game over message.

CRC Analysis – Version 2

V2: Update the player class and add a few classes to support the game. New: Game Object (Abstract Class), Enemy, Treasure.

Classes

- Game
- Window
- Game Object (Abstract Class)
- Player
- Enemy
- Treasure

Game (Main class)	
• Creation of objects	Everything

Window	
• Creates the window for the game	Game

Game Object (Abstract Class)	
• Object Location • Object Speed • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Objects displayed on the screen

Player	
• Object Location • Object Speed • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Enemy Treasure

Enemy	
• Object Location • Object Speed • Object ID	Game Object Enemy Treasure

Treasure (Gold)	
• Object Location • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Enemy

Class Diagram

In the second week, we have five object classes: Game, Window, Player, Enemy, and Treasure. We add the enemy, treasure and a new abstract game object as the new classes.

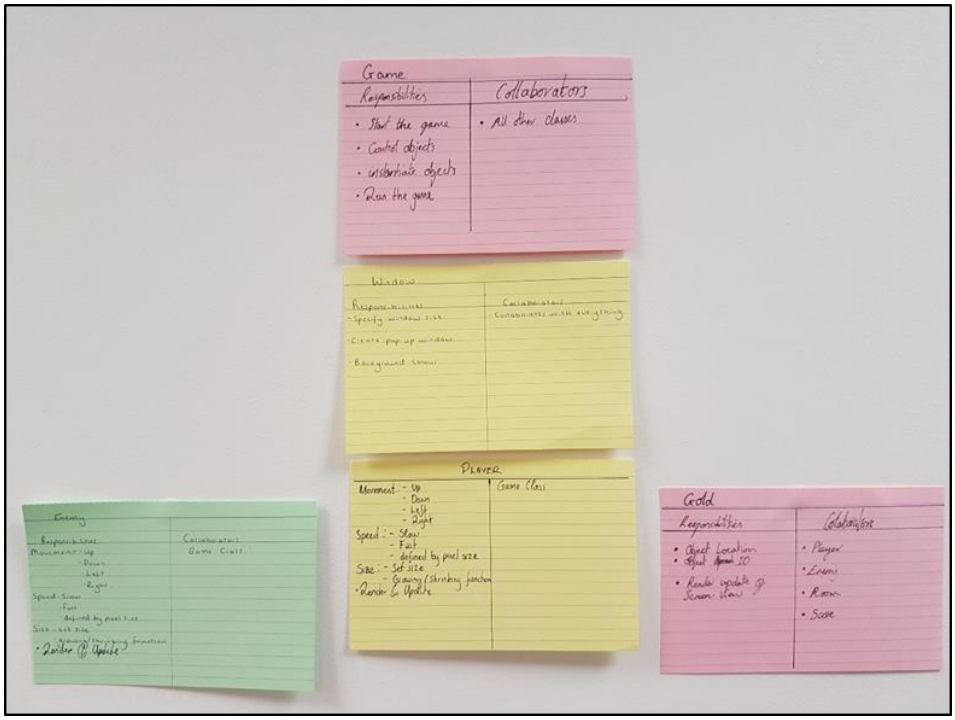


Figure 10: Week Two - Class Card Association

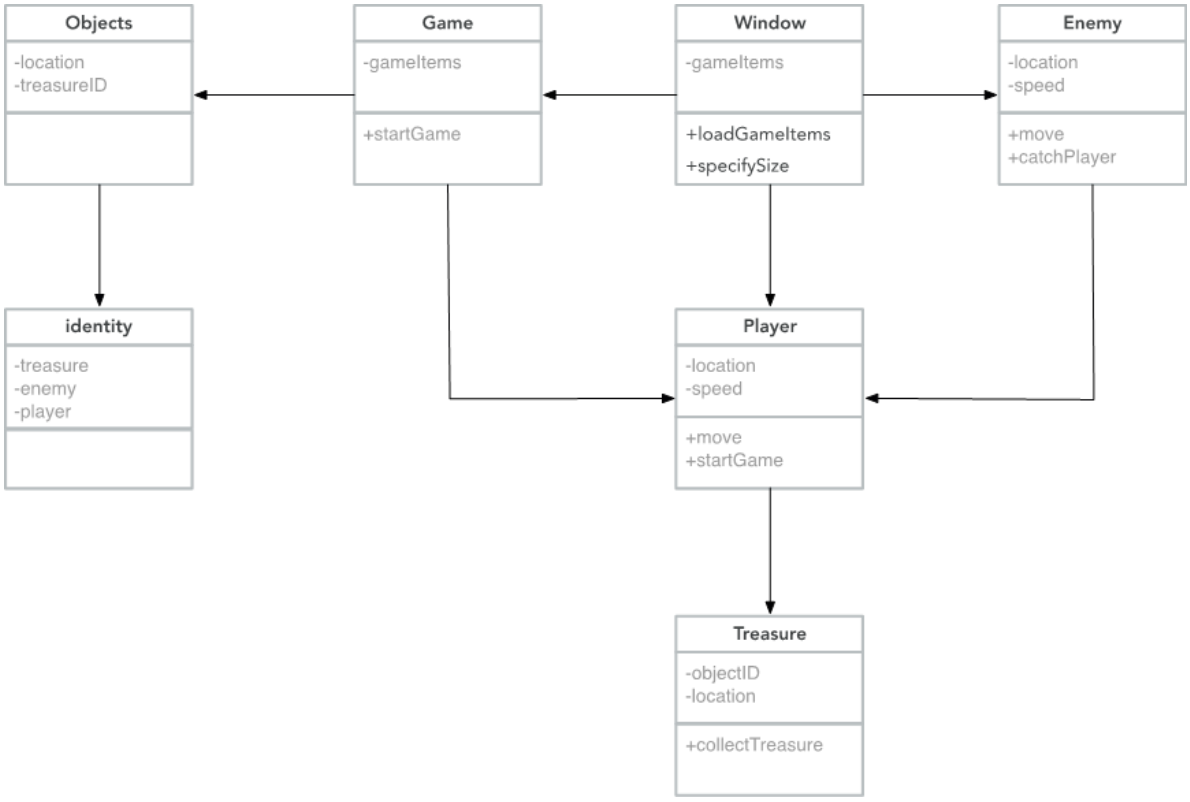


Figure 11: Week Two - Class Diagram

Prioritisation

Table 2: Week Two Priority Matrix

User Story ID	User Story Name	Primary Actor	Complexity[a]	Priority[b]	Completed[c]
US1	Open Game	Player	Medium	3	Partially
US2	Movement	Player	Low	1	Yes
US3	Exploration	Player	Low	2	Yes
US4	Score	Player	Medium	4	No
US5	Map	Player	High	5	No
US6	Win Condition	Player	High	4	No
US7	AI Bot	AI Bot	Medium	2	Partially
US8	Collision	AI Bot	Medium	2	No

[a]High, medium or low

[b]1(high) – 5 (low)

[c]Yes, no or partially

Justification

US1: We have added the criterion to create an ease of opening this game. We have therefore amended the complexity of this to Medium and decreased the priority. We do not have to worry about this until we release this product.

US2: We have added the criterion to create a starting location for this player. This was considered an easy task, so the complexity of the task was lowered.

US6: The main criterion that pushes this into the higher level of complexity is the inclusion of an exit to the room. However, whilst we are still working within one single room, trying to get a fully working room before multiple rooms is of essence. We have therefore given this a lower priority

US7: This is a main requirement for the game, and one that we can think about before creating further rooms. As we already have a parent class we can inherit functions from, this class should not be too difficult to create. The main problem is the seeking function we have specified to put in place.

US8: We will have to create a way to detect object collision. This should not be too difficult since each class as a clear identity we can access within our Game Object Linked List. We have therefore given this a medium complexity and high priority, as an enemy is no threat if it cannot effect the player.

Game Design & Implementation

Not too much changes in Version 2, a new element is BOUNDARY.

Words in blue is what is new in current version.

- A suitable size WINDOW.
- A PLAYER can move up, down, left and right, collect treasure.
- An ENEMY chases the player.
- A DOOR enables the player to access to further rooms.
- Some TREASURE can be collected by the player.
- BOUNDARY is added in the game, to stop both the player and enemy move outside.

User interface design

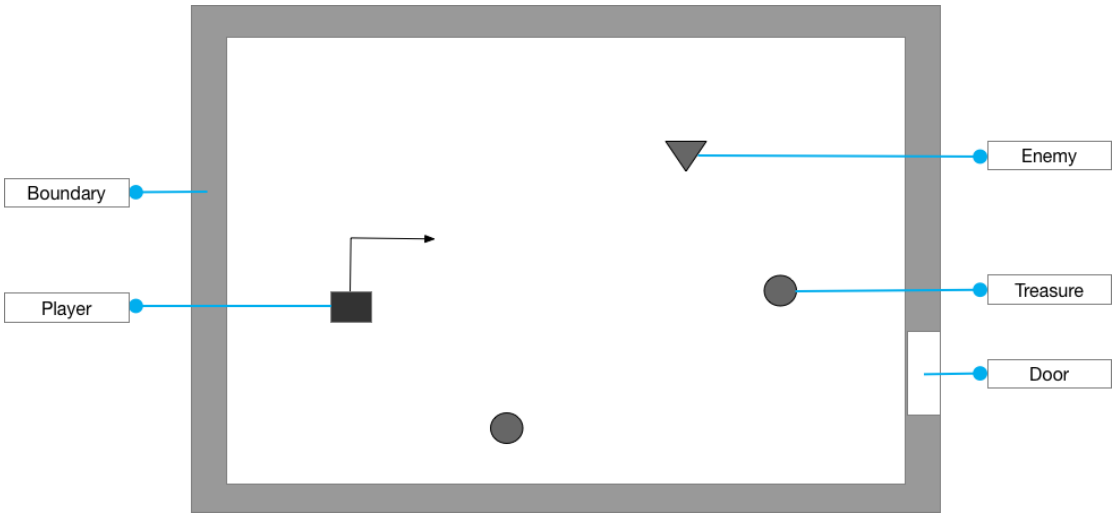


Figure 12: User Interface Design - Week Two

Current application view

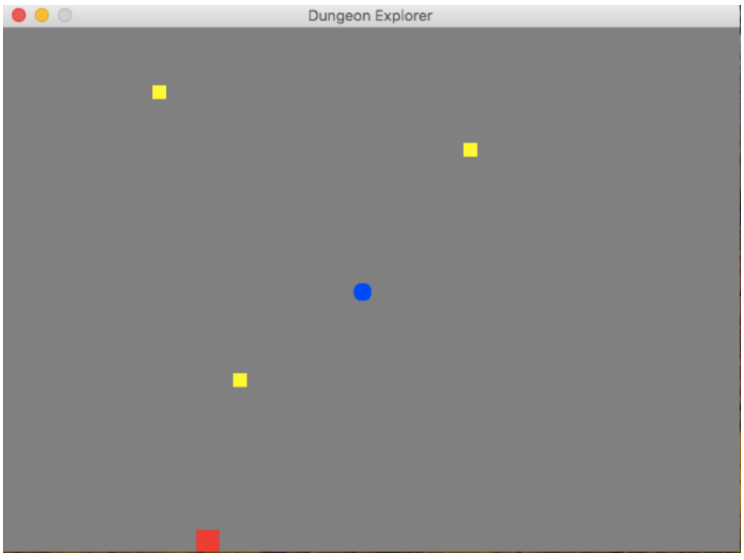


Figure 13: Implemented Game View - Week Two

Sprint Retrospective

Week two

Week two saw us get to grips with the process of how a sprint should run and familiarised the team with the foundational documents necessary for every sprint.

Improvements to the game narrative have delivered a basic framework from which we were able to start developing some basic class requirements that were unlikely to change. From the use cases developed it was clear that our game would require a game window to run in, and a character for the player to control.

To start the ball rolling with code implementation several of the team members individually researched methods for achieving the window task and character control. The team were successful in doing so and the end of this sprint we were able to start play testing the primitive model.

The resulting experimentation introduced our first selection of bugs and exceptions as shown in Figure 14. These can be added to future backlog but also used to draw inspiration from. The discovery of these issues has been very useful in highlighting new considerations for our existing user stories and the overarching game design.

By the end of this sprint all of the software channels we intend to use as a team had been established.

We have created:

- GitHub for an accessible code repository
- A collective GoogleDrive for documentation
- Trello for project and sprint planning
- Slack and WhatsApp group for communication

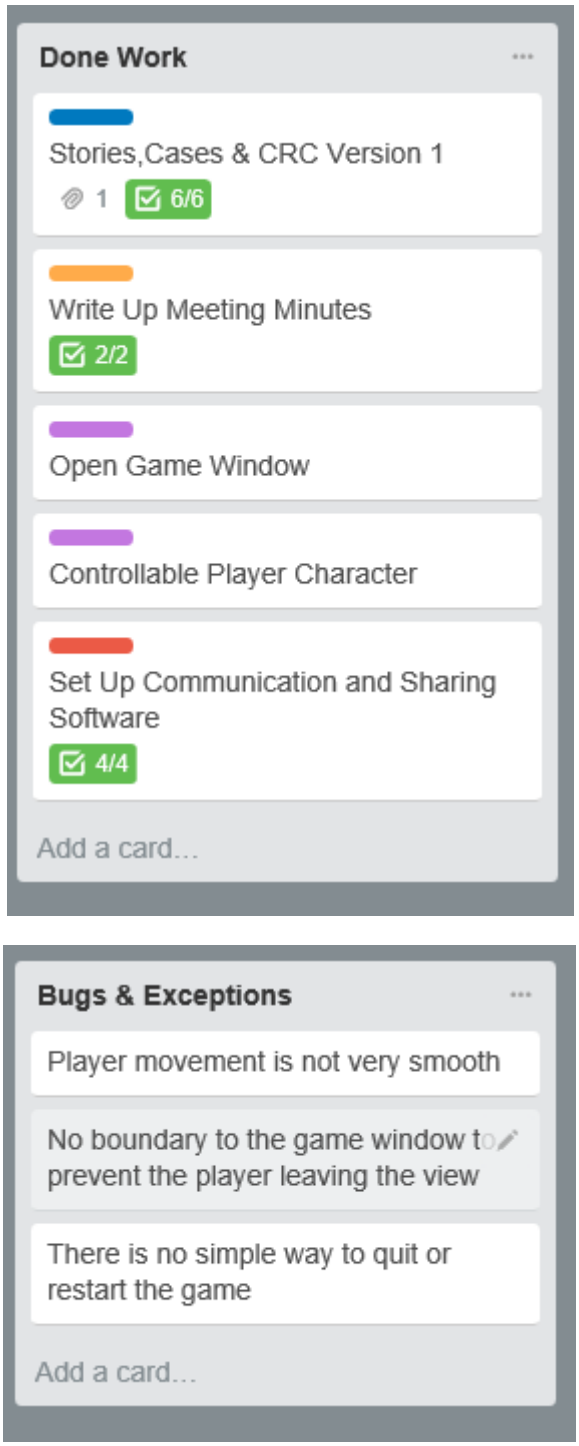


Figure 14: Trello Board Week Two - Done Work & Bugs

Week Three

Overview

As of now, we have a player character in the shape of a circle that can be moved around a black window. We will need to make sure the user cannot move this player beyond the boundaries of the window. Since we know the width and height of our window, we created a basic function that will check whether the player's coordinates are beyond the dimensions of the window. If they are, then we keep the players position at the width and height specified. We understand that this is not a very generalizable function, but it is a good starting point.

With the implementation of the enemy, which we specified in our user stories last week, we took a similar route as the player character, and this was a much more streamlined process. We already had a decent structure in place to create new objects from our parent class, and we would not have to worry about any key input for this object, as our aim is to have this moving autonomously. The difference here was that we would not have the speed of the enemy only moving within our tick method, but actually set this to a constant value within the objects constructor. This would leave the enemy moving at a fixed speed in a fixed direction. Unfortunately, this does not relate to our original user story, which specified that the enemy would move in a direction that tracked the location of the player character. This ended up being a hard problem to solve and so we have kept the enemy moving with a fixed direction, as opposed to a seeking direction.

A similar story applies to the inclusion of collectible treasure. Again, we implemented this as a subclass of our Game Object parent class, providing it with a location that can be set on each instantiation. However, with the treasure, we would be including multiple instances within the room, so we have made it a user story to do this at random locations within the map, instead of creating a new instance with a set location manually within the games constructor.

Whilst reviewing the presentation of the current game, it came to our attention that the room was rather bare. A dungeon is a setting that should contain obstacles and paths, with a sense of danger for the one who happens to navigate it. We therefore decided that the inclusion of a set of obstacles within the room would give the user a stronger sense of 'being within a dungeon', as opposed to merely exploring a barren, square shaped room.

We also have a set of objects that are only displayed as shapes in various colours. This is not a very presentable product for a game that is supposed to be a dungeon crawler. We would prefer to have a greater sense of surrounding for the user as they traverse the character around the window. We therefore decided to add a set of sprites to the presentation of each object, as opposed to the standard set of shapes we have within the graphics class.

Minutes of Customer Meeting

Group Name: Blue

Project Name: Dungeon Explorer- A multi-player online game
Date of Meeting: 15/11/2017
Start and End Time: 17:12-17:23

Objective
Weekly review to increase functionality of the game.

Attendees
Ben Boreham, Rikki Hodder, Ruowen Cheng, Yalin Shi

Absent
Lucy Nelson and Yu Jiaping sent apologies for being sick

Agenda
Documentation
Code Design
Moving Forward

Decisions
Documentation
User stories, cases analysis, CRC analysis and minutes all need to be included in process documentation. It is important to note that these all must be versioned as it is vital to the design process that these are added to and refined. Versioning is advised to not only be recorded in word documents but also in a physical form. It is also important to comment the priority of each user case to give direction.

In terms of documentation for next week's meeting, weekly reports should be written up as this is a clear way of seeing the progress of project each week. The end report will include all weekly reports with a narrative. Documentation can also include screenshots of Trello boards to add to archive. Slack is more used as a way of communication between members instead of documentation.

Code Design
Open source code was used for game loop. Members of the team need to familiarise themselves with coding and try to get up to the same standard. This will help with pair programming.

Action
Weekly report developed for next week and prioritising user stories for game design purposes.

Next Customer Meeting
Next meeting on the 22nd November 2017

Minutes of Scrum Meeting

Group Name: Blue

Project Name: Dungeon Explorer- A multi-player online game

Week 3: 15/11/2017 - 21/11/2017

Objective
Review of customer meeting and allocation of tasks

Attendees
Ben Boreham, Lucy Nelson, Rikki Hodder, Ruowen Cheng, Yalin Shi, Yu Jiaping

Agenda
Documentation
Game Overview

Decisions
Documentation
It was discussed in the customer meeting that weekly reports needed to put together to show progress of project. Weekly report consisted of versioned user stories, user cases and CRC analysis, sketches of user interface, scrum and customer meeting minutes and project Gantt chart

Game Overview
We have a window; a moving character and a boundary has already been implemented into the game. A wall was also created. The addition of an AI bot was the next prioritised user story. We decided that the bot was going to bounce off the boundary of the game. The bot moved at a fixed speed and a fixed direction. The bot following the player would have been too difficult to create.

Creating collectable treasure was also achieved this week. Random location of treasure was implemented in the code. This was also made into a user story. Image for objects were made

Action
Meeting minutes
User stories and user cases
Object/character aesthetic
Create AI enemy
Create collectable treasure
Project Gantt chart
Create weekly report
Fix bugs

Product Backlog

Week Three

Continuing on from week two, our user stories and use cases still needed expanding. We currently had eight of each but it had been suggested, in the most recent customer interview, that a project of this scope is likely to require at least a dozen. In response to this we set a target of twelve user stories to be documented by the end of this sprint.

The current incarnation of our executable game code consists of a controllable character in an empty window. This sprint will focus on bringing the game more in line with the basic game overview. Introducing enemies and gold, and containing the game area with walls are primary elements for implementation.

In addition to progressing the game and the relating use cases, we decided to dedicate a portion of this sprint to the organisation and formatting of the project documentation. As well as establishing an anticipated timeline for the remainder of the project. We hope that by producing a weekly report format it will be easier to convey the current position of development, and recent design decisions made, during customer meetings.

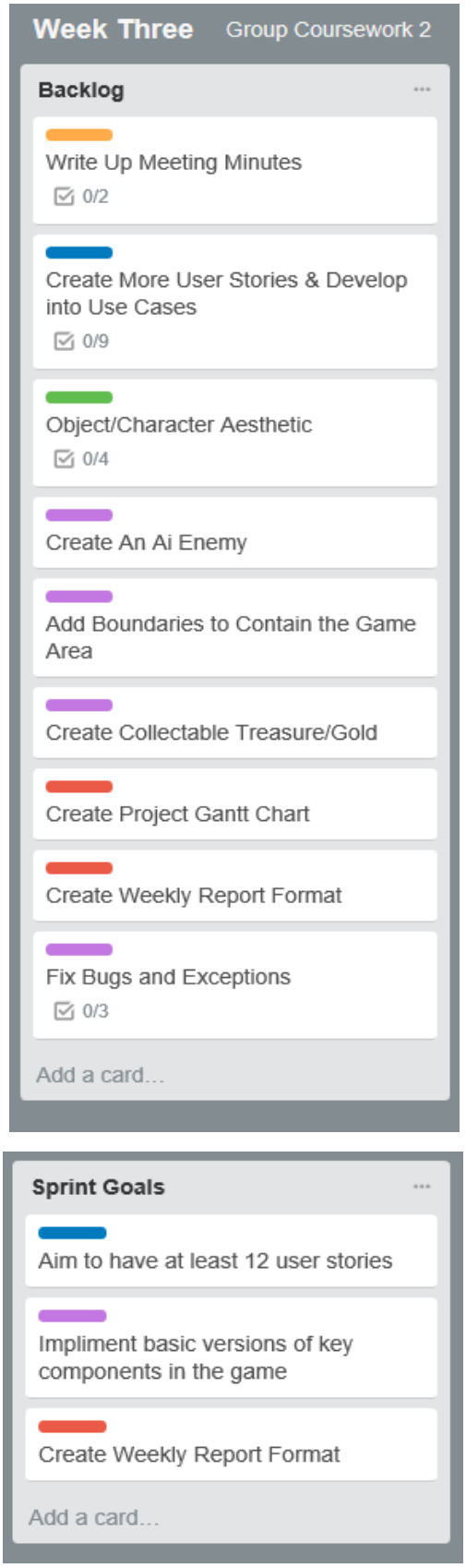


Figure 15: Trello Board Week Three - Sprint Backlog & Goals

User Stories – Version 3

Amendments

We have decided to remove User Story 1 (US1) for the time being as we have set the priority of having a file location for this game as low for the time being. We feel as though it would be better to implement new features and fine tune the current ones as this is the main part of the project the user will be interacting with.

We have also removed US4 due to the priority of this being low.

It was also decided that the starting location of the player was important. With the implementation of an enemy traversing the window, we needed to make sure that these two distinct objects started in locations that were on separate sides of the window. This would give the player an initial window of safety. This relates to US2.

There is currently one room within the game. To give a sense of novelty on each play through, we deemed it a good idea to add a randomised location of the treasure within each instantiation of the room. It was here that we amended US3 to have an additional criterion to implement this.

US4 specifies that the player can exit the current room once they have collected a sufficient amount of treasure. However, we have not yet specified what happens when the player decides to exit this room. We have therefore added the criterion to have this player spawn within a new room once they exit the current room.

US7 initially had the criterion for the bot to seek the player within the game. Whilst we would still like to implement this function, it seemed like a hard problem. We have therefore included the criterion to have this bot move in a specified direction in a specified speed.

The final user story from the previous version, US8, stated that the game should be over once the player has been caught by the bot. However, the criterion within this story did not specify a way to do this. The criterion for a game over screen has therefore been added.

Additions

We have added a further four user stories in addition to the previous version. These are US9, US10, US11, and US12.

User Stories

2. Movement

As a <player>, I want to <move a character around a room> in order to <carry out the tasks within the game>.

- Create a player for a user to interact with
- Allow this player to move around the window
- Create an appropriate starting location for this player

3. Exploration

As a <player>, I want to <explore a room> in order to <obtain treasure>.

- Create obtainable and collectible treasure
- Provide locations of this treasure in the window
- Provide randomised locations of treasure within the room

4. Score

As a <player>, I want to <keep score of the treasure I find> in order to <know when I can leave the current room>

- Create a scoreboard that can be referred to
- Create a message when a sufficient amount of treasure is found
- Provide a new location or room for the player to spawn

5. Map

As a <player>, I want to <track myself on a map> in order to <navigate through the room>

- Create a map that mirrors the action within the window
- Include location of player

7. AI Bot

As a <player>, I want <there to be an AI opponent> in order to <provide competition in the game>

- Create a bot
- Provide automatic movement for this bot
- Create a way for the bot to seek the player
- Create a specified speed and direction for this bot.

8. Collision

As an <AI opponent>, I want to <catch the player> in order to <end the game>

- Create a way to detect collision between the player and enemy
- Create a game over screen.

9. Controls

As a <player>, I want <to know how to move the character> so I can <navigate the room>

- Create a manual in-game to explain how the use can control the player character:
    - Either whilst playing the game, or
    - In an instruction manual within a menu screen

10. Exiting the game

As a <player>, I want <to be able to exit the game> so that I can <stop playing>

- Create a means to close the game
- Create a means to close the pop up window

11. Surrounding and Images

As a <player>, I want to <feel like I am controlling a person> so that <the game has a sense of realism>

- Create an image for player character, as opposed to a coloured dot.

12. Obstacles

As a <player>, I want to <navigate through paths and around obstacles> so that <I get a sense of being within a dungeon>

- Create walls within the room
- Create small paths for the player to go through

Use Cases – Version 2

NB: Blue text will highlight amendments within the use cases each week.

2. Movement

As a <player>, I want to <move a character around a room> in order to <carry out the tasks within the game>.

Use case	Move the character
Summary	This use case allows the player to move a character
Actor	Player
Trigger	The player presses a predefined button to move
Primary Scenario	7. The player character is instantiated within the game window and is visible to the user. 8. Each time the player is instantiated within the room, the specific location must be fixed in order to keep them at a distance from the starting location of the enemy AI bot 9. The player decides in which direction they would like the character to move 10. The player then presses the appropriate button to move the character. 11. The player will move in that particular direction, until 12. The player removes their finger from the button that determined movement. 13. The character comes to a halt and the action is terminated.
Exception Scenario	3. The buttons for movement are not defined or have not been instructed by the user 4. The character does not move. 5. The player character is removed from the map straight away as the enemy AI bot is instantiated at a location next to, or the same as, the player character. This does not give the player a chance to start the game.
Pre-conditions	The player character is instantiated within the window with a set of buttons that determine their movement.
Post-condition	The player will move in the direction specified by the player.

3. Exploration

As a <player>, I want to <explore a room> in order to <obtain treasure>.

Use case	To explore a room and collect treasure
Summary	We would like to give the user a sense of exploration by including collectibles such as treasure.
Actor	Player
Trigger	The player is instantiated within a room and is able to move towards collectibles.
Primary Scenario	5. The player will be placed at a starting location within the game window. 6. The player will be able to traverse the room as instructed by the user playing the game. 7. There will be distinct parts of the room, distinguished by obstacles, such as walls within the room, which will include collectibles, like treasure, that can be interacted with. 8. Within each instantiation of the room, the treasure will be located in randomized locations. This will ensure that the user cannot memorize the locations of the treasure and complete the game with ease. 9. Once interacted with, this will then disappear from the screen and will be the property of the player character
Exception Scenario	3. The buttons for movement are not defined or have not been instructed by the user 4. The instantiation of the walls within the room overlap with the treasure, making this unobtainable by the user. 5. Due to the treasure being randomized, it may be the case that some treasure is instantiated over already existing objects, such as an obstacle within the room. a. To combat this. We need to provide a function that checks whether these randomized treasure objects are intersecting with current objects within each room instantiation.
Pre-conditions	The player character is instantiated within the window with a set of buttons that determine their movement.
Post-condition	The player will interact with a collectible object and this will then disappear from the window.

4. Score

As a <player>, I want to <keep score of the treasure I find> in order to <know when I can leave the current room>

Use case	To keep track of treasure collected
Summary	This use case will allow the user to keep track of the treasure they have collected. This will allow them to know when they can progress within the game.
Actor	Player
Trigger	The player collects one unit of treasure
Primary Scenario	6. The character is adjacent to a piece of treasure in a room 7. The player directs the character to move in the direction of the treasure [Alternative: The player does not direct the character to move in the direction of the treasure] 8. The character moves in that direction thus interacting with the treasure 9. Once the two objects meet the treasure will disappear. 10. A counter will increment by one unit indicating that this has been stored within the player's inventory. 11. Once the player has collected the set amount of treasure within the room, a passage will appear allowing them to progress. 12. When the player has interacted with this passage. They will end up in a specific location within the successive room. This will ensure that they are not interacting with other objects within the room's instantiation.
Exception Scenario	The player does not direct the character in the direction of the treasure 4. The player will move in a direction towards empty space, and, the character will: 5. Either move to an empty space in the room, or not move, as specified in the use case alternative for use case 2. 6. The treasure is still present in its location. 7. When the player is initially located to the successive room, they may be overlapping with a current object, such as an obstacle within the room, causing the player to be immovable.
Pre-conditions	The player has started to navigate the room by moving the character
Post-condition	The player will obtain a piece of treasure

7. AI Bot

As a <player>, I want <there to be an AI opponent> in order to <provide competition in the game>

Use case	Inclusion of an AI opponent
Summary	We would like to include an AI opponent that also traverses the map within a specified movement function
Actor	AI opponent & Player
Trigger	Starting the game
Primary Scenario	3. The AI enemy constantly follows the player at a speed 75% that of the playable character a. There was an issue within this function. It was not as easy to implement as we had assumed. We will therefore look to implement the following function instead: The AI enemy will move about in a given path bouncing off of any of the objects in its path. For example, it can be that the set speed in the x direction is the same as the set speed in the y direction, so as to move diagonally about the screen from a given starting location 4. The enemy catches the player which results in a game over message.
Exception Scenario	1. The AI enemy becomes trapped in a part of the room and is unable to move. For example, if the player is directly in front of the enemy, but there is a wall in the way, the enemy will continue forward indefinitely whilst the player is in this position as they have been programmed to only move in the direction of the player. 2. The enemy is instantiated over a current obstacle within the room and causes this to become trapped in place, not being able to traverse the window.
Pre-conditions	The game is open and the player has started the game
Post-condition	A player-seeking enemy is in the room A random walk object is included in the room.

9. Controls

As a <player>, I want <to know how to move the character> so I can <navigate the room>

Use case	Controls for Gameplay
Summary	This use case will specify the need for the inclusion of control instructions to be accessible within the game.
Actor	System
Trigger	1. Choosing the instruction, or 2. A small display within the game specifying the controls.
Primary Scenario	1. The player opens the game and is taken to a menu screen. 2. The player has the option to open an instruction screen from the menu where the controls are displayed. 3. Or, the movement controls will be displayed within the game in a corner of the screen.
Exception Scenario	
Pre-conditions	The game is open
Post-condition	The player will be able to refer to an instruction guide on how to control the player.

10. Exiting the game

As a <player>, I want <to be able to exit the game> so that I can <stop playing>

Use case	A means to exit the game window or ‘gameplay’ mode.
Summary	To provide a way for the user to exit the game when they would like to stop playing or when they ‘die’ in the game.
Actor	Player
Trigger	Boredom, completion, or death. Hopefully the latter two.
Primary Scenario	1. The player clicks the exit button in-game and then clicks the X button on the top right of the window to close the game window. 2. The player is ‘caught’ by the AI bot and receives a game over message. They will then subsequently be returned to the menu screen where they can either reattempt or close the game.
Exception Scenario	
Pre-conditions	None
Post-condition	The game window is closed

11. Surrounding and Images

As a <player>, I want to <feel like I am controlling a person> so that <the game has a sense of realism>

Use case	To include an image in place of the player character
Summary	This use case will provide an image of a person in place of the current graphics object that displays a circle.
Actor	
Trigger	
Primary Scenario	The user will interact with a sprite as opposed to shapes defined in the graphics class.
Exception Scenario	
Pre-conditions	-
Post-condition	The game will have a set of images to interact with.

12. Obstacles

As a <player>, I want <navigate through paths and around obstacles> so that <I get a sense of being within a dungeon>

Use case	Instantiate obstacles within each room
Summary	We want to create objects in each room that will obstruct the player, forcing them to find a way around these.
Actor	Player
Trigger	The player will move towards an obstacle
Primary Scenario	The user will instruct the player to move in a particular direction. If they move in the direction of an obstacle, and finally reach this obstacle, once their boundaries intersect, the player will no longer be able to move in that particular direction.
Exception Scenario	The player character will be able to move through the obstacle: • This may happen due to the amount of pixels the player can move does not correlate with the location of the object. For example, if the player moves 6 pixels per 'move' and the wall objects size and location do not correlate with this as the play approaches.
Pre-conditions	A room and location for each object must be free
Post-condition	-

CRC Analysis – Version 3

V3: New: Handler, Room, Door, Obstacle.

Classes

- Game
- Window
- Handler
- Game Object (Abstract Class)
- Room
- Player
- Enemy
- Treasure (Gold)
- Door
- Obstacle

Game (Main class)	
• Creation of objects	Everything

Window	
• Creates the window for the game	Game

Game Object (Abstract Class)	
• Object Location • Object Speed • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Objects displayed on the screen

Handler	
• Tick • Render • Add Object • Remove Object	everything

Room	
• Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Enemy Treasure

Player	
• Object Location • Object Speed • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Enemy Treasure Room

Enemy	
• Object Location • Object Speed • Object ID	Game Object Enemy Treasure Room

Treasure (Gold)	
• Object Location • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Enemy **Room**

Door	
• Tick • Render • Change Room and Player location	Player Room Game

Obstacle	
• Object location • Detect collision	Player Enemy Room Game

Class Diagram

In the third week, we have ten classes. We add a handler for controlling objects. Also added are room, door and obstacle as the new classes.

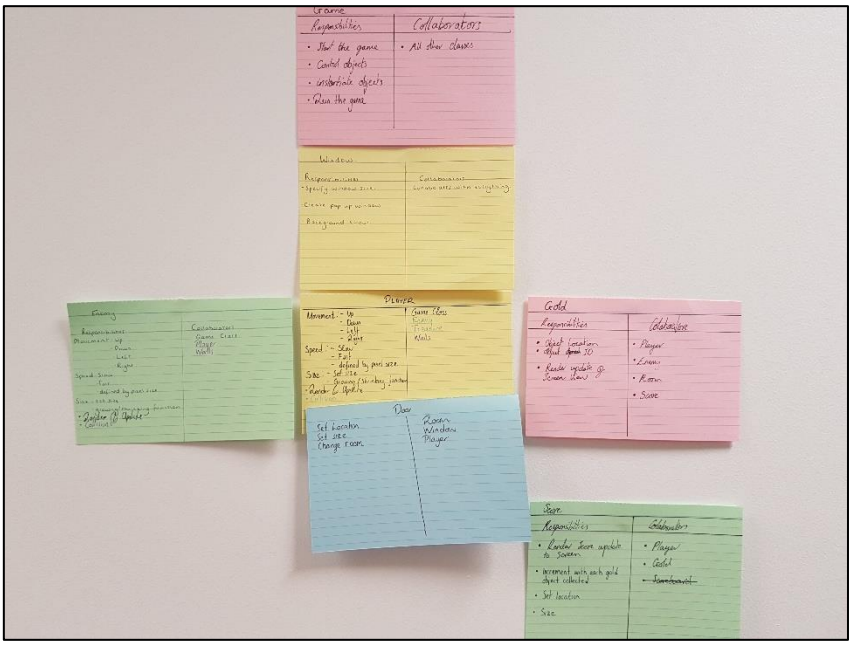


Figure 16: Week Three - Class Card Association

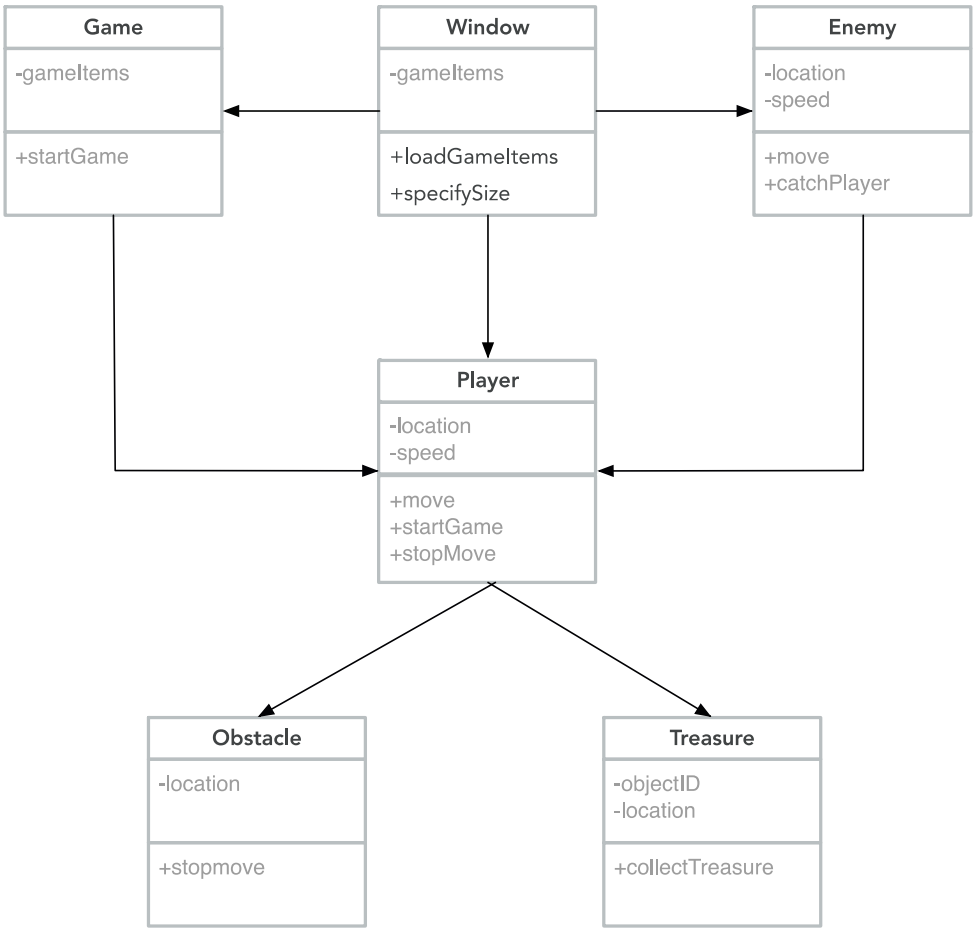


Figure 17: Week Three - Class Diagram

Prioritisation

Table 3: Week Three Priority Matrix

User Story ID	User Story Name	Primary Actor	Complexity[a]	Priority[b]	Completed[c]
US2	Movement	Player	Low	1	Yes
US3	Exploration	Player	Medium	3	Partially
US4	Score	Player	High	3	No
US7	AI Bot	AI Bot	Low	2	Yes
US8	Collision	AI Bot	Medium	4	Partially
US9	Controls	Player	Medium	3	No
US10	Exiting the game	Player	Medium	2	Partially
US11	Surrounding and images	System	Low	3	No
US12	Obstacles	System	Low	2	No

[a]High, medium, low

[b]1 (high) – 5 (low)

[c]Yes, no, partially

Justification

US2: We have decided to create an appropriate starting location for the player to prevent any objects overlapping upon room instantiation.

US3: The randomised location of treasure within the room is the new criterion added to this story. The Random class will make this easy, but problems may arise if these random locations overlap with specified locations of other objects. We therefore designated a medium complexity.

US4: The addition of creating a spawn location not within this room will be a hard problem, so we have decided to set this as a high complexity.

US7: The problem of creating a seeking function was a hard problem. As a temporary measure, we have created a set speed in the x and y direction to create a diagonal trajectory for this object.

US8: A Game Over screen should not be too difficult to display. We have the use of Java's AWT library to aid us with this. We have decided to keep to priority as a 3 as we can worry about the game over screen later down the line.

US9: The controls for the game are certainly necessary. But, again, as with US8, we do not need to worry too much about this until the game is set to be released.

US10: This story was included somewhat retrospectively. Whilst we can already close the game window, we would also like to be able to close the game without closing the window. We have set this to a medium complexity.

US11: The inclusion of an image in place of the graphics object should be an easy task. Whilst it will greatly improve the interface, it is not something of huge importance at this moment in time.

US12: We have already created boundaries for the room. The inclusion of extra objects with boundaries should not be a hard task. The priority has been set high and complexity low.

Game Design & Implementation

Words in blue is what is new in current version.

- A suitable size WINDOW.
- A PLAYER can move up, down, left and right, collect treasure.
- An ENEMY chases the player.
- A DOOR enables the player to access to further rooms.
- Some TREASURE can be collected by the player.
- BOUNDARY uses to stop both the player and bots move outside.
- OBSTACLE stops bots and the player moving.

User interface design

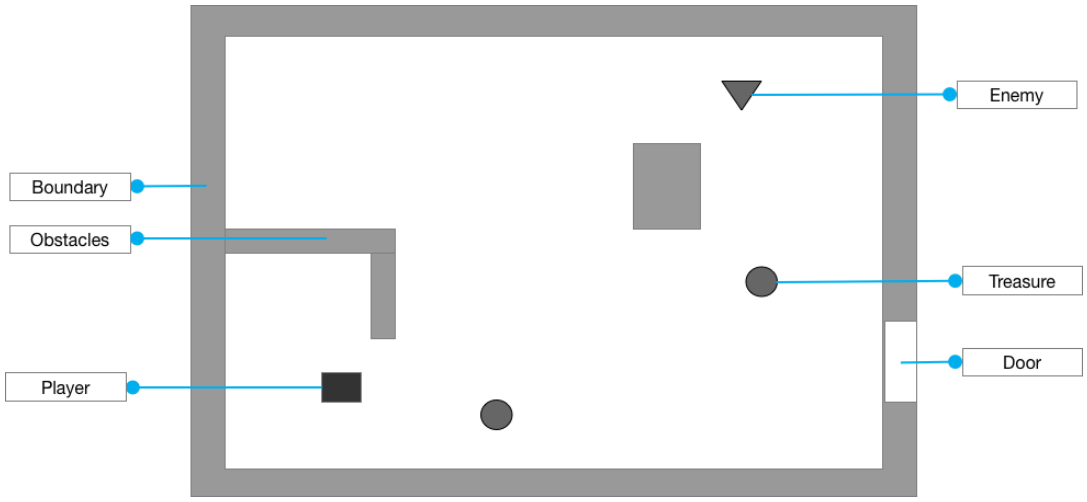


Figure 18: User Interface Design - Week Three

Current application view

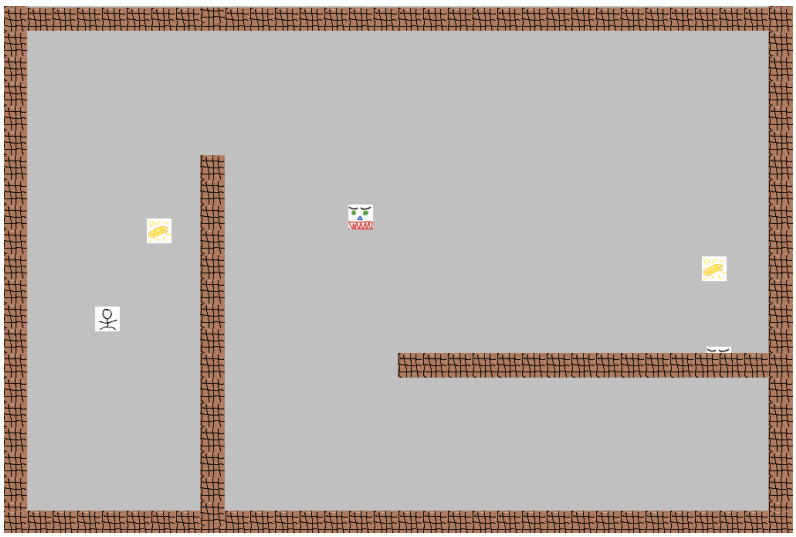


Figure 19: Implemented Game View - Week Three

Testing

Failed enemy-wall collision test.

Set up an enemy and a player. Four directions were tested, for example: enemy is moving to right and there is a wall on the right side of the enemy. (Before the method is finished, the bar on the top left is red representing failing)

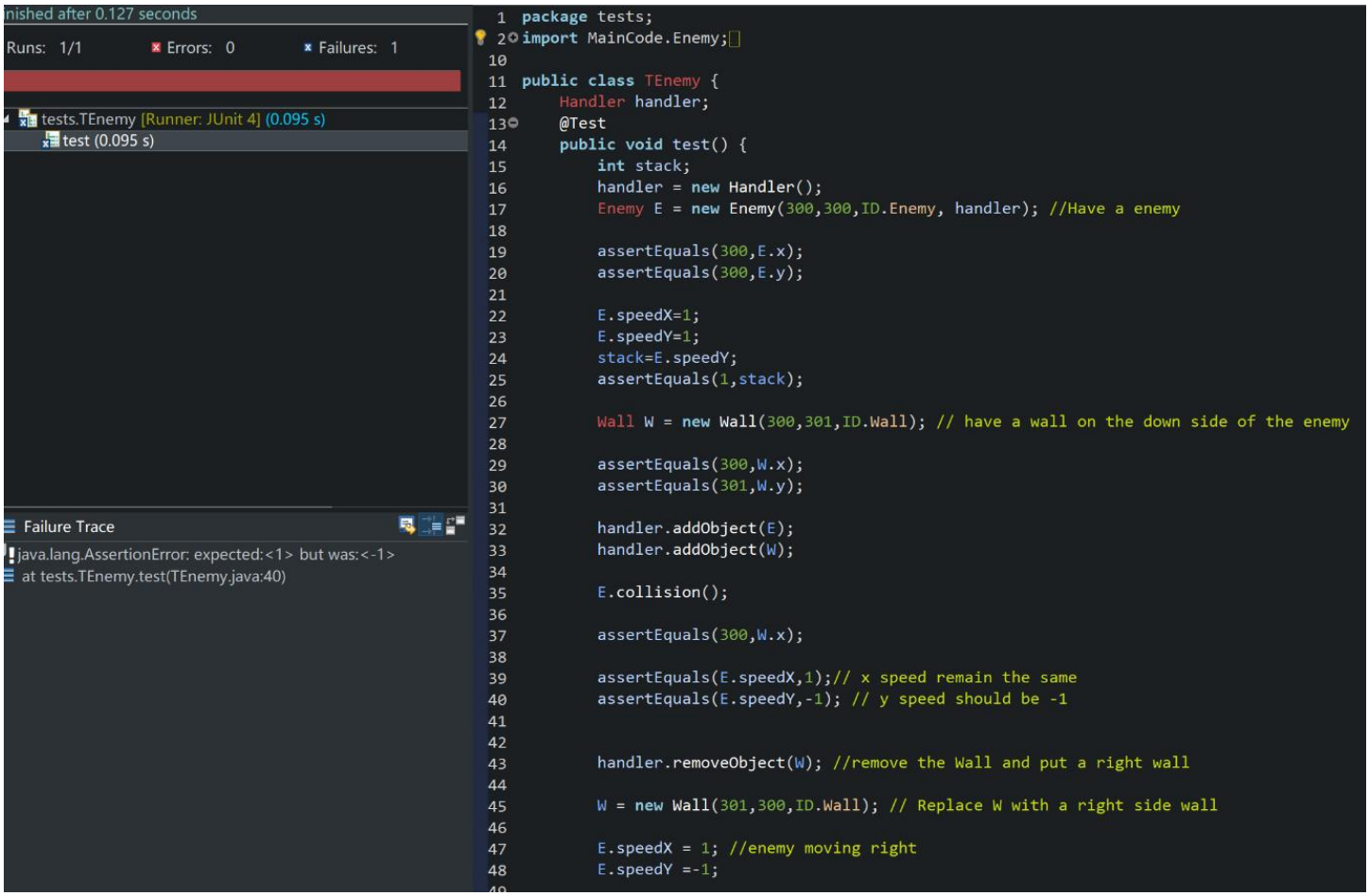


Figure 20: J Unit Test – Wall/Enemy Collision Failed Test

Enemy Collision method passed four directions test.

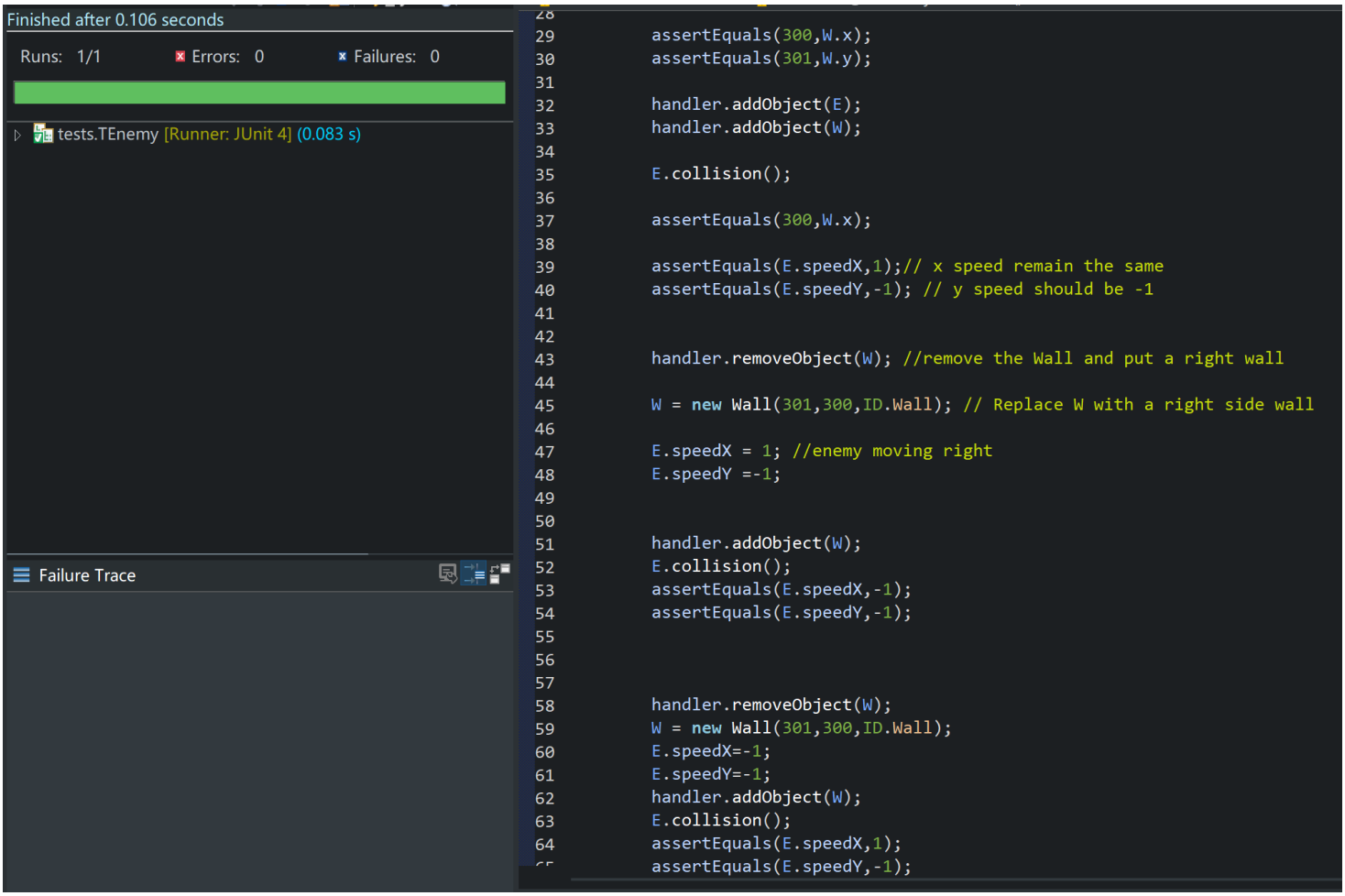


Figure 21: J Unit Test – Wall/Enemy Collision Passed Test

Sprint Retrospective

Week three

With a much more comprehensive set of user stories, use cases and CRC's now developed, a clearer course of action was becoming apparent. It was agreed that while it was still important to consider amendments and developments, the creation of new stories as a high priority could now stop.

The compilation of a weekly report enabled the team to collectively gain a more complete view of the project. So far the workload has generally divided team members into an either documentation or code creation centred role. Laying out progression and decision making on paper has helped members understand and engage with the side of the project they had been less familiar with. We decided that we should start to carry out this activity every week.

The running version of the game is starting to show the structure of a playable prototype. Introduction of an enemy and pieces of gold to collect has allowed us to start a certain level of play testing. By having all team members test the game the process of finding bugs and refining ideas has become more effective. We also coupled play testing sessions with pair/team programming so that new features and any recent refactoring of code could be explained, and fixes for smaller bugs could be dealt with quickly.

Some interesting bugs were observed this week. The boundary walls added to the game, successfully contained and guided player movement. But, when an enemy monster was introduced it was occasionally seen to pass through the boundary. Some experimenting with the monster class parameters identified that higher movement speed could solve the problem. This, however, in turn was deemed to negatively affect gameplay and the quick fix solution was discounted.

Pulling the game from our shared Github repository and importing it into an IDE was proving to be not as straight forward as anticipated. When it was successfully imported, the game behaved quite differently on different computers with lag occurring on less powerful laptops.

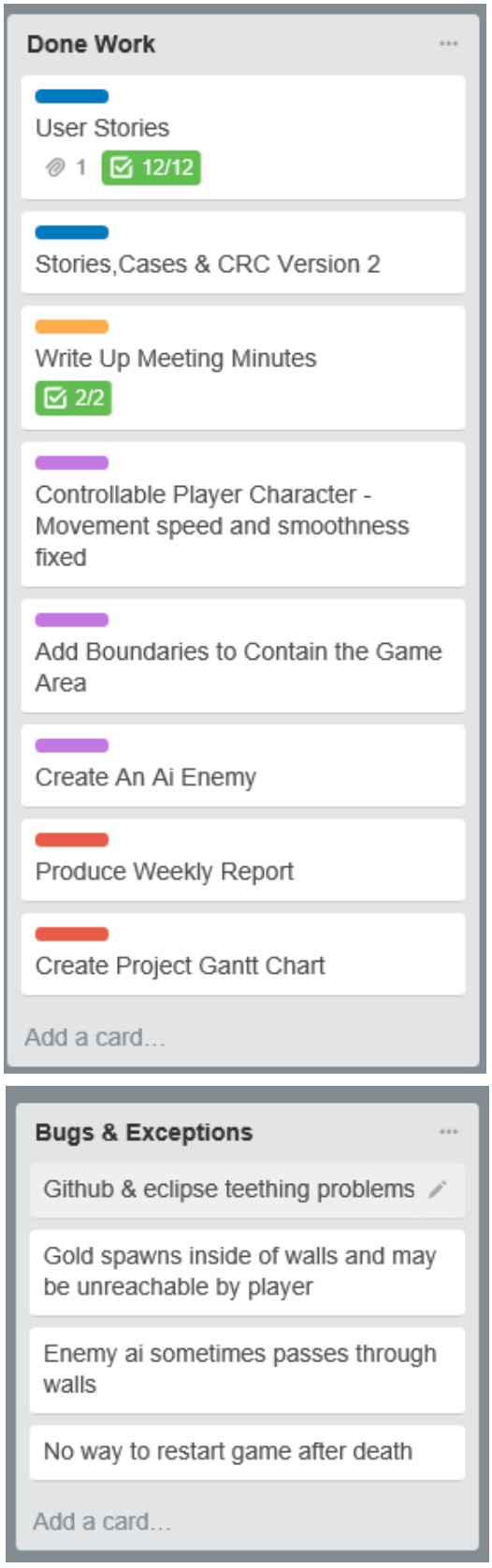


Figure 22: Trello Board Week Three - Done Work & Exceptions

Week Four

Overview

As of now, our game has the following features:

- A rectangular room contained by four walls on each side.
- A player character that can be controlled and moved by a user
- A set of walls within the room that obstruct the players movement
- An enemy that traverses the room with a specified speed and direction
- A set of treasure objects that can be collected by the player character
- An image in place of the shape object for the player within the graphics class

The player is currently restricted to one room, which comprises the whole dungeon. It was specified by the customer that there are to be multiple rooms included within the dungeon. It was therefore time for us to think about replicating what we have so far in order for the user to be able to progress. After some deliberation, we came to the decision to hold separate instances of room within an array. *Note the problems we had instantiating this within the bugs section.*

The inclusion of one enemy within a map of such a size does not seem to add too much difficulty to the game. We played around with amending the size and speed of the enemy, but this would often cause troubles with collision due to the pixel size and collision detection method that we had set up. We instead decided to provide a set of enemies, which would move in distinct directions and at differing speeds. This added to the difficulty of the game and also made it feel as though there was more happening within the game.

The image that was added for the player character was a nice addition as it gave the game a bit more substance towards a dungeon crawler-like game, and with the addition of the walls, the game started to have a presentable interface. However, we still had a red square and yellow circle representing the enemy and treasure, respectively. It was a clear progression to add images for these objects too. As a result, we would have a full user interface that was represented by images as opposed to predefined shapes. With this, we would have a product that represents the requirements of our customer.

In addition to the inclusion of multiple enemies, we also had the implementation where if the player is ‘caught’ by the enemy, then they would disappear from the map and the game would carry on until the user exited the game. This is partly what we want, where if the player is caught, then the game comes to an end, but this seems a bit abrupt. Instead, we decided to include a health bath within the game and have this decrease whenever the enemy interacts with the player. Upon too many interactions, we would like to have the player ‘die’ and disappear from the map, revealing a sign stating game over.

Minutes of Customer Meeting

Group Name: Blue
Project Name: Dungeon Explorer- A multi-player online game
Date of Meeting: 22/11/2017

Start and End Time: 17:06-17:07

Objective
Weekly review to increase on the functionality of the game.

Attendees
Ben Boreham, Lucy Nelson, Rikki Hodder, Ruowen Cheng, Yalin Shi, Yu Jiaping

Agenda
Review of documentation
Game overview

Decisions
Review of documentation
A copy of the weekly report was shown to the customer. Feedback included adding a contents page to make it easier to read. We must also ensure that all user cases, user stories and CRC analysis pages are versioned. Sections of the report need to be clearly labelled. The customer would like to review the weekly report next meeting to see how the project has evolved. Once all the weekly reports are constructed, reports can be integrated together to make them into a narrative for the final report.

It is vital that documentation corresponds to the code. This have been the case as user cases are prioritised. Cases with highest priority are implemented first.

Product documentation includes a user manual. This is a guide to how the player plays the game. You can include it in the game menu but would be better for consistency if this was produced in a document. Installation guide should be very straight forward. The game can be exported on eclipse. Maintenance guide is essentially there for a developer. Consider questions such as "what would I like to know if I needed to change the program to add sometime (e.g. a second monster)? What would I do if I wanted to change the size of the screen". The developer will need to be navigated to certain areas of the code that will be affected to find the key functionality and by considering various extension scenarios.

Game Overview
We have gold, health bar and a monster with random movement new to the customer this week. A user story which is being considered is whether the bot will follow the player. Next on the priority list is door/warp, rooms and scoreboard. Unsure whether the warp or passage that opens up when certain amount of gold is collected. Levels could be considered e.g. in the next room there will be more gold to collect, more monsters or increased speed of monsters.

Action
Weekly report, meeting minutes and prototype of game with added functionality (e.g. obstacles, door/warp and levelling) for next meeting.

Minutes of Scrum Meeting

Group Name: Blue

Project Name: Dungeon Explorer- A multi-player online game

Week 4: 22/11/2017 - 28/12/2017

Objective
Review of customer meeting and allocation of tasks

Attendees
Ben Boreham, Lucy Nelson, Rikki Hodder, Ruowen Cheng, Yalin Shi, Yu Jiaping

Agenda
Documentation
Game Overview

Decisions
Documentation
Weekly report needs to be created. Keep updating minutes and versioning of CRC, user stories and user cases.

Game Overview

- Addition of more enemies- there is an issue making the game harder with only one bot. The addition of more bots will make it harder for the player to complete the game
- More rooms via warp- to add a sense of adventure another room was added this week to the game. This room had obstacles, monsters and treasure.
- Produce images for treasure and enemy to give the game more substance
- Health bar- once the enemy hits the character we initially wanted to player to disappear. Health bar means that player can interact move with the bot before the game is over.

Action
Create multiple rooms
Version 3 of user stories, user cases and CRC analysis
Meeting minutes
User stories and cases
CRC analysis
Create and implement in game images
Create multiple enemies
Give player a health bar
Produce weekly report

Product Backlog

Week Four

In the week three customer meeting, the report format developed was vetted as a good template for documentation. The documentation for this week, and future weeks, would be provided in this report style.

Progress has been made on bringing together the game framework, but aesthetically it is still lacking. This week's sprint objective is to improve the game's image and make it look and feel more like a playable game. This will be done primarily by designing and overlaying pictures.

The addition of multiple enemies as well as the introduction of a health system needs to be brought into the game. Some consideration will be needed for how the multiple enemies will interact with each other and the amount of damage they cause.

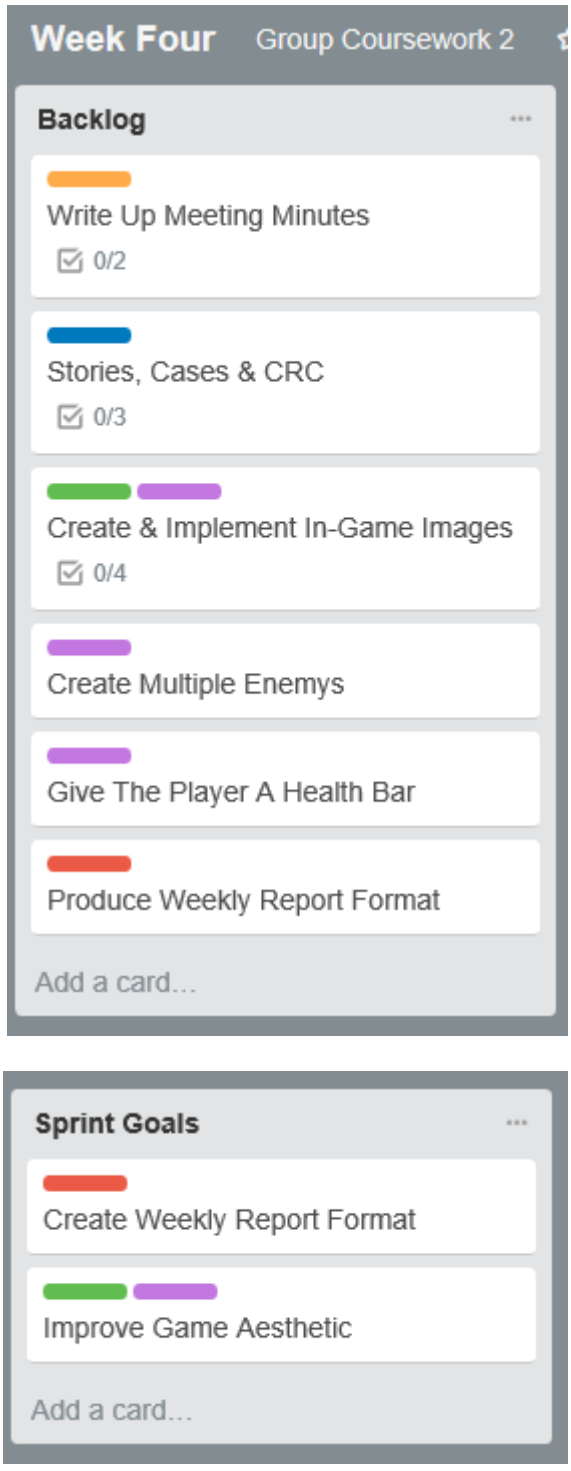


Figure 23: Trello Board Week Four - Product Backlog & Sprint Goals

User Stories – Version 4

Amendments

As of now, US2 has been fully implemented. The player has a starting location which is set at an appropriate distance from the enemies starting location, and the user is able to move a player around the room given the defined keys. We have therefore removed this from the list.

US5 is a feature that is specifically stated on our list of customer requirements, but with the implementation of the game that we have designed so far, this user story is becoming less and less desirable. For example, we are able to see the whole room at all times within the game, and any further room, at this current time, is going to be designed in the same way. We will therefore remove this from the user stories for now and hold a discussion with our customer to see if they are happy with this decision.

With US6, we would like to add a progress bar that increases in correlation with the amount of treasure that has been collected. This will reach the maximum when all of the treasure within the dungeon has been created. We deemed this to be appropriate as the player may forget which rooms have had all of their respective treasure collected once multiple rooms have been implemented.

Multiple enemies are to be included within our game, so it makes sense to update US7 and the criterion within this. Rather than specifying one enemy, we need to be able to provide functions for each separate enemy within the game.

We have removed US8 this week as we have deemed the Game Over screen to be a low priority in regard to where we are within the timeline. We hope to act on this within the next two weeks.

US9 has been removed as we have subsequently decided that while this is needed to inform the user, this is not as essential as some of the other user stories.

US10 has the addition that the player can return to the menu screen. This has not yet been added as a user story and so we will add this in as a user story in itself.

We have also decided to add an image to each of the objects included within the game, as opposed to just the player character that the user controls. This will give a dungeon-like feel to the game and make it seem like a more refined product. This has been represented within US11.

We finally add one more criterion to US12, which is to make sure that any object included within the game cannot surpass the walls that branch into the traversable room. This related to a glitch that we are currently having within our collision detection.

Additions

We have added one additional user story this week, US13. This specifies a menu screen upon the game opening.

User Stories

6. Win Condition

As a <player>, I want to <find a sufficient amount of treasure> in order to <escape the dungeon>

- Include multiple pieces of treasure
- Provide a way to keep count of the treasure collected
- Provide an exit to the room
- Provide a new location or room for the player to spawn
- Create a progress bar increases with amount of treasure collected.

7. AI Bot

As a <player>, I want <there to be multiple AI opponent> in order to <provide competition in the game>

- Create a bot
- Provide automatic movement for each bot within this game
- Create a way for the bot to seek the player
- Create a specified speed and direction for each bot within this game
- Create a starting location for each bot within the game

10. Exiting the Game

As a <player>, I want <to be able to exit the game> so that I can <stop playing>

- Create a means to close the game
- Create a means to close the pop up window
- Return to the menu screen once the player has quit the game.

11. Surrounding and Images

As a <player>, I want to <feel like I am exploring a dungeon> so that <the game has a sense of realism>

- Create an image for player character, as opposed to a coloured dot.
- Create a sprite for each object included within the game

12. Obstacles

As a <player>, I want to <navigate through paths and around obstacles> so that <I get a sense of being within a dungeon>

- Create walls within the room
- Create small paths for the player to go through
- Make sure all objects are restricted by these walls

13. Menu Screen

As a <player>, I want to <enter a menu screen> so that <the game does not start until I decide>

- Create a menu screen that opens when the game initially opens.
- Provide a set of options for the player to choose, e.g. instructions, play, etc.

Use Cases – Version 3

6. Win Condition

As a <player>, I want to <find a sufficient amount of treasure> in order to <escape the dungeon>

Use case	To create a lower limit for the amount of treasure a player can collect to escape the dungeon
Summary	This use case will provide a means for a player to finish after collecting a certain amount of treasure.
Actor	Player
Trigger	The player collects the set amount of treasure. We could set this to 75% of the total treasure in the dungeon.
Primary Scenario	5. The player obtains the piece of treasure that will exceed the lower limit needed to progress. 6. Each time the player obtains a piece of treasure within the whole game, there will be an instance of a progress bar that will increase in accordance with the amount of treasure collected in regard to the total amount of treasure included. 7. A door will appear at a set location in the room that will allow the player to leave the dungeon. 8. The player will move toward this door and interact with it. 9. Once it has been interacted with, the player will receive a completion message.
Exception Scenario	1. The player may be 'hit' by an enemy before the door image appears. In this case, the player will receive a game over message. 2. The progress bar is not linked to the total amount of treasure within the dungeon and therefore does not correlate to the correct degree. This could give the effect of the player completing before the total amount of treasure is collected.
Pre-conditions	The player is able to obtain a set amount of gold by navigating the room and avoiding the enemy
Post-condition	The player will obtain enough treasure to open up a doorway to a dungeon exit.

7. AI Bot

As a <player>, I want <there to be an AI opponent> in order to <provide competition in the game>

Use case	Inclusion of multiple AI enemies
Summary	We would like to include multiple AI opponents that also traverse the map within a specified movement function
Actor	AI opponent & Player
Trigger	Starting the game
Primary Scenario	5. The AI enemy constantly follows the player at a speed 75% that of the playable character a. There was an issue within this function. It was not as easy to implement as we had assumed. We will therefore look to implement the following function instead: The AI enemy will move about in a given path bouncing off of any of the objects in its path. For example, it can be that the set speed in the x direction is the same as the set speed in the y direction, so as to move diagonally about the screen from a given starting location b. We will also give each separate instantiation of the enemies a different speed and direction to move in. We may also amend the sizes of some of the enemies to have varying senses of danger. 6. The enemy catches the player which results in a game over message.
Exception Scenario	3. The AI enemy becomes trapped in a part of the room and is unable to move. For example, if the player is directly in front of the enemy, but there is a wall in the way, the enemy will continue forward indefinitely whilst the player is in this position as they have been programmed to only move in the direction of the player. 4. The enemy is instantiated over a current obstacle within the room and causes this to become trapped in place, not being able to traverse the window.
Pre-conditions	The game is open and the player has started the game
Post-condition	A player-seeking enemy is in the room A random walk object is included in the room. Multiple enemies are within each room.

10. Exiting the game

As a <player>, I want <to be able to exit the game> so that I can <stop playing>

Use case	A means to exit the game window or 'gameplay' mode.
Summary	To provide a way for the user to exit the game when they would like to stop playing or when they 'die' in the game.
Actor	Player
Trigger	Boredom, completion, or death. Hopefully the latter two.
Primary Scenario	3. The player clicks the exit button in-game and is returned to the menu screen. The player will then have the option to click the X button within the window frame to close the game window. 4. The player is 'caught' by the AI bot and receives a game over message. They will then subsequently be returned to the menu screen where they can either reattempt or close the game.
Exception Scenario	
Pre-conditions	None
Post-condition	The game window is closed

11. Surrounding and Images

As a <player>, I want to <feel like I am exploring a dungeon> so that <the game has a sense of realism>

Use case	Inclusion of images (sprites) in place of current objects
Summary	This use case will provide images (sprites) in place of all of the current objects within the room.
Actor	User
Trigger	
Primary Scenario	1. The user will interact with a sprite as opposed to shapes defined in the graphics class.
Exception Scenario	1. The images imported to be used as the sprites for each object do not have the correct dimensions to act appropriately for our current methods.
Pre-conditions	
Post-condition	The game will have a set of images to interact with.

12. Obstacles

As a <player>, I want <navigate through paths and around obstacles> so that <I get a sense of being within a dungeon>

Use case	Instantiate obstacles within each room
Summary	We want to create objects in each room that will obstruct the player, and every additional object, forcing them to find a way around these.
Actor	Movable objects
Trigger	An object will move towards, or be instantiated on, an obstacle.
Primary Scenario	1. A movable object will be traversing the room in the direction of one of the obstacles instantiated within the room 2. Once the objects collide, this will be detected and the moving object will rebound away from this in a natural manner, with the rebound being in the direction of the symmetry of the movement towards the obstacle.
Exception Scenario	The moving object will be able to move through the obstacle: • This may happen due to the amount of pixels the object can move does not correlate with the location of the object. For example, if the object moves 6 pixels per 'move' and the wall objects size and location do not correlate with this as the play approaches.
Pre-conditions	A room and location for each object must be free
Post-condition	The object rebounds from the obstacle

13. As a <player>, I want to <enter a menu screen> so that <the game does not start until I decide>

Use case	Create a menu Screen
Summary	This use case will detail the specifications for our main menu screen.
Actor	User
Trigger	The player has: 1. Opened the game for the first time 2. Successfully completed the game 3. Been unsuccessful. I.e. caught by the enemy.
Primary Scenario	The user will be navigated to the menu screen as specified in the trigger section. 1. The player will have the option to start the game 2. The player will have the option to exit the game 3. The player will have the option to view the gameplay controls
Exception Scenario	
Pre-conditions	The game has been started
Post-condition	A menu screen is available to the user

CRC Analysis – Version 4

V4: New class: Score, Menu Screen, Map

Technical classes – These will be the classes that organise the technical details of the game, such as handling each object and creating a window to open in order to play the game
Classes with instances involved in the game

Classes

- Game
- Game Object (Abstract Class)
- Handler
- Window
- Room
- Menu Screen
- Map
- Player
- Enemy
- Treasure (Gold)
- Door
- Obstacle
- Score (Have not built)

Game (Main class)	
• Creation of objects	Everything

Window	
• Creates the window for the game	Game

Game Object (Abstract Class)	
• Object Location • Object Speed • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Objects displayed on the screen

Handler	
• Tick • Render • Add Object • Remove Object	Everything

Room	
• Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Enemy Treasure

Menu Screen	
• Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	

Player	
• Object Location • Object Speed • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Enemy Treasure Room Door Score

Enemy	
• Object Location • Object Speed • Object ID	Game Object Enemy Treasure Room

Treasure (Gold)	
• Object Location • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Enemy Room **Score**

Door	
• Tick • Render • Change Room and Player location	Player Room Game

Obstacle	
• Object location • Detect collision	Player Enemy Room Game

Score (Have not built)	
• Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Treasure

Map	
• Object ID • Object Location • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Enemy

Class Diagram

In the fourth week, we have thirteen classes. We add score, menu screen and map as the new classes.

When the player collect the treasure, the score will increase, when the enemy touch the player, the health score will decrease.

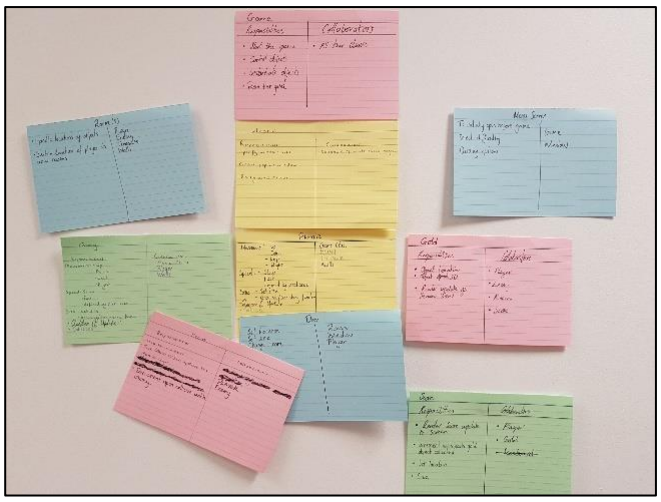


Figure 24: Week Four - Class Card Association

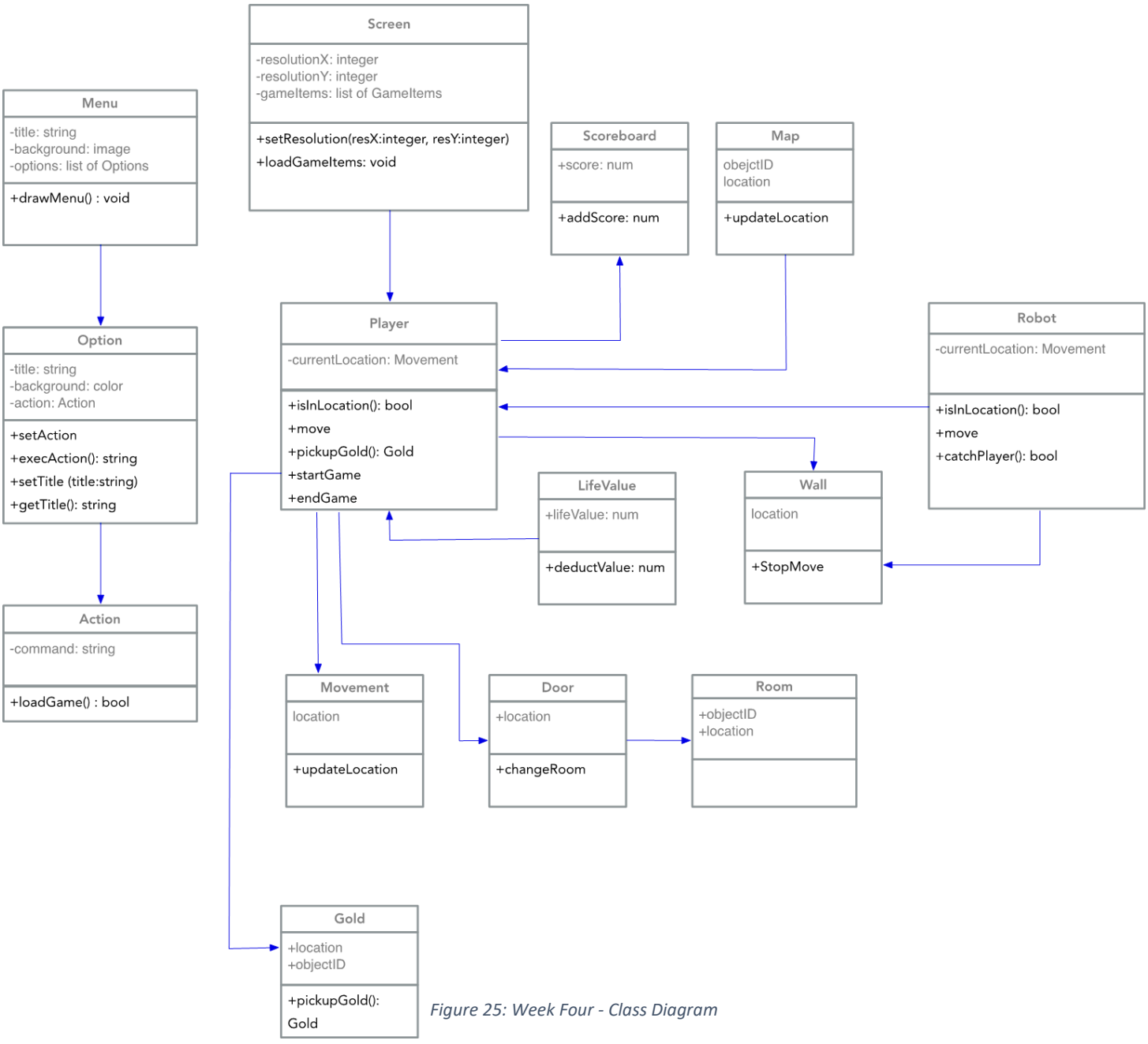


Figure 25: Week Four - Class Diagram

Prioritisation

Table 4: Week Four Prioritisation Matrix

User Story ID	User Story Name	Primary Actor	Complexity[a]	Priority[b]	Completed[c]
US6	Win condition	Player	Low	3	Partially
US7	AI bot	AI bot	Low	2	Yes
US10	Exiting the game	Player	Low	3	Partially
US11	Surrounding and images	System	Low	2	Yes
US12	Obstacles	System	Hard	1	Partially
US13	Menu screen	System	Medium	3	No

[a]High, medium, low

[b]1 (high) – 5 (low)

[c]Yes, no, partially

Justification

US6: With the knowledge of the graphics class and an implemented collision method, we should be able to create a health bar that decreases with each collision between the enemy and player. Therefore, the complexity has been set to low and priority as a 3.

US7: This task is of low complexity as each additional criterion has been successfully implemented already. We only need to add this to further enemies within the game.

US10: Since we have not yet created the menu screen, we will not be able to implement this function. However, with the code we have, we should be able to implement this once it has been created. We have therefore set this particular complexity to low with a medium priority.

US11: We have already implemented images within our game. We can therefore carry out this task with ease for the rest of the objects.

US12: Giving each object the same dimension should help us to create collisions and boundaries for all combinations of collisions, but this has yet to be implemented successfully. We have therefore deemed this a hard task.

US13: A main menu screen might be a relatively hard problem. We have set this to a medium complexity and a medium priority for now.

Game Design & Implementation

In version 4, some graphic elements are replaced by simple pictures. A warp door which enables a player to access to further rooms is also included.

Words in blue is what is new in current version.

- A suitable size WINDOW.
- A PLAYER can move up, down, left and right, collect treasure.
- An ENEMY chases the player.
- Some TREASURE can be collected by the player.
- BOUNDARY uses to stop both the player and bots move outside.
- OBSTACLE stops bots and the player moving.
- A WARP DOOR enables the player to access to further rooms.

User interface design

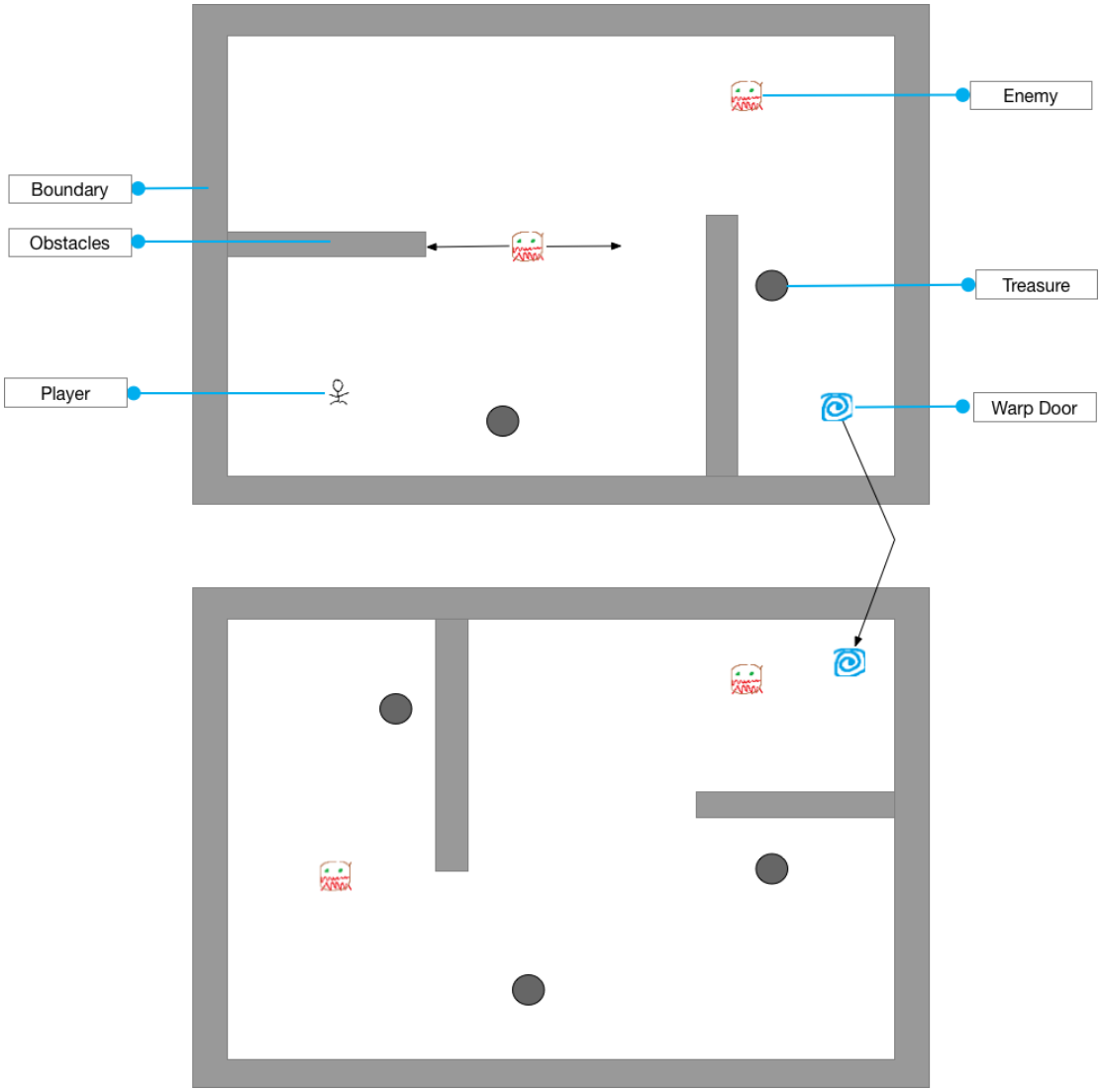


Figure 26: User Interface Design – Week Four

Current application view

View 1 (Room 1)

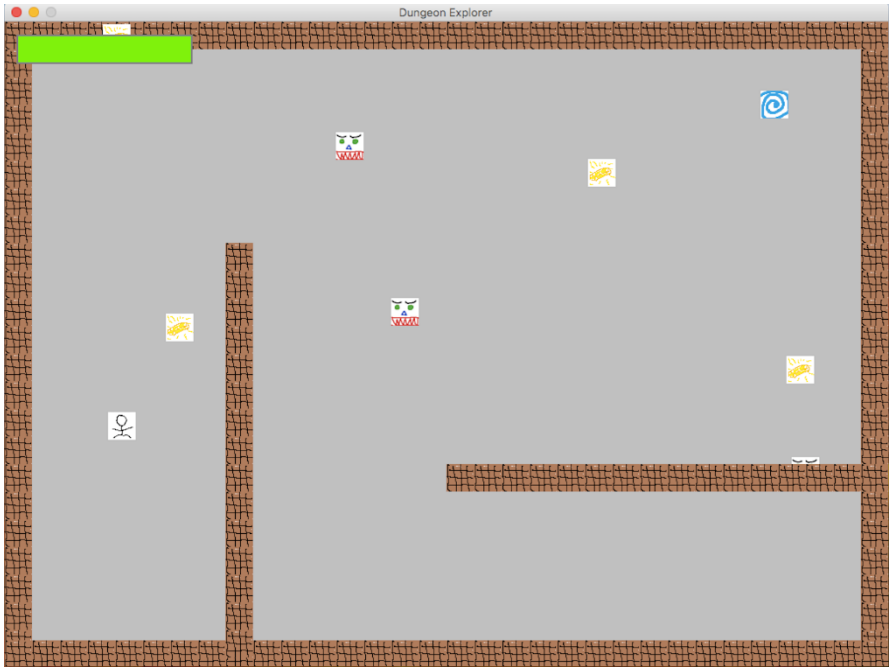


Figure 27: Implemented Game View - Week Four: Room 1

View 2 (A player access to Room 2 via the warp door)

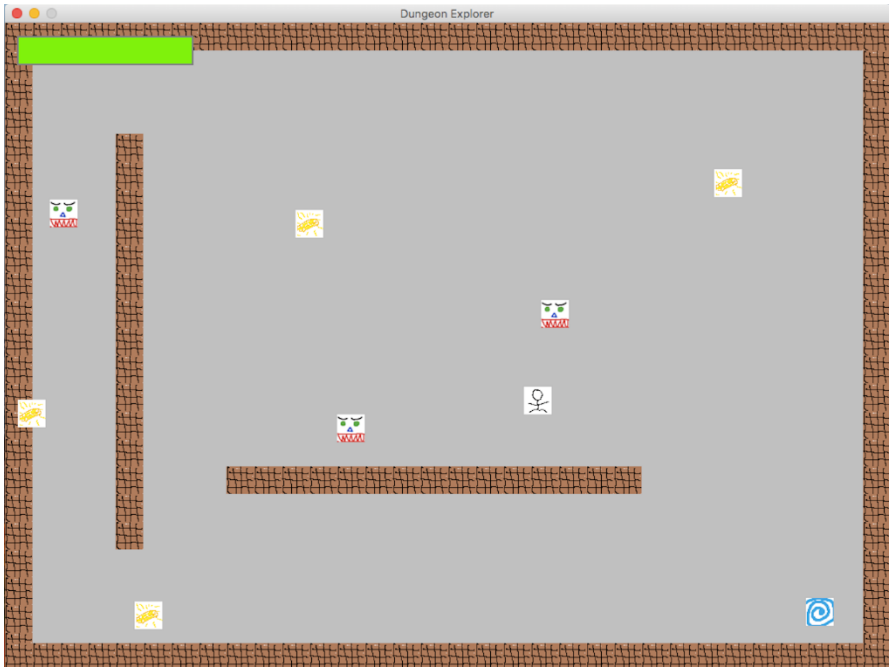


Figure 28: Implemented Game View - Week Four: Room 2

Testing

Health test. Have a Health bar, test 100 or not, have a player and enemy collide so that damage occurs, recheck the HEALTH.

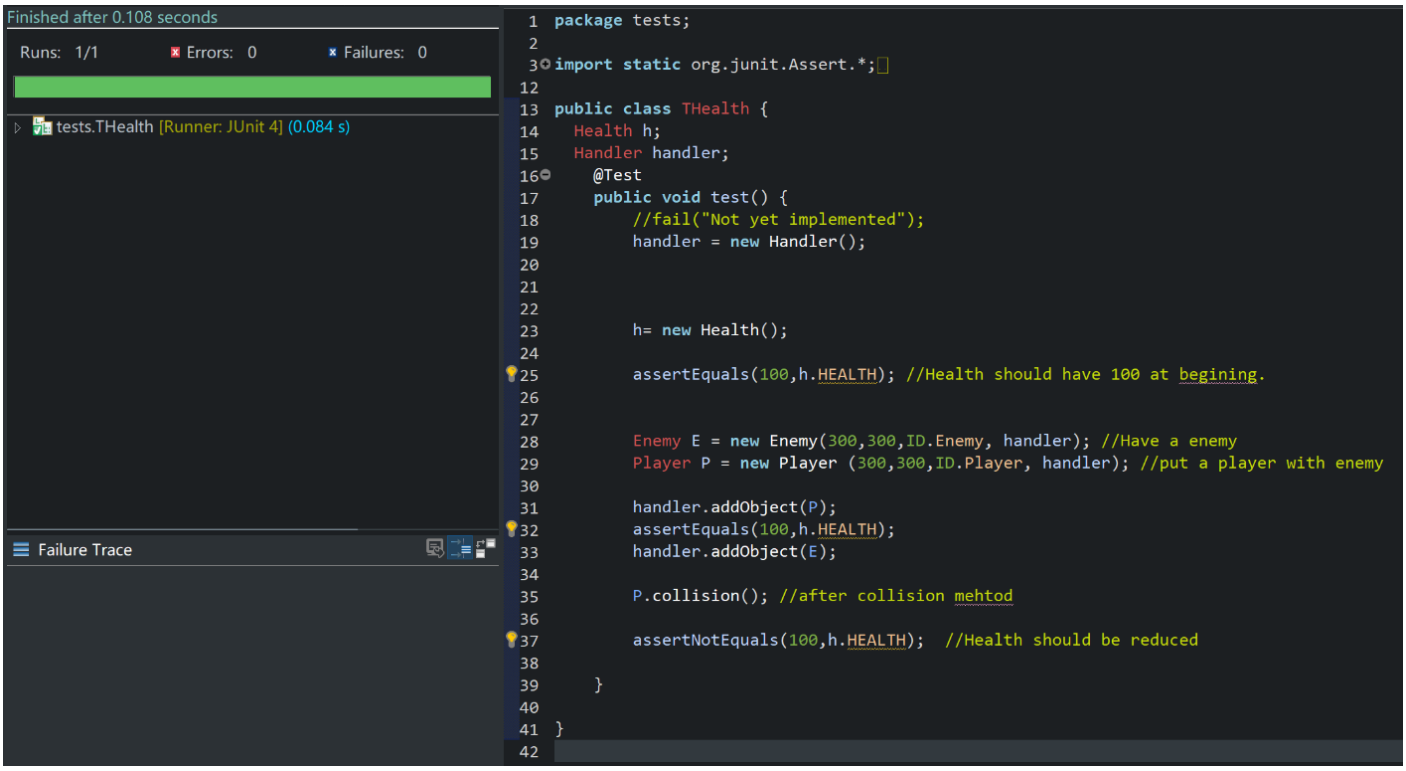


Figure 29: J Unit Test – Player Damage/Health Bar

Sprint Retrospective

Week four

A report of week four was successfully developed, with several amendments strengthening the user stories, use cases and CRC's. The process of ordering documentation helped make sure versions of each document are now in a concurrent style. Moving into week five, with only two weeks remaining for the project, it was decided that we would need to start preparing the final report.

The game images were created and implemented in the code. Together with the introduction of multiple enemies and multiple rooms to explore, the game is only a few features short of a finished prototype.

Introduction of a second room for the player to travel into prompted an issue with overlapping locations. This bug only occurs when the player enters another room. The monsters do not regenerate and maintain their respective locations. The new room's wall layout is different to the previous one and if the new location of a wall tile is within the same area as a monster, the monster may become stuck within the new wall.

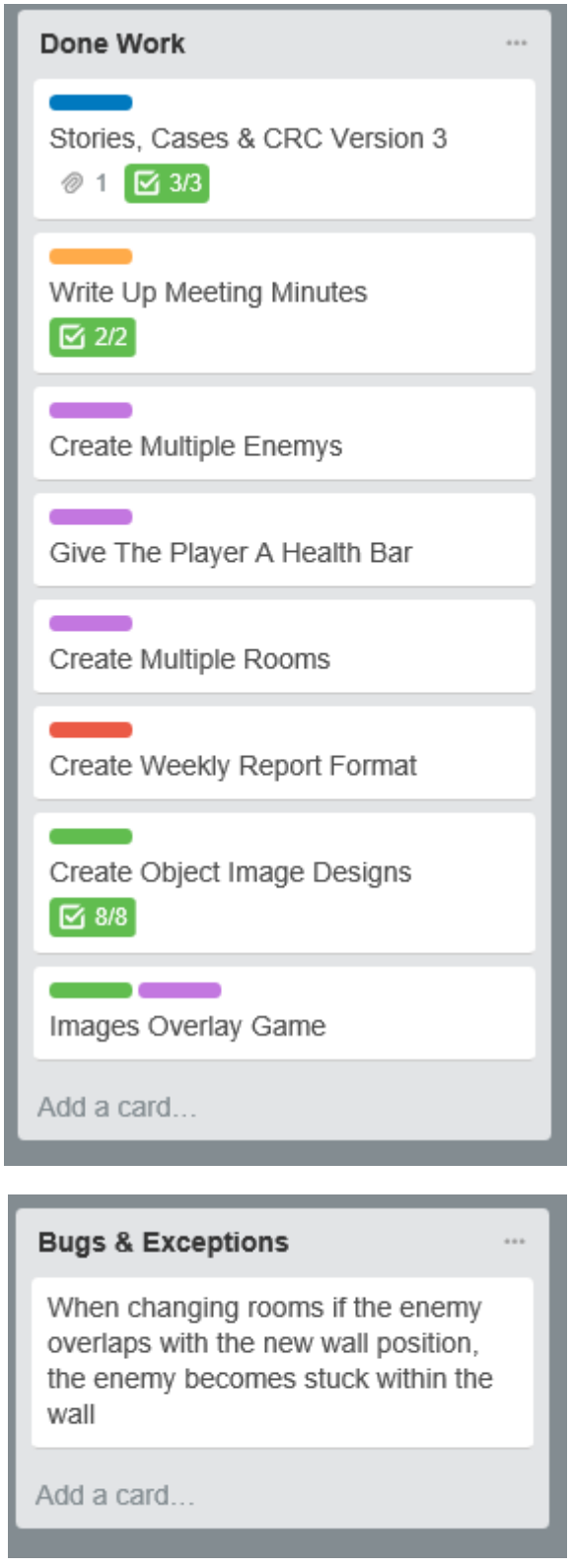


Figure 30: Trello Board Week Four - Done Work & Exceptions

Week Five

Overview

As we near the conclusion of this project, this is the first week that we feel as though the product we have created can be seen as a finished product, in a state ready for release. We do not have all of the requirements satisfied, but through careful judgment each week and through the use of our product content documentation, we have acted upon the most important set of requirements to have a game that is fully playable by a user. This is excluding a menu screen and a way for the user to open the game without having the source code within eclipse.

Now, at the close of last week, we had a set of two rooms, with each object within that room represented by a sprite, which helped give a dungeon-like sense of surrounding. We also had a health bar that would indicate to the user how many health points remained, giving them an indication of how many more times they would be able to be hit by the enemy. It arose in the customer meeting that the amount of collectibles within the game was lacking. Our customer stated that only having gold within the game didn't give it much purpose. They would like the user to be able to interact with more objects. However, they did not give much specification on what collectibles they would like to see, before briefly mentioning weapons and defence. It was therefore decided post-meeting, in a brainstorming session, that we would include a weapon within the game. Due to the current layout of the game, a firearm seemed like the natural weapon to choose. We therefore created a user story to have the player be able to defend themselves against the enemy.

It also came to light that the player could always see themselves within each room, as each room was specified to be the same size as the pop-up window. We had also not acted upon the 5th user story, which specified the inclusion of an in-game map on an HUD. It was therefore determined, in collaboration with the customer, that the map element of the game was to be scrapped. This gave us room to add further implementations within the game, as the implementation of the map was deemed to be a somewhat hard problem within our prioritisation matrix. In regard to the paragraph above, we would now have at least two sets of collectibles, weapons and treasure. Again, after careful team deliberation, we came to the conclusion that we should have some sort of scoreboard, along with a way of keeping a high score league table, as one would see on a classic arcade game.

Minutes of Customer Meeting

Group Name: Blue

Project Name: Dungeon Explorer- A multi-player online game

Date of Meeting: 29/11/2017

Start and End Time: 16:15-16:24

Objective
Weekly review to increase functionality of the game.

Attendees
Ben Boreham, Lucy Nelson, Rikki Hodder, Ruowen Cheng, Yalin Shi, Yu Jiaping

Agenda
Documentation overview
Game overview

Decisions
Documentation Overview
Final report must have separate versions of documentation. Essentially, it's the raw material of sprints which make up the final report. Make sure there is a narrative and retrospective before and after each sprint. Final report will be a chronological order of sprints. Cross referencing is very important. Make sure there is an understanding of how we broke the problem down and how we came up with a salutation. It will look strange if everything was perfect and the game did not come across any problems. Bugs must be documented. One current issue is that sometimes the speed at which we set the bots to move will affect how they interact with the boundary (sometimes it passes through it). User guides must also be written up.

Game Overview
The current game prototype has no map. We were considering implementing a restricted bird's eye view interface and will therefore will need a map. A map gives a sense of space to the game. Game was described as a platform by helpers. Customer advised that the game would have more direction if there were collectables e.g. a key to pass through the door or warp when collecting a certain amount of gold. We could also include a weapon and ammunition to pick up.

Action
Meeting minutes, weekly reports and added functionality to the game reviewed for next meeting. Final report starting to come together.

Next Customer Meeting
Next meeting on the 6th December 2017

Minutes of Scrum Meeting

Group Name: Blue

Project Name: Dungeon Explorer- A multi-player online game

Week 5: 29/11/2017 - 5/12/2017

Objective
Review of customer meeting and allocation of tasks

Attendees
Ben Boreham
Lucy Nelson
Rikki Hodder
Ruowen Cheng
Yalin Shi
Yu Jiaping

Agenda
Documentation
Game Overview

Decisions
Documentation
Weekly report needs to be put together. User guide, installation and maintenance guide mentioned in the customer meeting also need to be developed.

Game Overview
Gun with ammunition to give more depth to the game. Images need to be created. Monster can be shot which will also lead to points. This week a scoreboard was also created. First prototype of game needs to be produced for next week's customer meeting.

Action
Meeting minutes
User stories, user cases and CRC analysis
Create game over screen
Allow player to move in 8 directions
Add additional items to game
User guide, installation and maintenance
Game prototype version 1
Add gun and ammunition

Product Backlog

Week Five

By the end of week five, we aim to have a fully playable version of the game. This means that we can leave enough time in week six for enhancing the gameplay, fixing bugs and packaging the final product.

The remaining elements that need to be added are menu screens and an additional item for the player to interact with. We also wish to provide the player with diagonal movement.

As only two weeks of the project remain we must turn our attention back to the initial brief. As well as the final report, product documents such as the user manual, maintenance and installation guides need to be produced before completion. A starting place for researching user manuals is looking at existing ones from early arcade and computer games. For the maintenance guide we must consider the needs of someone who wishes to update or modify the game.

Judging by the length of the weekly reports that have been produced so far, the final report is likely to be quite long. Arranging this quantity of material will require a considerable amount of time so it is advisable that we make a start now. Although the format will chronologically connect the content of each weekly report, quite a lot of additional narrative and auxiliary documentation will be required.

A new approach to file sharing is going to be needed if multiple team members are to access and edit the same documents frequently, a revision of the GoogleDrive folder system and versioning conventions may be necessary.

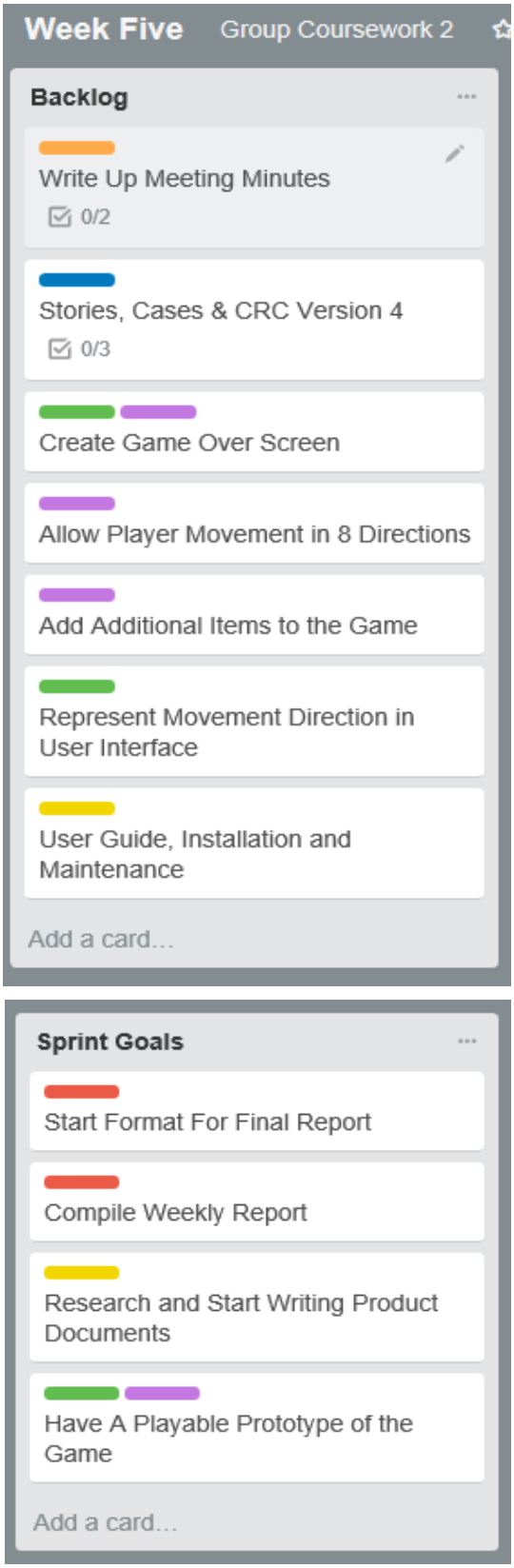


Figure 31: Trello Board Week Five - Product Backlog & Sprint Goals

User Stories – Version 5

Amendments

Due to more collectibles being added to the game, such as a weapon, we are going to need further controls added to the game. This relates to US9, where we specified the need to have movement instructions included within the game. This will now need to be amended to have a set of general control instructions that specify all of the actions that can be made within the game.

We will also need to amend US14, which was initially added last week. Rather than just have a list of items that the player can use as defence, we would now like to give the player the ability to fire a weapon, which will allow them to remove enemies from the same room.

Rather than just having the ability to destroy enemies within the game. We thought that it will make it more rewarding for the user to receive points for this functionality. So, as well as making the game easier for successfully removing enemies from the game, the user will also receive a set amount of points for each kill. We have therefore updated US4.

US11 will need to be amended as any new objects that we include will need to have an image attached to them. So, with the weapons that we are adding, we will need to provide an image for the weapon, and, if this is finally chosen to be a gun, we will need an image to represent a bullet firing across the screen or creating a blast radius, for example.

Additions

As discussed with our customer, the inclusion of one collectible within the game, currently being the treasure, is not enough. Our customer would like to see more things with the ability of being collected. We therefore decided to create a further user story, US14, which would aid us in implementing the use of weaponry or defence against the enemy.

User Stories

2. Movement

As a <player>, I want to <move a character around a room> in order to <carry out the tasks within the game and defend against enemies>.

- Create a player for a user to interact with
- Allow this player to move around the window
- Create a starting location for this player

4. Score

As a <player>, I want to <keep score of the point I achieve> in order to <know when I can leave the current room>

- Create a scoreboard that can be referred to
- Create a message when a sufficient amount of treasure is found
- Increase score for collecting treasure
- Increase score for killing enemies

7. AI Bot

As a <player>, I want <there to be multiple AI opponents> in order to <provide competition in the game>

- Create a bot
- Provide automatic movement for each bot within this game
- Create a way for the bot to seek the player
- Create a specified speed and direction for each bot within this game
- Create a starting location for each bot within the game
- Create removal of enemy object when killed by player

9. Controls

As a <player>, I want <to know which keys on the keypad control which action> so that I can <perform all tasks>

- Create a manual in-game to explain how the user can control the player character:
    - Either whilst playing the game, or
    - In an instruction manual within a menu screen
- Provide instructions on which keys provide which functionality.

11. Surrounding and Images

As a <player>, I want to <feel like I am exploring a dungeon> so that <the game has a sense of realism>

- Create an image for player character, as opposed to a coloured dot.
- Create a sprite for each object included within the game
- Create a sprite for weapons and weapon actions

14. Defence

As a <player>, I want to <have a means to defend myself against enemies> in order to <prevent the given threat>

- Create Weapon
- Create Armour
- Create Health Bar
- Create Damage Points
- Create a way for the player to use a weapon, such as firing a gun

Use Cases – Version 4

4. Score

As a <player>, I want to <keep score of the point I achieve> in order to <know what score I achieve>

Use case	To keep track of the points scored
Summary	This use case will allow the user to keep track of the points they have obtained. This will allow them to know what score they achieve by the time they complete the game or die.
Actor	Player
Trigger	The player obtains one score unit
Primary Scenario	13. The character is instantiated within the room at their start location. Either a. The user directs the player to move towards a piece of gold and ultimately interacts with this, collecting the gold object and increasing the score counter by one gold unit. The gold object will subsequently disappear b. The user directs the player to move towards the gun object within the room and collects this object. The player will then use this gun to fire towards an enemy. If successful, this enemy will be removed from the game and the score counter will increase by one enemy unit. Else, the enemy will remain within the map and the ammunition counter will be decreased by one. 14. Once the player has collected a set amount of points within the room, a passage will appear allowing them to progress. 15. When the player has interacted with this passage. They will end up in a specific location within the successive room. This will ensure that they are not interacting with other objects within the room's instantiation.
Exception Scenario	The player does not direct the character in the direction of the treasure 8. The player will move in a direction towards empty space, and, the character will: 9. Either move to an empty space in the room, or not move, as specified in the use case alternative for use case 2. 10. The treasure is still present in its location. 11. When the player is initially located to the successive room, they may be overlapping with a current object, such as an obstacle within the room, causing the player to be immovable.
Pre-conditions	The player has started to navigate the room by moving the character
Post-condition	The player will obtain a piece of treasure

7. AI Bot

As a <player>, I want <there to be multiple AI opponents> in order to <provide competition in the game>

Use case	Inclusion of multiple AI enemies
Summary	We would like to include multiple AI opponents that also traverse the map within a specified movement function
Actor	AI opponent & Player
Trigger	Starting the game
Primary Scenario	Enemy chasing player 7. The AI enemy constantly follows the player at a speed 75% that of the playable character a. There was an issue within this function. It was not as easy to implement as we had assumed. We will therefore look to implement the following function instead: The AI enemy will move about in a given path bouncing off of any of the objects in its path. For example, it can be that the set speed in the x direction is the same as the set speed in the y direction, so as to move diagonally about the screen from a given starting location b. We will also give each separate instantiation of the enemies a different speed and direction to move in. We will also amend the sizes of some of the enemies to have varying senses of danger. 8. The enemy catches the player, which results in a game over message. Player chasing enemy 1. When the player has obtained the gun object. They will have the ability to shoot the enemy. Once the user has performed this action successfully, the enemy will be removed from the map until the game has been complete or the player receives the game over message.
Exception Scenario	5. The AI enemy becomes trapped in a part of the room and is unable to move. For example, if the player is directly in front of the enemy, but there is a wall in the way, the enemy will continue forward indefinitely whilst the player is in this position as they have been programmed to only move in the direction of the player. 6. The enemy is instantiated over a current obstacle within the room and causes this to become trapped in place, not being able to traverse the window.
Pre-conditions	The game is open and the player has started the game
Post-condition	A player-seeking enemy is in the room A random walk object is included in the room. Multiple enemies are within each room.

9. Controls

As a <player>, I want <to know which keys on the keypad control which action> so I can <perform all tasks>

Use case	Controls for Gameplay
Summary	This use case will specify the need for the inclusion of control instructions to be accessible within the game.
Actor	System
Trigger	3. Choosing the instruction section within the menu screen, or 4. A small display within the game specifying the controls for all actions performable by the user.
Primary Scenario	4. The player opens the game and is taken to a menu screen. 5. The player has the option to open an instruction screen from the menu where the controls are displayed. 6. Or, the movement controls will be displayed within the game in a corner of the screen. The controls that need to be specified are: • Movement: Up, Down, Left, and Right • Weapon actions: Shoot, drop.
Exception Scenario	
Pre-conditions	The game is open
Post-condition	The player will be able to refer to an instruction guide on how to control the player.

11. Surrounding and Images

As a <player>, I want to <feel like I am exploring a dungeon> so that <the game has a sense of realism>

Use case	Inclusion of images (sprites) in place of current objects
Summary	This use case will provide images (sprites) in place of all of the current objects within the room.
Actor	User
Trigger	
Primary Scenario	2. The user will interact with a sprite as opposed to shapes defined in the graphics class. The following sprites will be included: a. Player b. Enemy c. Walls and obstacles d. Door e. Gun f. Ammo g. Firing
Exception Scenario	2. The images imported to be used as the sprites for each object do not have the correct dimensions to act appropriately for our current methods.
Pre-conditions	
Post-condition	The game will have a set of images to interact with.

14. Defence

As a <player>, I want to <have a means to defend myself against enemies> in order to <prevent the given threat>

Use case	Inclusion of weapons within the game
Summary	This use case will provide weapon objects within the game that can be collected by the player
Actor	System
Trigger	Instantiation of each room
Primary Scenario	A set of weapon objects will be instantiated within each room for the player to be able to collect. Once the player has collected these weapons, there will be a set of functions that will allow the player to use these weapons. 1. The player will navigate towards the gun object. One they have interacted with this, the gun will disappear. 2. The player will now have the ability to use an extra button within the game. This button will allow the player to fire the weapon, which can be solely used to attack the enemies within the game.
Exception Scenario	
Pre-conditions	Detection methods to determine whether two objects interact with each other.
Post-condition	The game will have a set of images to interact with.

CRC Analysis – Version 5

V5: Closer to the latest version of program, advanced flexibility, more expandable, scalable. New: Box, Health

Technical classes – These will be the classes that organise the technical details of the game, such as handling each object and creating a window to open in order to play the game
Classes with instances involved in the game

Classes

- Game
- Game Object (Abstract Class)
- Handler
- Window
- Room
- Menu Screen
- Map
- Health
- Box
- Player
- Enemy
- Treasure (Gold)
- Door
- Obstacle
- Score (Have not built)

Game (Main class)	
• Creation of objects	Everything

Window	
• Creates the window for the game	Game

Game Object (Abstract Class)	
• Object Location • Object Speed • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window • Return its collision boundary	Objects displayed on the screen

Handler	
• Tick • Render • Add Object • Remove Object	everything

Room	
• Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Enemy Treasure

Menu Screen	
• Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	

Health	
• Health number • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Changed by collision method in Player

Player	
• Object Location • Object Speed • Object ID • Tick (Abstract) to update the games logic and limit access area • Render (Abstract) to update what is seen in the window • Collision detection	Game Object Enemy Treasure Room Door Score Health

Enemy	
• Object Location • Object Speed • Object ID	Game Object Enemy Treasure Room

Treasure (Gold)	
• Object Location • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Enemy Room Score

Door	
• Tick • Render • Change Room and Player location	Player Room Game

Obstacle	
• Object location • Detect collision	Player Enemy Room Game

Score (Have not built)	
• Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Treasure

Class Diagram

In the fifth week, we have fifteen classes, we add box and health as the new classes. And we expand the old classes.

Figure 32: Week Five - Class Card Association

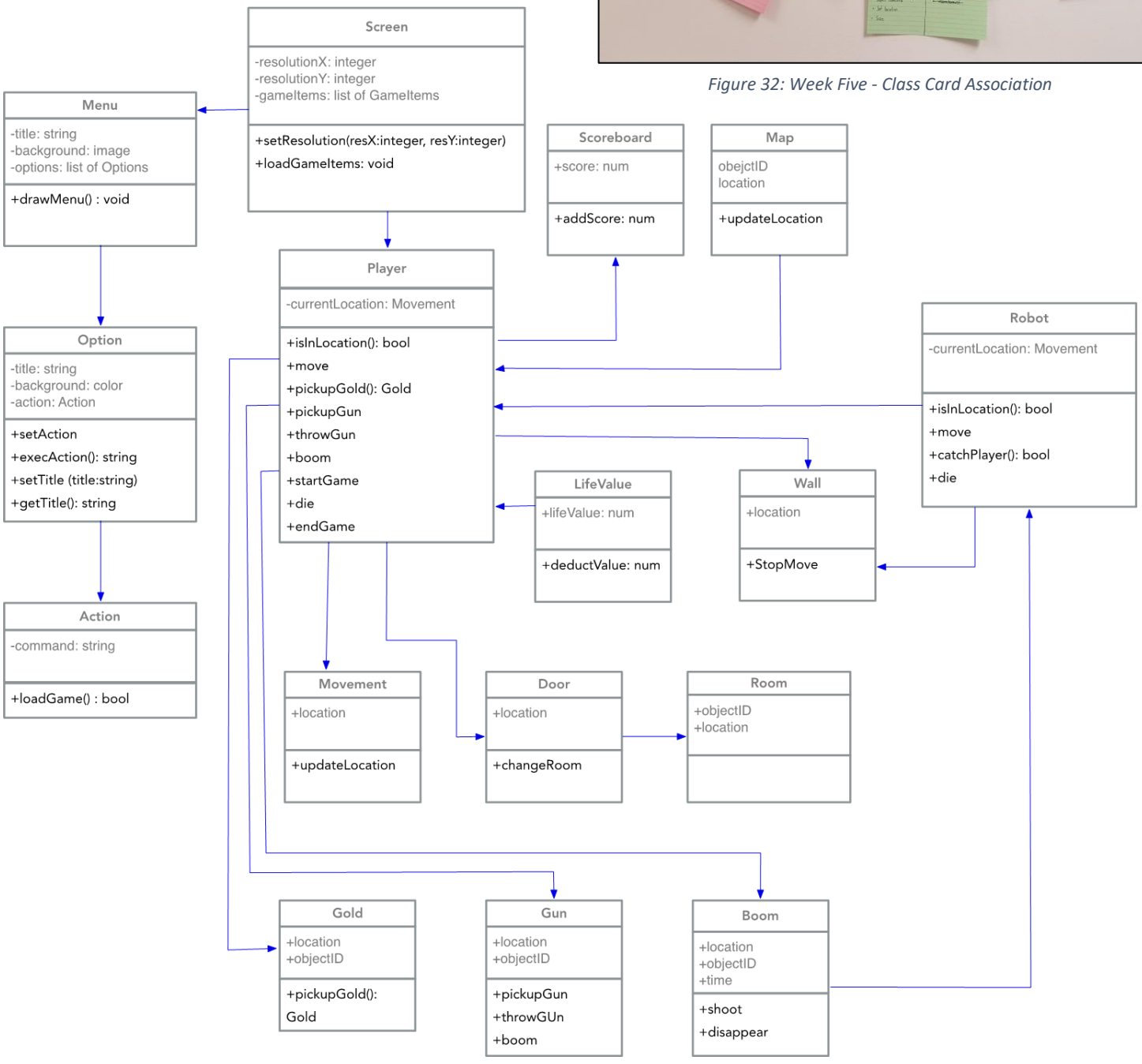


Figure 33: Week Five – Final Class Diagram

Prioritisation

Table 5: Week Five Prioritisation Matrix

User Story ID	User Story Name	Primary Actor	Complexity[a]	Priority[b]	Completed[c]
US2	Movement	Player	Low	2	Partially
US4	Score	Player	Low	2	Yes
US7	AI Bot	AI Bot	Low	1	Yes
US9	Controls	Player	Medium	4	No
US11	Surroundings and images	System	Low	2	Partially
US14	Defence	Player	Medium	2	Partially

[a]High, medium, low

[b]1 (high) – 5 (low)

[c]Yes, no, partially

Justification

US4: With the code in place, this was a simple problem to undertake. We have therefore designated this as low complexity.

US7: We can remove objects from the game within the handler class. This was implemented when the player was initially removed. So we can add this with ease.

US9: We have set the criterion to include the controls within the game. However, we may provide these within the user manual. For now, we have set this as a low priority.

US11: We just need to create images for these extra objects, the gun and the fire function, and then add these to the objects as we have done for previous objects.

US14: There has not been anything similar to this instantiated within the game, so this may be quite a hard task to implement. We have set the complexity to a medium with a high priority.

Game Design & Implementation

A number of new elements are added in version 5 in order to making the game integrity, artistic and more interesting, these are gun, bullet, health value, score and progress bar. Some functions are updated in PLAYER and ENEMY as well.

Words in blue is what is new in current version.

- A suitable size WINDOW.
- A PLAYER can move up, down, left and right, collect treasure and shoot bots.
- An ENEMY chases the player and will die when shot by a player.
- Some TREASURE can be collected by the player.
- BOUNDARY uses to stop both the player and bots move outside.
- OBSTACLE stops bots and the player moving.
- A WARP DOOR enables the player to access to further rooms.
- GUN can be pick up and throw away, can shoot to the enemy. In addition, the box will display 'EMPTY' when all bullets are used.
- There are five BULLETS when the player picks up the gun, BULLETS will decrease as a player shoot bots.
- HEALTH enables a player can be hit a few times, if the health value is empty, the player will die and the game is over.
- SCORE records how many treasures the player collected and how many bots are shot by the player. One treasure is valued 50 marks and a player can get 100 marks when shoot a bot.
- PROGRESS BAR is empty before the game starting, it will grow by percentage. When the bar is full, it means the player is win.

User interface design

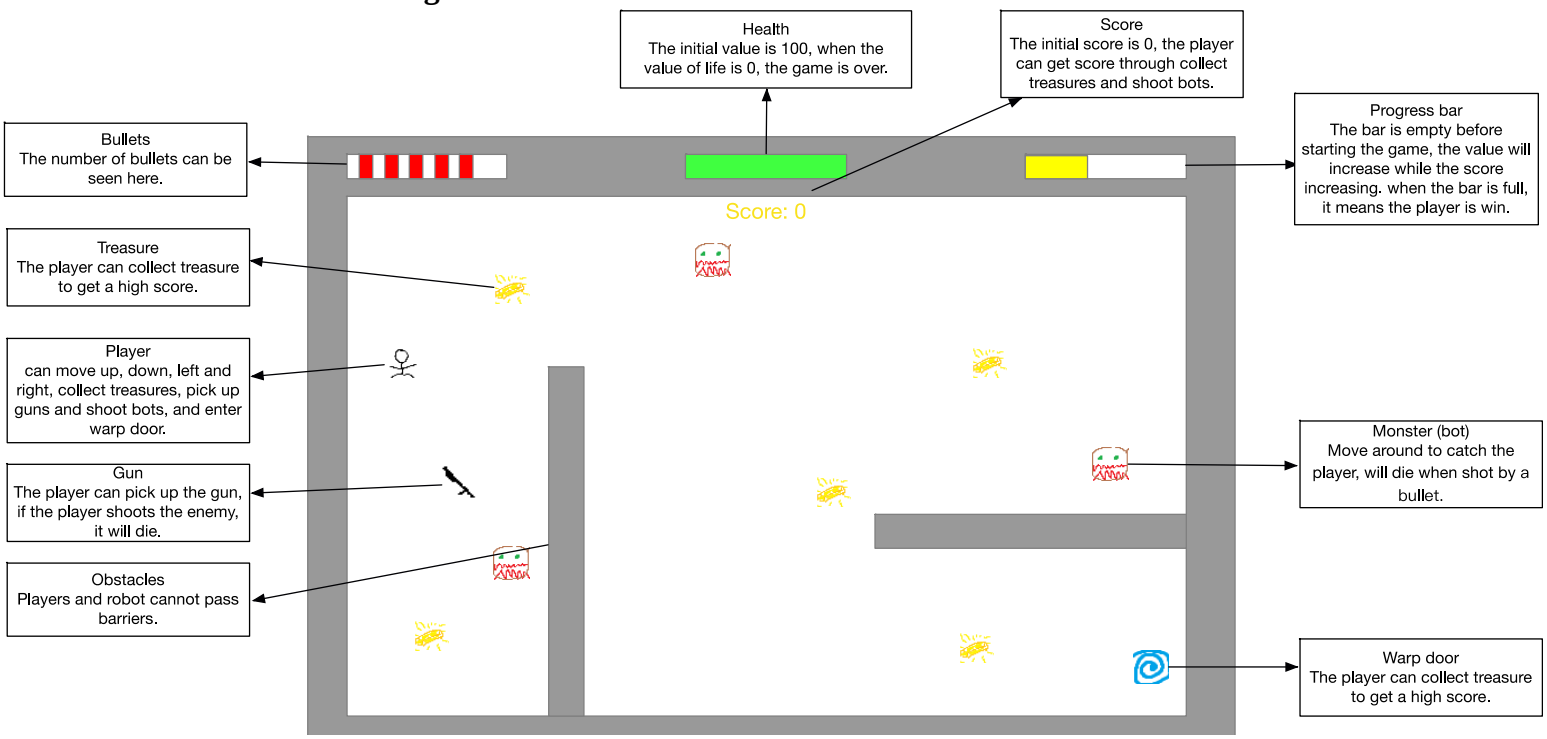


Figure 34: User Interface Design – Week Five

Current application view

View 1: Initial view

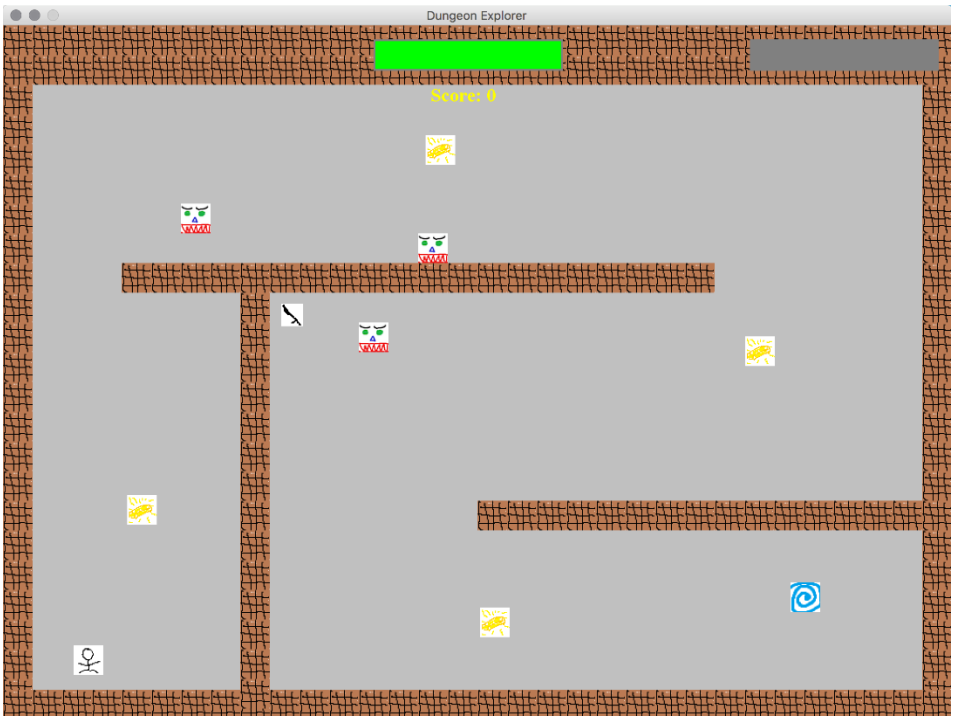


Figure 35: Implemented Game View – Week Five

View 2: The gun is picked up and some golds are collected.

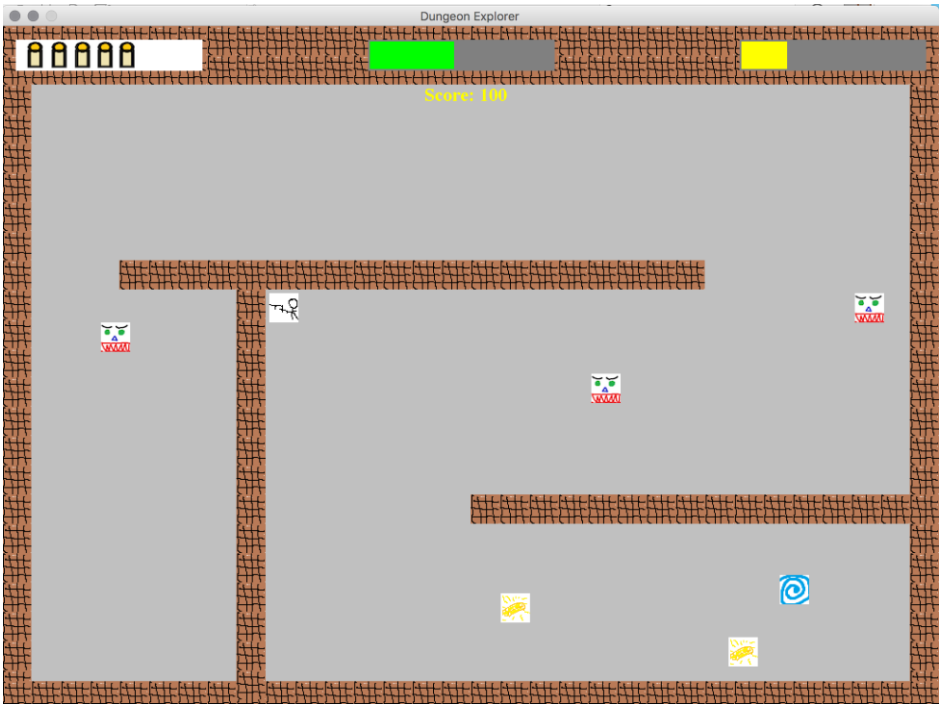


Figure 36: Implemented Game View – Week Five: Progress Bar, Health and Ammo

Testing

Room switch test. Collision method in player and variables in Game class were involved.

If there is a collision between Player and Door detected, the room layout should be changed.

In the test, the locations of the player and the door are changed to make a collision happen. As a result, the ‘currentRoomNo’ and ‘nextRoomNo’ in the Game class should change. The test has successfully passed as can be seen in Figure 37

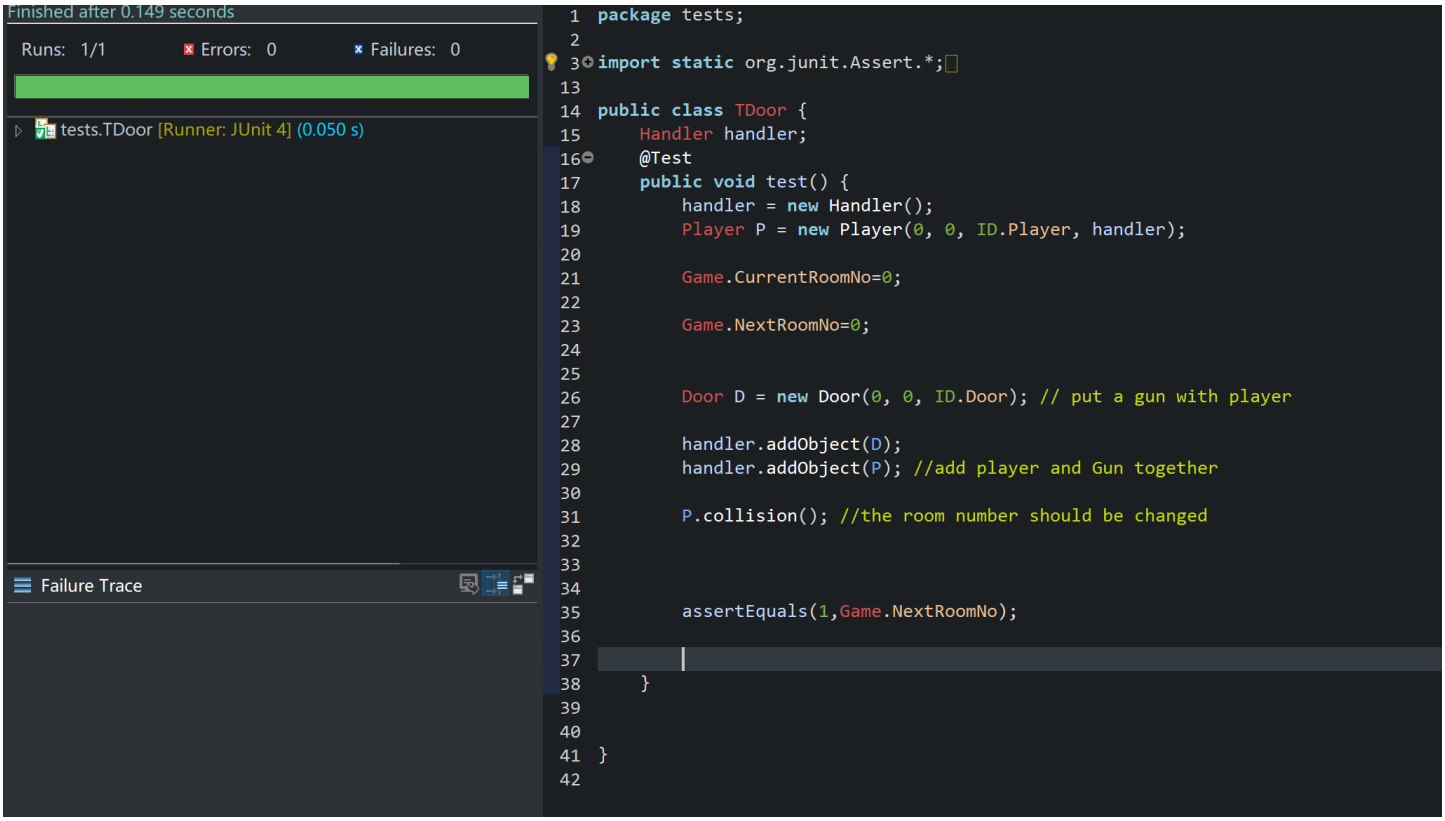


Figure 37: J Unit Test - Door/Room Change Test

Gun picking test. Player has no gun at the beginning.

Once the player touches a gun, the condition of player should be switched to ‘isHoldGun’ is true.

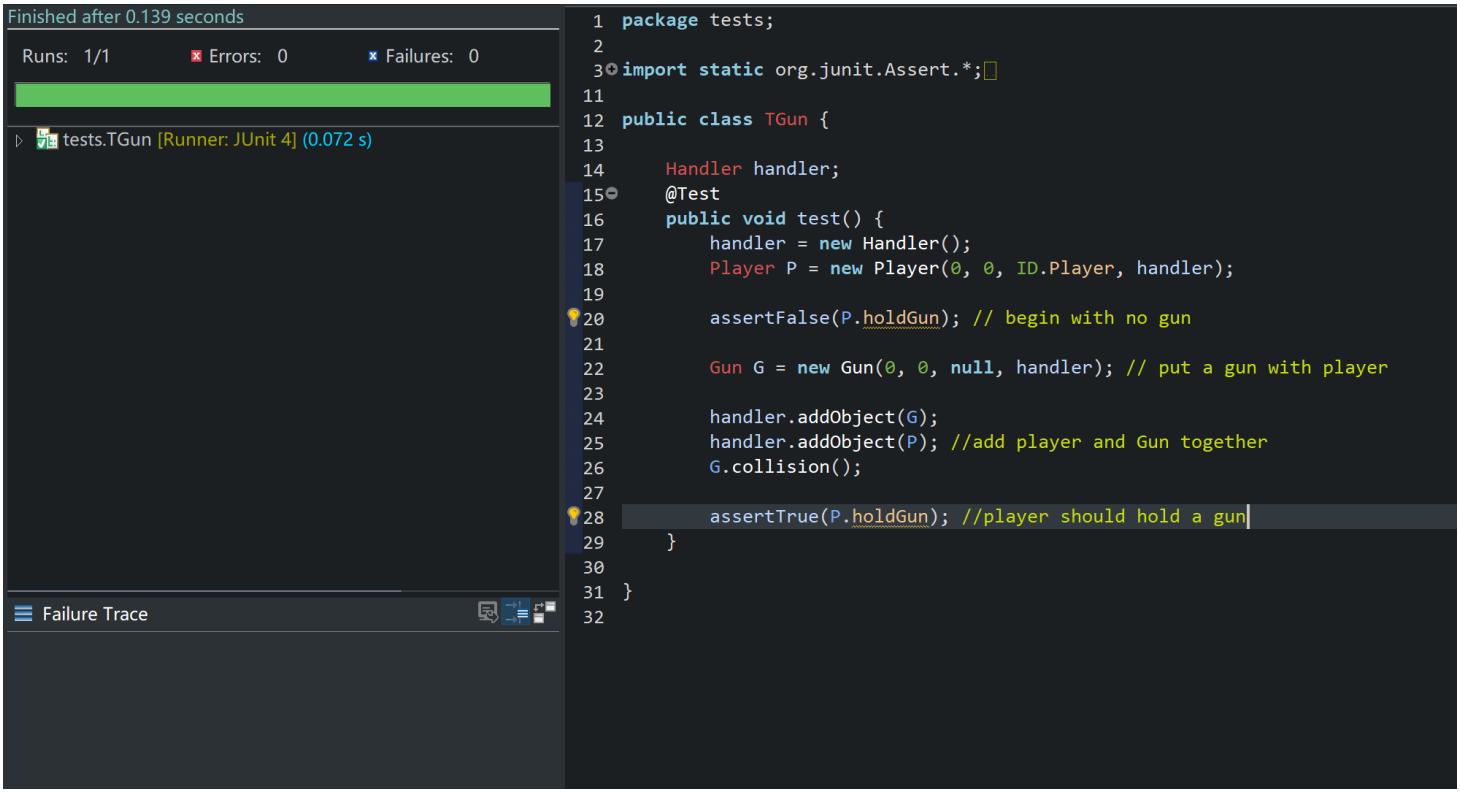


Figure 38: J Unit Test - Gun Pickup Test

Sprint Retrospective

Week Five

In the penultimate week we have completed a playable prototype of the game. There was a final addition to the game narrative of including a gun with which the player can pick up and shoot monsters for extra points. This feature has helped create a greater differential in the scoring potential of a player, making the game more competitive.

Once the game narrative was finalised it was possible to complete the user manual. Based on the manual for Pac-man, it contains explanations for all of the objects you will find in the game as well as instruction for scoring and other game mechanics. It may require some final adjustments to accommodate any changes made in week six.

Due to end of term deadlines, week five has not had the level of attention the project usually receives, as a result the final report status is slightly behind where we would like it to be.

A structure for the final report has been agreed upon and a couple of weeks of documentation have been completed. With only minimal alterations to the code required for the final week, the whole team is available for the compilation and checking of the remaining document.

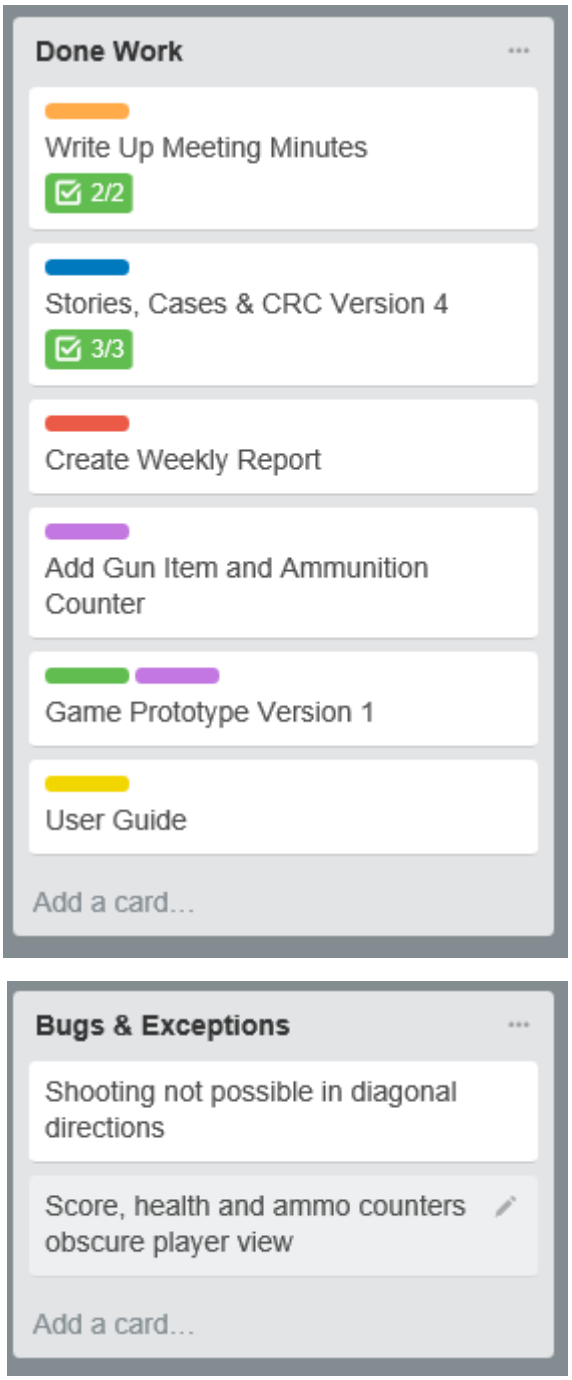


Figure 39: Trello Board Week Five - Done Work & Exceptions

Week Six

Overview

The progression from the previous week to this week includes a scoreboard to keep count of the treasure collected and enemies killed and a collectible weapon that can be used to remove the enemies from the map. We now have quite a few objects within the window that keep track of points, such as the health bar, progress bar, and scoreboard, and, as of this week, we also hope to have an ammunition counter that will only allow the user to have a set amount of bullets to 'kill' the enemies. There is not currently a sufficient amount of space to include all of these without obstructing the view of parts of the traversable room. It would not make much sense to make any of these smaller as it may be difficult for some users to make out exactly what they represent. To combat this problem, we have come up with a somewhat simple solution due to the time constraints of being in the final week. We will include an extra wall at the top of the room so that the mini displays only obstruct the non-traversable parts of the room. This will add a slight amount of difficulty to the game, as there is a smaller area for the player to move and dodge the enemies. However, amending the speed of the enemies, or reducing the amount of damage taken with each collision can easily rectify this.

We also realised that the player is currently able to fire the weapon they collect without limit. This makes it quite easy to kill the enemies and quickly have nothing to worry about when traversing the map. We therefore decided to limit the amount of ammunition within the gun object to prevent the user from button bashing to achieve results. This also relates to the glitch we had, mentioned in the exceptions section, which would cause the game to glitch after the user had instructed too many 'fires' from the gun.

There was also the problem that the gun would only fire in a specific direction if the user was currently moving in that direction. We included 7 extra images for the player, which would signify the direction they are travelling in, that would be updated and rendered whenever the player was to move in a particular direction. We would then fixate this position until they move in a different direction. This way, we would be able to have the gun fire in whichever direction the player is facing, despite the player being still.

Product Backlog

Week Six

With only small alterations left to make to the final game, the full team attention is directed at completion of the documentation. Generally each team member has taken on a different section. Documents are produced and saved in individual files before undergoing a peer review. Once the document has been finalised it is added into its respective section.

Minutes of Customer Meeting

Group Name: Blue

Project Name: Dungeon Explorer- A multi-player online game

Date of Meeting: 6/12/2017

Start and End Time: 16:31-16:40

Objective
Weekly review to increase functionality of the game.

Attendees
Ben Boreham
Lucy Nelson
Rikki Hodder
Ruowen Cheng
Yalin Shi
Yu Jiaping

Agenda
Documentation Overview
Game Overview

Decisions
Documentation Overview
The team are working on the final report. So far at least 5 sprints worth of material that needs to be combined as one narrative. Make sure to include overview and review. Four documents to be submitted- final report, installation guide, user guide and maintenance guide.

Game Overview
New to the same if picking up a gun and shooting the bot. Progression bar relates to the amount of gold collect. Score also increases when you shoot the bot. Possibility of high scoreboard. At the moment the game has no leveling and there is the potential of adding another few rooms. Customer agreed that the images in the same were fine and that a map for this game would be pointless as you can always see where the player is.

Action
Final report, installation guide, user guide and maintenance guides need to be completed.
Individual and team questionnaires need to be completed.

Minutes of Scrum Meeting

Group Name: Blue

Project Name: Dungeon Explorer- A multi-player online game

Week 6: 6/12/2017 - 15/12/2017

Objective
Review of customer meeting and allocation of tasks

Attendees
Ben Boreham
Lucy Nelson
Rikki Hodder
Ruowen Cheng
Yalin Shi
Yu Jiaping

Agenda
Documentation
Game Overview

Decisions
Documentation
Completion of final report, user guide, installation guide and maintenance guide.

Game Overview
Limit on ammunition as it was too easy to kill monsters in the game. The player is now able to move in 8 different directions and 7 extra character images were created. The gun was also able to fire whatever way the character was facing. Menu screen was implemented to allow the user to choose difficulty based on how fast the monster moves. Player can also exit the screen or restart the game with chosen difficult when you die.

Action
Final report
Final game
User guide
Installation guide
Maintenance guide

User Stories – Version 6

Amendments

It is now of utmost important to store the game in a location where a user can easily download this. We have therefore updated US1 to represent this.

US14 will need the inclusion of an ammunition counter. We found that the ability to have unlimited ammo reduced the difficulty and fun of the game. To combat this, we would like to limit the amount of ammunition a player can have.

The firing of the weapon would now related to the way the character is facing. To represent this, we would like to implement separate sprites that update when the player is moving in a particular direction, and to stay within that representation until the player decides to move in a separate direction. We have therefore updated US2.

The majority of the game is in a state ready to be released. However, we still lack a way for the user to restart the game once they have completed all of the tasks, or reached a game over screen. We have therefore set the task of having the menu screen show up as a subsequent action of the two scenarios just mentioned. This has been represented in US13.

Additions

1. Open Game

As a <player>, I want <to open a game on a PC from the desktop> in order to <play a game>.

- Make a window for a game to be played
- Create ease of use for opening game
- Create a file storage for this game that allows a user to download this with ease.

2. Movement

As a <player>, I want to <move a character around a **room**> in order to <explore>.

- Create a player for a user to interact with
- Allow this player to move around the window
- Create a starting location for this player
- Create a way for the player to use a weapon, such as firing a gun
- Create a further set of sprites to represent the player character moving in a specific direction

13. Menu Screen

As a <player>, I want to <enter a menu screen> so that <the game does not start until I decide>

- Create a menu screen that opens when the game initially opens.
- Provide a set of options for the player to choose, e.g. instructions, play, etc.
- Make the menu screen the default screen once the player wins or loses.
- Create difficulty settings for the game.

14. Defence

As a <player>, I want to <have a means to defend myself against enemies> in order to <prevent the given threat>

- Create Weapon
- Create Armour
- Create Health Bar
- Create Damage Points
- Create an ammunition counter to limit the weapon use

Use Cases – Version 5

1. Open Game

As a <player>, I want <to open a game on a PC from the desktop> in order to <play a game>.

Use case	Open the games user interface
Summary	We want to: • Create a window for the game to be played in • Create an action for the user to be able to open this game on a PC desktop
Actor	Player
Trigger	The player will try to open the game window from the PC desktop
Primary Scenario	2. The player downloads this game from a storage location yet to be decided. 3. The player finds the games shortcut on the desktop PC and subsequently clicks the shortcut. 4. The game window pops up on the screen in an appropriate position as specified by the developers.
Exception Scenario	3. The file has not been stored in an appropriate location. 4. The player cannot find the shortcut 5. The window does not open
Pre-conditions	Storage location availability
Post-condition	The game window is now open

2. Movement

As a <player>, I want to <move a character around a **room**> in order to <explore>.

Use case	Move the character
Summary	This use case allows the player to move a character
Actor	Player
Trigger	The player presses a predefined button to move
Primary Scenario	14. The player character is instantiated within the game window and is visible to the user. 15. Each time the player is instantiated within the room, the specific location must be fixed in order to keep them at a distance from the starting location of the enemy AI bot 16. The player decides in which direction they would like the character to move 17. The player then presses the appropriate button to move the character. a. Depending on the direction the user chooses to move the player. The player will face in that direction. There will be 8 separate direction

	images for the character to represent this. This also helps the user to know which way they are firing when they are stationary. 18. The player will move in that particular direction, until 19. The player removes their finger from the button that determined movement. 20. The character comes to a halt and the action is terminated.
Exception Scenario	6. The buttons for movement are not defined or have not been instructed by the user 7. The character does not move. 8. The player character is removed from the map straight away as the enemy AI bot is instantiated at a location next to, or the same as, the player character. This does not give the player a chance to start the game.
Pre-conditions	The player character is instantiated within the window with a set of buttons that determine their movement.
Post-condition	The player will move in the direction specified by the player.

13. Menu Screen

As a <player>, I want to <enter a menu screen> so that <the game does not start until I decide>

Use case	Create a menu Screen
Summary	This use case will detail the specifications for our main menu screen.
Actor	User
Trigger	The player has: 4. Opened the game for the first time 5. Successfully completed the game 6. Been unsuccessful. I.e. caught by the enemy.
Primary Scenario	The user will be navigated to the menu screen as specified in the trigger section. 4. The player will have the option to start the game 5. The player will have the option to exit the game ~~6. The player will have the option to view the gameplay controls~~ a. Note that the gameplay controls are now included within the user manual. This will be included within the games file. 7. The player will have the option to play in an easier mode and a harder mode. This can be amended within each different play through.
Exception Scenario	
Pre-conditions	The game has been started
Post-condition	A menu screen is available to the user

14. Defence

As a <player>, I want to <have a means to defend myself against enemies> in order to <prevent the given threat>

Use case	Inclusion of weapons within the game
Summary	This use case will provide weapon objects within the game that can be collected by the player
Actor	System
Trigger	Instantiation of each room
Primary Scenario	A set of weapon objects will be instantiated within each room for the player to be able to collect. Once the player has collected these weapons, there will be a set of functions that will allow the player to use these weapons. 3. The player will navigate towards the gun object. One they have interacted with this, the gun will disappear. 4. The player will now have the ability to use an extra button within the game. This button will allow the player to fire the weapon, which can be solely used to attack the enemies within the game. 5. Since the weapon can be fired an infinite amount of times, there is the exception of
Exception Scenario	The player has the ability to fire the gun an infinite amount of times. This has the problem of making the game too easy for the user. An ammunition counter will combat this problem so as to limit the amount of times the user can fire the weapon.
Pre-conditions	Detection methods to determine whether two objects interact with each other.
Post-condition	The game will have a set of images to interact with.

CRC Analysis – Version 6

V6: New: Gun

Technical classes – These will be the classes that organise the technical details of the game, such as handling each object and creating a window to open in order to play the game
Classes with instances involved in the game

Classes

- Game
- Game Object (Abstract Class)
- Handler
- Window
- Room
- Menu Screen
- Map
- Health
- Box
- Player
- Enemy
- Treasure (Gold)
- Door
- Obstacle
- Score (Have not built)
- Gun

Game (Main class)	
• Creation of objects	Everything

Window	
• Creates the window for the game	Game

Game Object (Abstract Class)	
• Object Location • Object Speed • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window • Return its collision boundary	Objects displayed on the screen

Handler	
• Tick • Render • Add Object • Remove Object	everything

Room	
• Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Enemy Treasure

Menu Screen	
• Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	

Health	
• Health number • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Changed by collision method in Player

Box	
• Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Player Room

Player	
• Object Location • Object Speed • Object ID • Tick (Abstract) to update the games logic and limit access area • Render (Abstract) to update what is seen in the window • Collision detection	Game Object Enemy Treasure Room Door Score Health

Enemy	
• Object Location • Object Speed • Object ID	Game Object Enemy Treasure Room

Treasure (Gold)	
• Object Location • Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Enemy Room Score

Door	
• Tick • Render • Change Room and Player location	Player Room Game

Obstacle	
• Object location • Detect collision	Player Enemy Room Game

Score (Have not built)	
• Object ID • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Treasure

Map	
• Object ID • Object Location • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window	Game Object Player Enemy

Gun	
• Object ID • Object Location • Tick (Abstract) to update the games logic • Render (Abstract) to update what is seen in the window • Pick up or throw • shoot bot	Game Object Player Enemy Room Game

Prioritisation

Table 6: Week Six Prioritisation Matrix

User Story ID	User Story Name	Primary Actor	Complexity[a]	Priority[b]	Completed[c]
US1	Open Game	Player	Low	1	Yes
US2	Movement	Player	Medium	3	Partially
US13	Menu screen	Player	Medium	2	Partially
US14	Defence	Player	Low	1	Partially

[a]High, medium, low

[b]1 (high) – 5 (low)

[c]Yes, no, partially

Justification

US1: There is an export function within eclipse that made this almost autonomous.

US2: Whilst we are familiar with uploading images to represent the object, it may be somewhat difficult to implement this within different directions the character is moving. We have set this to a medium priority.

US13: We are still without a menu screen. However, some research has suggested to use a JOptionPane to implement this. Whilst we have an idea of what to do, it may be difficult to use with our current code. We have therefore set this to a medium complexity.

US14: The ability to lower the weapon count will be an easy problem. Since this is causing a glitch within the game, causing the game to stop after the weapon has been fired too many times, we have given this a high priority.

Game Design & Implementation

In version 6, some dialog boxes are added in the game, which enable the game more intuitive and easy to operate.

Words in blue is what is new in current version.

- A suitable size WINDOW.
- A PLAYER can move up, down, left and right, collect treasure and shoot bots.
- An ENEMY chases the player and will die when shot by a player.
- Some TREASURE can be collected by the player.
- BOUNDARY uses to stop both the player and bots move outside.
- OBSTACLE stops bots and the player moving.
- A WARP DOOR enables the player to access to further rooms.
- GUN can be pick up and throw away, can shoot to the enemy. In addition, the box will display 'EMPTY' when all bullets are used.
- There are five BULLETS when the player picks up the gun, BULLETS will decrease as a player shoot bots.
- HEALTH enables a player can be hit a few times, if the health value is empty, the player will die and the game is over.
- SCORE records how many treasures the player collected and how many bots are shot by the player. One treasure is valued 50 marks and a player can get 100 marks when shoot a bot.
- PROGRESS BAR is empty before the game starting, it will grow by percentage. When the bar is full, it means the player is win.
- Some DIALOG BOXES will pop up at the beginning and the end of the game. At the beginning, the player can choose modes from 'DADY', 'HEAVEN' and 'HELL'. At the end, the DIALOG BOX will show 'You win!' or 'Game over' followed by scores the player gained so far, some buttons are available as well, includes 'Quit', 'Restart: DADY', 'Restart HEAVEN' and 'Restart HELL'

User interface design

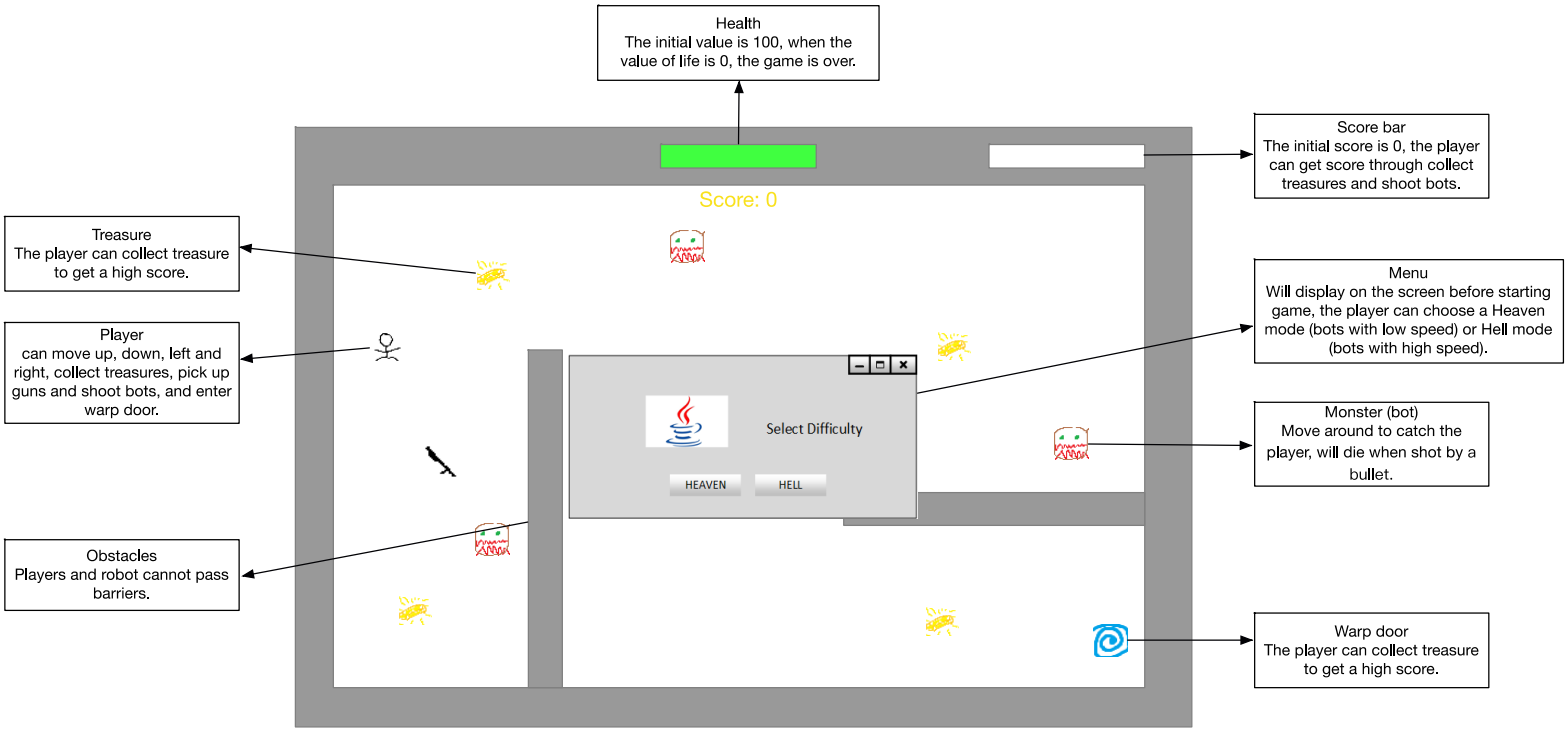


Figure 40: User Interface Design – Week Six: Main Menu Options

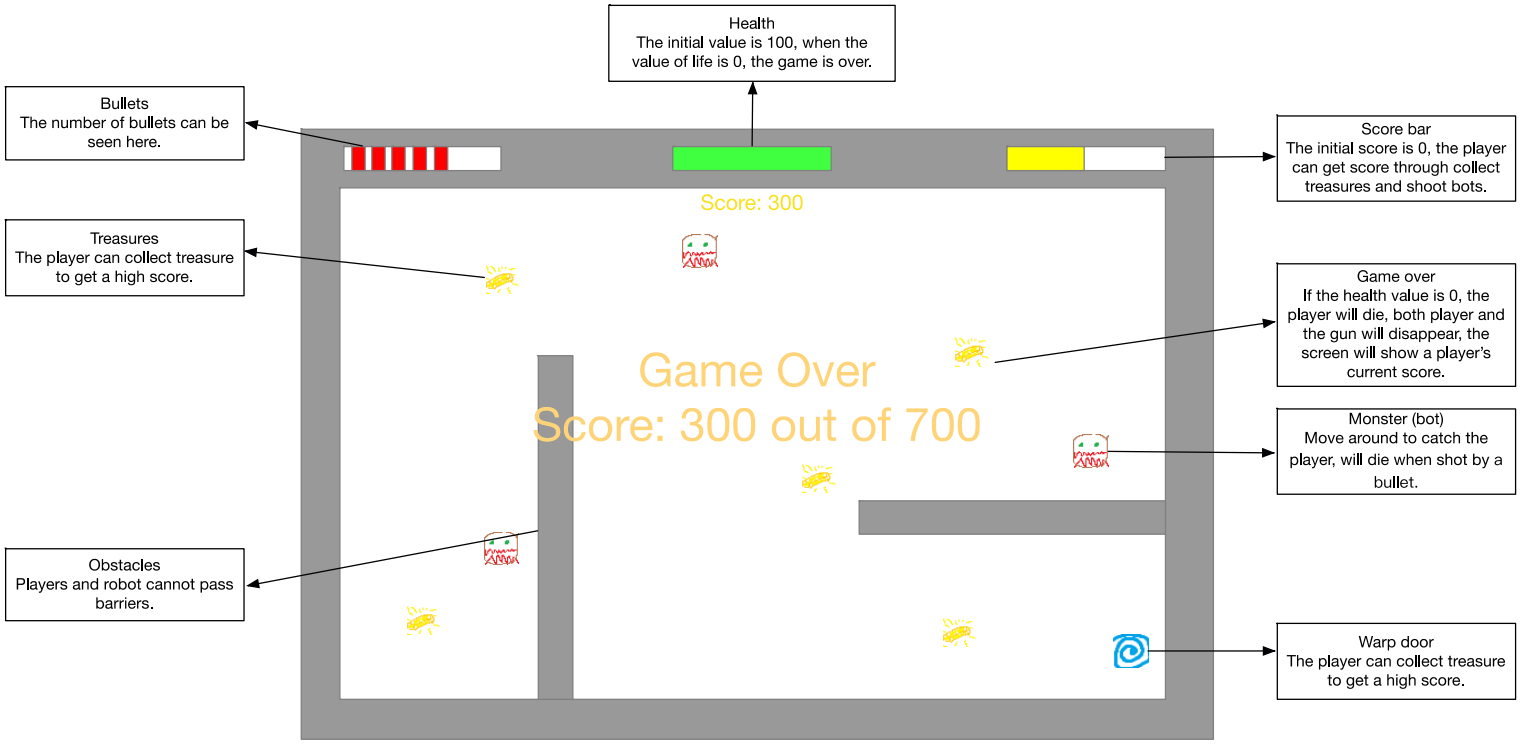


Figure 41: User Interface Design – Week Six: Game Over and Score

Current application view

View 1: Starting the game and choose the mode

Figure 42: User Interface Design – Week Six: Difficulty Selection

View 2: Player win

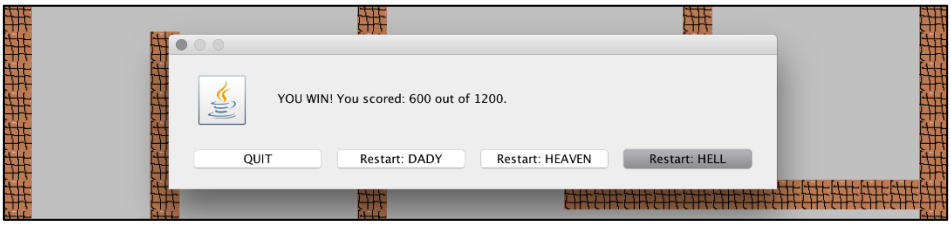


Figure 43: User Interface Design – Week Six: Game Completion Notification

View 3 Game over version 1 (This version is based on the interface design and it can be seen that there is a problem on displaying the score).

Figure 44: User Interface Design – Week Six: Game Over & Score Display –Score Error

View 4 Game over version 2 (The bug was fixed in Game over version 1 and will be modified in Game over version 3).

Figure 45: User Interface Design – Week Six: Game Over & Score (fixed)

View 5 Game over version 3 (This is the final version of Game over, in order to keep page consistency and make the game looks good)

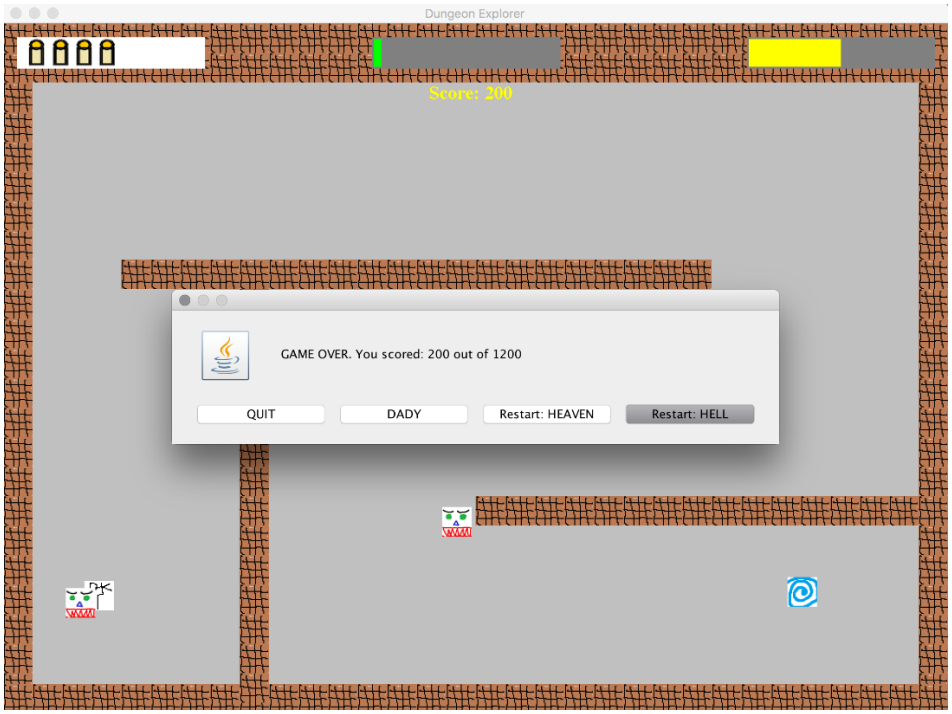


Figure 46: User Interface Design – Week Six: Game Over & Score Moved to Notification Window

Testing

‘isWin’ method test. The winning condition is that all gold has been collected.

If there is no more ‘Gold’ left in the room, then the Boolean method returns ‘Ture’, else returns ‘False’.

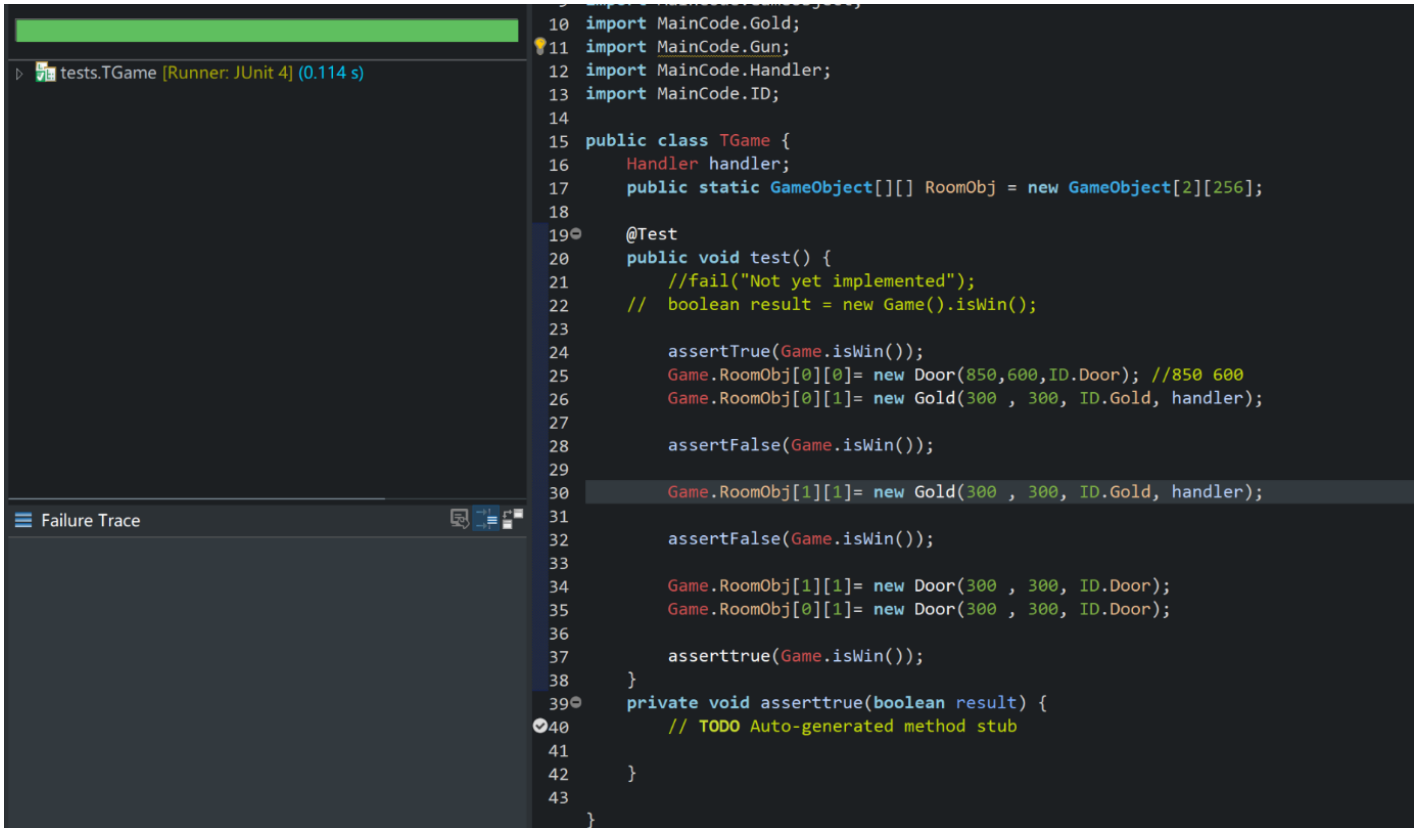


Figure 47: J Unit Test – Game Win Test