



# UNIVERSITY OF BIRMINGHAM

## LI TEAM PROJECT

[ GROUP C5]

## PROJECT REPORT

## [MAZETANK GAME]

NAME	STUDENT ID
SAKINAH MOHAMAD	1511442
KEVIN DEVABATTULA	1571591
TOM BYRNE	1333499
LEWIS DEAKIN-DAVIES	1561926
HUZEIFAH MOHAMMED	1567467

## Contents

1.	Overview	3
1.1.	Introduction	3
2.	Game Elements	4
2.1.	Tank	4
2.2.	Maze	4
2.3.	Score System	5
3.	Game Component	6
3.1.	User Interface	6
3.2.	Sound	6
3.3.	Artificial Intelligence	7
3.4.	Competitive Play & Networking	7
3.5.	Physics	7
4.	Final Requirements	8
4.1.	Major System Capabilities	
5.	Software Design	11
5.1.	Software Architecture design & Justification	11
5.2.	Design Pattern & Justification	12
5.3.	UML diagrams	13
6.	Interface Design	14
6.1.	Principles Of Design	14
6.2.	Game Interface Elements	15
6.3.	Interaction Style	16
6.4.	HCI Evaluation	16
7.	Software Engineering Processes	19
7.1.	Software Process,Principles and Methodology	19
8.	Risk Analysis	23
9.	Evaluation	24

9.1.	Game Strengths	24
9.2.	Game Weaknesses	25
9.3.	Improvement to Make	26
10.	Summary	27
11.	Contribution Assessment	28
12.	Individual Reflection	31
13.	References	33
14.	Appendix	34

# 1. OVERVIEW

## 1.1 Introduction

MazeTank is a 2D java-based tank shooting game. This game takes place in a maze, with walls as the obstacles. There are 3 pre-set mazes and a random maze generation option. To score a point, a player must hit the enemy tank by aiming and shooting a projectile. A projectile can bounce off walls two times before disappearing. When a player dies, the surviving tank gains a point and the game restarts. If the random maze map was chosen, a new random maze will be generated. The game will not end unless the exit button is clicked.

This game has 3 different modes;

1. **Player vs AI** - The user can play against an AI player
2. **2 Player Local** - 2 local users can play against each other.
3. **Online Play** - Up to 9 users can play against each other online.

# 2. GAME ELEMENTS

There are 4 important elements that we should focus on, they are the; Tank, Maze, Projectiles and Scoring System.

## 2.1 Tank

There are 3 types of tank:

1. **AI Tank** - A blue coloured tank controlled by AI.
2. **1st Player tank** - A green coloured tank, controlled by the 1<sup>st</sup> player, by pressing (W, S, A, D) to move the tank in the corresponding direction. The “spacebar” is pressed to shoot a projectile.
3. **2nd Player tank** - A red coloured tank controlled by the 2<sup>nd</sup> player, by pressing the (UP, DOWN, LEFT, RIGHT) arrows to move the tank in the corresponding direction. The “M” on the keyboard to make a shot.

## 2.2 Maze

There are 3 pre-set mazes, which were made by positioning wall objects in the desired position to create the map. The random maze generation option creates a unique, random maze each time a new game starts. The maze generation algorithm makes sure there are no blocked paths.

## 2.3 Projectiles

The circular projectiles are fired from the gun of each tank. The projectiles are fired at the same speed, however the velocity varies, depending on the direction the tank is facing. Only one projectile can be fired at any time, from a tank. Projectiles can bounce

off walls 2 times before disappearing, therefore precision and accuracy is needed.

## 2.4 Scoring System

In this game, we have a score system that keep track of the points scored by players. When a player dies, the opposing player gains a point. If the game is played locally or against AI, the players have a choice of quitting or starting a new game. On multiplayer, the game restarts automatically. As the game doesn't end until the exit button is clicked, the scoring system will keep tracking the scores.

# 3. GAME COMPONENTS

There are 5 important components:

1. User Interface
2. Sound
3. Artificial Intelligence
4. Competitive play & Networking
5. Physics

## 3.1 User Interface

The javaFx library was used to develop the menu and user interface. The following elements were used to create the game menu: Scene, pane, text, slider, line, linear gradient.

The user interface will be discussed in further detail in the 'Interface Design' section.

## 3.2 Sound

The sound effects used in this game were developed using the media-player element of javaFx. There are 4 sound effect files used in this game;

1. **click.mp3** is played whenever a button is clicked on the menu.
2. **explodetank.mp3** is played whenever a tank is hit by the bullet.
3. **pops.mp3** is played whenever a tank shoots a projectile.
4. **music.mp3** is played in the background of the start-up menu and is set to stop when the game start.

## 3.3 Artificial Intelligence

There are 2 components to the artificial intelligence in this game:

**Shooting AI-** The AI tank calculates where to shoot a projectile. If there is a direct shot available, the AI will shoot at the correct angle. If not, it will check to see if a projectile

can be bounced off the walls to hit the opposing tank.

**Path finding-** The depth first search algorithm is used to calculate a path through the randomly generated maze, from the AI tank to the opposing tank. The AI tank then follows this path.

### **3.4 Competitive play & Networking**

The competitive play concept is seen in the local and online options of the game. The users can keep playing against each other, to see who can gain the most amount of points.

Up to 9 players can play against each other in online mode.

### **3.5 Physics**

The concept of physics applied in this game are collision, acceleration & deceleration and projectile bounce.

1. **Collision** - Collision is applied between the tank and the walls of the maze. If a tank collides with a wall, the tank's velocity is set to 0.
2. **Acceleration & Deceleration** - The tank accelerates when setting off and decelerates when stopping, to make the game more realistic.
3. **Projectile bounce** - The projectiles can bounce off the walls 2 times before disappearing. When a projectile hits a wall, its speed remains the same but its direction is changed to the opposite direction to make it rebound correctly.

## **4. FINAL REQUIREMENTS**

### **4.1 Major System Capabilities**

In the SRS ,we have listed out our game's requirements. However, as this project was conducted, some changes have been made in these areas;

#### **1. Menu**

In previous SRS report, the game menu only had 2 options which were game mode and join game. We have made some great changes to the game menu when conducting this project. Players some more option to choose from now. These options are:

- Select play to choose 3 different game modes (Player vs AI, 2 Player Local, Online Multiplayer)
- Select a desired map type after selecting the game mode.
- Select the settings button which includes sound settings and volume settings.
- Select game instruction button which displays the game description and instructions.

## **2. Maze**

As planned, we have managed to implement the maze with walls into the game and have made 2 options available for the player; Random Map & Preset Map (3 preset mazes available).

## **3. Tank**

For the tank requirements, more functionality has been added where the tank can now move with acceleration and deceleration. Previously, tanks were set to have 3 lives, however, after some discussion, we decided to make the game go on continuously without lives and have the score system instead as having lives without resetting positions allowed for unfair gameplay.

## **4. Collision**

In this game we managed to apply collision between tanks and walls in the maze where tanks are prevented from going through walls in the maze.

In the early stage of the project, we defined our game to be able to detect collision between tanks and should prevent tanks to go through each other. However, we found that it could be annoying to collide with other tanks when trying to move through narrow areas.

## **5. Sound**

In the sound section of the game we successfully inserted the sounds effects on these events:

- Starting up the game.
- Tank shooting.
- Player dying when they are hit by the enemy's bullet
- Clicking on button.

User also can adjust the volume, switch off the sound effects and also switch on the sound effects after switching it off.

## **6. Weapons**

In the early stage of the project, we wanted to make this game with weapons available for the tanks. Weapons would be spawned randomly in the maze and the tank would be able to collect it for use. However, due to time constraints we were unable to implement this area in the game.

Note: The requirements documentations is attached on the appendix section.

## 5. SOFTWARE DESIGN

### 5.1 Software Architecture design & Justification

In this project, we used a combination of different architecture;

- Client/Server Architecture - To ensure players can play online, we used client/server architecture.
- Top-level Components:
  - The game engine is responsible for presenting the game to the player and for receiving player inputs.
  - The simulation is responsible for updating the state of the game in response to player inputs
  - The object system is responsible for maintaining the state information describing all objects in the game world.
  - In this project, there is no data manager involved.

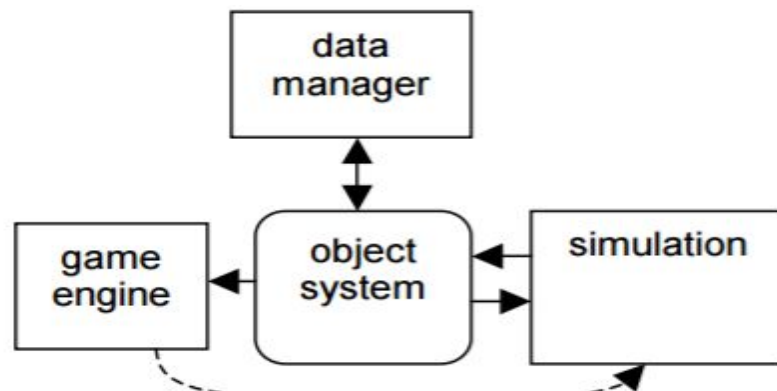


Figure 1: Top-level Components.

### 5.2 Design Pattern and Justification

For this project, we implemented the use of Behavioural patterns. We have different classes for different objects, namely:

**Player.java** - Represents the 1st player tank (green tank)

**Player2.java** - Represents the 2nd player tank (red tank)

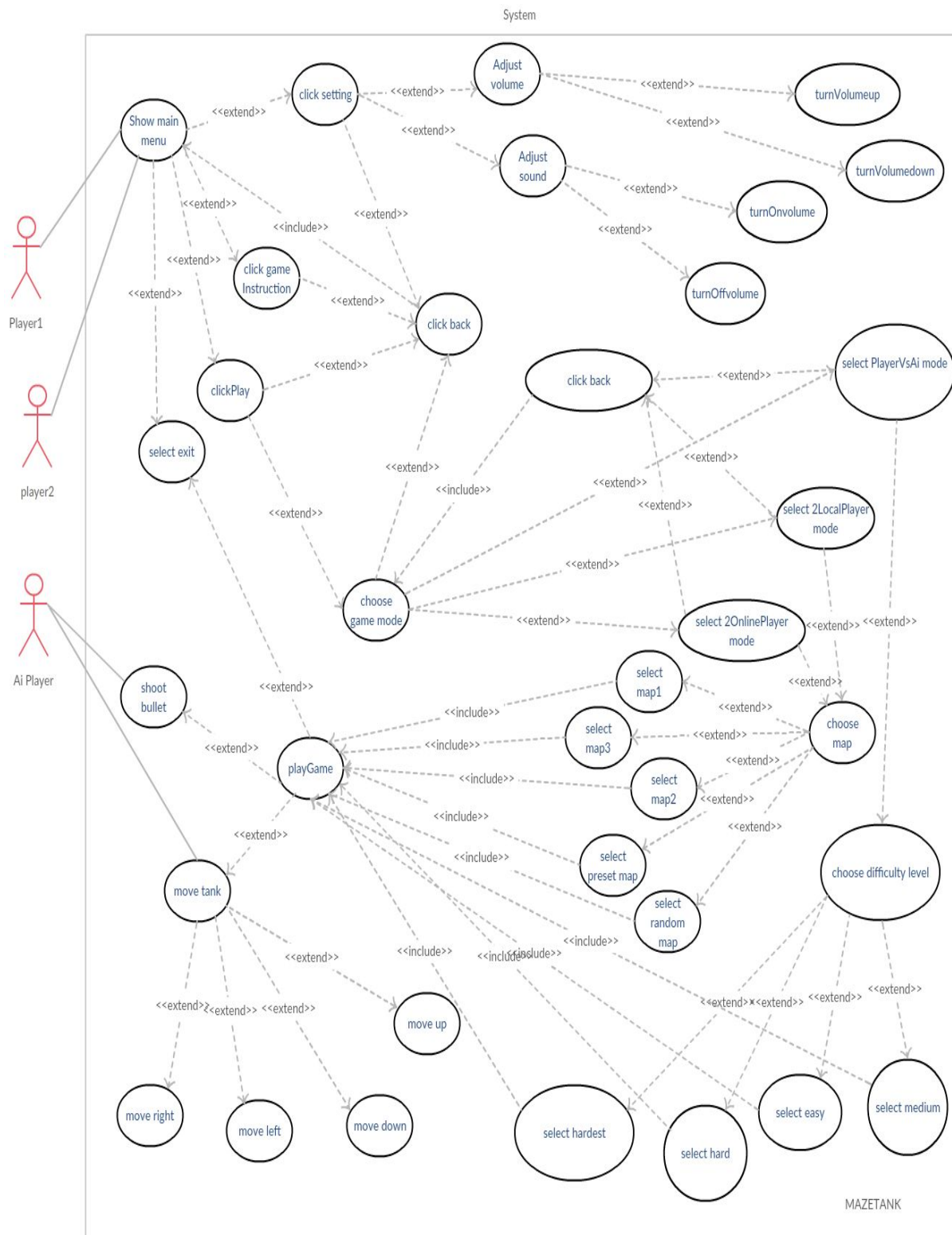
**PlayerAI.java** - Represents the AI player tank (blue tank)

**Projectile.java** - Represents the projectiles fired.



### 5.3 UML Diagrams

## 1. Use Case Diagram



## 6. INTERFACE DESIGN

### 6.1 Principles of Design

We considered these 2 aspects when planning for the HCI design of the game:

- Perceiving size and depth.

Perceiving size and depth will be important in a game where it helps players to visually sense on objects position and distance.

After some discussion on the game design, we decided to make the game as a top-down game. This means the size and depth of all objects in the game will not change and will be constant throughout the game.

- Perception of colour and brightness

We did some research on people's favourite colour and found that both men and women tend to like blue compared to other colours. However, as we applied the blue colour scheme to the game, the overall impression was not appealing. From a psychological perspective, blue is said to be a colour that brings a calming effect, but what we want is for the players to be enthusiastic and feel eager to play the game.

Therefore, we decided to choose brown as the background colour as it makes the game look aesthetically pleasing, and it matches the theme of the rural desert-like game we were aiming for.

As for the tanks colour, we decided to use blue, red and green as these are the top 3 preferred colours by men and women. Another reason for choosing these specific colours for the tank is because we want to make them more visible for the player to distinguish between their tank and the enemy's tank. Brighter colours make it easier for the objects to be seen.

### 6.2 Game Interface Elements

There are 3 elements of the interface:

**1. Text** - Both Serif Fonts and San Serif Fonts are used in this game. San Serif Fonts is used in the game name on the top of the game menu. This is to give a fun and energetic impression towards the game. On the other hand, Serif Fonts is used on the game menu items as it gives a delicate, warm aesthetic to the game.

**2. Icon/Pictures** - We wanted to give this game a natural and interesting look so we used an image of a wood veneer as the layout background. For the game description in the game menu, we made use of pictures to explain the tank's movement as the pictures give a better explanation than a block of text. The user can visualise the keys and recognize the keys instantly.

**3. The Game** - The game itself, which contains the tanks in a maze. The background of the game is a light brown to match the theme from the menu.

### 6.3 Interaction Style

The interaction style used in the game is Menu. In the game menu, a number of items are displayed and the player can click on the items which will result in a change of the interface state. The menu that our team created is a pop-up menu. When a player clicks on a menu item, it will make some items visible, which the user will click next, and some items invisible, which the user has already seen.

### 6.4 HCI Evaluation

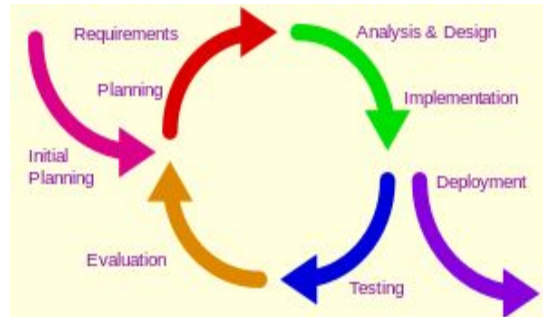
When we completed our project, we asked a small sample from the year group to test the game, so we could gain feedback. From the feedback that we got, the following table illustrates the heuristic evaluation that we can conclude for MazeTank.

HEURISTICS	EVALUATION
Provides consistent responses to the user's action	When a key is pressed, the tank moves and shoots accordingly. The menu responds to the user's click in a reasonable amount of time and acts accordingly.
Allows users to customize settings	The user can adjust the volume of the game. The user can turn the sound of the game on and off.
Provides predictable and reasonable behaviour for computer controlled units	The AI tank movement is reasonable and can be adjusted depending on the difficulty of the game. The aiming of the AI is very accurate but can also be avoided quite easily.
Provides controls that are easy to manage	The tank controls are easy to manage and easy to understand by players of all ages.
Provides instructions and help	The instructions are accessible in the game menu by clicking the 'instructions' button. The instructions help players understand the game.
Provides visual representations that are easy to interpret.	Players are able to interpret the game easily. We made a simple game menu, easily understandable game and mazes which are very easy to navigate through.

## 7. SOFTWARE PROCESS

### 7.1 Software Process, Principles and Methodology

For this project, we used the Iterative and Incremental Development model.



**Figure 1:** Iterative and Incremental Development cycle

Using this model, we first discussed on the planning and requirements of MazeTank. It was quite hard to capture specific requirements at the early stage of the project, so we tried our best to capture the major requirements of the project as the minor requirements could be added as the project progressed

We divided our project into sub-parts. These sub-parts were the goals that we wanted to achieve within the 11 weeks given to complete this project.

The summary of the steps taken are as below:

Phases	Description	Iterative and Incremental Development cycle
Initial Planning	During the early stage, we had a discussion on what style of game we would like to approach.	Initial planning
Capturing the Project's requirements	After deciding on the project, we discussed and captured the project's requirements.	Planning requirements
SRS documentation. Get a rectangle which	During week 3, we produced a SRS document. We also	Planning requirements Analysis & Design Implementation

able to move and shoots	started to create the game and had a rectangle that could move around.	Testing Evaluation (completed 1 cycle)
Refined rectangle movements. Started to develop the Game's networking Started to create a Game Menu. Started to create a random maze generator.	We divided our team to work on different sections of the project, which included: Refining the rectangle's movement, starting the networking, working on a random maze generator and the game menu	Planning requirements  Analysis & Design  Implementation
Integration of parts developed so far	Integrating the random maze generator and the rectangle into a game. Integrating the game menu into the game.	Implementation Testing Evaluation (Completed 2 cycles)
Started on creating an AI player. Integrating the Networking into the game	Some members worked on creating pathfinding and auto aiming for the AI player. Everyone else worked on bug fixes and integrating the networking.	Planning requirements Analysis & Design Implementation Testing Evaluation Deployment
Integration of AI Bug fixing Game improvement	The AI was then integrated into the game. Then the team worked on testing and bug fixing, whilst adding small improvements to the game.	Planning requirements Analysis & Design Implementation Testing Evaluation Deployment

By developing the project using this model, we could generate a working game during the early stages of the project, able to identify risks and carried out testing effectively

## **8. Risk Analysis**

Despite the benefit we got from following the Iterative and Incremental Development model, we also had some difficulties where we would sometimes spend too much time focusing on the sub-goals of the project and lose our focus on the main goals and criteria. We also had some problems when integrating the elements that had been developed separately. However, these issues had been solved with great teamwork and chemistry that we had in our team. Although we were sometimes off track from the time planning, we caught up with the help of each other.

## **9. GAME EVALUATION**

### **9.1 Game Strengths**

#### **1. The game is suitable for all ages as:**

- It does not contain any violent elements, therefore it's a game that can be played by all ages.
- It also has an easy and simple control which make it easier for the younger audience to play this game.
- It has a similar concept to the game “tank 1990”, so the older audience may be interested to play this game for the nostalgic feel.

#### **2. Straightforward and uncomplicated**

- Some people tend to like less challenging games because hard games make them feel the goals are out of reach, thus tend to give up on the game.
- In MazeTank, we considered this factor and made it so it matches the average ability level of anyone from our target audience.
- We also made this game in a way that it is not complicated for a new player to understand as the mazes are very easy to navigate through. So, no practice is required.
- For this reason, we believe that it will help to increase the maximum audience that can try this game.

#### **3. Not time consuming**

- There are some people who feel that playing games is a waste of time and a form of guilty pleasure.
- The game is suitable for those who are busy with life commitments, but still want to take a break and play simple games.
- Therefore, MazeTank is a suitable game for this audience as it is less time consuming.

#### **4. Difficulty level**

- In AI game mode, MazeTank has 4 options for the difficulty (easy, medium, hard, extremely hard). Therefore, players with different levels of ability can choose their level preference.

#### **5. More players in online game.**

- More players in a game means higher competition with friends which makes the game more fun. Therefore, it will be easier to influence others to start playing MazeTank.

### **9.2 Game Weaknesses**

#### **1. Game is too easy**

- Because of the current state of the game, some people might find it too easy and get bored easily.

#### **2. Not much variety**

- There is not much depth to the game and gets quite repetitive.

### **9.3 Improvement to make**

Due to time constraints and not much experience in making games, there are a few features that were interesting to have in our game but we were unable to implement.

These features are:

#### **1. Score Ranking**

MazeTank has a game score which keeps track of the points scored by the players in a game. If a ranking system can be implemented in the game, it would encourage players to invite friends and compete against each other in the game.

#### **2. Tank Weapons/Upgrades**

Having tank weapons would make the game much more interesting and would add a lot of variety. Different weapons could include having a tank with the power to go across a wall, make the bullets more advanced, having a tank shoot lasers pointing out to the target or have shield to protect the tank from enemy.

## 10. SUMMARY

In 11 weeks, we managed to complete the task provided and create MazeTank with all the features needed, namely User Interface, Sound, Artificial Intelligent, Networking and Competitive Play. We also applied physics concepts in the game such as collision, acceleration & deceleration and projectile bouncing. This project was a new experience for us and helped us to learn more, especially in using javaFx. Although the game is not perfect, in a short amount of time, our team managed to complete the task. In the early stages of planning, we had some features that we wanted to put in the game but we were unable to do so because of the time constraints. Lastly, this project helped us to improve our communication skills, teamwork and helped us to be more organized during the project planning.

## 11. CONTRIBUTIONS ASSESSMENT

Team Project individual contributions assessment

Your name: **SAKINAH MOHAMAD**

Your team: C5

Team member (include your own name in this list)	Contribution (%) (should sum to 100%)
SAKINAH MOHAMAD	20
KEVIN DEVABATTULA	20
TOM BYRNE	20
LEWIS DEAKIN-DAVIES	20
HUZEIFAH MOHAMMED	20



Your name: **Huzeifah Mohammed**

Your team:C5

Team member (include your own name in this list)	Contribution (%) (should sum to 100%)
SAKINAH MOHAMAD	20
KEVIN DEVABATTULA	20
TOM BYRNE	20
LEWIS DEAKIN-DAVIES	20
HUZEIFAH MOHAMMED	20

Your name: **Tom Byrne**

Your team:C5

Team member (include your own name in this list)	Contribution (%) (should sum to 100%)
SAKINAH MOHAMAD	20
KEVIN DEVABATTULA	20
TOM BYRNE	20
LEWIS DEAKIN-DAVIES	20
HUZEIFAH MOHAMMED	20

Your name: **Kevin Devabattula**

Your team:C5

Team member (include your own name in this list)	Contribution (%) (should sum to 100%)
SAKINAH MOHAMAD	20
KEVIN DEVABATTULA	20
TOM BYRNE	20
LEWIS DEAKIN-DAVIES	20
HUZEIFAH MOHAMMED	20

Your name: **Lewis Deakin-Davies**

Your team:C5

Team member (include your own name in this list)	Contribution (%) (should sum to 100%)
SAKINAH MOHAMAD	20
KEVIN DEVABATTULA	20
TOM BYRNE	20
LEWIS DEAKIN-DAVIES	20
HUZEIFAH MOHAMMED	20

## **12. INDIVIDUAL REFLECTION**

**NAME: SAKINAH MOHAMAD**

In this project, I was given a task to create the user interface for the game, which is the game menu and I also responsible to implement the sound into the game. The user interface and sound are both implement into the game using javaFx. This is a great opportunity for me to enhance my skills and learn about javaFx as I have no experience using it before. As for the sound, I'm using import java.io.File to read the sound files and all of the sound files are in mp3 format. In the game menu, I've created a setting button which contain Sound button and Volume button. Player can click on the sound button to switch on or to switch off the sound. As for the volume button, I've implement a slider which player can use to adjust the volume of the sound. In the game itself, I contributes in the game physics (collision, acceleration, deceleration) and game AI together with my team mates. Lastly, I play a great role in writing the final report. I'm certainly have advantage on this one as I'm taking the software engineering modules and had learn about requirement, software processes and UML diagrams. Knowing about these things certainly help the group with the project planning process.

**NAME: HUZEIFAH MOHAMMED**

During this project my main roles were, creating the random maze generator and working on AI alongside Kevin. Both of these roles were new experiences for me. Before we started to created the game, I had doubts on whether I would be able to fulfil these roles, however with the help of my teammates we managed to create a very fun game, with quite advanced AI mechanics.

Me and Kevin split the AI into 2 parts, the pathfinding and the shooting. We decided that I would work on the shooting and Kevin would work on the pathfinding. This method worked perfectly and we created a very convincing AI bot. After creating the AI player, we saw some more potential and thought playing against the AI could be made more difficult. Kevin thought of increasing the speed of the AI bot, depending on the difficulty chosen. I thought of increasing the number of projectiles fired by the AI, depending on the difficulty. We decided between the 2 ideas by voting in the group, and it was decided we would use Kevin's idea.

Lewis worked on the creation of the main game. He worked on creating the tanks so they could move and shoot in any direction, making the projectiles bounce off walls and he also worked on most of the physics in the game. We would see his constant progress, throughout the 11 weeks, on Whatsapp through his video messages. He also helped Tom with the Networking and Multiplayer when he was having some problems.

Originally Kevin worked on Networking and Multiplayer however the game was not

developed enough for the networking to be fully functional, so Kevin moved onto AI. Later on Lewis took over networking and with the help of Tom, they were able to fix all the issues and managed to create a working multiplayer game.

Sakinah created the User Interface and also implemented sound into the game. Whenever we needed something added into the UI, she was ready to help and implemented it very quickly. She also contributed to the game physics and AI.

Tom contributed to various parts of the project including networking, UI and also did a lot of bug fixing. He made a UI for the players to host or join a game and did all the JUnit testing.

As mentioned before, Kevin worked on the AI. He also integrated everyone's code into one game and contributed to the networking.

Altogether, as a team, we worked together smoothly. Everyone contributed equally to the project and each team member was key to the success of the project. Whenever a team member had any problems, we would all discuss the problems and solve it together. We had team meetings twice a week, every Tuesday and Friday, where we would discuss the progress made from the last meeting and what needs to be done by the next meeting. We also created a Whatsapp group where we discussed various issues related to the project and shared images/videos of what we had developed.

Due to our good communication and teamworking skills, we managed to create an impressive game and we all learnt from each, building on each others strengths.

## **NAME: TOM BYRNE**

Throughout the creation of MazeTank I felt that our group worked very well as a team. We made sure to frequently communicate with each other, explaining what we were working on and when we thought it would be finished – this prevented multiple team members from working on the same piece of code at once and essentially repeating work. We had team meetings twice a week where we would demo what we've been working on and ask for help if we were having difficulties. I found these meetings particularly beneficial as it was much easier to explain and resolve problems while working together in person rather than over the internet. During these meetings, we would often employ the technique "pair programming", I found that this helped to overcome more difficult parts of the coding that we were struggling with as individuals. During our first meeting, we discussed each other's strengths and weaknesses and made a rough plan as to who was going to work on each component of the game. To conclude each subsequent meetings, we would all agree on what tasks needed to be completed next and select which tasks we were going to do.

Everyone on the team contributed to the final product in various ways. Sakinah created the base game menu/UI after learning how JavaFX works and implemented the sound engine for the game, she also used her software engineering skills to produce various useful UML diagrams and put a lot of effort into the final report.

Lewis created the initial game in SWING, it consisted of 2 squares that could move around

on screen. Over the next week or so with some assistance from the rest of the team the game was built up to feature shooting, physics and improved graphics. Kevin then went on to port the game over from SWING to JavaFX as we had decided in our SRS.

With the main game being basically finished we split our efforts into two main directions, networking and AI.

Kevin had already started to develop the networking code but got stuck after a certain point, Lewis took this task over with support from me. After a lot of hard work we managed to produce functional networking code.

Kevin and Huzeifah took on the majority of the AI workload, splitting it into 2 components: movement and shooting. They developed a few different methods for doing this that all worked differently, we assessed each of the methods as a group before deciding what would be used in the final product. Their AI code was later improved to feature various difficulty settings.

Once we had all of the game components working we fully integrated the code and continued testing, only making small code changes to improve performance/playability.

## **NAME: LEWIS DEAKIN-DAVIES**

The development of MazeTank started off quickly and effectively, we laid down the rules and the logic for our game and began. I started setting the foundations for the game by putting together a playable base game in swing that implemented some physics and playability. This involved being able to move two players around whilst shooting projectiles in a window. As this was being worked on Sakinah developed her menu in the already decided on JavaFX. The base game I made was to just get something working with plans to be changed to FX. Once we had a working menu, Kevin was able to change the code I had written to be compatible with FX and not use swing. This allowed our game to seem more functional. Tom monitored different areas of the code and was constantly communicating bugs and fixes, ensuring the code was functional at the start of development.

Once we had the basic game running, Kevin made an attempt to learn about networking and managed to set up the foundations for the networking code we use now. The game needed to be more developed for the networking to work properly so Kevin and Huzeifah started working on AI. With some help from Sakinah they managed to get a functional AI system in place with varying difficulties, even managing to implement the complex calculations needed for the tanks to be able to determine the rebounding projectile's trajectory.

After I had sufficiently implemented what was needed in the main game, I took on networking. The code that Kevin had written before was very useful in understanding how it worked as I had no prior knowledge of how networking should function. He taught me how it worked properly and I spent a lot of time trying to get it functional. Tom helped me check my code and often fixed bugs and ironed out issues with me. Tom implemented a UI to make for easier access to the networking options and allowed for a proper host and client system. As

most of our group do joint honours, we hadn't learnt software engineering, so Tom took it upon himself to sort out the unit testing.

Along with the meetings twice a week we had constant git pushes and pulls and continuous communication in our group chat, helping each other out whenever needed and ensuring we were always all on the same page. Overall, everyone worked very hard when needed and cooperated effectively.

## **NAME: KEVIN DEVABATTULA**

At the beginning of the project, I took on the role of networking and tried to make a start on all the networking items. This proved to be a challenging task and I wasn't making much progress. At that point in our game we had created a basic version of the game with moving objects and a game menu, but the game was made in Swing while the game menu was made in JavaFX, so I decided to move everything over from Swing to JavaFX. This was an opportunity for me to teach myself the ins and outs of JavaFX to move everything over and was another challenge which I managed to complete so everything was integrated into one. While we were still working in Swing, we also had a random maze generator which would generate the walls of a maze of certain dimensions. This wasn't part of our game yet so I pulled the maze into our actual game and added the walls so collision could take place with the tanks and walls.

After this, I then set to work on the AI. The plan was to do the AI pathfinding first so the AI tank could get to the player tank and then work on the AI tank shooting at the player. Creating the pathfinding of the AI bot was difficult and took a bit of time and required me to use and look at many different java techniques, but eventually we built a solid AI bot that goes to the player as the player moves and then shoot at the player at an angle such that the player would die. After the basic AI bot was created we decided we needed various levels of difficulty which we agreed would be through the speed of the bot being increased.

In our team, Sakinah made the game user interface. She created the game menu in JavaFX and added the images and sounds of the game, such as the background of the game menu and the sounds that occur when a button is clicked or a player is shot. She also made a lot of contributions to the SRS and the final report as she had a lot of knowledge with that through her Software Engineering module.

Lewis made the game logic and was responsible for the movement and shooting of the tanks. His work was the first piece of code that we built our game off. He started off with 2 squares on an empty plain map which then could shoot smaller rectangles as bullets. He also added the acceleration and deceleration of the tanks and the collision of bullets with the walls as well. Him and Tom also did the networking together.

Tom monitored different areas of the code and found and fixed whatever bugs that he found. He also worked on the networking with Lewis, making sure that it was properly functional and that players could face each other on the game mode. He also made a UI for the players to host or join a game. He also did all the JUnit testing, since he had experience with the Software Engineering module and could understand and complete the testing.

Huzeifah created the algorithms and methods needed to generate a random maze for our game. He also worked with me to create the AI and mostly worked on trying to make the AI shoot at the target player.

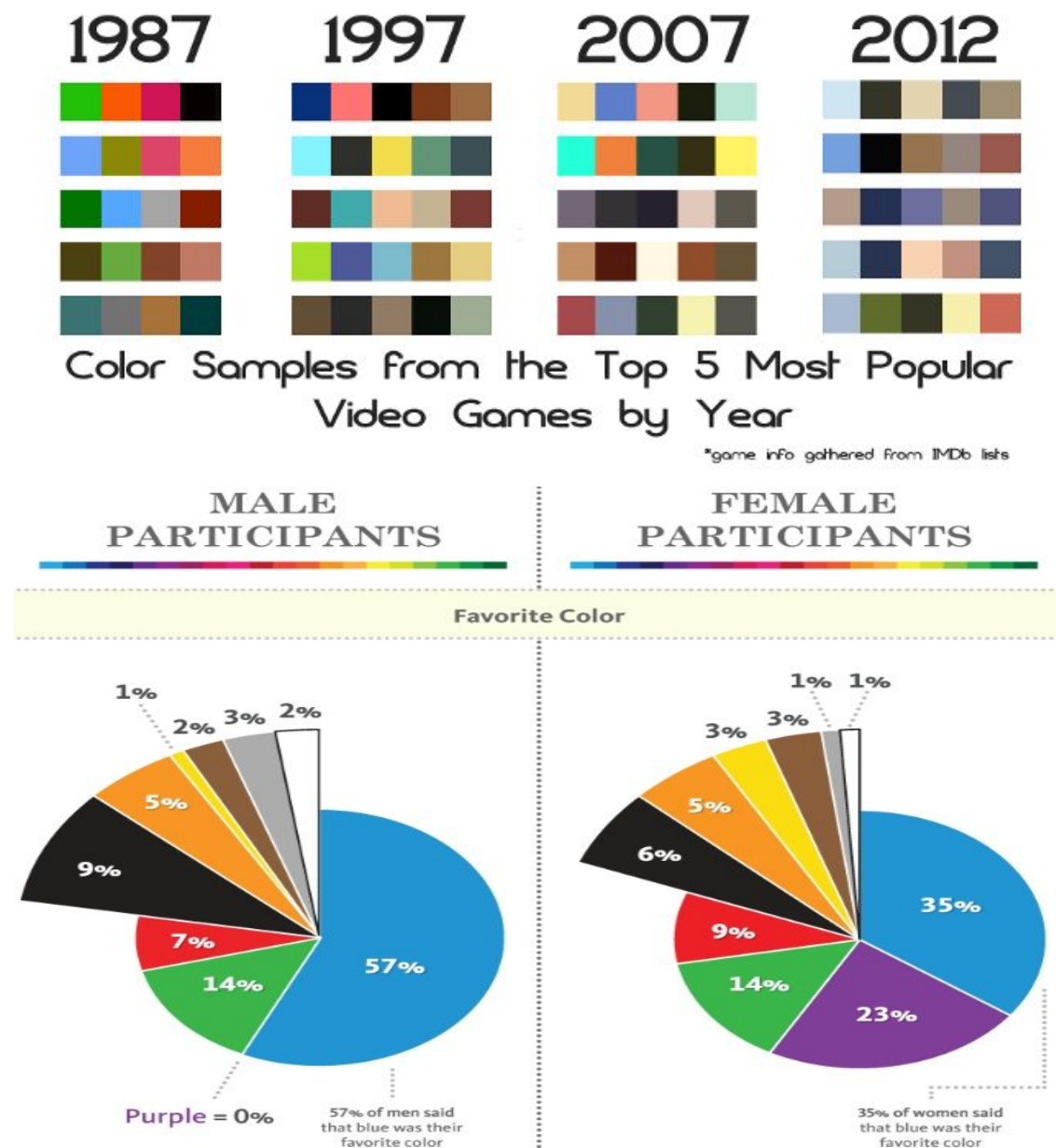
Altogether as a team I believe that we worked very well. We were constantly communicating with each other and communicating different bugs and other stuff across. We met twice a week and everyone worked hard and cooperatively.

## 13. REFERENCES

[https://www.theseus.fi/bitstream/handle/10024/43234/Nguyen\\_Hung.pdf?sequence=1](https://www.theseus.fi/bitstream/handle/10024/43234/Nguyen_Hung.pdf?sequence=1)

[http://www.gamasutra.com/blogs/HermanTulleken/20150729/249761/Color\\_in\\_games\\_An\\_indepth\\_look\\_at\\_one\\_of\\_game\\_designs\\_most\\_useful\\_tools.php](http://www.gamasutra.com/blogs/HermanTulleken/20150729/249761/Color_in_games_An_indepth_look_at_one_of_game_designs_most_useful_tools.php)

<https://www.templatemonster.com/blog/font-psychology/>





## **14. APPENDIX**

### **14.1 System Requirements**

#### **1. Menu**

- 1.1. The game should have menu system which allow players to click play.
- 1.2. The game should have menu system which allow players to click setting.
- 1.3. The game should have menu system which allow players to click game instruction.
- 1.4. The game should have menu system which allow players to click exit.
- 1.5. The game should have menu system which allow players to select game mode: PLAYER vs AI.
- 1.6. The game should have menu system which allow players to select game mode: 2 Local Player.
- 1.7. The game should have menu system which allow players to select game mode: 2 Online Player.
- 1.8. The game should have menu system which allow players to select Map 1 after selecting game mode.
- 1.9. The game should have menu system which allow players to select Map 2 after selecting game mode.
- 1.10. The game should have menu system which allow players to select Map 3 after selecting game mode.
- 1.11. The game should have menu system which allow players to select Random Map after selecting game mode.

#### **2. Maze**

- 2.1. The game shall have a maze map with wall obstacles.
- 2.2. The game shall have random maze option available for player.
- 2.3. The game shall have 3 preset maze option available for player.

#### **3. Tank**

- 3.1. The game shall allow players to move the tank in all direction (left,right,up,down)
- 3.2. The game shall allow a players to fire bullets with a tank.
- 3.3. The game shall allow the tank's bullets to reflect for 5 times if it's hit the wall.
- 3.4. The game shall allow tanks to move with acceleration.
- 3.5. The game shall allow tanks to move with deceleration.

#### 4. Collision

- 4.1. The game shall detect a collision between tank and a wall in maze.
- 4.2. The game should be able to prevent tank from going through the walls in maze.

#### 5. Sound

- 5.1. The game should play shooting sound effect when tank shoot a bullet.
- 5.2. The game should play background sound effect when player is on game menu.
- 5.3. The game should play exploding sound effect when tank got hit by a bullet.
- 5.4. The game should play click sound effect when player click on the menu buttons.
- 5.5. The game shall allow player to adjust the volume by sliding the volume slider .
- 5.6. The game shall allow player to switch off the sound effect when sound button is clicked
- 5.7. The game shall allow player to switch on the sound effect when sound button is clicked

## **Test Report -**

Initially when we were working on MazeTank, we used incremental integration testing. Whenever any new functionality was added we would manually test it under different circumstances to make sure it worked as expected. As the development of MazeTank went on it became clear that this type of testing was inefficient; simple code modifications would completely break the tests, they were cumbersome to run and difficult to keep track of/updated.

To combat these problems, we decided to use JUnit for automated testing. Once the test cases were written they could all be added to a test suite and ran simultaneously, this saved us a lot of time as we no longer had to figure out which tests our code changes might have effected before running the tests. We continued to write JUnit tests as we went along for each testable method, giving us good test coverage and a large test suite that could quickly and easily be ran before each git push. Having a large test suite became increasingly beneficial as the project went on as we could quickly identify errors that would normally have gone unnoticed and created a bigger problem further down the line.

Some types of testing still had to be performed manually such as stress testing the networking code for example. In this case, we used real world manual testing/play testing, by loading up multiple versions of the game, across multiple computers and networks to see if it could handle the artificially load limit we put in place.

### **JUnit Test Classes:**

#### **GameMainJUnitTests -**

CellEastWallTests, CellSouthWallTests, CellVertWallTests, CellHoriWallTests  
CellEastWallAiTests, CellSouthWallAiTests, CellVertWallAiTests, CellHoriWallAiTests  
CellTests, GameAiTests, GameEngineAiTests, GameEngineTests, GameObjectTests,  
GameTests, HandlerTests, MapTests, Player2Tests, PlayerAiTests, PlayerTests,  
ProjectileTests.


#### **NetworkingJUnitTests-**

CellEastWallTests, CellSouthWallTests, CellVertWallTests,  
CellHoriWallTests, GameEngineTests, ConcreteNetGameObject, GameObjectTests,  
GameTests, HandlerTests, PlayerIDTests, PlayerTests, ProjectileTests, ServerPlayerTests,  
NetGameEngineTests, NetworkingInterfaceTests.

Figure 1: EcIEMMA coverage report.



































▼  MazeTankIntegrated	 80.9 %	11,861	2,801
▼  src	 80.9 %	11,861	2,801
>  Networking	 61.8 %	2,179	1,348
>  GameMain	 87.7 %	6,495	914
>  GameMenu	 85.5 %	3,187	539

Figure 2: Completed JUnit Test Suite Run

 JUnit

Finished after 0.718 seconds

Runs: 95/95
Errors: 0
Failures: 0

- >  NetworkingJUnitTests.ServerPlayerTests [Runner: JUnit 4] (0.002 s)
- >  GameMainJUnitTests.HandlerTests [Runner: JUnit 4] (0.000 s)
- >  GameMainJUnitTests.PlayerAiTests [Runner: JUnit 4] (0.377 s)
- >  GameMainJUnitTests.MapTests [Runner: JUnit 4] (0.005 s)
- >  GameMainJUnitTests.CellSouthWallTests [Runner: JUnit 4] (0.001 s)
- >  GameMainJUnitTests.Player2Tests [Runner: JUnit 4] (0.006 s)
- >  GameMainJUnitTests.GameEngineAiTests [Runner: JUnit 4] (0.004 s)
- >  GameMainJUnitTests.GameAiTests [Runner: JUnit 4] (0.047 s)
- >  NetworkingJUnitTests.NetworkInterfaceTests [Runner: JUnit 4] (0.039 s)
- >  NetworkingJUnitTests.ProjectileTests [Runner: JUnit 4] (0.004 s)
- >  NetworkingJUnitTests.NetGameEngineTests [Runner: JUnit 4] (0.003 s)
- >  GameMainJUnitTests.CellSouthWallAiTests [Runner: JUnit 4] (0.001 s)
- >  NetworkingJUnitTests.CellHoriWallTests [Runner: JUnit 4] (0.005 s)
- >  GameMainJUnitTests.CellHoriWallTests [Runner: JUnit 4] (0.001 s)
- >  NetworkingJUnitTests.GameEngineTests [Runner: JUnit 4] (0.007 s)
- >  NetworkingJUnitTests.CellVertWallTests [Runner: JUnit 4] (0.000 s)
- >  GameMainJUnitTests.ProjectileTests [Runner: JUnit 4] (0.001 s)
- >  GameMainJUnitTests.CellVertWallTests [Runner: JUnit 4] (0.000 s)
- >  NetworkingJUnitTests.CellEastWallTests [Runner: JUnit 4] (0.000 s)
- >  GameMainJUnitTests.GameEngineTests [Runner: JUnit 4] (0.000 s)
- >  GameMainJUnitTests.CellTests [Runner: JUnit 4] (0.000 s)
- >  GameMainJUnitTests.CellEastWallAiTests [Runner: JUnit 4] (0.000 s)
- >  NetworkingJUnitTests.PlayerIDTests [Runner: JUnit 4] (0.000 s)
- >  NetworkingJUnitTests.NetGameTests [Runner: JUnit 4] (0.000 s)
- >  NetworkingJUnitTests.GameObjectTests [Runner: JUnit 4] (0.004 s)
- >  GameMainJUnitTests.CellEastWallTests [Runner: JUnit 4] (0.000 s)
- >  NetworkingJUnitTests.HandlerTests [Runner: JUnit 4] (0.001 s)
- >  NetworkingJUnitTests.PlayerTests [Runner: JUnit 4] (0.002 s)
- >  GameMainJUnitTests.GameObjectTests [Runner: JUnit 4] (0.004 s)
- >  GameMainJUnitTests.GameTests [Runner: JUnit 4] (0.001 s)
- >  GameMainJUnitTests.CellHoriWallAiTests [Runner: JUnit 4] (0.000 s)
- >  GameMainJUnitTests.CellVertWallAiTests [Runner: JUnit 4] (0.000 s)
- >  NetworkingJUnitTests.CellSouthWallTests [Runner: JUnit 4] (0.000 s)
- >  GameMainJUnitTests.PlayerTests [Runner: JUnit 4] (0.001 s)