Requirements and Analysis Document for Tank Revolution

Version: 2.0

Date: 2017-05-22

Author: Group 12

This version overrides all previous versions.

1 Introduction:
People love to play games, especially on their phone when they have some minutes to spare while for example waiting for the bus. When scouting the competition in this sort of games we have discovered that there is potential of making this game better and more friendly to play during short periods of time. We want this game to entertain you whenever you need it to.

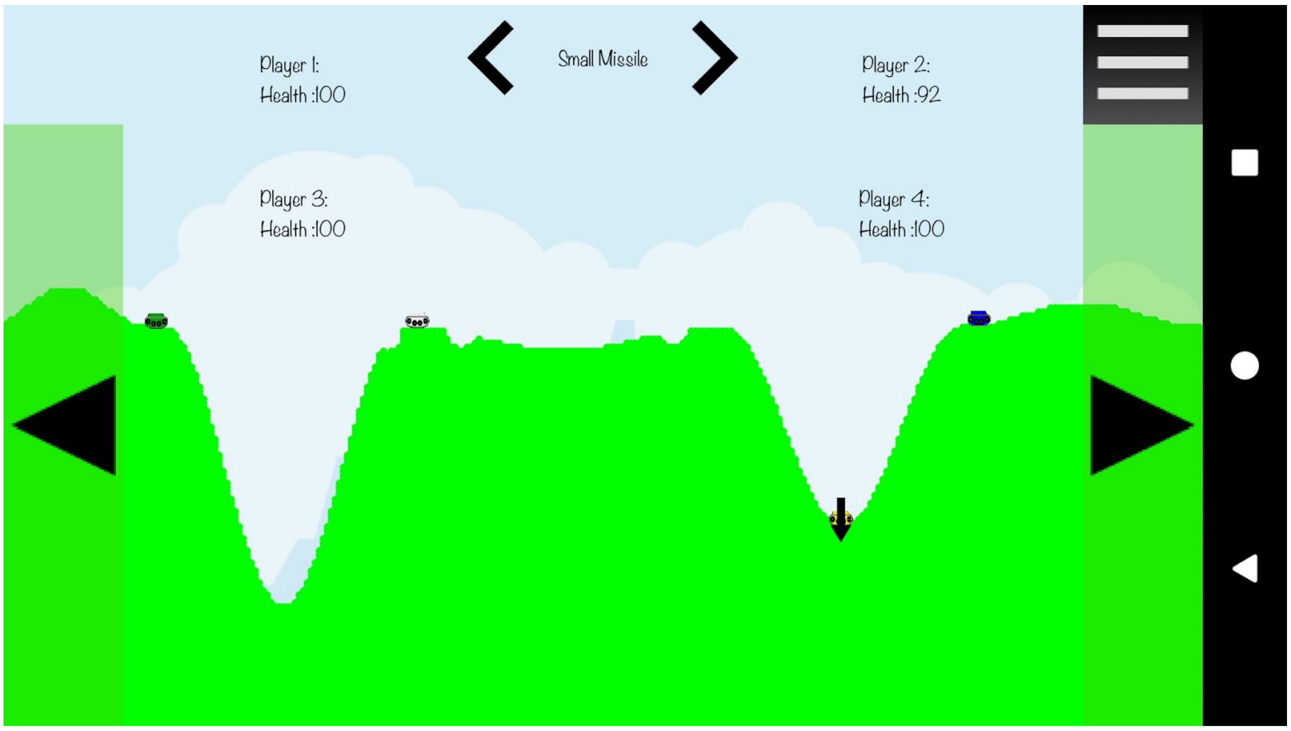
1.1 Definition:
A 2D, turn based artillery Android game where you are a tank. You take turns with your opponents of making moves. In one move you can move your tank, you have gas that will reduce when you are moving. You can also switch weapons and fire shots. The goal of the game is hit the other tanks with your shots, when a tank is hit its health will be reduced. When it doesn't have any health left the tank explodes and the game is over for that player. In each turn you get new gas so that you can move. The game ends when there's only one tank remaining.

1.2 Scope:
The game has two modes: "Quick Game" and "Custom Game".
In Quick Game the user plays against a computer NPC of difficulty medium on the first map in our list.
In Custom Game the user gets to pick between 5 different maps, can choose to have between two and four players, pick whether the players should be controlled by human or by NPC and set the NPC difficulty of each tank at four different levels.
There are four different weapons to chose from, differing in damage and blast radius but not in appearance.
The user can, at any time, pause the game and in that menu chose to resume, restart or end the game. When one tank is remaining the only option is to return to the main menu.

1.2 Definitions, acronyms and abbreviations:
NPC: "Non-player character", a character controlled by the game's A.I.
Player: Character controlled by a human user.

2 Requirements:

2.1 User interface:



An in-game screenshot showing four players, a bit of destroyed terrain and an arrow pointing to the current player.

2.2 Functional requirements:

The user should be able to:

Execute a turn:

- Pause the game if they want to:
    - Resume the game.
    - Restart the game.
    - End the game.
- Move the tank into a position they want.
    - See their remaining fuel.
- Switch to a different weapon.
    - See the name of their selected weapon.
- Fire a shot towards the other tank.
    - See the angle they're aiming at.
    - Abort the shot by dragging their finger close to where they started.
    - See the projectile follow the trajectory.

2.3 Non-functional requirements:

2.3.1 Usability:

This is a mobile game and it should be easy to play without having it explained, the controls and gameplay should be intuitive and not require a tutorial.
It might be played for example on a bus so the controls must be precise even in a bumpy environment.
The game should be easy to play even if you're on a bus or elsewhere where the surrounding is moving or is not stable.
The implementation should be in Java.
It will be on a small screen and scalable for different screen sizes.

2.3.2 Reliability:
The reliability of the app is not a high priority since this is a casual game and not a system which other systems are dependent on. The worst thing that can happen if the game encounters a critical error is that the user has to start a new game.

2.3.3 Performance:
Since this is a 2D game with exclusively 2D-models the performance was not really an issue. The one thing that mainly prevents the performance is the calculations performed on the terrain every time a projectile hits it. This could be improved with a better algorithm but that would take up more time than we have.

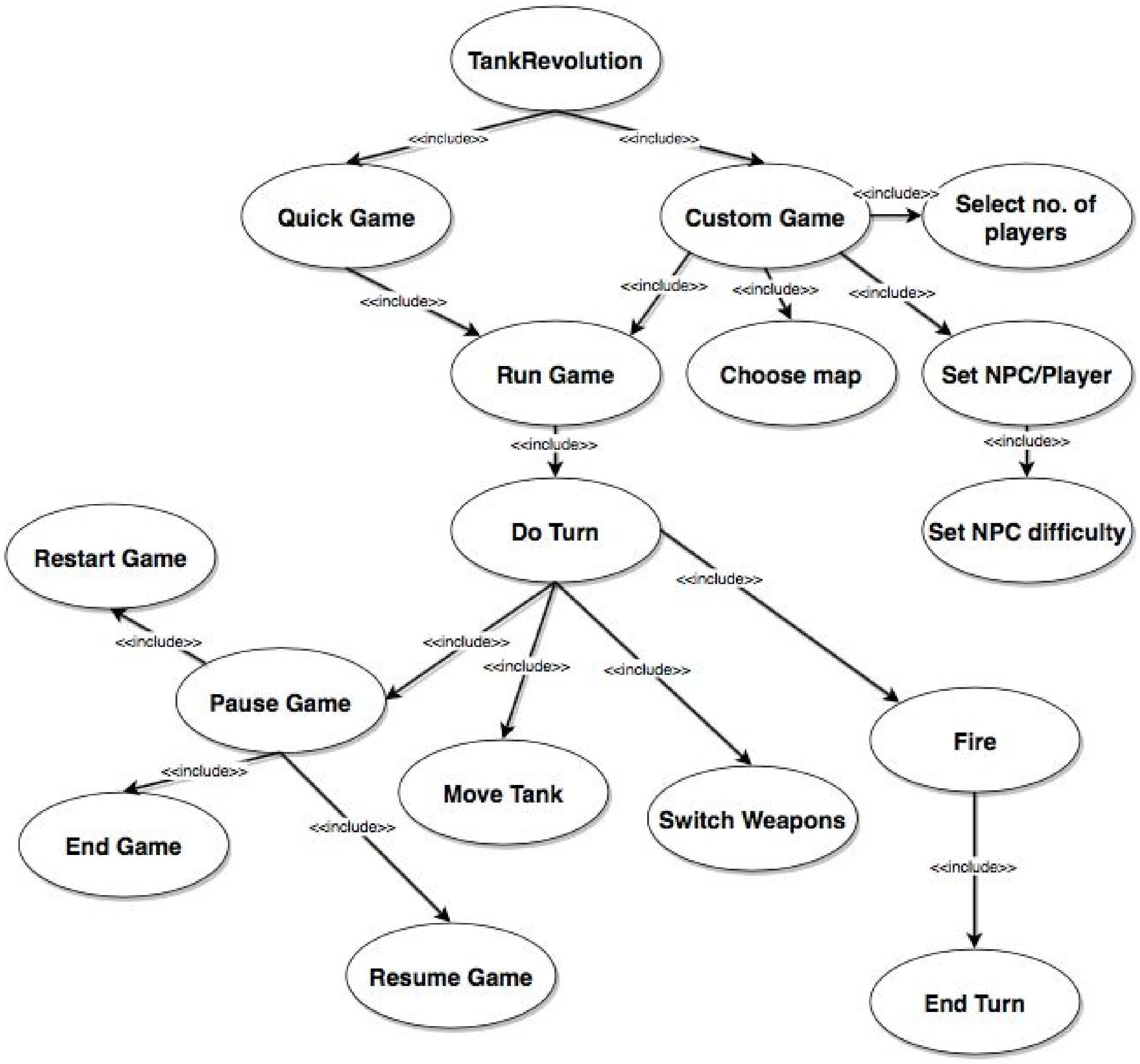
2.3.4 Supportability:
Since the game is heavily dependent on the libGDX framework and its graphic component, writing automated unit tests is not optimal and we have to instead rely on manual tests.
Extensions of the game are possible as long as the framework supports it.

2.3.5 Implementation:
The application will be written in Java using the libGDX game engine(BadLogic Games 2017).

2.3.6 Packaging and Installation:
This is an application developed for Android. The game will be distributed as an .apk-file(Android application package file). This file contains the whole game, including all necessary assets and can be installed directly onto any Android phone.
The desktop version is only for development and testing purposes and will therefore not be distributed as an executable.

2.3.7 Legal:
Since we do not have a lot of time to develop this project, some assets and some imported libraries will be made by someone else and protected by copyright law. However, since this project was strictly for educational purposes and not for monetary gain, this falls under fair use and no laws were broken.

3 Use cases:



A UML-style Use case diagram.

3.1 Use cases in order of priority:

1. Fire
2. Move
3. Start Game
4. Quick Game
5. Custom Game
6. Pause Game
7. Restart Game
8. Resume Game
9. End Game
10. Select no. of players
11. Set NPC/player
12. Switch weapon
13. Chose map.
14. Set NPC difficulty

Use Case: Fire

Summary: The player fires projectile by dragging and dropping on the screen and the system checks if the attack hits something.
Priority: High
Extends: Do Turn
Includes:
Participants: The actual player

	Actor	System
1	Drags finger on screen to decide the angle and velocity of the projectile.	
2		Shows a shoot vector on the screen corresponding to the input from user. The vector starts where the initial touch took place and mirrors the present position.
3	Lets go of the screen away from where he initialized the fire.	
4		Vector disappears. Shows animation of the projectile flying through the air.
4.1 Tank hit		Explosion animation is played. Enemy tank's health is reduced, terrain is partly destroyed. End turn.
4.2 Tank missed		
4.2.1 Terrain hit		Explosion animation is played. Terrain is partly destroyed. End turn.
4.2.2 Out of bounds		End turn.

Use Case: Move Tank

Summary: The player moves its tank by using one of the arrows on the sides of the screen. The tank can move as long as there is gas left.
Priority: High
Extends: DoTurn
Includes:
Participants: The actual player

	Actor	System
1	Click (and holds) on the arrow button on the side of the screen.	
2		Shows animation of the tank moving in the direction of the arrow being clicked. Shows the gas level in the arrow keys. Arrow keys are less transparent.
3	Lets go of the screen.	
4		Shows animation of the tank stopping. Shows the gas level in the arrow keys. Arrow keys are more transparent.

Alternative flow:
The player is out of gas.

1	Click (and holds) on the arrow button on the side of the screen.	
2		The system shows that the gas-level is empty by making the entire arrow-key gray and less transparent. Tank doesn't move.

Alternative flow:
The player moves into a steep wall.

1	Click (and holds) on the arrow button on the side of the screen.	
2		Show animation of the tank moving

		into the wall and bouncing back. Shows the gas level in the arrow keys. Arrow keys are less transparent.
3	Lets go of the screen.	

<u>Alternative flow:</u>
The player moves into the edge of the screen.

1	Click (and holds) on the arrow button on the side of the screen.	
2		Show animation of the tank moving into the wall and bouncing back. Shows the gas level in the arrow keys. Arrow keys are less transparent.
3	Lets go of the screen.	

<u>Alternative flow:</u>
Player collide with another tank.

1	Click (and holds) on the arrow button on the side of the screen.	
2		Your tank pushes the other tank in the same direction that your tank is moving with half the speed.

<u>Alternative flow:</u>
Player moves into a steep fall.

1	Click (and holds) on the arrow button on the side of the screen.	
2		The tank move over the edge. The tank starts to fall.
2.1		The tank hits the ground after the fall.
2.2		The tank doesn't hit the terrain (i.e. fall out of the screen) and the tank is destroyed.

Use Case: Start Game

Summary: A new session is started and the game UI is shown.
Priority: High
Extends:
Includes:
Participants: The actual player.

1		The menu screen disappears and the game UI is shown. See Custom Game/Quick Game

Use Case: Quickstart

Summary: The player starts a game session without changing any settings.

Priority: Medium

Extends:

Includes: Start Game

Participants: The actual player

	Actor	System
1	Clicks on the icon for quickstart	
2		System starts a new game with one human controlled player and one NPC with medium difficulty.

Use Case: Start Custom Game

Summary: The player starts a custom game session.
Priority: Medium
Extends:
Includes: Start Game
Participants: The actual player.

1	Presses the big blue button named “Start Game”	
2		Closes the menu.

3		Launches a game session with the selected settings.

Use Case: Pause Game

Summary: The user pauses the game, bringing up a pause menu.
Priority: Low
Extends: Do Turn
Includes: Restart game, resume game, exit game.
Participants: The actual player

	Actor	System
1	Clicks on the menu button in the top right corner of the screen	
2		Brings up the pause menu with three different buttons: "Resume Game", "Restart Game" and "Main menu".

Use Case: Restart Game

Summary: The user presses the "Restart Game" button and the game restarts.
Priority: Low
Extends:
Includes:
Participants: The actual player

1	Clicks on the "Restart Game"-button	
2		The view changes to the game and a new game is started with the same options as the last game.

Use Case: Resume Game

Summary: The user presses the "Resume Game" button and the game resumes.
Priority: Low

Extends:
Includes:
Participants: The actual player

1	Clicks on the “Resume Game”-button	
2		The view changes to the game and the game resumes.

Use Case: End Game

Summary: The user presses the “Main menu” button and the game ends and goes to the start menu.
Priority: Low
Extends:
Includes:
Participants: The actual player

1	Clicks on the “Main menu”-button	
2		The game ends and the view changes to the start menu.

Use Case: Select no. of players.

Summary: The players selects how many players there should be in the game, between two and four.
Priority: Medium
Extends:
Includes:
Participants: The actual player.

1	Presses a number on the switch marked “2 3 4”	
2		Highlights the selected number.
3		Sets the number of players in game to the number selected.

Use Case: Switch Weapons

Summary: The player changes the actual weapon being used to fire attacks.

Priority: Medium
Extends: Do Turn
Includes:
Participants: The actual player

	Actor	System
1	Clicks on the arrows for switching weapon.	
2		Shows the name of the next or previous weapon.
3		Selects the weapon shown.

Use Case: Set NPC/player

Summary: The player chooses if the characters should be controlled by the A.I. or a human.
Priority: Low/Medium.
Extends:
Includes: Set NPC Difficulty
Participants: The actual player.

1.1	Presses NPC for a certain player.	
1.2	Presses Player for a certain player.	
2.1		Sets the player to NPC. To Select Difficulty.
2.2		Sets the player to player.

Use Case: Chose Map

Summary: The players picks between the different maps the game has to offer.
Priority: Low/Medium.
Extends:
Includes:
Participants: The actual player.

1	Presses one of the arrows on the top of the screen	
2		Shows the name of the next or

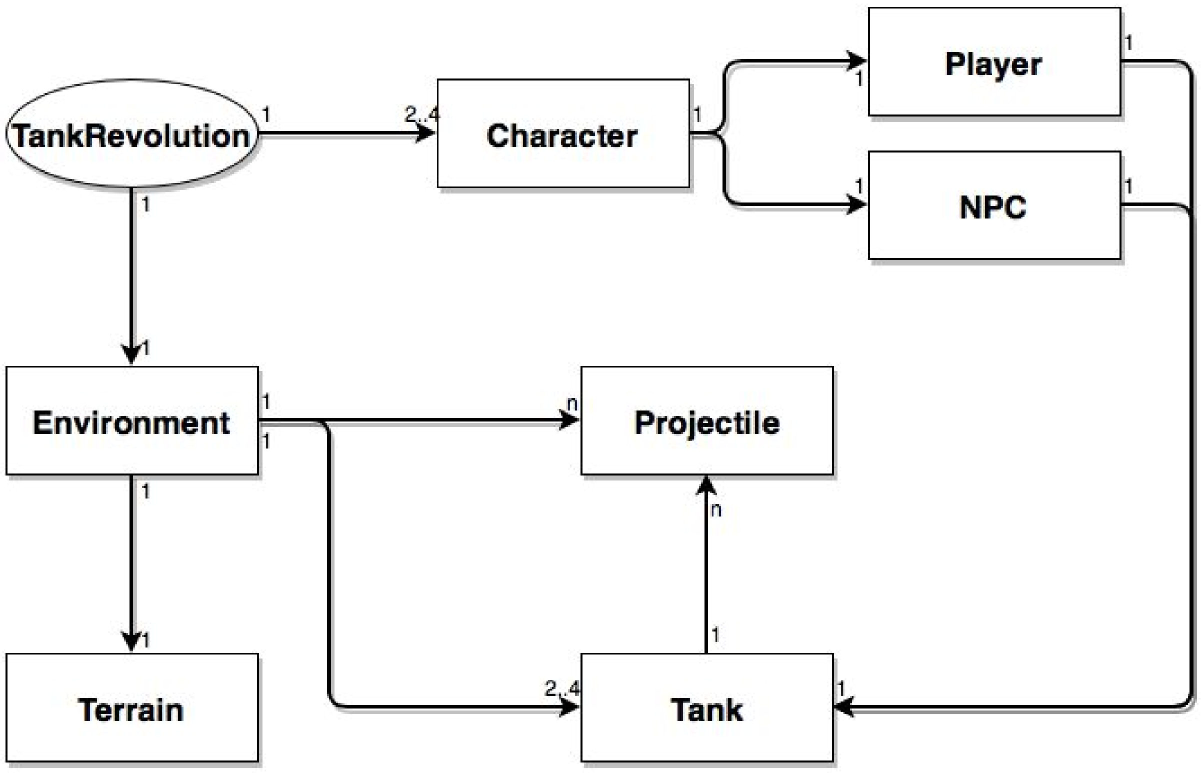
		previous map. and sets it as the selected map.
3		Sets the map shown as the selected one for the game.

Use Case: Set NPC Difficulty

Summary: Sets the difficulty of the NPCs if there are any. Expert difficulty will pretty much never miss.
Priority: Low
Extends:
Includes:
Participants: The actual player.

1	Presses one of the four buttons marked “Easy”, “Medium”, “Hard” or “Expert”.	
2		Gives the NPC the selected difficulty.

4 Domain model:



A simplified UML class diagram.

4.1 Class responsibilities:

TankRevolution: The Main Game Class. Handles starting the game and all its internal logic. Holds one and only one Environment and two to four Characters
.
Character: An abstract player of the game, can be controlled by either a human or the computer. Each of these holds one and only one Tank.

Player: A player controlled by a human being.

NPC: “Non-player character”. A player controlled by the computer.

Tank: Represents the tanks on the playing field. Has a position, speed, health, a type of weapon and creates the Projectiles.

Projectile: Represents the projectiles flying through the air. Has a position, damage and blast radius.

Environment: Represents the in-game world. Handles the terrain, the tanks and the projectiles.

Terrain: One and only one is created each game. Can be destroyed by projectiles hitting it.