Bilkent University

**CS319 Object-Oriented Software Engineering Project**

*Siege*

Analysis Report

Project Group 26: İrem Yüksel, Gökhan Şimşek, Emir Acımış, Furkan Küçükbay

Course Instructor: Uğur Doğrusöz

October 15, 2016

**Table of Contents**

[**1. Introduction**](#_40646dcp6vz9)[**3**](#_40646dcp6vz9)

[**2. Proposed System**](#_f0u7urfs0v2m)[**3**](#_f0u7urfs0v2m)

[2.1. Overview](#_h98qil1k74hb) [3](#_h98qil1k74hb)

[2.1.1. Gameplay and Control](#_rq71xjog8vh) [3](#_rq71xjog8vh)

[2.1.2. Obstacles and Objects](#_4q84wwcb382f) [4](#_4q84wwcb382f)

[2.1.2.1. Brick Wall](#_vflutj2uuaql) [4](#_vflutj2uuaql)

[2.1.2.2 Iron Wall](#_2al2lzp9q54o) [4](#_2al2lzp9q54o)

[2.1.2.3. Steel Wall](#_jd4p8f3huf0s) [4](#_jd4p8f3huf0s)

[2.1.3. Characters](#_i3bom0pljyju) [4](#_i3bom0pljyju)

[2.1.3.1. The Player Tank](#_oagokbgpx6qk) [5](#_oagokbgpx6qk)

[2.1.3.2. The Basic Enemy Tank](#_dozlyvi3ip96) [5](#_dozlyvi3ip96)

[2.1.3.3. The Crazed Tank](#_4u16iskwhad) [5](#_4u16iskwhad)

[2.1.3.4. The Panzer](#_r1jbmktmqtd2) [5](#_r1jbmktmqtd2)

[2.1.4. Power-Ups](#_hxsbr9wouhg0) [5](#_hxsbr9wouhg0)

[2.1.4.1. Shield](#_flua3tbb5bbv) [5](#_flua3tbb5bbv)

[2.1.4.2. Double Shots](#_2ssf91yty54k) [6](#_2ssf91yty54k)

[2.1.4.3. Extra Life](#_7vx9rwnrotjp) 6

[2.1.4.4. Ultimate Protection](#_wkryqmce6yse) [6](#_wkryqmce6yse)

[2.1.5. Scoring](#_dl5nh1hgt460) [6](#_dl5nh1hgt460)

[**3. Functional Requirements**](#_3dkw2fsbp564)[**7**](#_3dkw2fsbp564)

[3.1. Play Game](#_pkoheusim4bk) [7](#_pkoheusim4bk)

[3.2. Help](#_ojdxoc753zzq) [7](#_ojdxoc753zzq)

[3.3. Pausing the Game](#_m9h824v9ruh9) [7](#_m9h824v9ruh9)

[3.4. Continue the Game](#_tktc7j5o7ig1) [7](#_tktc7j5o7ig1)

[3.5. Settings](#_3b4oo15vv4v0) [7](#_3b4oo15vv4v0)

[3.5.1 Level Selection](#_qpieefrolg9x) [7](#_qpieefrolg9x)

[3.6. High Score](#_lym1h2xrdl5e) [8](#_lym1h2xrdl5e)

[**4. Non-Functional Requirements**](#_iqr3jtcaqv1o)[**8**](#_iqr3jtcaqv1o)

[4.1. Performance](#_8zi7dzxajb6c) [8](#_8zi7dzxajb6c)

[4.2. Usability](#_73sgrzgjbdge) 8

[4.3. Extendability](#_uwoetby07qr) [8](#_uwoetby07qr)

[**5. Constraints**](#_dt95bgnmwxjq)8

[**6. System Model**](#_qhau3m383uop)9

[6.1. Use-Case Model](#_kuej8jxu4bb4) 9

# 1. Introduction

The game “Siege” that is proposed is a revival of the classical game Tank 1990, with additions. The goal of the game is protecting a ‘sacred object’ from the attacks of the incoming tanks, by shooting and blowing them up.

The game consists of 5 levels, each one with a different layout - a different map - with increasing difficulty. To pass a level, the player must destroy every tank that is attacking, which will be a fixed number for every level and will be available for the user to see. As each level is passed, the number of assaulting tanks will increase, new types of more durable and more dangerous tanks will be introduced, and different types of walls will constitute the map.

The player will have 3 lives. In case of losing all their lives, the game will end. There is also another way of losing the game, thus ending it, namely the destruction of the ‘sacred object’. While this object can be destroyed by the enemy tanks, a misfire can also cause its doom.

In the following sections, the game above will be described in detail.

# 2. Proposed System

## 2.1. Overview

### 2.1.1. Gameplay and Control

The player controls the tank by using WASD keys, W meaning forward (or up in the case of a 2D perspective), A means left, S means right, and D means down. The player can fire using the SPACE key. The ESC key causes the game to pause.

### 2.1.2. Obstacles and Objects

The obstacles in the game are walls. While some of these walls can be destroyed, either with a single shot or requiring 3 shots, some of them are indestructible. Once any wall is destroyed, the path will be cleared and the tanks will be able to pass through that tile. The three types of walls are as follows:

#### 2.1.2.1. Brick Wall

The most basic type of wall. It can be destroyed with a single shot, either by enemy tanks or by the player.

#### 2.1.2.2 Iron Wall

This type of wall can be destroyed only after three shots. Its appearance will change after each shot, making it seem weaker.

#### 2.1.2.3. Steel Wall

This wall cannot be destroyed, effectively acting as a shield from the incoming attacks, if stayed behind.

The only object that the game will have that differs from the obstacles but still constitutes the layout is the ‘sacred object’. Protection of this object is the main goal of the game. This object can be destroyed with a single shot, by either the enemy tanks or the player’s tank. It is protected by surrounding brick walls, which cover its every side.

### 2.1.3. Characters

The characters that are proposed are the player, i.e. the controlled tank, and the enemy tanks. Both types of these characters can destroy the destructible obstacles in the game, and the ‘sacred object’. All these characters have the same movement speed, except for the crazed tank after it has been hit. The enemy tanks will change their directions every time they encounter an obstacle, if they had not broken it before, or every 5 seconds if they do not encounter any obstacles. The types of tanks are described in detail below:

#### 2.1.3.1. The Player Tank

The tank that player will control will be the most basic tank in the game. As mentioned earlier, it will have 3 lives. After losing a life, it will spawn every time in the same spot, near the ‘sacred object’ that it protects. The maximum number of lives the player can have is 3. It has a fixed fire rate, same as the basic enemy tank, which shall be 1 shot per second. An advantage that the player tank has over the enemy tanks is its ability of gathering power-ups, which will be discussed in the upcoming sections.

#### 2.1.3.2. The Basic Enemy Tank

This tank can be destroyed with a single shot. It also has a fixed fire rate, 1 shot per second.

#### 2.1.3.3. The Crazed Tank

This tank can be destroyed with two shots, but its most important property is that after being hit for the first time, it gets crazy and both its movement speed and its fire rate doubles, i.e. its fire rate will be 1 shot per half a second.

#### 2.1.3.4. The Panzer

This is the most durable tank of the game. It can be destroyed only by four shots, and it has the same movement speed and fire rate as the basic enemy tank.

### 2.1.4. Power-Ups

The player will never know when a power up will show up. The power ups will appear on random areas and disappear if not taken by the player. With these features, the game will be more enjoyable for the player.

#### 2.1.4.1. Shield

When taken by the player, the shield will protect the player’s tank from the attacks of the enemy tanks. The shield will be available for 10 seconds and then disappear.

#### 2.1.4.2. Double Shots

This power up will make the player’s shots to the walls or to the enemy tanks two times faster. This will help the player to kill the stronger enemy tanks easier. Double shots power up will also be available for 10 seconds and then the player will shoot with the initial fire rate.

#### 2.1.4.3. Extra Life

The player will increase their lives by 1 by taking this power up. So the player will be available to play longer. Also there is no time limit for this feature. If the player has 3 lives already, their lives will not increase, as the limit is 3.

#### 2.1.4.4. Ultimate Protection

Since the main goal of the player is to protect the ‘sacred object’, this feature will be very helpful. It changes the type of the walls around the ‘sacred object’ to the steel wall, making them unbreakable for 10 seconds, to protect it from the enemy tanks’ attacks.

### 2.1.5. Scoring

Player will get extra points after shooting an enemy tanks or breaking a wall. This way, the player will be more motivated to play, and beat their own high-score. The scoring will be done in the following way:

* Destroying a brick wall will give 10 points.
* Destroying an iron wall will give 25 points.
* Destroying a basic enemy tank will give 100 points.
* Destroying a crazed tank will give 150 points.
* Destroying a panzer will give 200 points.
* At the end of the game, remaining lives will be multiplied by 1000 and will be added to the end score.

# 3. Functional Requirements

## 3.1. Play Game

This is the most basic requirement of our project. After opening the game, the player will be able to play the game, direct the tank according to the controls specified above, pass levels if successful, and reach the end, either by winning or losing.

## 3.2. Help

In this feature, the player will be able to see how s/he can play the game, the functions of each key and what is the goal of the game. Also, the kinds of enemy tanks, obstacles and power-ups will be introduced and explained.

## 3.3. Pausing the Game

With this feature the player will have the opportunity to pause the game whenever s/he wants. Also, the status of the obstacles and the score will be saved. Being able to pause the game anytime will make the game more appealing since the player can be interrupted by something during the game and does not have to start over.

## 3.4. Continue the Game

This feature will allow the player to continue playing the game from where s/he left after the pause. Also, the advantage of this feature is that it will prevent players from losing any points or lives and starting over to the level.

## 3.5. Settings

This feature will help the player to make adjustments on the game such as turning on/off sounds, adjusting the volume and changing the level.

### 3.5.1 Level Selection

Tank Game will five different levels where each level has different difficulties. The player will be able to change the level in settings.

## 3.6. High Score

After finishing and being successful in each level, the player’s score will be saved if it is one of the 5 highest scores so far. The player will be able to see the highest 5 scores.

# 4. Non-Functional Requirements

## 4.1. Performance

Project needs to respond the player’s commands quickly since the player’s tank’s lives are dependent on it. The enemy tanks and the player’s tanks must move smoothly.

## 4.2. Usability

In the Tank Game, we will keep the interface as simple as possible for the users so that the possible players could learn it easily.

## 4.3. Extendability

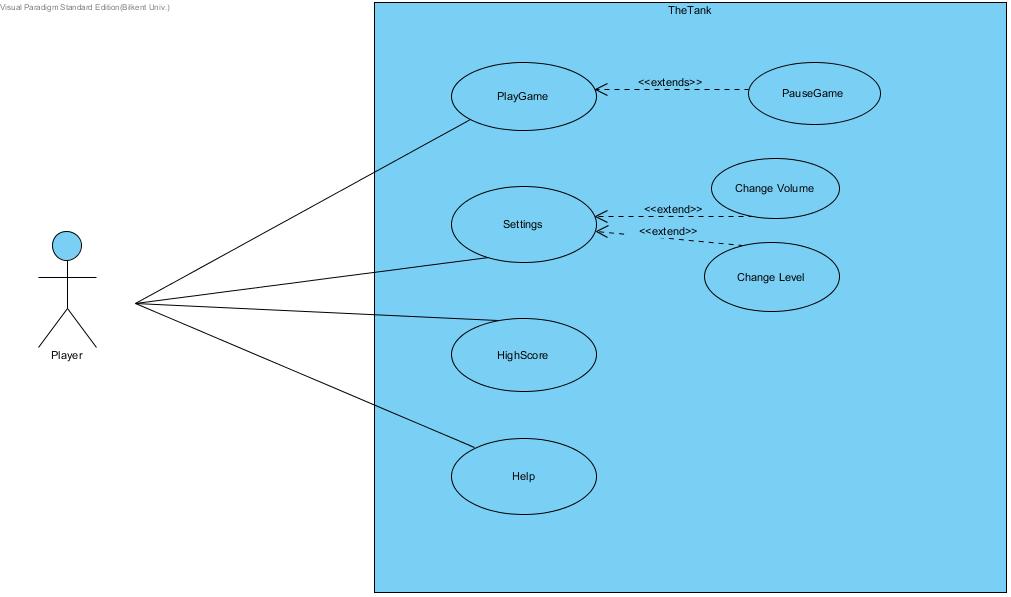
We will structure the program in a way that will enable us to implement any further improvements.

# 5. Constraints

Our program will be a desktop application. The game will be implemented in Java.

# 6. System Model

## 6.1. Use-Case Model



**UseCase #1**

**Use case name:** PlayGame

**Participating Actors:** Player

**Entry Condition:** Player opens the game.

**Main Flow of Case:**

* Player starts the game.
* The levels of the game start to flow while playing the game until the last level.
* The score of the player displayed while playing and when the levels are finished the game display the final score of the player.
* If the score is a high score, the score is written on the high score system and recorded.
* The game finishes.

**Exit Condition of Case:**

* Player loses all their lives.
* Player wants to exit.
* Player finishes the game by completing every level.

**Alternative Flow of Case:**

* Player can exit the game anytime.
* Player loses all their lives.

**UseCase #2**

**Use case name:** HighScore

**Participating Actors:** Player

**Main Flow of Case:**

* Player pushes the high score button.
* Player sees the high score list.
* Exit from the high score.

**Exit Condition of Case: None**

**Alternative Flow of Case: None**

**UseCase #3**

**Use case name:** Settings

**Participating Actors:** Player

**Main Flow of Case:**

* Player push to the settings button.
* Player changes the game settings that he/she wants OR changes the level of the game
* Exit from the settings menu.

**Exit Condition of Case:-**

**Alternative Flow of Case:-**

**UseCase #4**

**Use case name:** Help

**Participating Actors:** Player

**Main Flow of Case:**

* Player pushes the help menu button.
* Player sees the controls, the in-game buttons, the descriptions of different types of enemies and obstacles, and information about the general gameplay.
* Exit from help.

**Exit Condition of Case: None**

**Alternative Flow of Case: None**

**UseCase #5**

**Use case name:** PauseGame

**Participating Actors:** Player

**Entry Condition:** Player can access to the pause menu while playing the game

**Main Flow of Case:**

* Player starts the game and push the pause button
* Player pause the game encounter with the pause menu
* Changing the settings of the game OR change the level of the game
* Player continues playing the game

**Exit Condition of Case:**

* Player wants to continue playing.
* Player wants to exit the game.

**Alternative Flow of Case:**

* Player exits the game directly if player doesn’t want to play
* Player wants to play a different level

**UseCase #6**

**Use case name:** Change Volume

**Participating Actors:** Player

**Main Flow of Case:**

* Player pushes the settings button.
* After reaching settings sees the volume icon and change the volume.
* Exit from the settings.

**Exit Condition of Case: None**

**Alternative Flow of Case: None**

**UseCase #7**

**Use case name:** Change Level

**Participating Actors:** Player

**Main Flow of Case:**

* Player push to the settings button.
* After reach the settings the user see the Change Level button and pushes it.
* User decide which level they want to play
* User plays the game
* Finishes the level and if the level is not last level keep going on the game, if not the game over massage occur.

**Exit Condition of Case: None**

**Alternative Flow of Case: None**