**CS673S16 Software Engineering** 

**Team 1 - TankInATank**

**Software Design Document**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Baoxiang Yang | Team Leader | *Baoxiang YANG* | 11/17/2017 |
| Haotian Wu | Requirement Leader | *Haotian Wu* | 11/17/2017 |
| Shuhan Liu | QA Leader | *Shuhan Liu* | 11/17/2017 |
| Lei Yang | Implementation Leader | *Lei Yang* | 11/17/2017 |
| Qiwei Zheng | Environment and Integration Leader | *QZ* | 11/17/2017 |
| Linlan Chen | Configuration Leader | *Linlan Chen* | 11/17/2017 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
|  | **team1** | **2017/12/03** | Make it more complete |
|  |  |  |  |

[Introduction](#_gjdgxs)

[Software Architecture](#_30j0zll)

[Design Patterns](#_1fob9te)

[Key Algorithms](#_3znysh7)

[Classes and Methods](#_2et92p0)

[References](#_tyjcwt)

[Glossary](#_3dy6vkm)

# Introduction(Baoxiang Yang)

In this section, give an overview of this document, and also address the design goals of your software system.

This Software Design Document is a document which will be used to help for software development by providing the details of implementation process. In this document, we provide the narrative and graphical instruction of the software design of the whole project. We will introduce our software architecture, design patterns, key algorithms, classes, methods and other supporting requirement in details in this document.

1. Goal

The Main goal of this software Design Document is to clarify details enough for software development to proceed with an understanding of what should be built and how it is expected to be built of our project by provide a description of the whole design.

1. Scope

This software design document is focused on the base level system and critical parts of the system. All team members in team1 will have full access to make changes, as he/she deems necessary.

1. Glossary
   1. DB: Database
   2. SDD: Software Design Document
   3. TBD: To be decided
   4. Tank Game: The main program of our project.
2. References
3. Overview
   1. Introduction
   2. Software Architecture
   3. Design Patterns
   4. Key Algorithms
   5. Classes and Methods
   6. References

# Software Architecture (Lei Yang) In this section, you will describe the decomposition of your software system, which include each component (which may be in terms of package or folder) and the relationship between components. You shall have a diagram to show the whole architecture, and class diagram for each component. The interface of each component and dependency between components should also be described. If any framework is used, it shall be defined here too. Database design should also be described if used. High level speak, we use the traditional MVC architecture to rule the software.

# 

# We have 5 models to handle all the calculation.

# Damage processor is the model calculate damage between tanks

# Map updater is the model update the UI change. E.g. tank movement

# Camera Processor is the model update the camera angle when tank is moving

# Game Manager controls the main menu

# Shell Manager controls shells shot by tanks

# View controller takes the responsibility to transfer data between models and views

# We have 3 views to handle the UI part.

* Map view is the map UI user sees.
* Tank View is the tank user controls, theoretically it’s inherited from map view. In practice, to make the system has better scalability, we separate them.
* Menu View is directed by Game Manager, which shows game start and end.

# Design Patterns (Linlan, Haotian)

In this section, you shall describe any design patterns used in your software system.

There are three main design patterns in our project:

**Factory:**

By using factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface. In our project, we create shells by factory, each time the tank fires, the factory creates a new shell. And we also create two tanks through the factory design pattern at the beginning of each round.

**Observer:**

If the health point of a tank changes, the health point display slider should change automatically, thus, we apply observer pattern here. The display slider is an observer of the tank’s health point. Once the health point changes, the slider will be notified and change accordingly. We also use observer pattern on game manager. Game manager is an observer of tank status, player status, player input and game status. When tank is destroyed, game manager will be notified, change the status of the game to stop, add scores, show the score of players and start a new round.

**Singleton:**

By using singleton pattern, we create only one “Game Manager” object in the game.

# Key Algorithms (Haotian, Shuhan)

In this section, you shall describe any key algorithms used in your software system, either in terms of pseudocode or flowchart.

Camera:

For the camera part, we wrote an algorithm that the camera will get the positions of all characters(tanks) and calculate an average position which camera will focus on. And camera will also auto zoom in or out depends on the distance between two tanks.

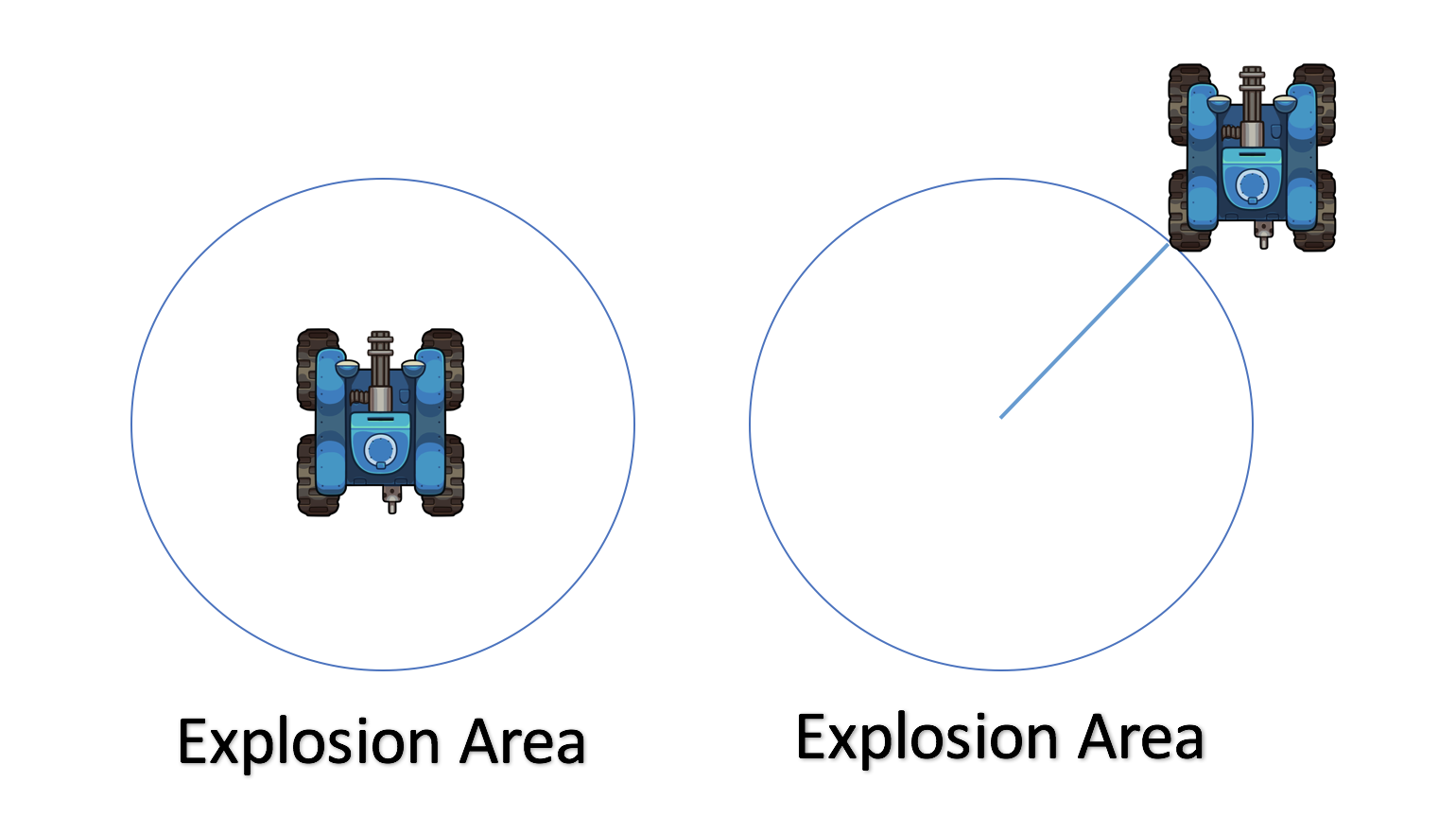
Shooting:

For the shooting part, out shell is affected by gravity, so the time it hits the ground (assume it doesn’t hit other objects) remains the same. However, we still can change the distance of the shell by using different force when firing. The player can adjust the shooting force by changing the time they hold the fire button.



Explosion:

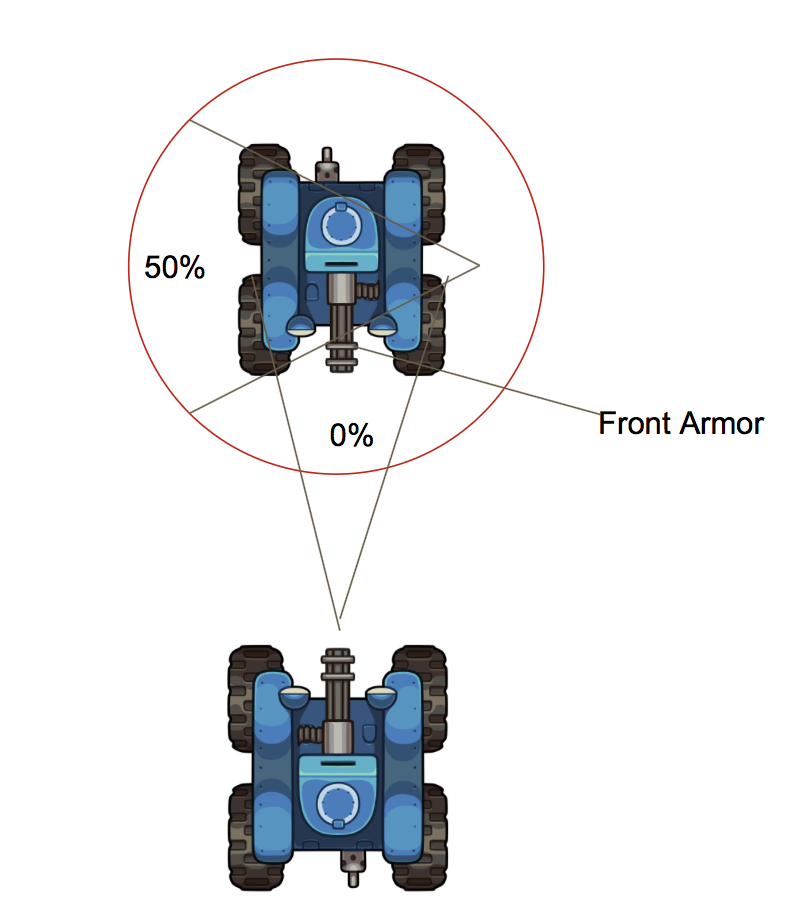
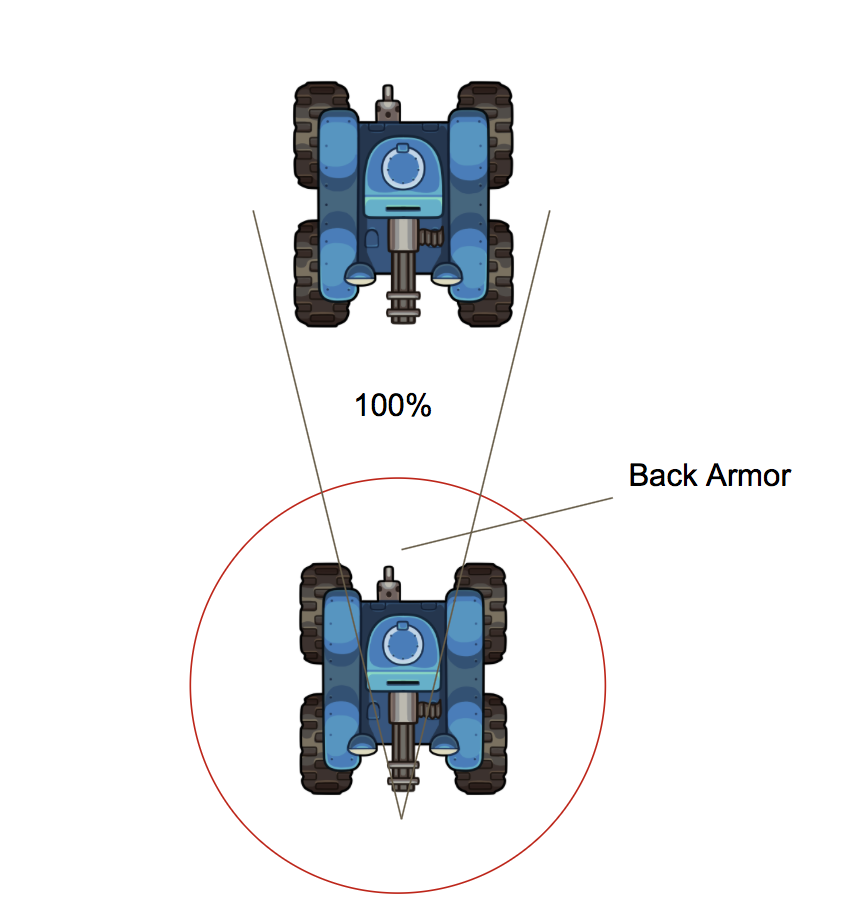
The tank will gets hit not only when the shell hits the tank directly, moreover, it will be affected by explosions.



When the shell explodes, it will land effect in a small area which we call it explosion area. After we compute the distance between the center of the explosion area to the edge of the tank (based on its box collider), we will compare it to the diameter of the explosion area to decide whether it will get damage or not.

Health：

For the damage, when a shell hits different part of the tank or gets the tank gets affected by other explosion, the damage varies.

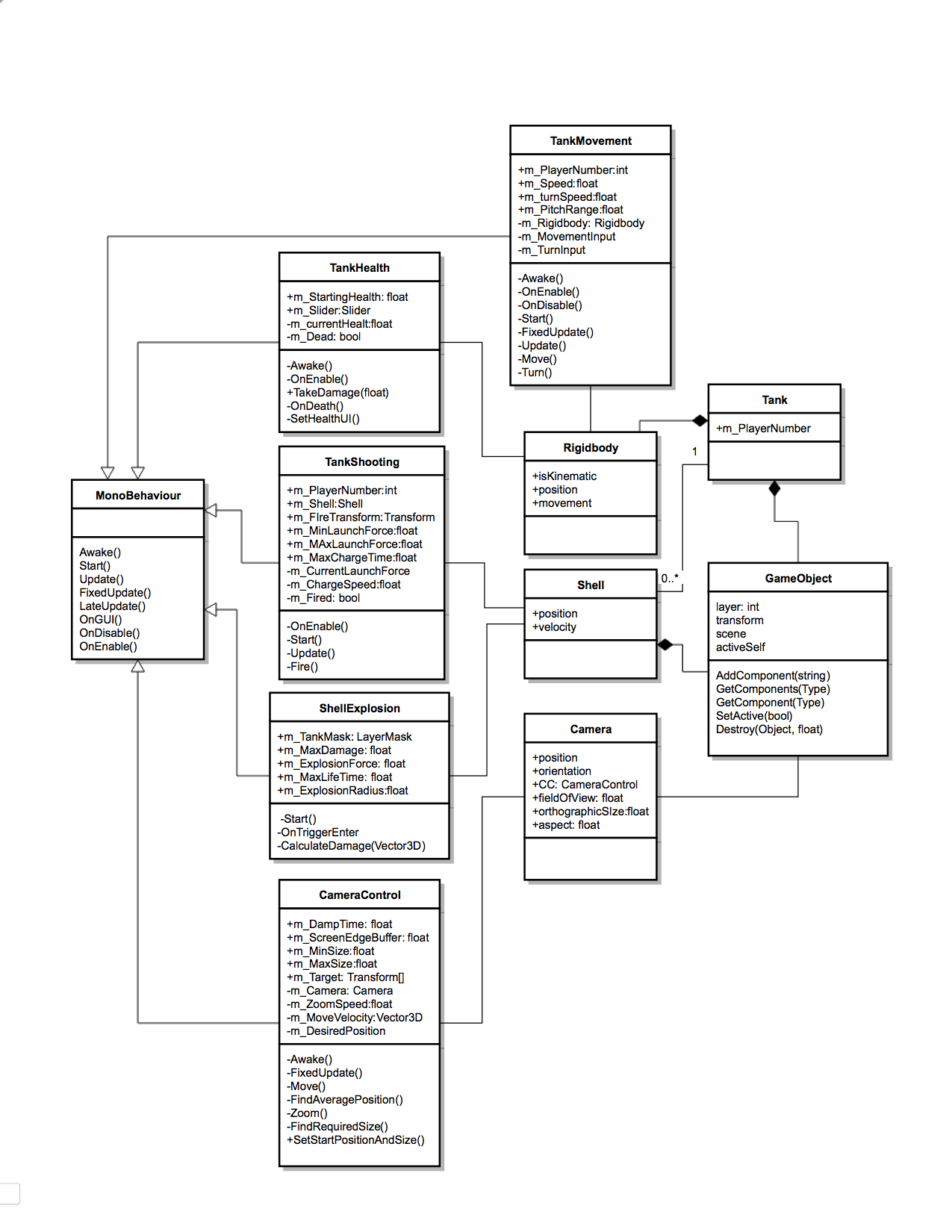
According to our design, the front armor is strong enough to defend the tank from the explosion, while the side and back are more fragile, which will take 50% and 100% of the damage caused by the shell explosion.

# Classes and Methods (Qiwei and Linlan)

This part can be a reference to automatic generated document for all classes and methods.

Here we have the following classes: GameObject, Shell, Tank, Camera, TankMovement, TankHealth, TankShooting, ShellExplosion, CameraController, MonoBehavior, Rigidbody.

GameObejct and MonoBehavior are base classes provided in Unity, GameObject is the base class for objects and MonoBehavior is the base class for scripts. Since Tank and Shell are all composited by some GameObject, there exists “composition” relation between them. And TankHealth, TankShooting, ShellExplosion, TankMovement and CameraController are all scripts, thus they inherited the base class MonoBehavior. We call the body part of the tank “Rigidbody” and TankHealth and TankMovement both works on Rigidbody. TankShooting and ShellExplosion works on Shell.



# References

# Glossary