# Game Project Documentation

# (Tank Game and Katch&Pop)

## CSC-413 Spring 2018

## Professor Anthony Souza

**May 24th, 2018**

**Aye Sandy Win & Shan Kwan Cho**

**Table of Contents**

**Tank Game and Katch&Pop Game**
**Software Documentation**

Contributor: Aye Sandy Win and Shan Kwan Cho
GitHub Repository:
**Tank Game**:
https://github.com/csc413-03-sp18/csc413-03-tankgame-Team16-ShanKwanCho.git
**Katch&Pop (Second Game):**
https://github.com/csc413-03-sp18/csc413-secondgame-Team16-AyeSandyWinShanKwanCho.git

## 1. Introduction:

There are two projects to implement which both are 2D game projects. The first game project is TankGame Project and the second one is Katch&Pop. For the TankGame Project, there are two players game and the objective of the game is to attack and destroy the opponents of each players and to get the higher scores. For the Katch&Pop Game which is second game project, the objective of the game is to hit the Octopus(Big Leg) to get the highest score.

### 1.1 Overview:

The main idea of the both game projects are introducing and implementing one of the most essential strategies in software development industry which is understanding the knowledge of the Object Orient Programming (OOP) such as hierarchy of inheritance, encapsulation and polymorphism. The structures of both game project are similar which can be reused most of the classes ideally using the strategies of the inheritance, encapsulation and polymorphism.

Note: This project is running on Mac OS High Serra and IDE is NetBeans
More info IDE of NetBeans: https://netbeans.org

### 1.2 Project using Programming Language:

Both Game projects are written in Java language.

### 1.2 User Interface

Both games allow the user to play the two-dimensional expressions of games and featuring of attacking, shooting and getting up the total scores by pressing control keypads respectively.

## 2. Resources

There are two completed project which include Airplane Shooter Game. Resources such as supporting files (ie. Images and audio files for both games) can be found via iLearn. As well, resources from professor's lectures, notes, implementations, examples, discussions, collaboration inside and outside of the class such as Slack Channel can be useful respectively.

Reference : Instruction guide for the TermProject2018

## 3.Scope of Work

Tank Game and Katch&Pop Game Projects can be allowed to reuse the game engine of the Plane Game and resources from iLearn which implement and program for both game projects.

## 4. Assumptions

Development and implementation of program files were performed in Netbeans IDE 8.0.2 environment on Mac OS. Tank Game and Katch&Pop Game Projects use the game engine of Plane Game as a base and all the resources from iLearn.

## 5. Technical Guide

### 5.1 Project Components:

Technically, Tank Game project includes major of eighteen java files and a folder of resources file which includes all the files that stores the supporting files for the project.

1. MapLoader.java
2. Bullet.java
3. CollisionDetector.java
4. DestructableWall.java
5. Explosion.java
6. GameApplication.java
7. GameEvents.java
8. GameWorld.java
9. KeyControl.java
10. Location.java
11. MovableObject.java
12. Player.java
13. PowerUp.java
14. Tank.java
15. Thing.java
16. Unit.java
17. Wall.java
18. SoundPlayer.java

As well as, Katch&Pop Game project also includes major of eighteen java files and a folder of resources file which includes all the files that stores the supporting files for the project.
Note: Some of the classes are reused from the Tank Game Project.

1. BigLeg.java
2. Block.java
3. CollisionDetector.java
4. Katch.java
5. GameApplication.java

6. GameEvents.java
7. GameWorld.java
8. KeyControl.java
9. Location.java
10. MovableObject.java
11. MovingBigLeg.java
12. Player.java
13. Pop.java
14. Tank.java
15. Thing.java
16. Unit.java
17. Wall.java
18. SoundPlayer.java

## 5.2 Basic Structure (Hierarchy):

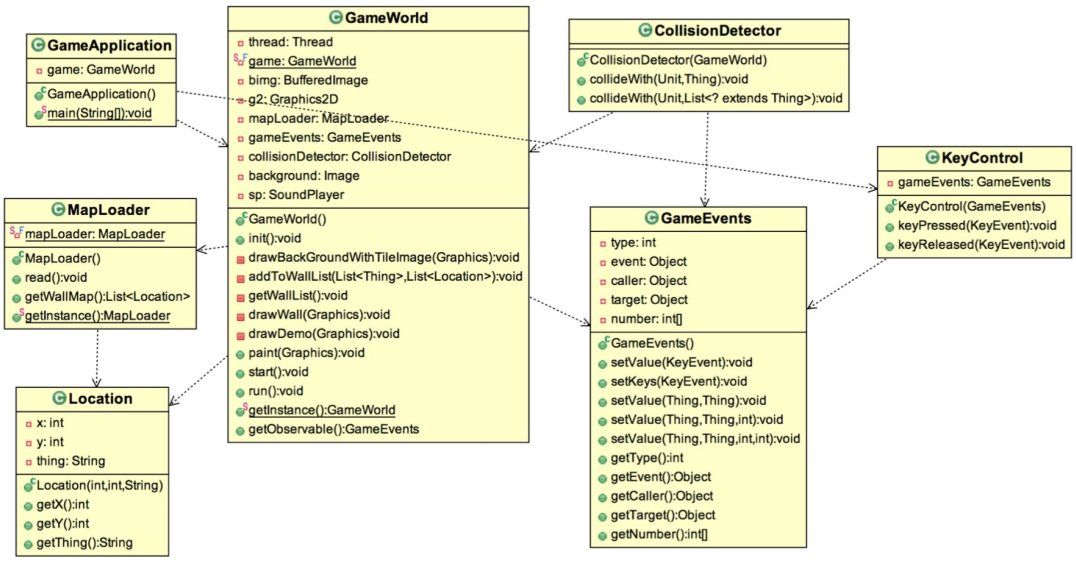
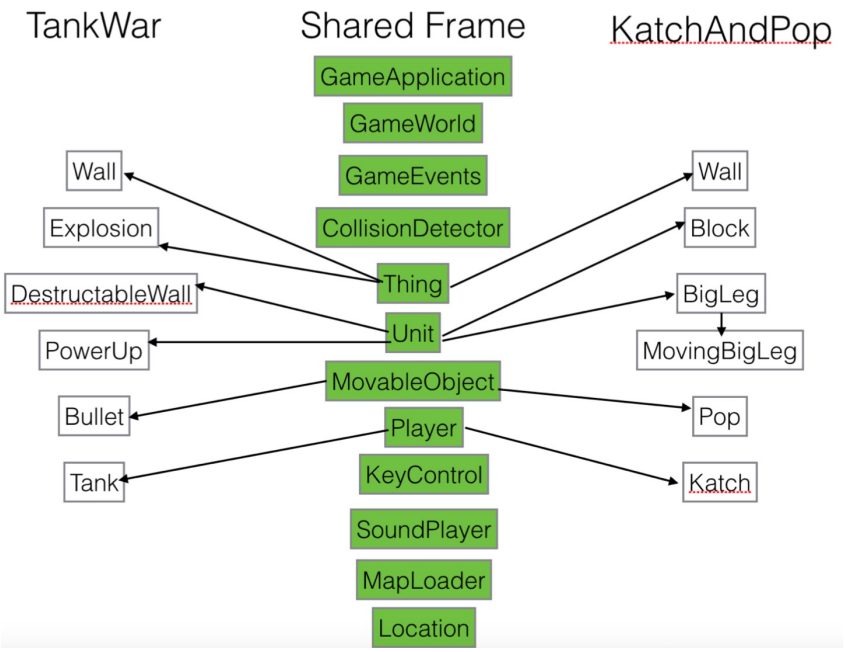



Fig 5.2.1 Tank Game Class Hierarchy

**The Structure of the two game:**




Note: Some classes are reusing for both game projects

Fig 5.2.2 Katch&Pop Game Class Hierarchy

**5.3 Game Project Development**

In the middle is the main part of the both games. The shared part could be divided into 3 categories. The first four (GameApplication class, GameWorld class, GameEvents class and CollisionDetector class) are the main engine for the both games project. Thing, Unit, MovableObject and Player are the inherited abstract classes used by the both game project. KeyControl, SoundPlayer, MapLoader and Location are the helper classes for both games.

Most functions in these classes are shareable and reusable for both game projects. These classes perform as the game engine. Some of base codes provided from PlaneGame (iLearn resources).

| | |
|---|---|
| GameApplication.java: | Game Application class is the main classes also public class for each which is JFrame class and initialized the starting parameters of the game,<br>Starting game and running the game. |
| GameEvents.java: | GameEvents class, public class extends from observable class and watching by all Thing objects which can be designed to update when it needs. |

| | |
|---|---|
| CollisionDector.java: | CollictionDetector class is the one to detect if there is a collision. If there is a collision, it will pass all the information to the GameEvents to enable GameEvents to notify all of the observers respectively. |
| GameWorld.java: | GameWorld class is key part of both games which includes methology of drawing both of the game of the background and existing objects of thing and implements Runnable interfaces. There are two essential methods in GameWorld: -drawBackGroundWithTileImage( ) to draw the backgrounds and -addToThingsList( ) includes Loop through an ArrayList of ArrayLists Things, to draw all the things appropriately in the games. |
| Thing,java: | Thing class is the base abstract class which represents all the objects to Be drawn in game projects. All things inside this class are capable of being collision however all things are not able to cause collisions. |
| Unit.java: | Unit class extends Thing.java. All units are capable of causing Collisions when a thing touches the Unit.It needs to update after a collision, such as destructible wall in the Tank Game Project or block in the Katch&Pop Game Project. |
| KeyControl.java: | KeyControl is the public class extends KeyAdapter performing as a keystroke reader, which supports determining when a key is pressed and when it is released. |
| Wall.java: | Wall class extends Thing class and draw the rectangle of background with implementing Image.IO.read to get the pictures from resources folder of game project. |
| MoveableObject.java: | MoveableObject class extends unit Class. It could move, such as Pop extends it. |
| Player.java: | Player class extends MoveableObject class which includes additional data fields. |

**5.4 Additional Classes**

In TankGame project, Tank.java includes the properties of tank. The contributors inpu the methods of draw, update, stay, move, fire and damaged which need for the entire functions of the tank game. As well Bullet.java which extends Moveableobject, set the speed of the tanks and move around, get the rectangle and update the position of myTank and Enemy Tank. For Explosion.java, extending Thing class, using the strategies of try-catch and be able restart which means re-draw the new rectangles for tank after explosion and setting time back.

For Katch&Pop game project, BigLeg.java extends Unit class, get/set the method of draw, get the rectangle and update those along with using try-catch methology. MovingLeg.java extends BigLeg and draw, update and the rectangle. Katch.java extends Player and be able to get/set the width and height of the rectangle, draw and update those. Pop.java extends MoveableObject class which get/set from the resources and getRec, update also the sound form the resources folder.

## 6. User Guide

### 6.1: Tank Game

Game Description: This game is be able to play with the opponents (2 players total) each other. This tank game include 2 players which can fight with bullet one to another.
One player (Blue Tank) can control the keyboard with the keys :
back arrow = backward movement
forward arrow = Forward movement
up arrow = upward movement
down arrow = downward movement
space = shoot the bullet
-
The opponent (Red Tank) can control the keyboard with the keys:
A = backward movement
D = Forward movement
W = upward movement
S = downward movement
enter = shoot the bullet

### 6.2 Katch&Pop Game

Game Description: The Goal of the game is to hit the octopus(BigLeg) to get the highest score.
Keys Control : Use Arrow Keys to move left and right.

## 7.Implementation discussion / Outcome after struggles

The contributors used Collection Detector to support separate different classes. Observer Pattern could reduce coupling to minimize the effect of change. For instance, the contributor may modify the Katch class or even delete that class in Katch&Pop Game Project in the future however, Pop class wouldn't be changed anymore.

CollisionDetector

```
public void collideWith(Unit caller, List<? extends Thing> things) {

}

public void collideWith(Unit caller, Thing target) {
   if (caller.getRec().intersects(target.getRec())) {
       …
       gameEvents.setValue(caller, target, katch.getX(), …);
   }
}
```

Figure 7.1 CollisionDetector

Both contributors tried to figure out with using polymorphism, override and over load methods and finally came up with “use collideWith(Thing, thing)” instead of collideWithTank(). As well for override and overload, “collideWith(List<Thing> things)” and collideWith(Thing, thing)”.

Again, in CollisionDector.java, contributors got the error message of

```
Exception in thread "main" java.util.ConcurrentModificationException
```

the fact that using index method to modify the same collection in another place. After debugging, the contributor decided to use iterator in one place as shown below.

```
                }
            }
        }

        public void collideWith(Unit caller, List<? extends Thing> things) {

            for (int i = 0; i < things.size(); i++) { // important! don't use iterator loop, otherwise show up ConcurrentModificationException
                collideWith(caller, things.get(i));
            }
        }
    }
```

Figure 7.2 collideWith

After contributing both game projects, the contributors had end up with some weakness area of both game projects and had sum up with some improvements for the future. Some improvements are bounding box in other words bounding circles should get smaller to get more accurate collision in Tank Game Project, additional weapons should be clear after tank gets a new life, adding new levels, more weapons and challenges of both game, adding more features such as higer difficulties, challenges, to be able to let the player gets great experiences and game simulation while playing each game.

## 8. Screenshots of Tank Game Project And Katch&Pop Game Project

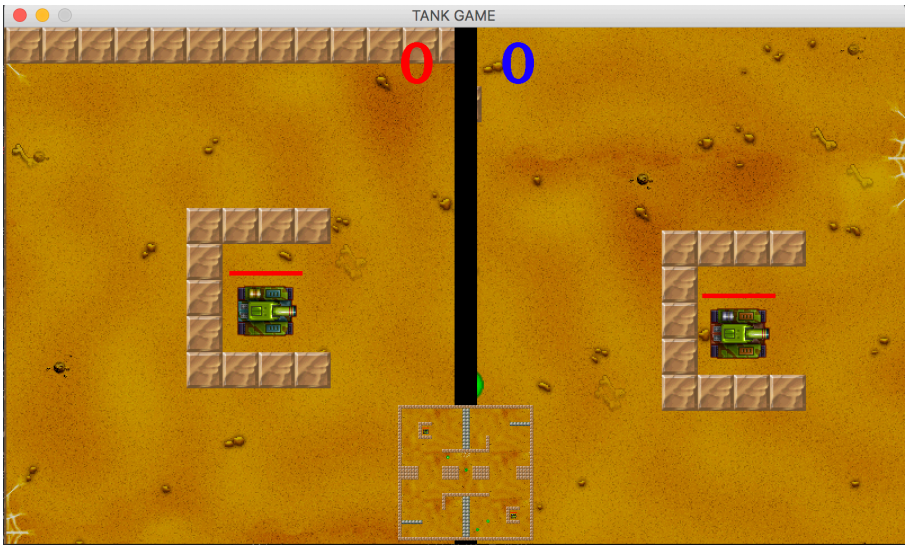



Figure 8.1 Screenshot of Tank Game

Figure 8.2 Katch&Pop Game Project

## 9. Conclusions

In conclusion, the idea of this project is understanding the essential strategies in software development (i.e. knowledge of Object Orient Programming (OOP), interfaces, hierarchy of inheritance, encapsulation and polymorphism. These projects are practicing those methology and reusability. Both contributors discussed and followed Plane Game design pattern most likely same java files. The contributor had two separate of Tank file before which are enemies and myTank. Those are very similar so that contributors decided to create a single Tank.java.
By what contributors collaborated and implemented, these information can be used for the second game. The thing that contributor made at first was separating all classes into the java file. It was waste to create different classes for walls, bricks and enemies for both games, reusing classes made much easier and save effort and time. For creating different unit, simply created an instance of the wall class with different image arrays, different scores and parameters.

## 10.Credit and Reference

Instruction: Instruction guide for the Term Project 2018
Slack: private channel: csc413-spring2018
Collaboration on Slack/ iLearn
IDE NetBeans: https://netbeans.org

## 11. Support, Contact and FAQ

If you have issue with this gam project, please let us know
by email: scho4@mail.sfsu.edu, awin@mail.sfsu.edu

## 12. License

This project is licensed under [Tank Game Project Version 1.0 and Katch&Pop Game Project Version 1.0]