Module Code: **CS2JA16**
Assignment Report Title: **Android Game**
Student Number: **28010336**
Date Completed: **04/05/2021**
Actual hours spent for the assignment: **30 hours**
Assignment evaluation (3 key points):
1) I have a better understanding on Android App Game Development
2) I have gained knowledge on how to use Firebase Realtime databases
3) I have learnt how to use Threading in my classes

TANKWARS 2D ANDROID GAME

Abstract

This report contains the results of developing an Android 2D tank wars gaming app that has three levels of play. The player tank scores points when it hits the enemy bullets and tanks. The score is displayed on the screen and a timer controls how long the game lasts.

Introduction and Showcase

The 2D TankWars Android app game that has been developed in Android Studio IDE using Java. The three levels are, Easy, Medium and Hard. The aim of the game is for the player tank to hit targets and enemy tanks where they score points which are updated on the top right corner of the screen. The player can also see the high score list for each level by clicking the 'High Score' text that is displayed on the main menu. The scores are stored in a Firebase database with the key node being the Levels, Easy(1). Medium(2) and Hard(3). Under each key node there are two child nodes, Username and Score. When a game ends in any one level, the current score is compared with the score in the database and if it is higher then it is overwritten. If it is lower, then the score in the database remains the same.

Easy Level

In this level, the player tank moves left/right/up/down at the bottom of the screen and shoots one bullet at a time. There is an enemy tank at the top of the screen which moves from side to side, and if the player bullet hits this tank the player scores a point. There is also a missile on the screen which is created dynamically and bounces off the sides and top of the screen, and off the three obstacles on the screen as well. If the missile hits the enemy tank, it will score a point on the total score. The aim of this game is to score as many points as possible in 30 seconds, and the Timer implemented will finish the game once the time is reached. There are also three static obstacles on the screen on which the bullet and missile rebound off, the points are scored if the player bullet hits the enemy tank at the top of the screen. If the player bullet hits an obstacle, the player bullet will be destroyed. If the enemy bullet hits an obstacle, the enemy bullet will be destroyed.

Medium Level

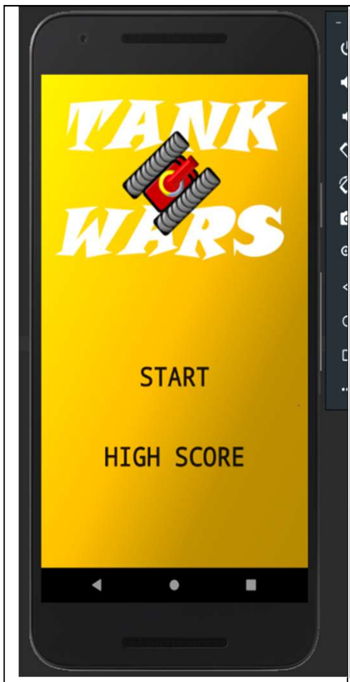
Two enemy tanks at the top of the screen, move from side to side on their half of the screen on the top line, at different speeds from one another. There are four obstacles in the arena.
Enemy tanks fire bullets, if the enemy bullet hits an obstacle, the bullet is destroyed and another one is spawned. If the enemy bullet collides with the player bullet, both bullets are destroyed, and a new enemy tank bullet is spawned. The player can spawn another bullet if he touches the area on or around his tank.
There is a missile which moves around the screen, the missile rebounds off the top, bottom and sides of the screen. If the missile hits either of the enemy tanks, it will rebound, but will also score a point for the player.
The player tank can move in all 4 directions in a limited area at the bottom of the screen. If the player touches the screen on or very near to the player tank, the player tank will shoot a bullet. The player bullet will rebound off the obstacles, the sides of the screen and the bottom, but if it gets to the top of the screen the player bullet will go out of bounds and the player is then able to fire another bullet. If the player bullet hits either of the enemy tanks, it will score a point for the player.

Hard Level

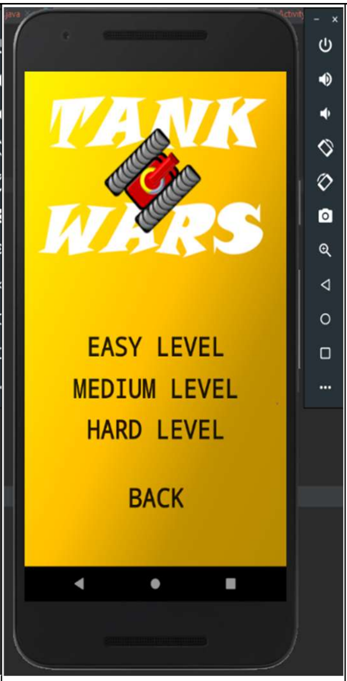
Three enemy tanks at the top of the screen, two on the top line and one underneath. All move from side to side at different speeds from one another. There are three obstacles in the arena.
Enemy tanks fire bullets, if the enemy bullet hits an obstacle, the bullet is destroyed and another one is spawned. If the enemy bullet collides with the player bullet, both bullets are destroyed, and a new enemy tank bullet is spawned. If the enemy bullet hits the player tank, the bullet is destroyed, and the player loses one point off his score if he has more than zero points.
There is a missile which moves around the screen, the missile rebounds off the top, bottom and sides of the screen. If the missile hits either of the enemy tanks, it will rebound, but will also score a point for the player.
The player tank can move in all 4 directions in a limited area at the bottom of the screen. If the player touches the screen on or very near to the player tank, the player tank will shoot a bullet. The player bullet will rebound off the obstacles, the sides of the screen and the bottom, but if it gets to the top of the screen the player bullet will go out of bounds and the player is then able to fire another bullet. If the player bullet hits either of the enemy tanks, it will score a point for the player.

Game Screens

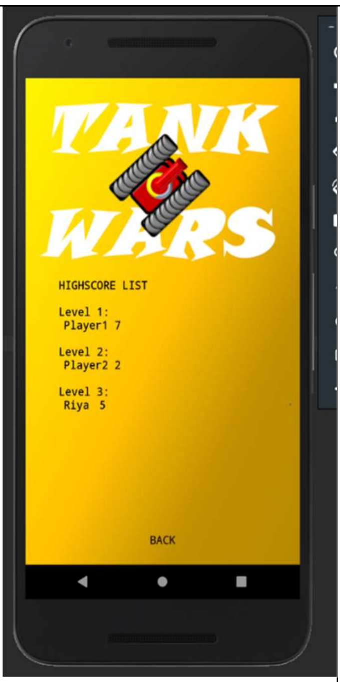
The game screens are as below.
The first screen shows the start of the game and on selecting start, the game menu screen is displayed. On selecting the High Score list the High Score screen is displayed which shows the highest scores saved in the database for each level. After selecting a game level the Ready screen is displayed. The next screens depend on what level was selected from the Menu.



Main Menu Screen



Game Menu Screen



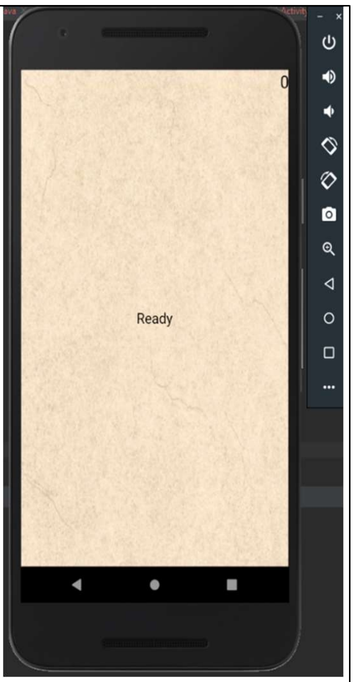
HIGHSCORE LIST

Level 1:
Player1 7

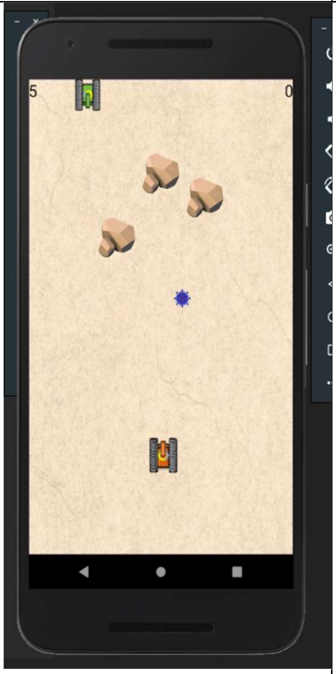
Level 2:
Player2 2

Level 3:
Riya 5

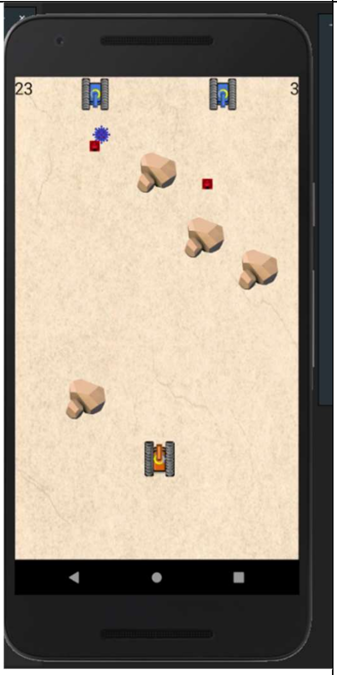
HighScore List Screen



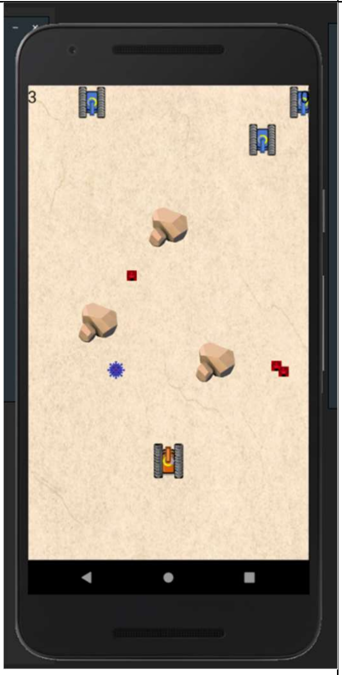
Ready to Play Screen



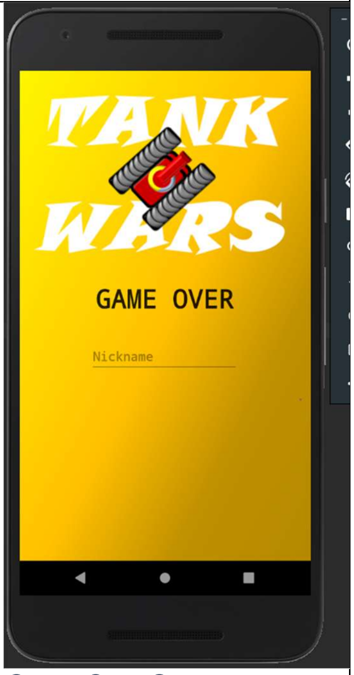
Easy Level Screen



Medium Level Screen



Hard Level Screen



Game Over Screen

OOP Design

The TankWars app uses OOP design principles, Abstraction, Encapsulation, Inheritance and implements interfaces.

Inheritance is an important concept in OOP and it's a mechanism that allows a class to inherit the properties of another class. I have implemented this in most of the classes that I have created. In the EasyGameThread class code snippet as shown below, I have inherited the properties of the Thread class. In Easy Game View class, extends keyword inherits the properties of the SurfaceView class which implements the SurfaceHolder.Callback and SensorEventListener methods.

```
public abstract class EasyGameThread extends Thread {
public class EasyGameView extends SurfaceView implements SurfaceHolder.Callback,
SensorEventListener {
```

Abstraction

The three levels of the game have abstract class which extend the Thread class. This class handles the start up of the games and has abstract methods like StartTimer and non abstract methods like setTimerView.

```
public abstract class HardGameThread extends Thread {
        …….
        abstract protected void startTimer();
        public void setTimerView(TextView oTimerView) { this.oTimerView = oTimerView; }
```

Use of API class/methods

CountDownTimer – Counts how long the game will last. In this game it will run for 30 seconds

```
protected void startTimer() {
    iCount = 0;
    //Start countdown timer (milliseconds)
    oCountDownTimer = new CountDownTimer(30000, 1000){
```

Android Database API used to connect to the Firebase Realtime database. The HireScoreActivity class implements the FireBase database where the scores are stored. When data is saved to the database, an instance of the FireBase database is initiated and a reference to the root node made using getReference(). The child nodes are then accessed by getting the values of those nodes.

```
database = FirebaseDatabase.getInstance();
myRef = database.getReference("UserID");
```

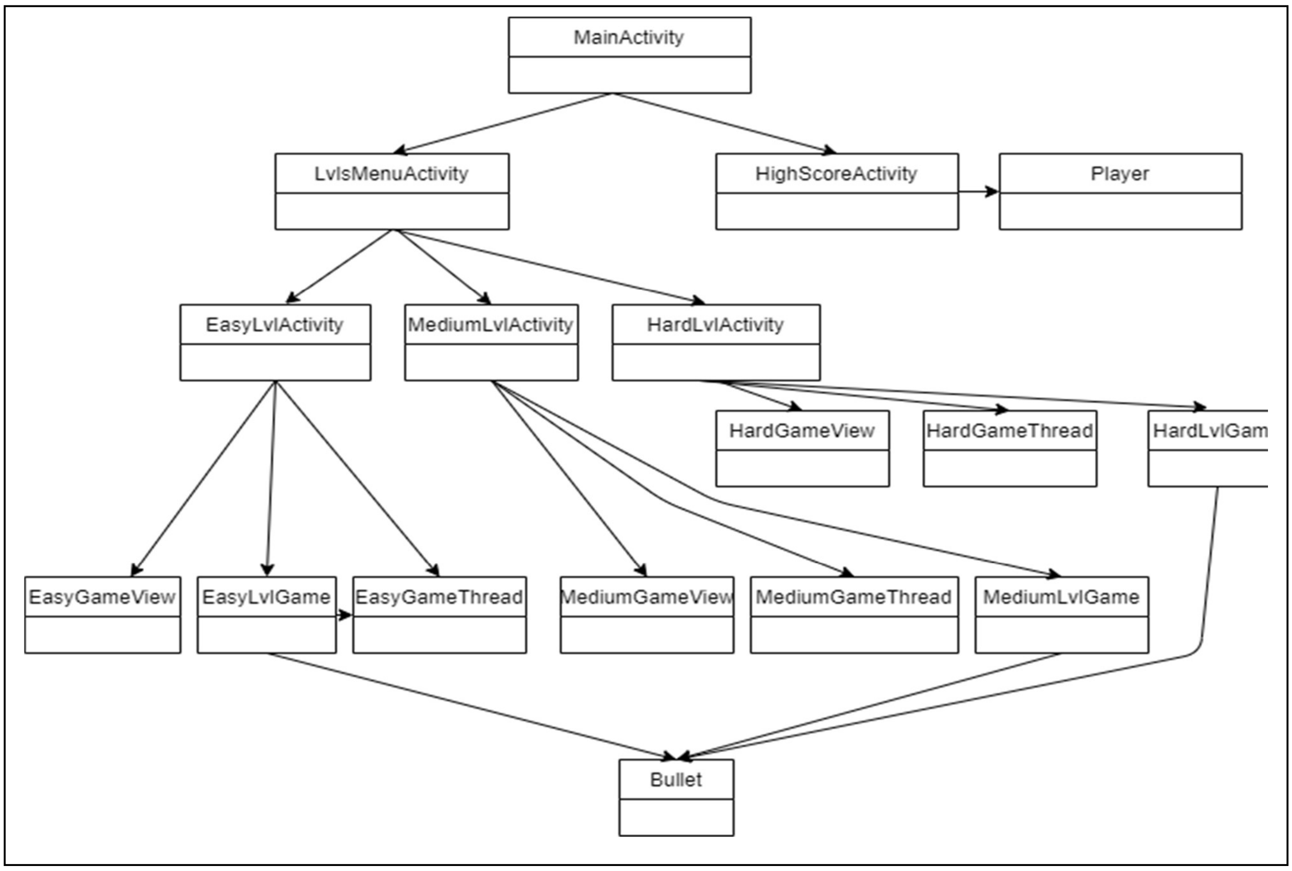
To read data from the database the dataSnapshot object is used. The data is stored in an arraylist and then displayed on the screen when the user wants to view the high scores.

An **ArrayList** has been used to hold the properties of the bullet and tanks classes which controls the size, speed, dimensions and direction. An example can be found in the HardLvlGame class:

```
private ArrayList<clHardETank> arETank;
//Initialise enemy array list
arETank = new ArrayList<>();
arETank.add(new clHardETank(oTankBmp, fTankX, fTankY, fTankSpeedX, bBulletFired,
oTankBulletBmp, fBulletX, fBulletY, fBulletSpeed, fBulletSpeedX, fBulletSpeedY));
```

Class Diagram

Below is a diagram showing the classes used in the project. I have shown a high level view to show how the classes interact with each other.



Class Diaggram

Memory Usage and Speed Improvements

The following solutions would improve memory usage and improve speed of the game:
I have reduced the use of external libraries as the porting and optimising uses memory so is not recommended for Android apps
Use of threads to run background processes and releasing the memory usage once finished

- Reduce the high score list to 3
    - Reduce memory usage
    - Hence increases speed
- Use double or int instead of float
    - Floating-point is 2 times more slower than int
    - Double takes up much more space than float
    - Use of final keyword to directly access values stored in classes
    - Using enhanced loop syntax eg
      ```
      ArrayList value = new ArrayList();
      for (DataSnapshot ds : dataSnapshot.getChildren()) {
      ```
- Avoid allocating objects like paint in onDraw()
    - Prepare everything in the constructor, so it's ready when drawing.
    - Use DrawText() for drawing the test
- Most of the Bitmaps used in the code have not been as this would use more memory
    - If full resolution images are used, too much memory will be used
- Destroy listeners
    - This prevents memory leak
- Avoid using static variables

- This prevents memory leak as well

Improvements/Extensions

Sound when bullets fired
Implement more tanks that destroy each other
Implement a joystick to enhance playing experience
Developing a 3D tank wars game where the tanks can randomly move in the arena and fire bullets

Conclusions

I have enjoyed developing this Android gaming app and learnt a lot about OOP design and how to link to an online database. There were a few challenges with using the Emulator but used the different Emulators available to make the program work effectively. I would like to enhance this game and spend time understanding how to implement a 3D game.

References

https://www.toptal.com/android/android-performance-tips-tools
https://android-developers.googleblog.com/2019/04/improving-app-performance-with-art.html
https://developer.android.com/training/articles/perf-tips
https://www.toptal.com/android/android-performance-tips-tools
https://heartbeat.fritz.ai/increasing-performance-in-an-android-application-1086640aeef#e81f

GitLab Link

https://csgitlab.reading.ac.uk/fe010336/android-tankwars.git

Demo self-assessment table

Each tick-box represents one mark unless otherwise specified.		Range
Code style (2 marks each points): Following Java code conventions for all the project ✓ [ ] variable and class names ✓ [ ] method names ✓ [ ] indentation rules ✓ [ ] Using inline comments frequency ✓ [ ] Javadoc comments for every function (except getters/setters)	Overall OOP design and API usage: ✓ [ ] Appropriate use of inheritance (2 marks) ✓ [ ] Appropriate use of Abstract classes (2 marks) ✓ [ ] Use of Android API classes/methods (Other than in base code) (3 marks) ✓ [ ] Greater than 3 original classes (3 marks)	0-20
Functionality: ✓ [ ] On-screen menus (1 mark) ✓ [ ] Scores (1 mark) ✓ [ ] Controllable character (1 mark) ✓ [ ] Sensor interaction (touch/ accelerometer/ etc) (1 mark) Game has levels ✓ [ ] Works (4 marks) [ ] Attempted (2 marks) Use of standardised data structures (e.g., List, Vector, Collection, Date): ✓ [ ] Works (10 marks) [ ] Attempted (5 marks) Randomly/procedurally generated features: ✓ [ ] Works (10 marks) [ ] Attempted (5 marks) Opponents (AI): ✓ [ ] Works (5 marks) [ ] Attempted (3 marks)	Design quality (both game and other game screens) (1 mark each): ✓ [ ] Professional looking ✓ [ ] Understandable game flow ✓ [ ] Feedback to user instead of crashing, or recover ✓ [ ] Runs smoothly without interruptions ✓ [ ] Installs without error ✓ [ ] Starts/exits without error ✓ [ ] Program does not crash ✓ [ ] Produced in video form	0-40
Improvements marks: Online high score list ✓ [ ] Works (10 marks) [ ] Attempted (5 marks) Multithreading improvements [ ] Works (10 marks) ✓ [ ] Attempted (5 marks) Memory optimisation [ ] Works (5 marks) ✓ [ ] Attempted (3 marks) User Level Creation [ ] Works (5 marks) [ ] Attempted (3 marks)	Other extension/improvement / innovation [ ] Works (10 marks) ✓ [ ] Attempted (5 marks) Note: either the attempted marks or the works marks will be given (so you can only receive 5 marks per attempt max, and there is a max of 10 marks for this section but tick off any extra work done anyway!) For attempted marks to be awarded the feature must be at such a state, that the code could have worked but does not due to bugs or similar.	0-40