Android App Development

Summer Training Report

Submitted By

Kainaat Singh

STIP ID: STIP/17/23374

Course Date:06/06/2017

Under the Guidance of

Shubham Gupta

Engineer

Wingfotech Pvt. Ltd.

Branch Name: Computer Science

College Name:PEC University of Technology,Chandigarh

Submitted to:

Trainee Name: Shubham Gupta

Designation : Engineer

Submitted By:

Student Name: Kainaat Singh

STIP ID: STIP/17/23374

Declaration

I, Kainaat Singh hereby declare that the work which is being presented in this project report titled "Dungeon Runner" by me, in partial fulfillment of the requirements for the award of Android App Development Degree in "Computer Science", is an authentic record of my own work carried out under the guidance of Shubham Gupta, Engineer, Wingfotech Pvt. Ltd.

To the best of my knowledge, the matter embodied in this report has not been submitted to any other University/ Institute for the award of any degree or diploma.

Kainaat Singh

Acknowledgement

It is a pleasure to acknowledge many people who knowingly and unwittingly helped us, to complete our Summer Training. First of all let us thank God for all the blessings, which carried us through all these years.

I express my gratitude to Shubham Gupta, Engineer for providing this opportunity of getting one month summer training.

I would like to expresses my gratitude to other faculty members of Computer Science Department for providing academic inputs, guidance & encouragement throughout the training period.

Kainaat Singh

List of Figures

List of Code Snippets

NOTE TO READER

Due large number of lengthy files present in the source code of this project only small code snippets have been included in this project report.

For viewing the entire source code of this project please follow the link given below:

https://github.com/svensevenslow/Dungeon-Runner

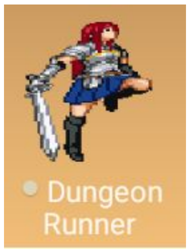


Table of Contents

Chapter 1

Introduction

1.1 Wingfotech Pvt. Ltd.

- ✓ This company works vividly into various domains, education being 1 of them. The brand name is stated as 'Wingfotech'.
- ✓ This brand is a leading brand in Online Education and holds its specialization in Science & Technology.
- ✓ Help to prepare educational professionals recognized for the quality and significance of their teaching, research, scholarship, service, outreach, and leadership.
- ✓ Enhance the commitment of faculty, staff, and students to the centrality of diversity. Sustain a caring, supportive climate throughout the event.

About Wingfotech Pvt. Ltd.

Wingfotech Pvt. Ltd. is the leading training provider firm, that have been making a name of itself with its systematic & scientific approach in the field of latest technology covering Electronics/Electrical , CS/ IT, Mechanical , Mechatronics and other Engineering branches ,founded to provide a safe, educational, valuable, enjoyable, and life-changing environment for Innovating and learning minds. We organize seminars, workshops and training programs that are aimed at encouraging students to think deeply about innovation.

1.2 About Company's Mission & Vision

Mission:

Our mission is to inspire young people to be science and technology leaders, by engaging them in exciting mentor-based programs that build science, engineering and technology skills, that inspire innovation, and that foster well-rounded life capabilities including self-confidence, communication, and leadership

Vision:

To Transform our culture by creating a world where science and technology are celebrated and where young people dream of becoming science and technology leaders. That breed inventions in technology . Our vision is of a new world of learning based on the compelling truth that improving education is the key to the survival of the human race services.

1.3 Various Services provided by the Company:

- ✓ Robotics Lab
- ✓ Research Work
- ✓ Online Electronics Store
- ✓ Workshops, Summer Training, Winter Training
- ✓ Industrial Solution
- ✓ Engineering Solution
- ✓ Training and Certification
- ✓ Job Oriented Program

Chapter 2

SOFTWARE REQUIREMENTS

1 Eclipse IDE

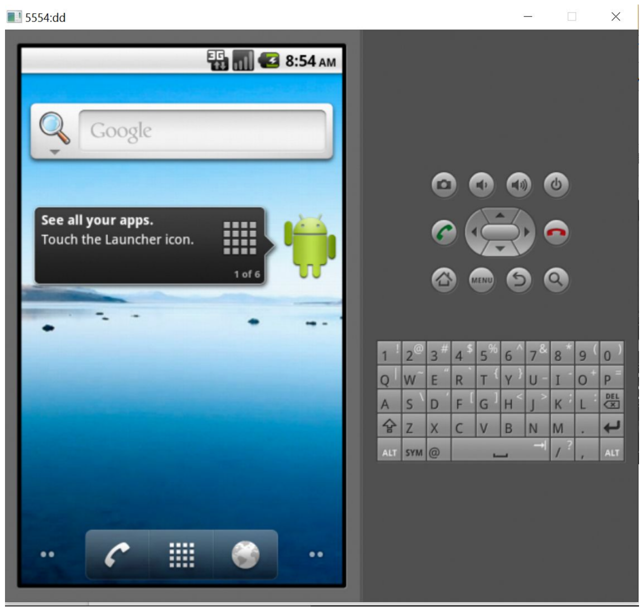
2 JDK(version 7 or above)

3 Android SDK

4 Piskel IDE

5 Windows Paint

eclipse
HELIOS
(c) Copyright Eclipse contributors and others, 2000, 2011. All rights reserved. Eclipse is a trademark of the Eclipse Foundation,Inc. Oracle and Java are registered trademarks



Chapter 3: SOFTWARE DESCRIPTION

I. SOFTWARE FOR PROGRAMMING;

1.The programming part of the Android Application “Runner” was done via Eclipse IDE version 3.8(Helios)

2 For using Eclipse IDE for Android development the following software needs to be installed

a. Android SDK
b. JDK (version 7 or above)

3 The basic program is written in java.The layout is designed using XML.

4 On running the program .apk file is generated which is run on the emulator or installed on the mobile phone as an application.

II. SOFTWARE FOR GRAPHICS

1 Piskel:An online IDE for pixel art. It has been used for the design of sprites,power-ups and tiles in the game.

2 Microsoft Paint: It is a basic graphics/painting utility.It has been used for designing the background of the game

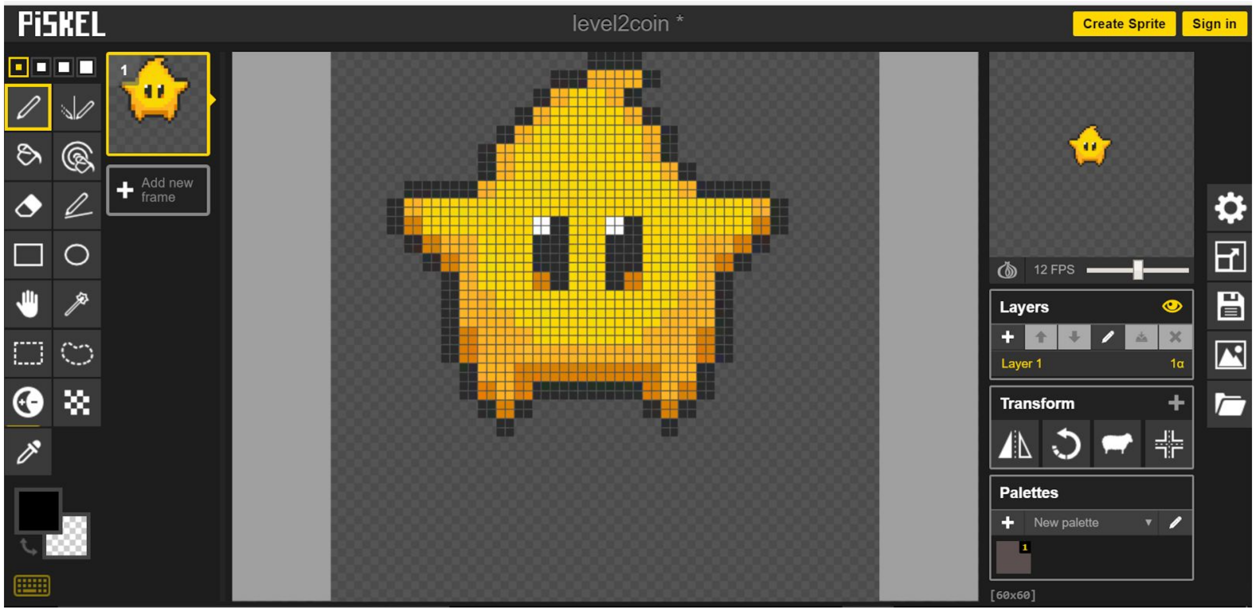


Fig: Designing Golden fruit using piskel

Chapter 4: Coding

Introduction

Let me introduce you to an exciting new game called “Dungeon Runner” . It is an Android Application which is a Java game ported to Android.

Dungeon Runner:An Overview

I. Game description

1 The Game consists of two levels.Level 2 can be accessed only after a successful completion of Level 1.

2 It is a platform game .It involves guiding a sprite(character) to jump between suspended platforms and/or over obstacles to traverse their environment.

3 The aim of the game is to collect stars in Level 1 and the golden fruit in Level 2.

4 While attempting to collect stars and golden fruit the player also has to fight arch enemies Heliboy in Level 1 and Iron Man in Level 2.

5 The player scores a point every time he collects a star or the golden fruit.

6 Ideally, the game ends with successful completion of Level 2. However falling off the platform will lead to sprite death and end of the game.

7 The weapon used by the sprite is a pistol- gun . It can be used both as a gun and a sword.

II. Playing the Game

1. The left side of the screen consists of three buttons
   - **a.** The top button is used for making the sprite jump up to traverse the platforms .
   - **b.** The middle button is used for shooting bullets aimed at the arch enemies.
   - **c.** The bottom button is used for making the sprite duck.

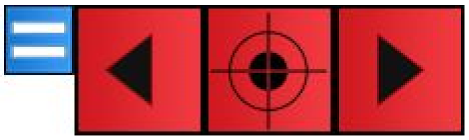


Fig1 : Buttons on the screen

2. A long press on the right side of the screen will enable the player to control the movement of the sprite towards the right.

III. SCREEN SHOTS OF THE WORKING GAME

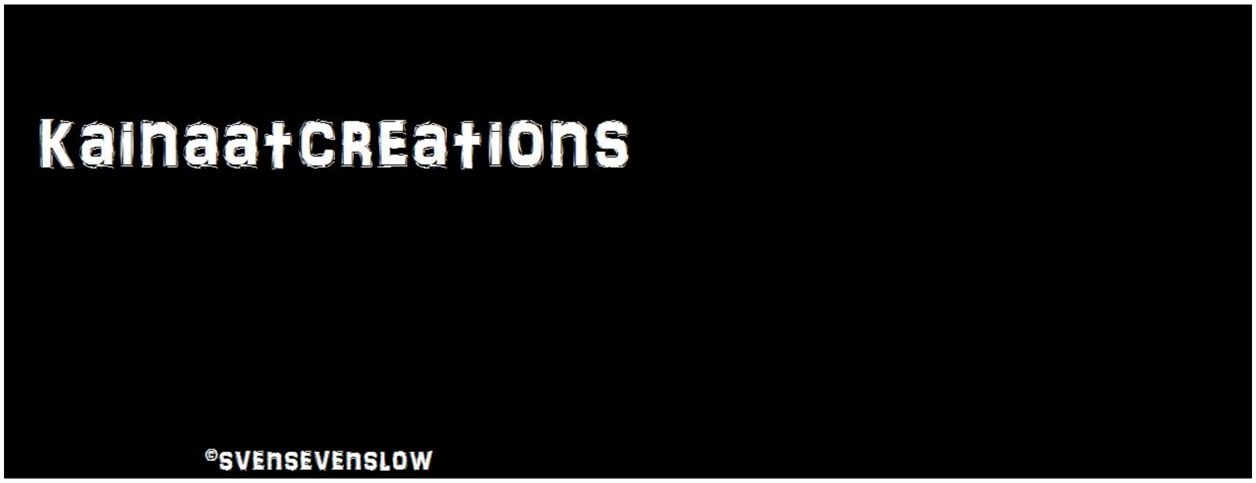


Fig 2 : Splash Screen

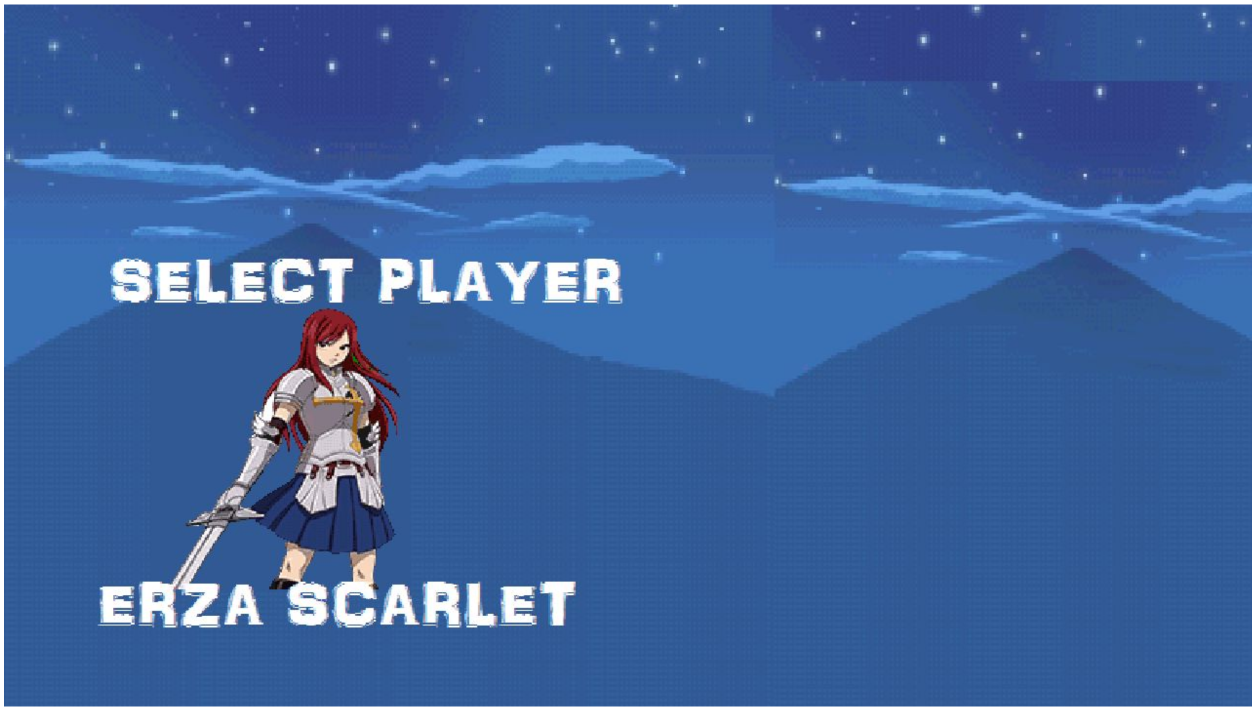


Fig 3 : Main Menu Screen

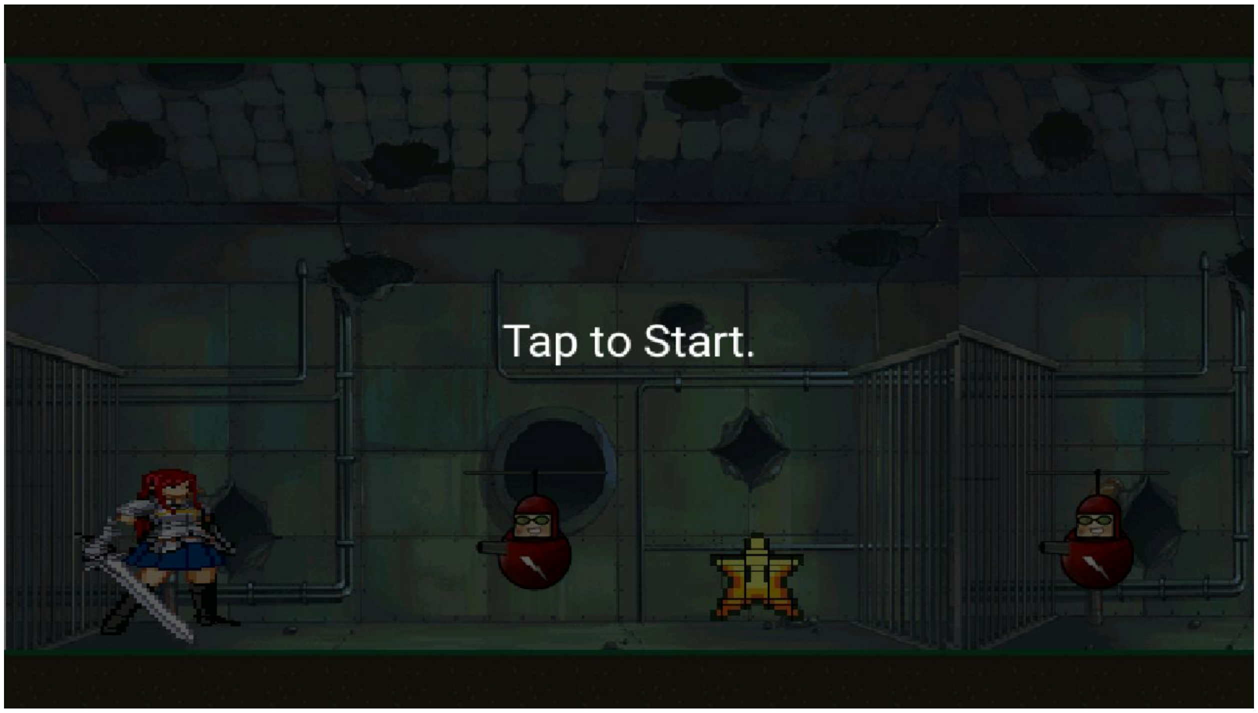


Fig 4: Start Screen

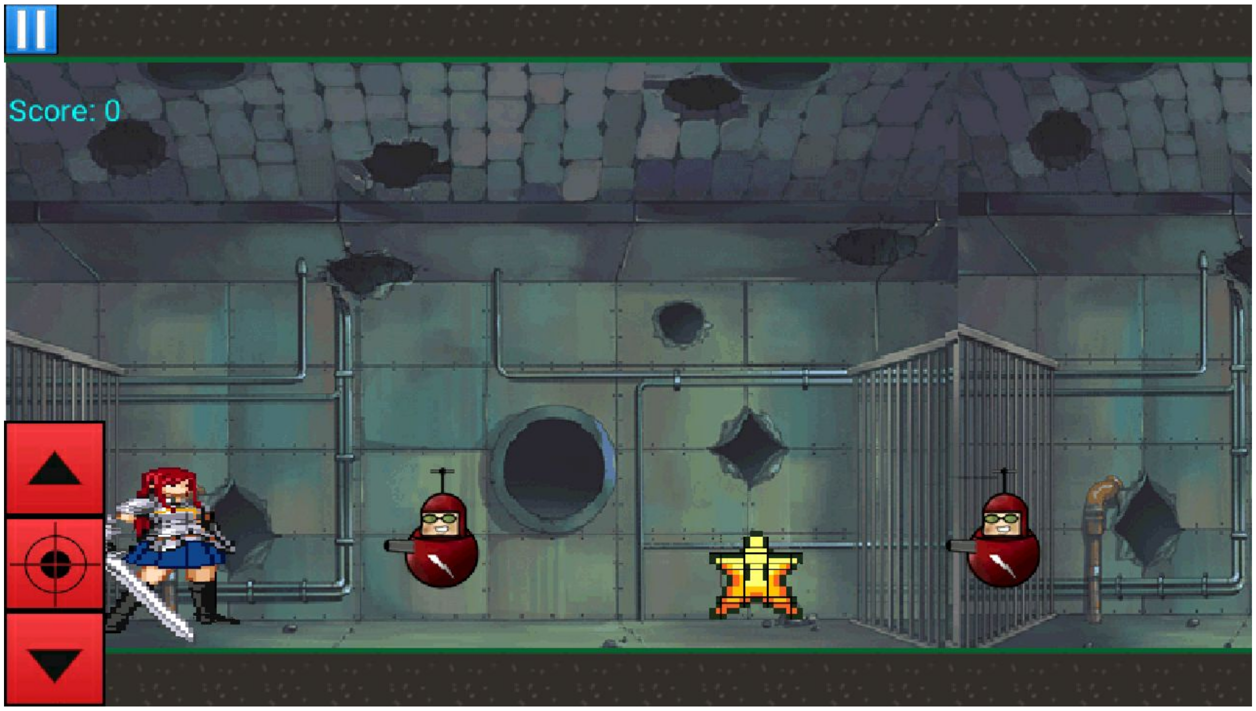


Fig 5 :Start of Level 1

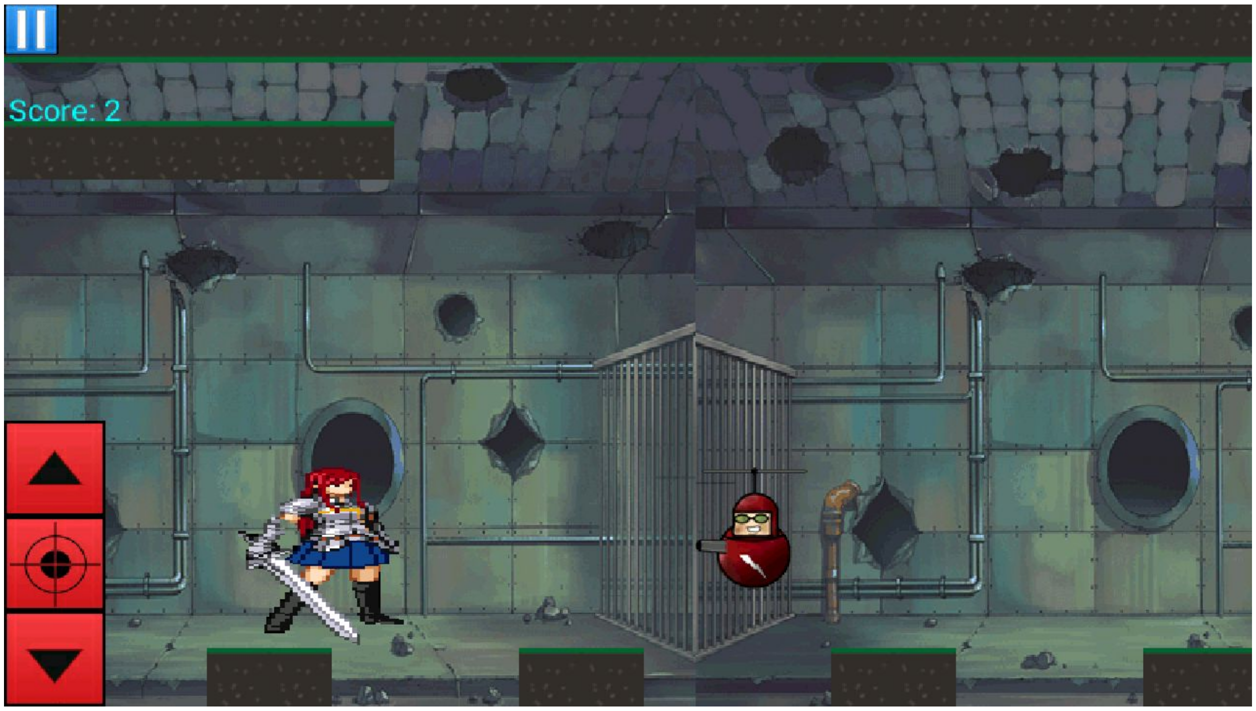


Fig 6 : Continuing Level 1

Fig 7: Start Screen for Level 2

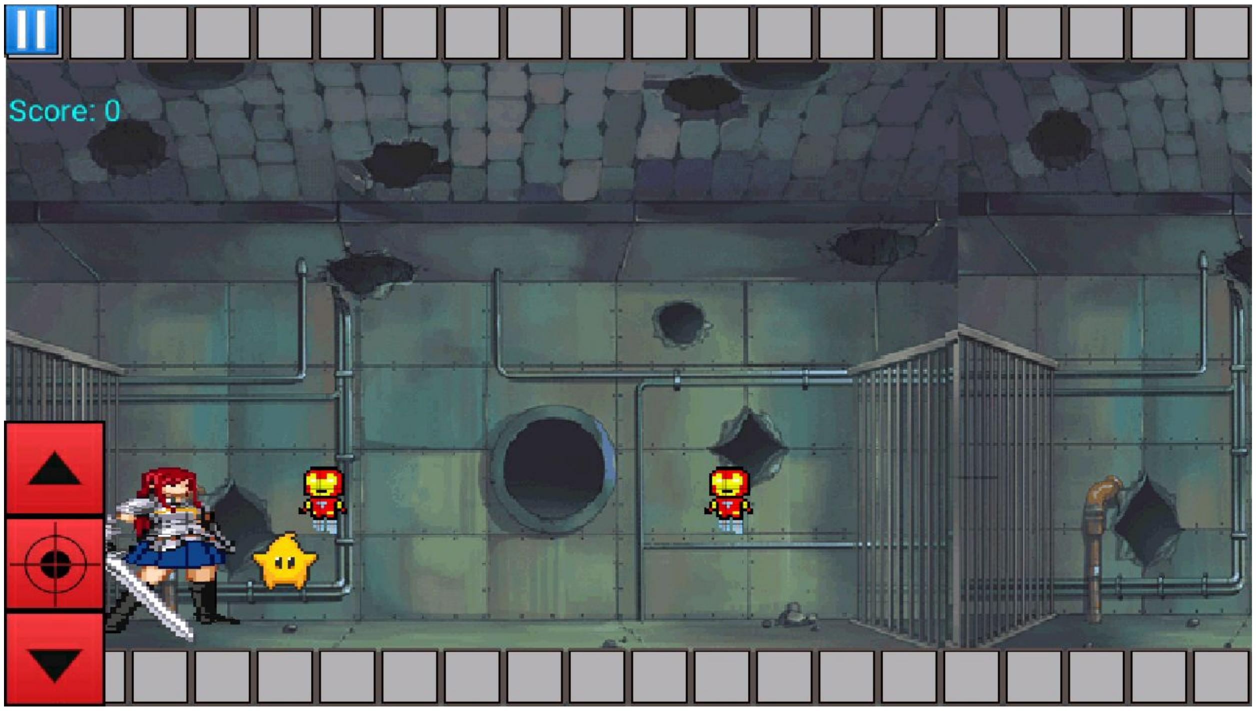


Fig 7: Start of Level 2

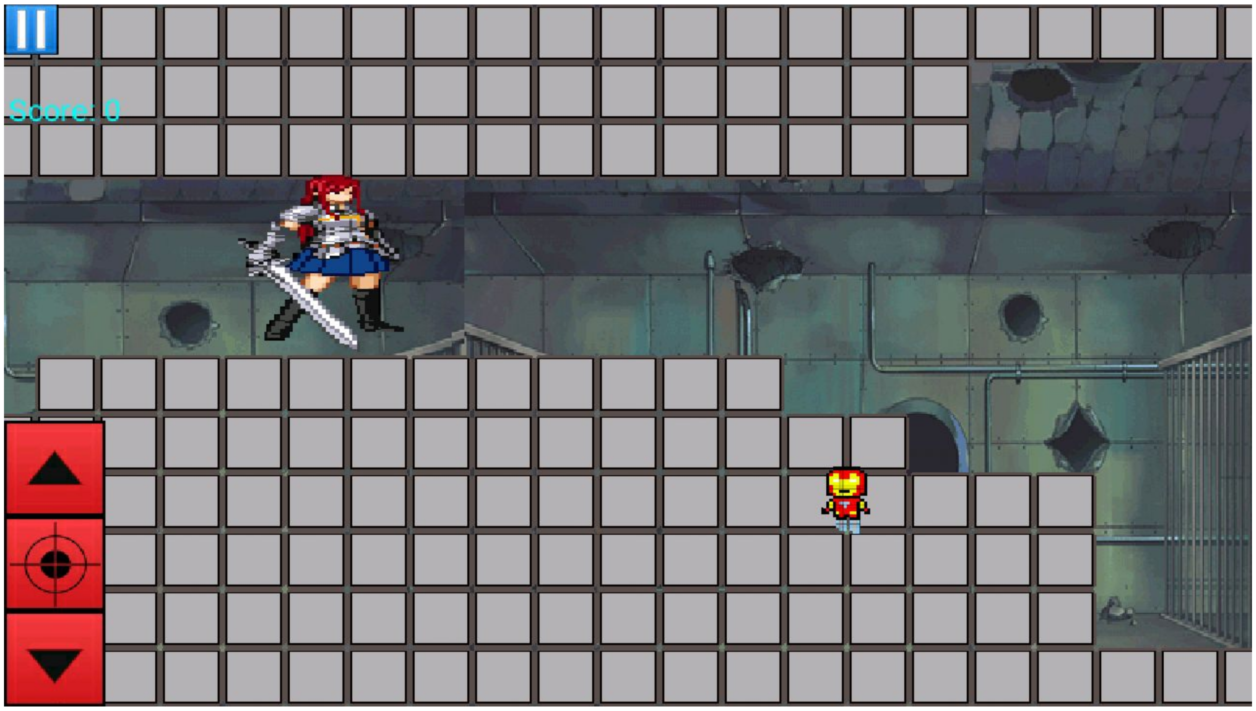
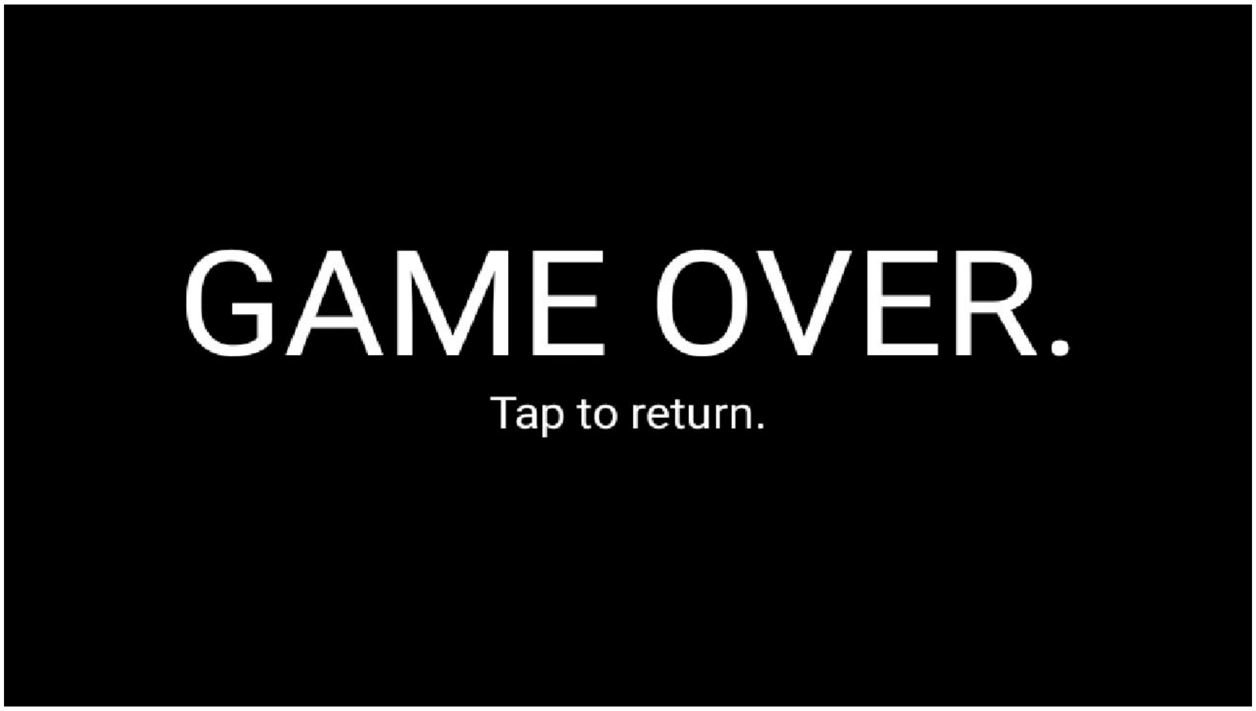
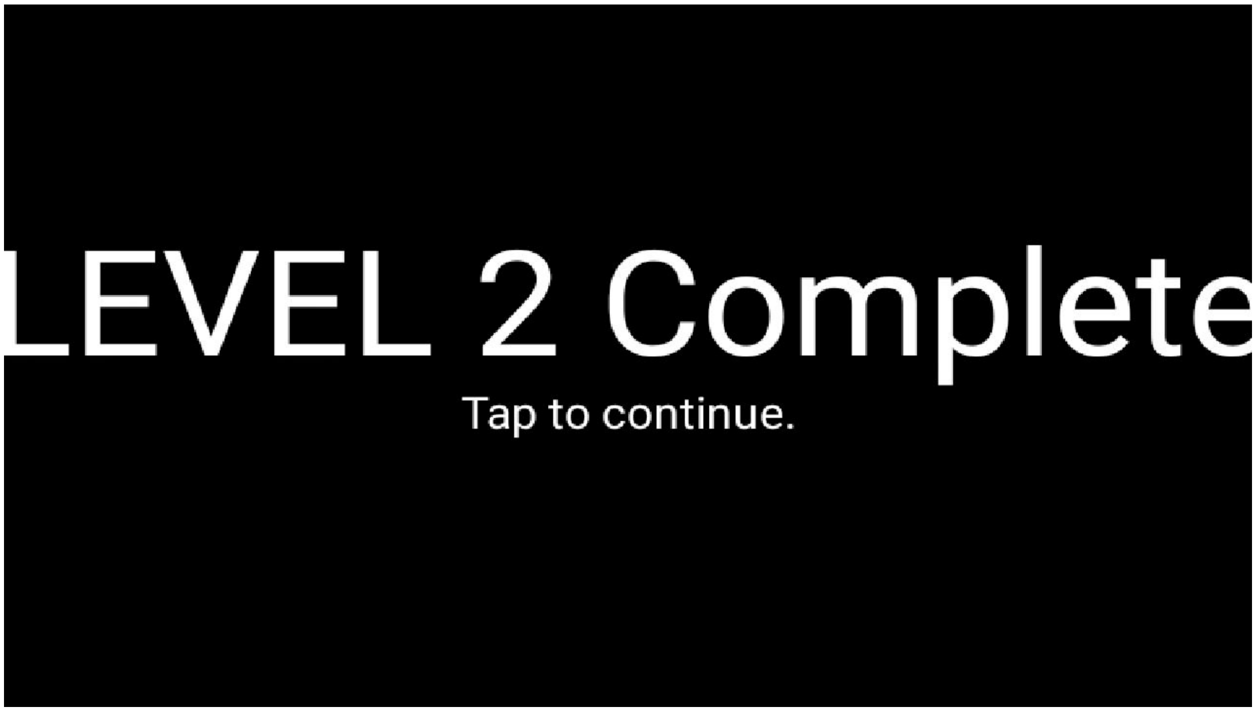


Fig 8: Continuing Level2


GAME OVER.

Tap to return.

Fig 9: Game Over Screen


LEVEL 2 Complete

Tap to continue.

Fig 10 Level 2 Completion Screen

4.2) BASIC GAME LOGIC

1.The Game Loop

This is an infinite loop which runs in the game until the game gets over.All variables, methods etc used in the game are repeatedly updated here.

2. Animation

Animation is created by change of frames after every 17 milliseconds .Research has proved that change of frame every 17 milliseconds allows the eye to perceive it as an animation.

3 Handling Collision

For handling collisions the player and enemies are considered as rectangles. When the rectangles overlap it is considered a collision. Similarly collisions with tiles are handled.

4. On Collision changes

A. **Collision with a bullet:** When an enemy is hit by bullets it is considered a collision. After the collision the enemy is supposed to be removed from the screen.To do this the position of the enemy on the screen is changed such that it is not visible on the screen.

For eg: If the position of its y- coordinate in pixels is 360 then it is made -360 so that it is not visible on the screen any more.The top-most left corner of the screen is considered as (0,0).

B. Collision with stars and golden fruit: On collision with a star or a golden fruit the star or golden fruit is no longer visible on the screen as its y-coordinate in pixels is changed in the program .The score is also increased on collecting stars and golden fruit. A variable is used to keep count of the score,such as

```
collection = GameScreen.getRobot().getScore() + 1;
```

every time a collision takes place;

C. On collision with tiles: On collision with a tile the x or y coordinate of the sprite is changed such that it is above the tile .

```
public void checkVerticalCollision(Rect rtop, Rect rbot) {

                if (Rect.intersects(rbot, r) && type == 8) {

                        robot.setJumped(false);

                        robot.setSpeedY(0);

                        robot.setCenterY(tileY - 63);

                }

        }



public void checkSideCollision(Rect rleft, Rect rright, Rect leftfoot, Rect rightfoot) {
                if (type != 5 && type != 2 && type != 0 && type != 3 && type != 1){
                        if (Rect.intersects(rleft, r)) {
                                robot.setCenterX(tileX + 102);

                                robot.setSpeedX(0);

                        }else if (Rect.intersects(leftfoot, r)) {

                                robot.setCenterX(tileX + 85);
                                robot.setSpeedX(0);
                        }

                        if (Rect.intersects(rright, r)) {
                                robot.setCenterX(tileX - 62);

                                robot.setSpeedX(0);
                        }

                        else if (Rect.intersects(rightfoot, r)) {
                                robot.setCenterX(tileX - 45);
                                robot.setSpeedX(0);
                        }
                }

}
```

Code Snippet: Collision with tiles

4 Dealing with sprite death

If the y coordinate in pixels of sprite goes beyond a certain value the game is considered to be over

```
        if (robot.getCenterY() > 500) {
                        state = GameState.GameOver;
                }
```

Code Snippet: Sprite Death

5 Switching between levels

If the y coordinate in pixels of sprite goes below a certain value the sprite is moved on to the next level

```
if (robot.getCenterY() < 80) {
                        state = GameState.LevelUp;
}
```

Code Snippet: Switching Levels

6 Some AI touches in the game

The enemy will always move in the direction of the movement of the sprite.

```
public void follow() {
            if (centerX < -95 || centerX > 810) movementSpeed = 0;
            else if (Math.abs(robot.getCenterX() - centerX) < 5) movementSpeed = 0;
            else {
                  if (robot.getCenterX() >= centerX) {
                        movementSpeed = 1;
                  } else {
                        movementSpeed = -1;
                  }
                  }
            }
```

7 Handling touch events

1 Touch Handler Class

```
public interface TouchHandler extends OnTouchListener {

    public boolean isTouchDown(int pointer);


    public int getTouchX(int pointer);


    public int getTouchY(int pointer);


    public List<TouchEvent> getTouchEvents();

}
```

Code Snippet: Java class for handling touch events

```
private boolean inBounds(TouchEvent event, int x, int y, int width,

                      int height) {

              if (event.x > x && event.x < x + width - 1 && event.y > y

                             && event.y < y + height - 1)

                      return true;

              else

                      return false;

       }
```

Code Snippet:The inBounds method for handling touch events depending on the position of contact on the screen

4.3) Displaying multiple enemies ,stars and golden fruit in the game

A thread has been utilized for implementing a timer in the game so that after regular intervals of time enemies,stars and golden fruit can be displayed on the screen.

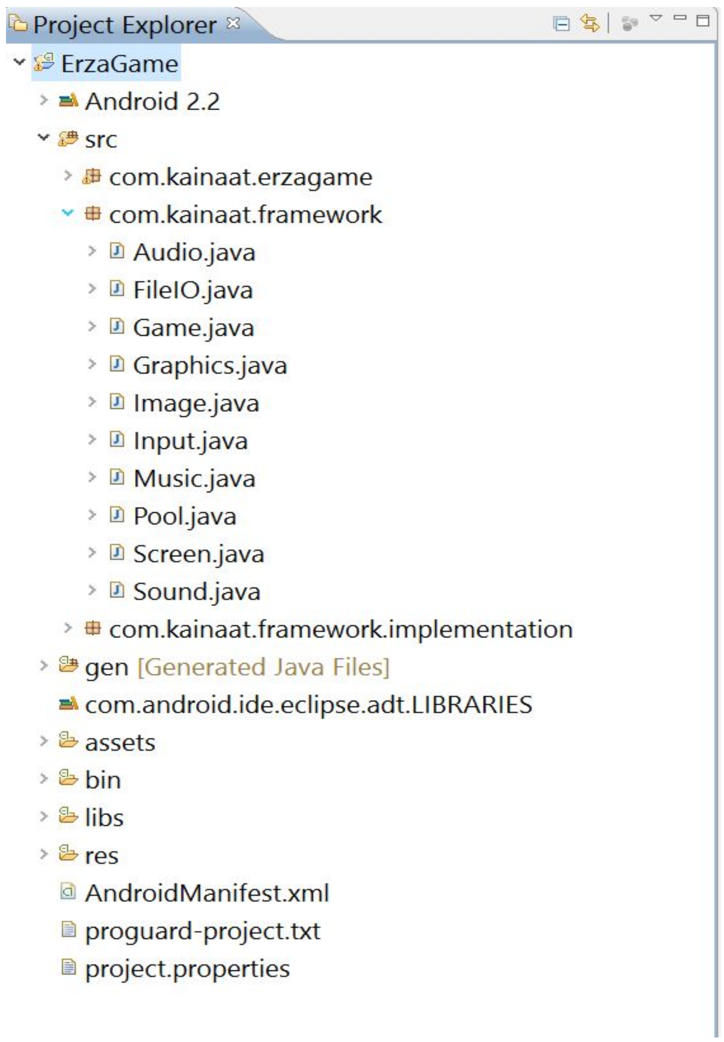
```
       Thread timer = new Thread() {

                      public void run() {
                             try {
                                    sleep(3000);
                                    visible = true;
                                    GameScreen.hb.setCenterY(360);
                                    GameScreen.hb2.setCenterY(360);
                                    GameScreen.hb.setCenterX(bg2.getBgX() - 300);
                                    GameScreen.hb2.setCenterX(bg2.getBgX() - 600);

                             } catch (InterruptedException e) {
                                    e.printStackTrace();
                             }
                      }
};
timer.start();
```

4.4) Implementing the game in Android

I. The basic layout

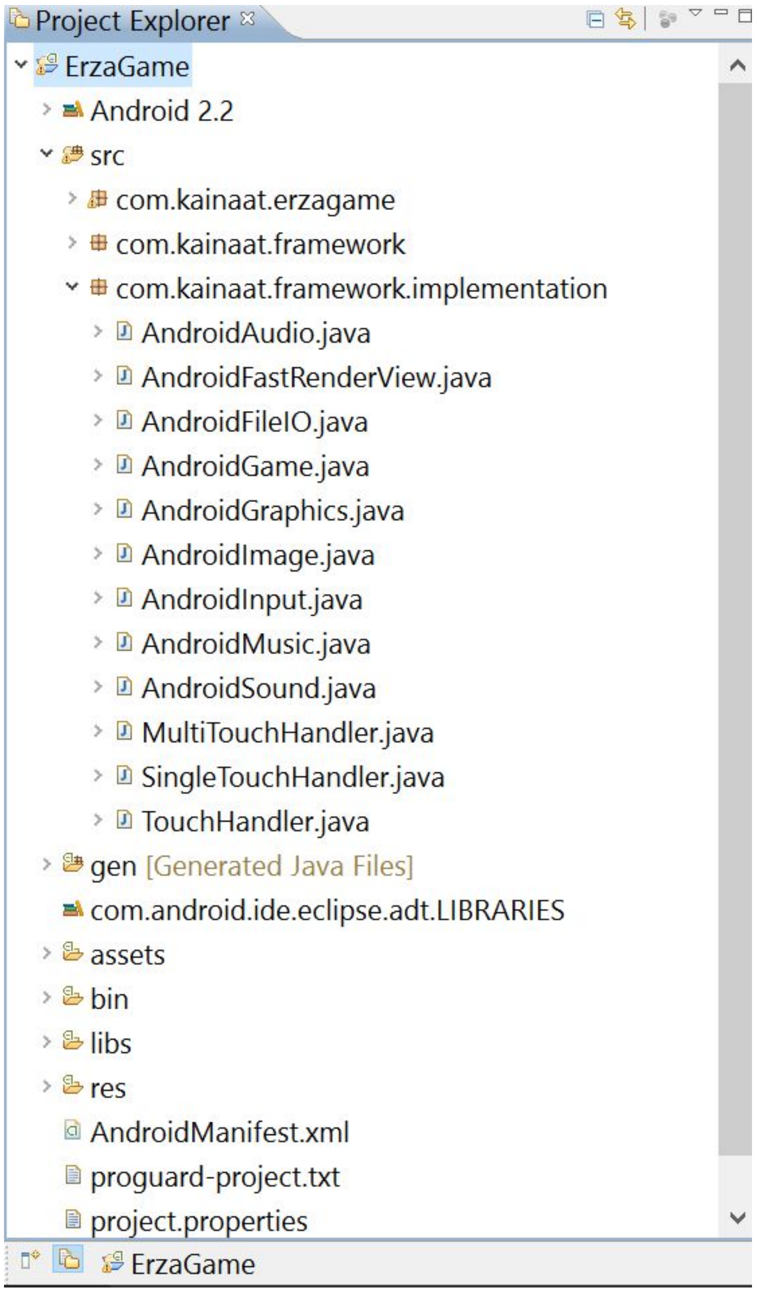
The source code is divided into three packages

a. **The framework package:** This consists of java interfaces in which all the methods are defined. These are used by the java classes that implement these interfaces.



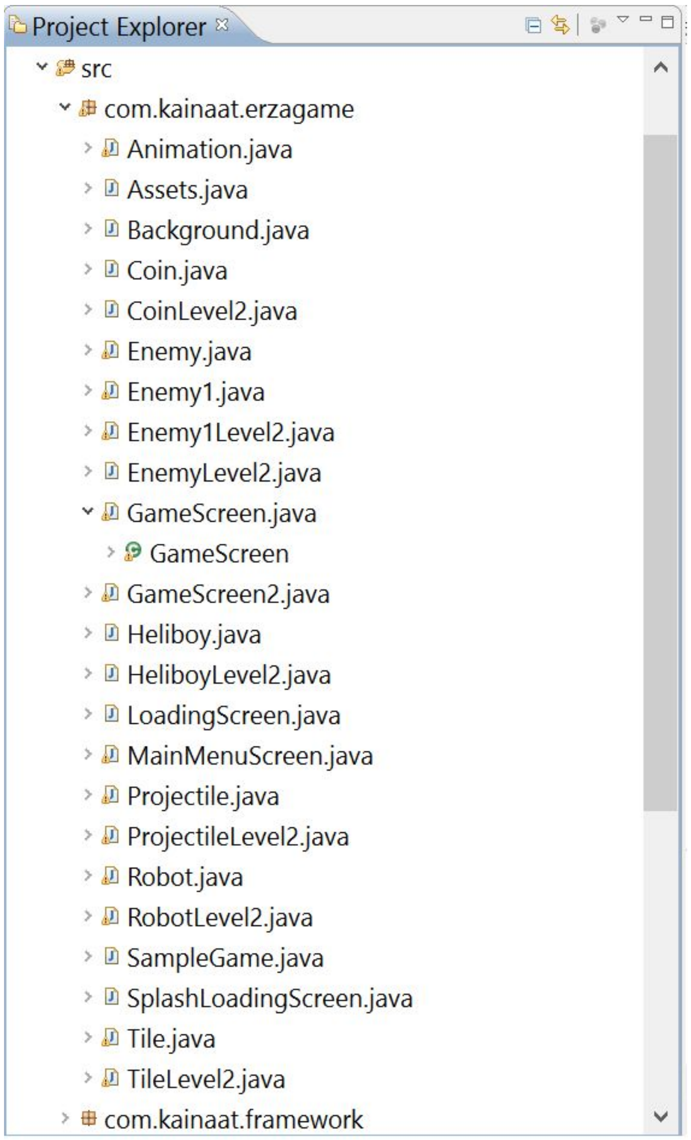
```
Project Explorer
ErzaGame
  Android 2.2
  src
    com.kainaat.erzagame
    com.kainaat.framework
      Audio.java
      FileIO.java
      Game.java
      Graphics.java
      Image.java
      Input.java
      Music.java
      Pool.java
      Screen.java
      Sound.java
    com.kainaat.framework.implementation
  gen [Generated Java Files]
  com.android.ide.eclipse.adt.LIBRARIES
  assets
  bin
  libs
  res
  AndroidManifest.xml
  proguard-project.txt
  project.properties
```

b. The framework implementation package : This package consists of java classes implementing the interfaces written in the framework package.The methods from the interfaces that it implements are given functionality here depending on our requirement.



```
Project Explorer
ErzaGame
  Android 2.2
  src
    com.kainaat.erzagame
    com.kainaat.framework
    com.kainaat.framework.implementation
      AndroidAudio.java
      AndroidFastRenderView.java
      AndroidFileIO.java
      AndroidGame.java
      AndroidGraphics.java
      AndroidImage.java
      AndroidInput.java
      AndroidMusic.java
      AndroidSound.java
      MultiTouchHandler.java
      SingleTouchHandler.java
      TouchHandler.java
  gen [Generated Java Files]
  com.android.ide.eclipse.adt.LIBRARIES
  assets
  bin
  libs
  res
  AndroidManifest.xml
  proguard-project.txt
  project.properties
ErzaGame
```

c. The game package: It consists of java classes that are subclasses of classes defined in the framework implementation package.This is where the actual game is coded


Project Explorer

```
src
  com.kainaat.erzagame
    Animation.java
    Assets.java
    Background.java
    Coin.java
    CoinLevel2.java
    Enemy.java
    Enemy1.java
    Enemy1Level2.java
    EnemyLevel2.java
    GameScreen.java
      GameScreen
    GameScreen2.java
    Heliboy.java
    HeliboyLevel2.java
    LoadingScreen.java
    MainMenuScreen.java
    Projectile.java
    ProjectileLevel2.java
    Robot.java
    RobotLevel2.java
    SampleGame.java
    SplashLoadingScreen.java
    Tile.java
    TileLevel2.java
  com.kainaat.framework
```

As an example, the Code for the Image Interface and AndroidImage class is shown below

1 The Image interface in the framework package

```
package com.kainaat.framework;

import com.kainaat.framework.Graphics.ImageFormat;

public interface Image {

    public int getWidth();

    public int getHeight();

    public ImageFormat getFormat();

    public void dispose();

}
```

2 The implementation of the Image Interface

```
package com.kainaat.framework.implementation;


import android.graphics.Bitmap;

import com.kainaat.framework.Image;

import com.kainaat.framework.Graphics.ImageFormat;

public class AndroidImage implements Image {

    Bitmap bitmap;

    ImageFormat format;


 public AndroidImage(Bitmap bitmap, ImageFormat format) {

        this.bitmap = bitmap;

        this.format = format;
    }
    @Override
    public int getWidth() {
        return bitmap.getWidth();
    }

    @Override
    public int getHeight() {
        return bitmap.getHeight();
    }

    @Override
    public ImageFormat getFormat() {
        return format;
    }
   @Override
    public void dispose() {
        bitmap.recycle();
    }
}
```

4.4) II. A Brief description of the Game package

The GameScreen Class is where the entire game runs.The update methods of all classes in this package are called in the update method of the GameScreen Class

a. Class: Animation

1.The ArrayList frames will contain AnimFrame objects that have two values: an image and a duration it is displayed.

2. **animTime** and **currentFrame** are **synchronized**, the two will always be called sequentially .

```
public Animation() {
            frames = new ArrayList();
            totalDuration = 0;

            synchronized (this) {
                animTime = 0;
                currentFrame = 0;
            }
        }
```

Code Snippet: Constructor for the Animation class

1 currentFrame refers to the integer value index of the current frame in the ArrayList (the current image shown).
2 animTime will keep track of how much time has elapsed since the current image was displayed, so that when animTime reaches a certain value, we can switch over to the next frame.
3. totalDuration, refers to the amount of time that each frame (image) will be displayed for.
4. elapsedTime is the amount of time that passed since the last update.

```
public synchronized void update(long elapsedTime) {
		if (frames.size() > 1) {
			animTime += elapsedTime;
			if (animTime >= totalDuration) {
				animTime = animTime % totalDuration;
				currentFrame = 0;

			}

			while (animTime > getFrame(currentFrame).endTime) {
				currentFrame++;

			}
		}
	}
```

Code Snippet: Update method for Animation class

b. Class:Background

1 The background class gets the current background from the GameScreen class and enables Rightwards scrolling

```
public Background(int x, int y){
            bgX = x;
            bgY = y;
            speedX = 0;
      }
```
Code Snippet : Constructor for the background class

```
public void update() {
            bgX += speedX;

            if (bgX <= -2160){
                  bgX += 4320;
            }
      }
```
Code Snippet : update method of the background class

c. Class: Projectile

The projectile class handles bullets in the game .It takes care of creating bullets on the screen and then removing them on collision with the enemy

```
private void checkCollision() {

             if (Rect.intersects(r, GameScreen.hb.r)) {
                    visible = false;

                    if (GameScreen.hb.health > 0) {
                           GameScreen.hb.health -= 1;

                    }
                    if (GameScreen.hb.health == 0) {
                           GameScreen.hb.setCenterX(-360);
                           bonus = GameScreen.getRobot().getBonus() + 5;
                           GameScreen.hb.health = 5;

                    }
```

Code Snippet: Checking collision of bullets with the enemy

d. Class: Robot

1 The Robot class defines the game sprite/character

```
private Background bg1 = GameScreen.getBg1();
private Background bg2 = GameScreen.getBg2();
Code Snippet 3: Recieving the current background from the GameScreen class

public void update() {
      // Moves Character or Scrolls Background accordingly.

      if (speedX < 0) {
            centerX += speedX;
      }
      if (speedX == 0 || speedX < 0) {
            bg1.setSpeedX(0);
            bg2.setSpeedX(0);

      }
      if (centerX <= 200 && speedX > 0) {
            centerX += speedX;
      }
      if (speedX > 0 && centerX > 200) {
            bg1.setSpeedX(-MOVESPEED / 5);
            bg2.setSpeedX(-MOVESPEED / 5);
      }

      // Updates Y Position
      centerY += speedY;

      // Handles Jumping

            speedY += 1;

      if (speedY > 3){
            jumped = true;
      }

      // Prevents going beyond X coordinate of 0
      if (centerX + speedX <= 60) {
            centerX = 61;
      }

      rect.set(centerX - 34, centerY - 63, centerX + 34, centerY);
      rect2.set(rect.left, rect.top + 63, rect.left+68, rect.top + 128);
      rect3.set(rect.left - 26, rect.top+32, rect.left, rect.top+52);
```

```
            rect4.set(rect.left + 68, rect.top+32, rect.left+94, rect.top+52);
            yellowRed.set(centerX - 110, centerY - 110, centerX + 70, centerY + 70);
            footleft.set(centerX - 50, centerY + 20, centerX, centerY + 35);
            footright.set(centerX, centerY + 20, centerX+50, centerY+35);


        }
```

Code Snippet : Update method of Robot class

e. Class : Tile

```
2222222222555555555

        88555555888


8888888   888855555
```

Fig :Sample map1.txt file which was used for making tile layout

Draws tiles on the screen after reading from the map file and checks for collision with sprite

```
	private Robot robot = GameScreen.getRobot();
	private Background bg = GameScreen.getBg1();
	Code Snippet 5:Recieving the current bakcground and sprite from the GameScreen
class

	public void update() {
			speedX = bg.getSpeedX() * 5;
			tileX += speedX;
			r.set(tileX, tileY, tileX+40, tileY+40);

			if (Rect.intersects(r, Robot.yellowRed) && type != 0) {
				checkVerticalCollision(Robot.rect, Robot.rect2);
				checkSideCollision(Robot.rect3, Robot.rect4,
Robot.footleft, Robot.footright);
			}

		}
Code Snippet : Update Method of tile class

	if (type == 5) {
			tileImage = Assets.tiledirt;
		} else if (type == 8) {
			tileImage = Assets.tilegrassTop;
		} else if (type == 4) {
			tileImage = Assets.tilegrassLeft;

		} else if (type == 6) {
			tileImage = Assets.tilegrassRight;

		} else if (type == 2) {
			tileImage = Assets.tilegrassBot;
```

```
            }else if(type == 1){
                  tileImage = Assets.coin;
            }else if(type == 3){
                  tileImage = Assets.levelUp;
            }else {type = 0;}
```

Code snippet :assigning tiles

f. Class : GameScreen

1. The GameScreen class is where the entire game runs.
2. Instances of objects belonging to the various classes like Enemy,Robot,Background,Coin,Animation etc are defined here

```
enum GameState {
             Ready, Running, Paused, GameOver,LevelUp
       }

public void update(float deltaTime) {
             List<TouchEvent> touchEvents = game.getInput().getTouchEvents();


             if (state == GameState.Ready)
                    updateReady(touchEvents);
             if (state == GameState.Running)
                    updateRunning(touchEvents, deltaTime);
             if (state == GameState.Paused)
                    updatePaused(touchEvents);
             if (state == GameState.GameOver)
                    updateGameOver(touchEvents);
             if(state == GameState.LevelUp){
                    updateLevelUp(touchEvents);
             }

       }
```

Code Snippet: Update method of the GameScreen class

```
private void updateGameOver(List<TouchEvent> touchEvents) {
             int len = touchEvents.size();
             for (int i = 0; i < len; i++) {
                    TouchEvent event = touchEvents.get(i);
                    if (event.type == TouchEvent.TOUCH_DOWN) {
                           if (inBounds(event, 0, 0, 800, 480)) {
                                  nullify();
                                  game.setScreen(new MainMenuScreen(game));
                                  return;
                           }
                    }
             }

       }
```

Code Snippet: Method called when Game state is Game Over

```
public void paint(float deltaTime) {
		Graphics g = game.getGraphics();

		g.drawImage(Assets.background, bg1.getBgX(), bg1.getBgY());
		g.drawImage(Assets.background, bg2.getBgX(), bg2.getBgY());
		paintTiles(g);

		//the bullets are constructed on the screen considering an array
		ArrayList projectiles = robot.getProjectiles();
		for (int i = 0; i < projectiles.size(); i++) {
			Projectile p = (Projectile) projectiles.get(i);
			g.drawRect(p.getX(), p.getY(), 10, 5, Color.YELLOW);
		}

		g.drawImage(currentSprite, robot.getCenterX() - 61,
				robot.getCenterY() - 63);
		g.drawImage(hanim.getImage(), hb.getCenterX() - 48,
				hb.getCenterY() - 48);
		g.drawImage(hanim.getImage(), hb2.getCenterX() - 48,
				hb2.getCenterY() - 48);
		g.drawImage(hcoin.getImage(), c1.getCenterX() - 48, c1.getCenterY());


		if (state == GameState.Ready)
			drawReadyUI();
		if (state == GameState.Running){
			drawRunningUI();
			updateScore();
		}
		if (state == GameState.Paused)
			drawPausedUI();
		if (state == GameState.GameOver)
			drawGameOverUI();
		if(state == GameState.LevelUp)
			drawLevelUpUI();


	}
```

Code Snippet: Painting objects on the screen

```
private void drawGameOverUI() {
		Graphics g = game.getGraphics();
		g.drawRect(0, 0, 1281, 801, Color.BLACK);
		g.drawString("GAME OVER.", 400, 240, paint2);
		g.drawString("Tap to return.", 400, 290, paint);

	}
```
Code Snippet: Method called to update the state of the screen when game is over

```
private void loadMap() {
		ArrayList lines = new ArrayList();
		int width = 0;
		int height = 0;

		Scanner scanner = new Scanner(SampleGame.map);
		while (scanner.hasNextLine()) {
			String line = scanner.nextLine();

			// no more lines to read
			if (line == null) {
				break;
			}

			if (!line.startsWith("!")) {
				lines.add(line);
				width = Math.max(width, line.length());

			}
		}
		height = lines.size();

		for (int j = 0; j < 12; j++) {
			String line = (String) lines.get(j);
			for (int i = 0; i < width; i++) {

				if (i < line.length()) {
					char ch = line.charAt(i);
					Tile t = new Tile(i, j,
Character.getNumericValue(ch));
					tilearray.add(t);
				}
```

```
                        }
                }

}
```

Code snippet: Loading map file for displaying tiles on the screen in the GameScreen constructor

```
paint4 = new Paint();
             paint4.setTextSize(20);
             paint4.setTextAlign(Paint.Align.LEFT);
             paint4.setAntiAlias(true);
             paint4.setColor(Color.CYAN);
```

Code snippet: Defining the size and alignment of the text

4 XML file

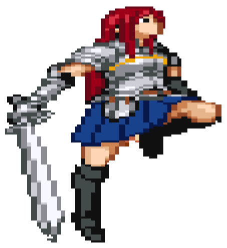
This game does not require an xml file for layout design

5 The Manifest file

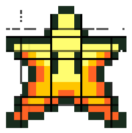
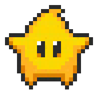
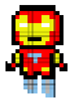
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="com.kilobolt.robotgame"

    android:versionCode="1"

    android:versionName="1.0" >


    <uses-permission android:name="android.permission.WAKE_LOCK" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.VIBRATE" />


    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />


    <application
        android:icon="@drawable/jumpp"
        android:label="Dungeon Runner" >
        <activity
            android:name="com.kainaat.erzagame.SampleGame"
            android:configChanges="keyboard|keyboardHidden|orientation"
            android:label="Dungeon Runner"
            android:screenOrientation="landscape" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />


                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>


</manifest>
```

Chapter 5: Graphics and Character Images

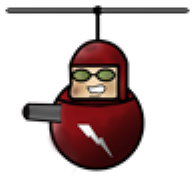
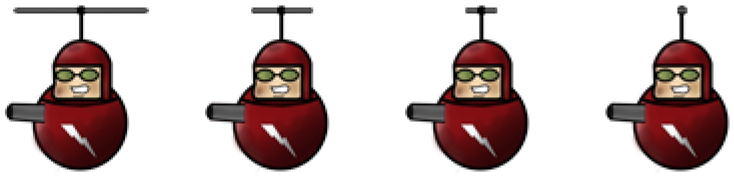
Sprite Name: Erza Scarlet



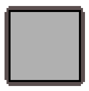
Crouch Position Standing position Jumping Position

Level1 Star **Level 2:** Golden Fruit **Level2 Enemy:**Iron Man

Level 1 Enemy : Heliboy



Level 1 Tiles **Level2 Tiles**

BIBLIOGRAPHY

Websites

1. http://www.kilobolt.com/game-development-tutorial.html
2. https://www.tutorialspoint.com/java
3. https://www.javatpoint.com/java-tutorial
4. https://www.tutorialspoint.com/android/
5. https://developer.android.com

Books

1. Java For Dummies, 7th Edition by **Barry A. Burd**

Online Communities

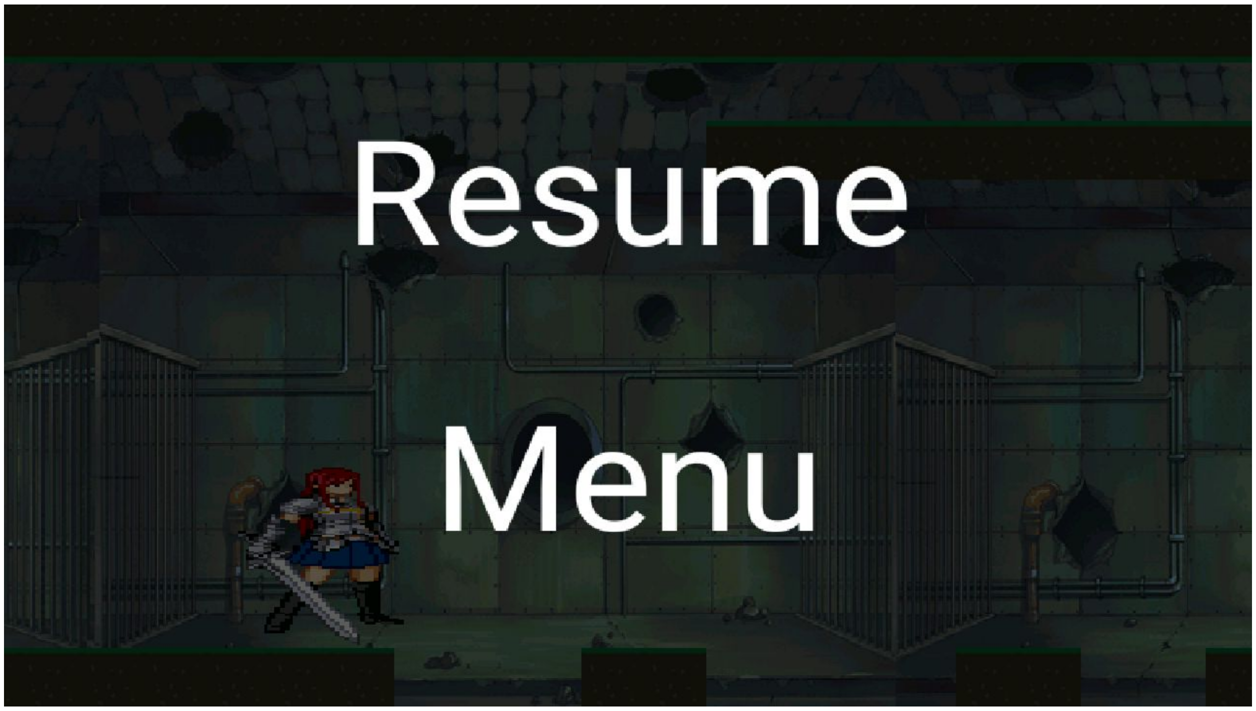
1. Stack - overflow

Guidance for Software Installation

1. https://eclipse.org/
2. http://www.oracle.com/technetwork/java/javase/

Graphics reference

1. http://spritedatabase.net


Resume
Menu