IT2010 – Mobile Application Development
BSc (Hons) in Information Technology
2nd Year
Faculty of Computing
SLIIT

2023 – Lab Exam 01
Report

Student ID	IT22898920
Batch	2024_Y2_S2_GROUP_02.02
Marks	
1. Code Quality and Organization (2 Points)	
2. Functionality (4 Points)	
3. Creativity and User Interface Design (2 Points)	
4. Performance and Stability (2 Point)	
Total: 10 Marks	
Evaluator	

Description:

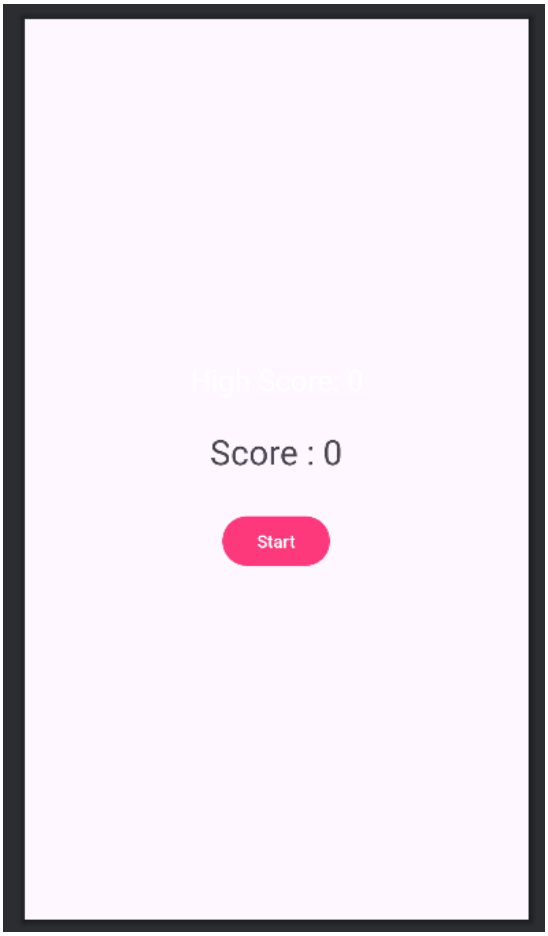
City Chase is an arcade-style game Android application. In this game, the name of the player is the “Thief” running across the city where the “Police” do not offer the “Thief” anything. In this game, the player’s goal is to keep the thief alive as long as possible by avoiding the police, moving through three lines to collect points for the scandal. A challenging game because as the “Thief’s” score increases, so does the game strategy.

Instructions:

1. **Starting the Game:** Press the 'Start' button on the main screen to initiate a new game session.
2. **Game Controls:**
    - Tap on the left half of the screen to move the Thief one lane to the left.
    - Tap on the right half of the screen to move the Thief one lane to the right.
3. **Scoring:** Avoid police and survive as long as possible to increase your score. Points are awarded for each police car dodged.
4. **High Scores:** Your highest score is saved automatically. Try to beat your own high score in subsequent games!

5. **Viewing High Scores:** Tap on the score at the top of the screen to toggle the visibility of the high score during gameplay.
6. **Ending the Game:** The game ends if you collide with any of the police obstacles. You can then choose to start a new game or exit.

Screenshots:



activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:id="@+id/rootLayout"
    tools:context=".MainActivity">


    <TextView
        android:id="@+id/highScoreTextView"
        android:textSize="24sp"
        android:textColor="@color/white"
        android:text="@string/high_score_0"
        android:textAlignment="center"
        android:layout_margin="8dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:visibility="visible" />
    <TextView
        android:id="@+id/score"
        android:textSize="28sp"
        android:text="@string/score_0"
        android:textAlignment="center"
        android:layout_margin="15dp"
```

```
        android:layout_margin="15dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/startBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="15dp"
        android:gravity="center"
        android:text="@string/start"
        app:backgroundTint="@color/button_colour" />

</LinearLayout>
```

MainActivity.kt

```kotlin
package com.example.new_game

import android.annotation.SuppressLint
import android.content.Context
import android.content.SharedPreferences
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.LinearLayout
import android.widget.TextView
import com.example.new_game.GameTask
import com.example.new_game.GameView

class MainActivity : AppCompatActivity(), GameTask {
    private lateinit var rootLayout: LinearLayout
    private lateinit var startButton: Button
    private lateinit var mGameView: GameView
    private lateinit var uscore: TextView
    private lateinit var highScoreTextView: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        startButton = findViewById(R.id.startBtn)
        rootLayout = findViewById(R.id.rootLayout)
        uscore = findViewById(R.id.score)
        highScoreTextView = findViewById(R.id.highScoreTextView)
        mGameView = GameView( c: this,  gameTask: this)
```

```kotlin
    startButton.setOnClickListener { it: View!
        startGame()
    }

    uscore.setOnClickListener { it: View!
        toggleHighScoreVisibility()
    }
}

private fun startGame() {
    mGameView.setBackgroundResource(R.drawable.gamebackground)
    rootLayout.addView(mGameView)
    startButton.visibility = View.GONE
    uscore.visibility = View.GONE
    highScoreTextView.visibility = View.GONE
}

@SuppressLint("SetTextI18n")
private fun toggleHighScoreVisibility() {
    highScoreTextView.visibility = if (highScoreTextView.visibility == View.VISIBLE) View.GONE else View.VISIBLE
    highScoreTextView.text = "High Score: ${getHighScore()}"
}

@SuppressLint("SetTextI18n")
override fun closeGame(mScore: Int) {
    uscore.text = "Score: $mScore"
```

```
        val currentHighScore = getHighScore()
        if (mScore > currentHighScore) {
            saveHighScore(mScore)
            highScoreTextView.text = "High Score: $mScore"
        }

        rootLayout.removeView(mGameView)
        startButton.visibility = View.VISIBLE
        uscore.visibility = View.VISIBLE
        highScoreTextView.visibility = View.VISIBLE

        mGameView.resetGameState()
    }

    private fun saveHighScore(score: Int) {
        val sharedPreferences = getSharedPreferences( name: "game_preferences", Context.MODE_PRIVATE)
        val editor = sharedPreferences.edit()
        editor.putInt(HIGH_SCORE_KEY, score)
        editor.apply()
    }

    private fun getHighScore(): Int {
        val sharedPreferences = getSharedPreferences( name: "game_preferences", Context.MODE_PRIVATE)
        return sharedPreferences.getInt(HIGH_SCORE_KEY,  defValue: 0)
    }

    companion object {
        private const val HIGH_SCORE_KEY = "high_score"
    }
}
```

GameView.kt

```
package com.example.new_game

import android.annotation.SuppressLint
import android.content.Context
import android.content.SharedPreferences
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.view.MotionEvent
import android.view.View
import com.example.new_game.R

@SuppressLint("ViewConstructor")
class GameView(var c: Context, var gameTask: GameTask) : View(c) {
    private var myPaint: Paint = Paint()
    private var speeds = 1
    private var time = 0
    private var score = 0
    private var police = ArrayList<HashMap<String, Any>>()

    var viewWidth = 0
    var viewHeight = 0
    var myThiefPosition = 0

    private val preferences: SharedPreferences = c.getSharedPreferences( name: "GamePreferences", Context.MODE_PRIVATE)

    init {
        myPaint = Paint()
    }
```

```
    fun resetGameState() {
        police.clear()
        score = 0
        speeds = 1
    }

    @SuppressLint("DrawAllocation", "UseCompatLoadingForDrawables")
    override fun onDraw(canvas: Canvas) {
        super.onDraw(canvas)
        viewWidth = measuredWidth
        viewHeight = measuredHeight

        // Generate other ships randomly
        if (time % 700 < 10 + speeds) {
            val map = HashMap<String, Any>()
            map["lane"] = (0 ≤ .. ≤ 2).random()
            map["startTime"] = time
            police.add(map)
        }

        // Update game time
        time += 10 + speeds

        // Set up drawing properties
        myPaint.style = Paint.Style.FILL

        // Draw the player's ship
        val shipWidth = viewWidth / 5
        val shipHeight = shipWidth + 10
```

```
val d = resources.getDrawable(R.drawable.thief,  theme: null)
d.setBounds(
    left: myThiefPosition * viewWidth / 3 + viewWidth / 15 + 25,
    top: viewHeight - 2 - shipHeight,
    right: myThiefPosition * viewWidth / 3 + viewWidth / 15 + shipWidth - 25,
    bottom: viewHeight - 2
)
d.draw(canvas)
myPaint.color = Color.GREEN
var highScore = getHighScore()

for (i in police.indices) {
    try {
        val shipX = police[i]["lane"] as Int * viewWidth / 3 + viewWidth / 15
        val shipY = time - police[i]["startTime"] as Int
        val d2 = resources.getDrawable(R.drawable.polic,  theme: null)

        d2.setBounds(
            left: shipX + 25,  top: shipY - shipHeight,  right: shipX + shipWidth - 25, shipY
        )
        d2.draw(canvas)
        if (police[i]["lane"] as Int == myThiefPosition) {
            if (shipY > viewHeight - 2 - shipHeight && shipY < viewHeight - 2) {
                gameTask.closeGame(score)
            }
        }
```

```
            if (shipY > viewHeight + shipHeight) {
                police.removeAt(i)
                score++
                speeds = 1 + Math.abs(score / 8)
                if (score > highScore) {
                    highScore = score
                    saveHighScore(highScore)
                }
            }

        } catch (e: Exception) {
            e.printStackTrace()
        }
    }
    myPaint.color = Color.WHITE
    myPaint.textSize = 40f
    canvas.drawText( text: "Score : $score",  x: 80f,  y: 80f, myPaint)
    canvas.drawText( text: "High Score : $highScore",  x: 80f,  y: 140f, myPaint)
    canvas.drawText( text: "Speed : $speeds",  x: 380f,  y: 80f, myPaint)
    invalidate()
}

@SuppressLint("ClickableViewAccessibility")
override fun onTouchEvent(event: MotionEvent?): Boolean {
    when (event!!.action) {
        MotionEvent.ACTION_DOWN -> {
            val x1 = event.x
            if (x1 < viewWidth / 2) {
                if (myThiefPosition > 0) {
```

```
                if (x1 < viewWidth / 2) {
                    if (myThiefPosition > 0) {
                        myThiefPosition--
                    }
                }
                if (x1 > viewWidth / 2) {
                    if (myThiefPosition < 2) {
                        myThiefPosition++
                    }
                }
                invalidate() // Redraw the view after updating ship position
            }
            MotionEvent.ACTION_UP -> {

            }

        }
        return true
    }

    private fun saveHighScore(score: Int) {
        preferences.edit().putInt("HighScore", score).apply()
    }

    private fun getHighScore(): Int {
        return preferences.getInt( key: "HighScore",  defValue: 0)
    }
}
```

GameTask.kt

```kotlin
package com.example.new_game



interface GameTask {
fun closeGame(mScore:Int)
}
```

Colors.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="button_colour">#FD397C</color>
    <color name="background_color">#FD397C</color>

</resources>
```

Strings.xml

```xml
<resources>
    <string name="app_name">New_Game</string>
    <string name="high_score_0">High Score: 0</string>
    <string name="score_0">Score : 0</string>
    <string name="start">Start</string>
</resources>
```