FACULTY OF COMPUTING

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

2023 – Lab Exam 03 Report

Student ID	IT22105134
Batch	WD 1.1
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:

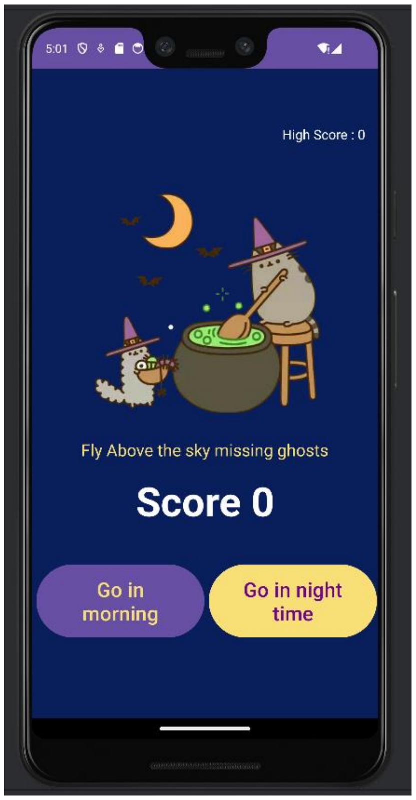
Title: Pusheen Quest

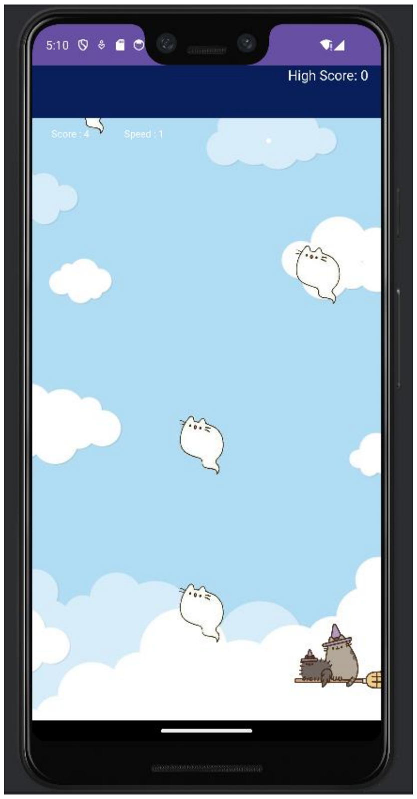
This mobile application is a mobile game that takes players on an adventure of going forward through ghosts. players control a pusheen cat character as they navigate through ghosts as challenges, dodging ghosts and striving to achieve the highest score possible. Pusheen quest game has a simple yet interactive and cute interface with tow game modes; light view (morning view) and dark view (night view). And also app is capable of saving the high score through a singleton object

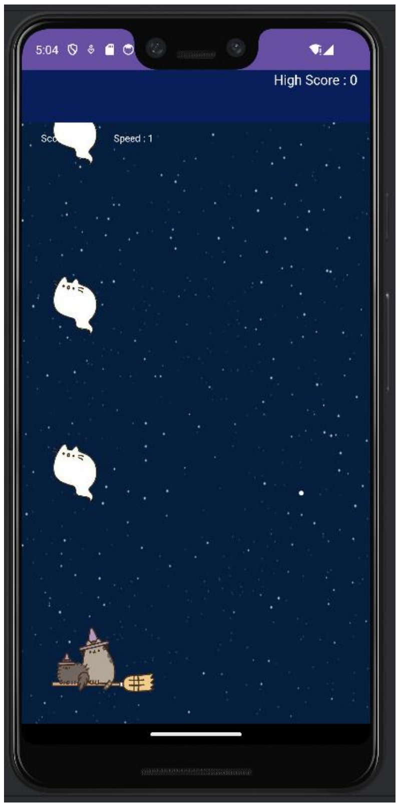
Instructions:

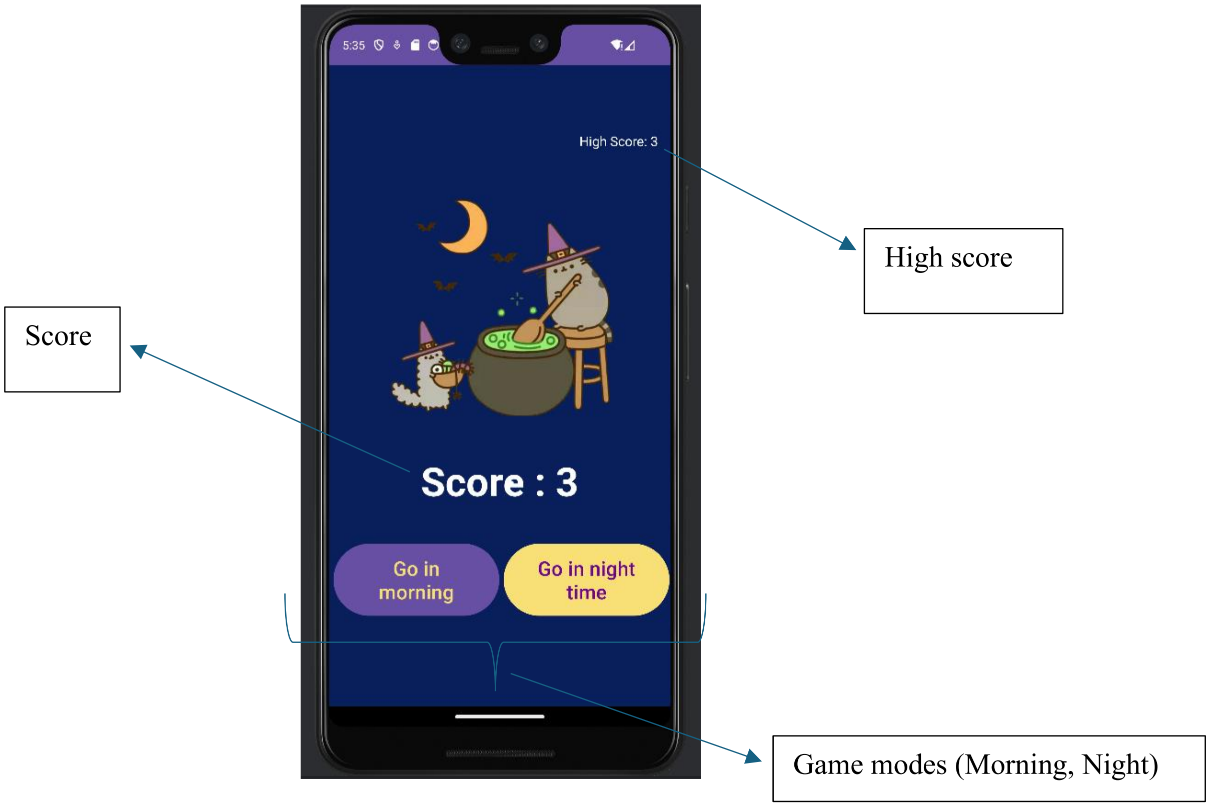
1. Launch the game Pusheen quest.
2. Choose a desired game mode (morning, night).
3. Play game.
4. After loosing high score will be displayed and allows user to start game again by choosing a mode

Screenshots:









Codes

Activity_main.xml

```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:id="@+id/rootLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#081f5c"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/highscore"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:layout_marginRight="15dp"
        android:layout_marginBottom="40dp"
        android:gravity="right"
        android:text="High Score : 0"
        android:textColor="#ffffff"
        android:textSize="16sp" />

    <ImageView
        android:id="@+id/image"
        android:layout_width="273dp"
        android:layout_height="291dp"
        android:layout_marginBottom="20sp"
        app:srcCompat="@drawable/maincat" />

    <TextView
        android:id="@+id/gametext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Fly Above the sky missing ghosts"
        android:textColor="#f9e076"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/score"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="15dp"
        android:text="Score 0"
        android:textAlignment="center"
        android:textColor="#ffffff"
        android:textSize="48sp"
        android:textStyle="bold" />
    <LinearLayout
        android:id="@+id/layout2"
        android:layout_width="match_parent"
        android:layout_height="145dp"
        android:orientation="horizontal">

        <Button
            android:id="@+id/button"
            android:layout_width="200dp"
            android:layout_height="95dp"
            android:layout_gravity="center"
            android:layout_marginStart="5dp"
            android:gravity="center"
            android:text="Go in morning"
            android:textColor="#f9e076"
            android:textSize="24sp" />

        <Button
            android:id="@+id/startBtn1"
            android:layout_width="200dp"
            android:layout_height="95dp"
            android:layout_gravity="center"
            android:layout_marginStart="5dp"
            android:text="Go in night time"
            android:textColor="#710393"
            android:textSize="24sp"
            app:backgroundTint="#f9e076" />
    </LinearLayout>

</LinearLayout>
```

MainActivity.kt

```
package com.example.mygame

//MainActivity.kt
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.TextView

class MainActivity : AppCompatActivity(),GameTask {
    lateinit var rootLayout : LinearLayout
    lateinit var layout2 : LinearLayout
    lateinit var startBtn1 : Button
    lateinit var startBtn2 : Button
    lateinit var mGameView : GameView
    lateinit var mGameView2 : GameView2
    lateinit var score : TextView
```

```kotlin
lateinit var image: ImageView
lateinit var gametext: TextView
lateinit var highscore: TextView

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

    startBtn1 = findViewById(R.id.startBtn1)
    startBtn2 = findViewById(R.id.button)
    rootLayout = findViewById(R.id.rootLayout)
    layout2 = findViewById(R.id.layout2)
    score = findViewById(R.id.score)
    image = findViewById(R.id.image)
    gametext = findViewById(R.id.gametext)
    highscore = findViewById(R.id.highscore)

    HighScoreManager.initialize(applicationContext)

    startBtn1.setOnClickListener{
        mGameView = GameView(this, this) // Initialize the GameView here
        mGameView.setBackgroundResource(R.drawable.nightsky3)
        rootLayout.addView(mGameView)
        startBtn1.visibility = View.GONE
        layout2.visibility = View.GONE
        score.visibility = View.GONE
        image.visibility = View.GONE
        gametext.visibility = View.GONE
        startBtn2.visibility = View.GONE
    }

    startBtn2.setOnClickListener{
        mGameView = GameView(this, this) // Initialize the GameView here
        mGameView.setBackgroundResource(R.drawable.sky1)
        rootLayout.addView(mGameView)
        startBtn1.visibility = View.GONE
        score.visibility = View.GONE
        image.visibility = View.GONE
        gametext.visibility = View.GONE
        startBtn2.visibility = View.GONE
        layout2.visibility = View.GONE
    }
}



override fun closeGame(mScore: Int) {
    val currentHighScore = HighScoreManager.getHighScore()
    if (mScore > currentHighScore) {
        HighScoreManager.updateHighScore(mScore)
    }
    score.text = "Score : $mScore"
    highscore.text = "High Score: ${HighScoreManager.getHighScore()}"
    HighScoreManager.updateHighScore(mScore)
    rootLayout.removeView(mGameView)
    startBtn1.visibility = View.VISIBLE
```

```
        score.visibility = View.VISIBLE
        image.visibility = View.VISIBLE
        gametext.visibility = View.GONE
        startBtn2.visibility = View.VISIBLE
        layout2.visibility = View.VISIBLE
    }

    override fun closeGame2(mScore: Int) {
        val currentHighScore = HighScoreManager.getHighScore()
        if (mScore > currentHighScore) {
            HighScoreManager.updateHighScore(mScore)
        }
        score.text = "Score : $mScore"
        highscore.text = "High Score: ${HighScoreManager.getHighScore()}"
        HighScoreManager.updateHighScore(mScore)
        rootLayout.removeView(mGameView2)
        startBtn1.visibility = View.VISIBLE
        score.visibility = View.VISIBLE
        image.visibility = View.VISIBLE
        gametext.visibility = View.GONE
        startBtn2.visibility = View.VISIBLE
        layout2.visibility = View.VISIBLE
    }

}
```

PlayGame.kt

```
package com.example.mygame

//GameView.kt
import android.content.Context
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.view.MotionEvent
import android.view.View

class GameView(var c :Context, var gameTask: GameTask):View(c)
{
    private var myPaint: Paint? = null
    private var speed = 1
    private var time = 0
    private var score = 0
    private var myCatPosition = 0
    private val ghostCats = ArrayList<HashMap<String,Any>>()

    var viewWidth = 0
    var viewHeight = 0
    init {
        myPaint = Paint()
```

```
}

override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    viewWidth = this.measuredWidth
    viewHeight = this.measuredHeight

    // Calculate the width and height of the witchcat
    val witchcatWidth = viewWidth / 3
    var witchcatHeight = witchcatWidth + 10

    // Calculate the smaller size for the ghostcat
    val ghostcatWidth = witchcatWidth / 2
    val ghostcatHeight = witchcatHeight / 2

    if(time % 700 < 10 +speed){
        val map = HashMap<String,Any>()
        map["lane"] = (0..2).random()
        map["startTime"] = time
        ghostCats.add(map)
    }
    time = time + 10 + speed

    myPaint!!.style = Paint.Style.FILL
    val d = resources.getDrawable(R.drawable.witchcat,null)

    d.setBounds(
        myCatPosition * viewWidth / 3 + viewWidth / 15 + 25,
        viewHeight-2 - witchcatHeight,
        myCatPosition * viewWidth / 3 + viewWidth / 15 + witchcatWidth - 25 ,
        viewHeight - 2
    )
    d.draw(canvas!!)
    myPaint!!.color = Color.GREEN
    var highScore = 0

    for (i in ghostCats.indices){
        try {
            val ghostcatX = ghostCats[i]["lane"] as Int * viewWidth / 3 + viewWidth / 15
            var ghostcatY =  time - ghostCats[i]["startTime"] as Int
            val d2 = resources.getDrawable(R.drawable.ghostcat,null)

            // Draw the ghost cat with smaller dimensions
            d2.setBounds(
                ghostcatX + 25 , ghostcatY - ghostcatHeight , ghostcatX + ghostcatWidth - 25 , ghostcatY
            )
            d2.draw(canvas)
            if (ghostCats[i]["lane"] as Int == myCatPosition){
                if (ghostcatY > viewHeight - 2 - witchcatHeight
                    && ghostcatY < viewHeight - 2 ){

                    gameTask.closeGame(score)
                }
            }
            if (ghostcatY > viewHeight + witchcatHeight)
            {
                    ghostCats.removeAt(i)
                    score++
                    speed = 1 + Math.abs(score / 8)
                    if (score > highScore){
                        highScore = score
                    }
                }
            }
            catch (e:Exception){
                e.printStackTrace()
            }
        }
        myPaint!!.color = Color.WHITE
        myPaint!!.textSize = 40f
        canvas.drawText("Score : $score",80f,80f,myPaint!!)
        canvas.drawText("Speed : $speed",380f,80f,myPaint!!)
        invalidate()
    }

    override fun onTouchEvent(event: MotionEvent?): Boolean {
        when(event!!.action){
            MotionEvent.ACTION_DOWN ->{
                val x1 = event.x
                if (x1 < viewWidth/2){
                    if (myCatPosition> 0){
                        myCatPosition--
                    }
                }
                if (x1 > viewWidth / 2){
                    if (myCatPosition<2){
                        myCatPosition++
                    }
                }
                invalidate()
            }
            MotionEvent.ACTION_UP ->{}
        }
        return true
    }

}
```

PlayGame2.kt

```
package com.example.mygame

//GameView.kt
import android.content.Context
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.view.MotionEvent
import android.view.View
```

```kotlin
class GameView2(var c :Context, var gameTask: GameTask):View(c)
{
    private var myPaint: Paint? = null
    private var speed = 1
    private var time = 0
    private var score = 0
    private var myCatPosition = 0
    private val ghostCats = ArrayList<HashMap<String,Any>>()

    var viewWidth = 0
    var viewHeight = 0
    init {
        myPaint = Paint()
    }

    override fun onDraw(canvas: Canvas) {
        super.onDraw(canvas)
        viewWidth = this.measuredWidth
        viewHeight = this.measuredHeight

        // Calculate the width and height of the witchcat
        val witchcatWidth = viewWidth / 3
        var witchcatHeight = witchcatWidth + 10

        // Calculate the smaller size for the ghostcat
        val ghostcatWidth = witchcatWidth / 2
        val ghostcatHeight = witchcatHeight / 2

        if(time % 700 < 10 +speed){
            val map = HashMap<String,Any>()
            map["lane"] = (0..2).random()
            map["startTime"] = time
            ghostCats.add(map)
        }
        time = time + 10 + speed

        myPaint!!.style = Paint.Style.FILL
        val d = resources.getDrawable(R.drawable.flyingcat,null)

        d.setBounds(
            myCatPosition * viewWidth / 3 + viewWidth / 15 + 25,
            viewHeight-2 - witchcatHeight,
            myCatPosition * viewWidth / 3 + viewWidth / 15 + witchcatWidth - 25 ,
            viewHeight - 2
        )
        d.draw(canvas!!)
        myPaint!!.color = Color.GREEN
        var highScore = 0

        for (i in ghostCats.indices){
            try {
                val ghostcatX = ghostCats[i]["lane"] as Int * viewWidth / 3 + viewWidth / 15
                var ghostcatY =  time - ghostCats[i]["startTime"] as Int
                val d2 = resources.getDrawable(R.drawable.fish,null)
                // Draw the ghost cat with smaller dimensions
                d2.setBounds(
                    ghostcatX + 25 , ghostcatY - ghostcatHeight , ghostcatX + ghostcatWidth - 25 , ghostcatY
                )
                d2.draw(canvas)
                if (ghostCats[i]["lane"] as Int == myCatPosition){
                    if (ghostcatY > viewHeight - 2 - witchcatHeight
                        && ghostcatY < viewHeight - 2 ){

                        gameTask.closeGame2(score)
                    }
                }
                if (ghostcatY > viewHeight + witchcatHeight)
                {
                    ghostCats.removeAt(i)
                    score++
                    speed = 1 + Math.abs(score / 8)
                    if (score > highScore){
                        highScore = score
                    }
                }
            }
            catch (e:Exception){
                e.printStackTrace()
            }
        }
        myPaint!!.color = Color.WHITE
        myPaint!!.textSize = 40f
        canvas.drawText("Score : $score",80f,80f,myPaint!!)
        canvas.drawText("Speed : $speed",380f,80f,myPaint!!)
        invalidate()
    }

    override fun onTouchEvent(event: MotionEvent?): Boolean {
        when(event!!.action){
            MotionEvent.ACTION_DOWN ->{
                val x1 = event.x
                if (x1 < viewWidth/2){
                    if (myCatPosition> 0){
                        myCatPosition--
                    }
                }
                if (x1 > viewWidth / 2){
                    if (myCatPosition<2){
                        myCatPosition++
                    }
                }
                invalidate()
            }
            MotionEvent.ACTION_UP ->{}
        }
        return true
    }

}
```

GameIdea.kt

```kotlin
package com.example.mygame

//GameTask.kt
interface GameTask {
    fun closeGame(mScore:Int)
    fun closeGame2(mScore:Int)
}
```

highScore.kt

```kotlin
import android.content.Context
import android.content.SharedPreferences

object HighScoreManager {
    private const val HIGH_SCORE_KEY = "high_score"
    private lateinit var sharedPreferences: SharedPreferences
    private var isInitialized = false

    fun initialize(context: Context) {
        sharedPreferences = context.getSharedPreferences("HighScorePref", Context.MODE_PRIVATE)
        isInitialized = true
    }

    private fun checkInitialized() {
        if (!isInitialized) {
            throw IllegalStateException("HighScoreManager must be initialized before use")
        }
    }

    fun updateHighScore(score: Int) {
        checkInitialized()
        val editor = sharedPreferences.edit()
        editor.putInt(HIGH_SCORE_KEY, score)
        editor.apply()
    }

    fun getHighScore(): Int {
        checkInitialized()
        return sharedPreferences.getInt(HIGH_SCORE_KEY, 0)
    }
}
```