PROJECT - SNAKE GAME

AIM: To develop and launch snake game using Android studio.

ABSTRACT:

The Snake Game in Android Studio is a classic arcade-style mobile game that provides entertainment and challenges to users of all ages. This project involves the development of a mobile application that allows players to control a virtual snake on the screen, guiding it to consume food items to grow in length while avoiding collisions with the walls and its own body. The game leverages Android Studio's features and functionalities to create a visually appealing and interactive user experience.

The primary objective of this project is to create a user-friendly and engaging game interface that integrates seamlessly with Android devices. Through continuous interaction, players must strategize their movements to collect food items that appear randomly on the grid, with each consumed item leading to the snake's growth and an increase in the player's score.

The Snake Game's development involves various components, including user interface design, game mechanics implementation, collision detection, scoring system, and sound effects integration. The Android Studio environment offers the necessary tools to design graphical elements, define user input handling, and manage the game's logic. Furthermore, the project emphasizes the

importance of optimizing performance and responsiveness to ensure smooth gameplay even on devices with varying hardware capabilities.

Key features of the Snake Game include:

Intuitive Controls: The game incorporates intuitive touch and swipe controls, allowing players to guide the snake in various directions with ease.

Dynamic Gameplay: As the snake grows longer with each food item consumed, the challenge increases, requiring players to plan their movements carefully to prevent collisions.

Scoring System: Players are awarded points for each food item eaten. The game maintains a scoreboard to track and display the highest scores achieved.

Visual Feedback: Visual effects and animations enhance the gaming experience, providing feedback for successful actions and collisions.

Game Over Conditions: The game includes conditions that lead to a "game over" state, such as the snake colliding with the walls or its own body. Upon game over, players can view their final score and attempt to beat their previous records.

In conclusion, the Snake Game developed in Android Studio showcases the successful implementation of a classic arcade game on modern mobile platforms. By combining creative design, efficient programming, and user-centered development, this project offers an entertaining and nostalgic gaming experience that highlights the capabilities of Android Studio for game development.

CODE:

MainActivity.kt

```
package com.example.gamemode

import android.app.Activity
import android.os.Bundle
import android.os.Handler
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.RelativeLayout
import android.widget.Toast
import java.util.*
import kotlin.math.pow
import kotlin.math.sqrt

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

        val board = findViewById<RelativeLayout>(R.id.board)
        val border = findViewById<RelativeLayout>(R.id.relativeLayout)
        val lilu = findViewById<LinearLayout>(R.id.lilu)
        val upButton = findViewById<Button>(R.id.up)
        val downButton = findViewById<Button>(R.id.down)
        val leftButton = findViewById<Button>(R.id.left)
        val rightButton = findViewById<Button>(R.id.right)
        val pauseButton = findViewById<Button>(R.id.pause)
        val newgame = findViewById<Button>(R.id.new_game)
        val resume = findViewById<Button>(R.id.resume)
        val playagain = findViewById<Button>(R.id.playagain)
```

```
val score = findViewById<Button>(R.id.score)
val score2 = findViewById<Button>(R.id.score2)
val meat = ImageView(this)
val snake = ImageView(this)
val snakeSegments =
    mutableListOf(snake) // Keep track of the position of each snake segment
val handler = Handler()
var delayMillis = 30L // Update snake position every 100 milliseconds
var currentDirection = "right" // Start moving right by default
var scorex = 0
board.visibility = View.INVISIBLE
playagain.visibility = View.INVISIBLE
score.visibility = View.INVISIBLE
score2.visibility = View.INVISIBLE
newgame.setOnClickListener {

    board.visibility = View.VISIBLE
    newgame.visibility = View.INVISIBLE
    resume.visibility = View.INVISIBLE
    score2.visibility = View.VISIBLE

    snake.setImageResource(R.drawable.snake)
    snake.layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.WRAP_CONTENT,
        ViewGroup.LayoutParams.WRAP_CONTENT
    )
    board.addView(snake)
    snakeSegments.add(snake) // Add the new snake segment to the list

    var snakeX = snake.x
    var snakeY = snake.y

    meat.setImageResource(R.drawable.meat)
    meat.layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.WRAP_CONTENT,
        ViewGroup.LayoutParams.WRAP_CONTENT
    )
```

```
board.addView(meat)
val random = Random() // create a Random object
val randomX =
    random.nextInt(801) - 400 // generate a random x-coordinate between -400 and 400
val randomY =
    random.nextInt(801) - 400 // generate a random y-coordinate between -400 and 400

meat.x = randomX.toFloat()
meat.y = randomY.toFloat()

fun checkFoodCollision() {
    val distanceThreshold = 50
    val distance = sqrt((snake.x - meat.x).pow(2) + (snake.y - meat.y).pow(2))
    if (distance < distanceThreshold) {
        val newSnake =
            ImageView(this) // Create a new ImageView for the additional snake segment
        newSnake.setImageResource(R.drawable.snake)
        newSnake.layoutParams = ViewGroup.LayoutParams(
            ViewGroup.LayoutParams.WRAP_CONTENT,
            ViewGroup.LayoutParams.WRAP_CONTENT
        )
        board.addView(newSnake)
        snakeSegments.add(newSnake) // Add the new snake segment to the list
        val randomX =
            random.nextInt(801) - -100
        val randomY =
            random.nextInt(801) - -100
        meat.x = randomX.toFloat()
        meat.y = randomY.toFloat()
        delayMillis-- // Reduce delay value by 1
        scorex++
        score2.text = "score : " + scorex.toString() // Update delay text view
    }
}
val runnable = object : Runnable {
    override fun run() {
```

```
                for (i in snakeSegments.size - 1 downTo 1) { // Update the position of each snake segment except for the head
                    snakeSegments[i].x = snakeSegments[i - 1].x
                    snakeSegments[i].y = snakeSegments[i - 1].y
                }
                when (currentDirection) {
                    "up" -> {
                        snakeY -= 10
                        if (snakeY < -490) { // Check if the ImageView goes off the top of the board
                            snakeY = -490f
                            border.setBackgroundColor(getResources().getColor(R.color.red))
                            playagain.visibility = View.VISIBLE
                            currentDirection = "pause"
                            lilu.visibility = View.INVISIBLE
                            score.text =
                                "your score is  " + scorex.toString() // Update delay text view
                            score.visibility = View.VISIBLE
                            score2.visibility = View.INVISIBLE
                        }
                        snake.translationY = snakeY
                    }
                    "down" -> {
                        snakeY += 10
                        val maxY =
                            board.height / 2 - snake.height + 30
                        if (snakeY > maxY) {
                            snakeY = maxY.toFloat()
                            border.setBackgroundColor(getResources().getColor(R.color.red))
                            playagain.visibility = View.VISIBLE
                            currentDirection = "pause"
                            lilu.visibility = View.INVISIBLE
                            score.text =
                                "your score is  " + scorex.toString() // Update delay text view
                            score.visibility = View.VISIBLE
                            score2.visibility = View.INVISIBLE
                        }
                        snake.translationY = snakeY
```

```
}
"left" -> {
    snakeX -= 10
    if (snakeX < -490) { // Check if the ImageView goes off the top of the board
        snakeX = -490f
        border.setBackgroundColor(getResources().getColor(R.color.red))
        playagain.visibility = View.VISIBLE
        currentDirection = "pause"
        lilu.visibility = View.INVISIBLE
        score.text =
            "your score is  " + scorex.toString() // Update delay text view
        score.visibility = View.VISIBLE
        score2.visibility = View.INVISIBLE
    }
    snake.translationX = snakeX
}
"right" -> {
    snakeX += 10
    val maxX =
        board.height / 2 - snake.height + 30 // Calculate the maximum y coordinate
    if (snakeX > maxX) {
        snakeX = maxX.toFloat()
        border.setBackgroundColor(getResources().getColor(R.color.red))
        playagain.visibility = View.VISIBLE
        currentDirection = "pause"
        lilu.visibility = View.INVISIBLE
        score.text =
            "your score is  " + scorex.toString() // Update delay text view
        score.visibility = View.VISIBLE
        score2.visibility = View.INVISIBLE
    }
    snake.translationX = snakeX
}
"pause" -> {
    snakeX += 0
    snake.translationX = snakeX
}
```

```
            }
            checkFoodCollision()
            handler.postDelayed(this, delayMillis)
        }
    }
    handler.postDelayed(runnable, delayMillis)
// Set button onClickListeners to update the currentDirection variable when pressed
    upButton.setOnClickListener {
        currentDirection = "up"
    }
    downButton.setOnClickListener {
        currentDirection = "down"
    }
    leftButton.setOnClickListener {
        currentDirection = "left"
    }
    rightButton.setOnClickListener {
        currentDirection = "right"
    }
    pauseButton.setOnClickListener {
        currentDirection = "pause"
        board.visibility = View.INVISIBLE
        newgame.visibility = View.VISIBLE
        resume.visibility = View.VISIBLE
    }
    resume.setOnClickListener {
        currentDirection = "right"
        board.visibility = View.VISIBLE
        newgame.visibility = View.INVISIBLE
        resume.visibility = View.INVISIBLE
    }
    playagain.setOnClickListener {
        recreate()
    }
  }
 }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/score2"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:background="@color/green"
        android:textStyle="bold"
        android:textColor="@color/black"
        android:textSize="15dp"
        android:text="score"
        app:layout_constraintBottom_toTopOf="@+id/lilu"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent">
    </Button>

    <RelativeLayout
        android:id="@+id/relativeLayout"
        android:layout_width="390dp"
        android:layout_height="390dp"
        android:gravity="center"
        android:background="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:ignore="MissingConstraints">

        <Button
            android:id="@+id/score"
            android:background="@color/black"
```

```
    android:layout_width="wrap_content"
    android:textAlignment="center"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_marginEnd="120dp"
    android:layout_marginTop="140dp"
    android:textSize="15dp"
    android:text="Game Over ! Play Again">
</Button>

<Button
    android:id="@+id/new_game"
    android:background="@color/purple_700"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_marginEnd="130dp"
    android:layout_marginTop="130dp"
    android:text="New Game">
</Button>

<Button
    android:id="@+id/resume"
    android:background="@color/purple_700"
    android:layout_width="150dp"
    android:textAlignment="center"
    android:layout_height="wrap_content"
    android:layout_below="@id/new_game"
    android:layout_alignParentEnd="true"
    android:layout_marginEnd="130dp"
    android:layout_marginTop="10dp"
    android:text="Resume Game">
</Button>

<Button
    android:id="@+id/playagain"
    android:background="@color/red"
```

```
        android:layout_width="150dp"
        android:textAlignment="center"
        android:layout_height="80dp"
        android:layout_below="@id/resume"
        android:layout_alignParentEnd="true"
        android:layout_marginEnd="120dp"
        android:layout_marginTop="60dp"
        android:textSize="20dp"
        android:textStyle="bold"
        android:text="Game Over ! Play Again">
    </Button>

    <RelativeLayout
        android:id="@+id/board"
        android:layout_width="380dp"
        android:layout_height="380dp"
        android:background="@color/black"
        android:gravity="center"
        android:layout_marginLeft="5dp"
        tools:ignore="MissingConstraints">
    </RelativeLayout>


</RelativeLayout>

<LinearLayout
    android:id="@+id/lilu"
    android:layout_width="330dp"
    android:layout_height="330dp"
    android:orientation="vertical"
    android:layout_alignParentBottom="true"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    tools:ignore="MissingConstraints">
```

```
<LinearLayout
    android:gravity="center"
    android:layout_width="match_parent"
    android:layout_height="100dp">

    <Button
        android:id="@+id/up"
        android:layout_margin="10dp"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_gravity="center"
        android:gravity="center"
        android:text="UP">
    </Button>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="100dp">

    <Button
        android:id="@+id/left"
        android:layout_margin="10dp"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_gravity="center"
        android:gravity="center"
        android:text="LEFT">
    </Button>

    <Button
        android:id="@+id/pause"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_gravity="center"
        android:gravity="center"
        android:text="pause">
```

```
        </Button>

        <Button
            android:id="@+id/right"
            android:layout_margin="10dp"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:layout_gravity="center"
            android:gravity="center"
            android:text="RIGHT">
        </Button>

    </LinearLayout>

    <LinearLayout
        android:gravity="center"
        android:layout_width="match_parent"
        android:layout_height="100dp">

        <Button
            android:id="@+id/down"
            android:layout_margin="10dp"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:layout_gravity="center"
            android:gravity="center"
            android:text="DOWN">
        </Button>
    </LinearLayout>
  </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Snake.xml

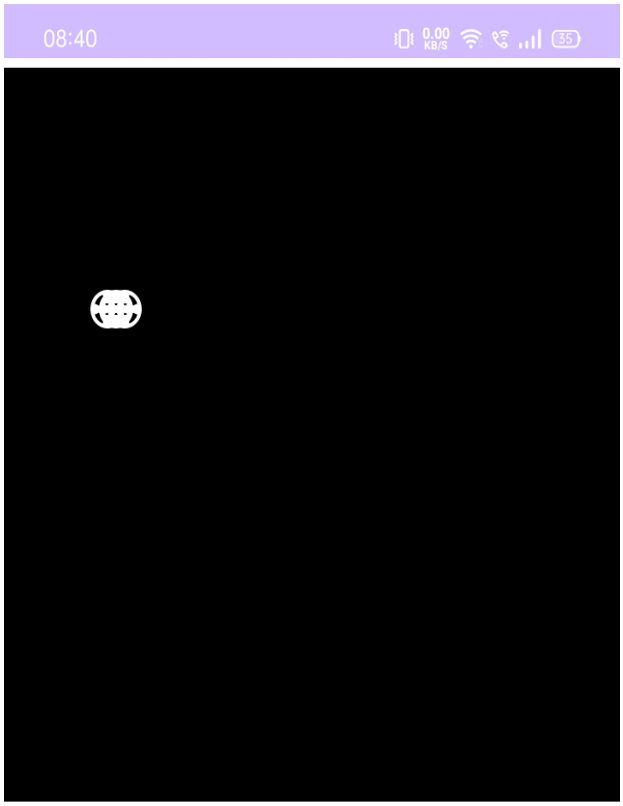
```
<vector android:height="24dp" android:tint="#FFFFFF"
```

```
   android:viewportHeight="24" android:viewportWidth="24"
   android:width="24dp"
 xmlns:android="http://schemas.android.com/apk/res/android">

   <path android:fillColor="@android:color/white"
  android:pathData="M12,2C6.48,2 2,6.48 2,12c0,5.52 4.48,10 10,10s10,-4.48 10,-10C22,6.48
17.52,2 12,2zM19.46,9.12l-2.78,1.15c-0.51,-1.36 -1.58,-2.44 -2.95,-2.94l1.15,-2.78C16.98,5.35
18.65,7.02 5.35,16.98 4.54,14.87zM14.88,19.46l-1.15,-2.78c1.37,-0.51 2.45,-1.59
2.95,-2.97l2.78,1.17C18.65,16.98 16.98,18.65 14.88,19.46z"/>

</vector>
```

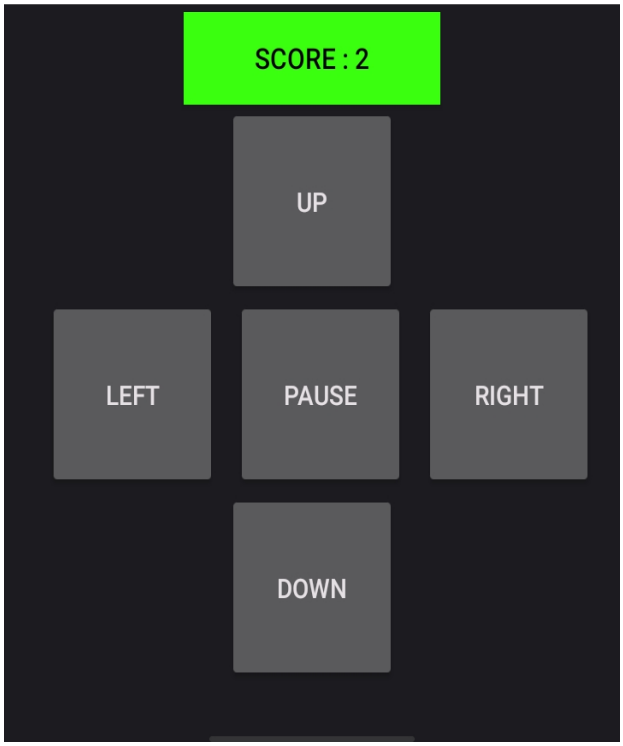
Meat.xml

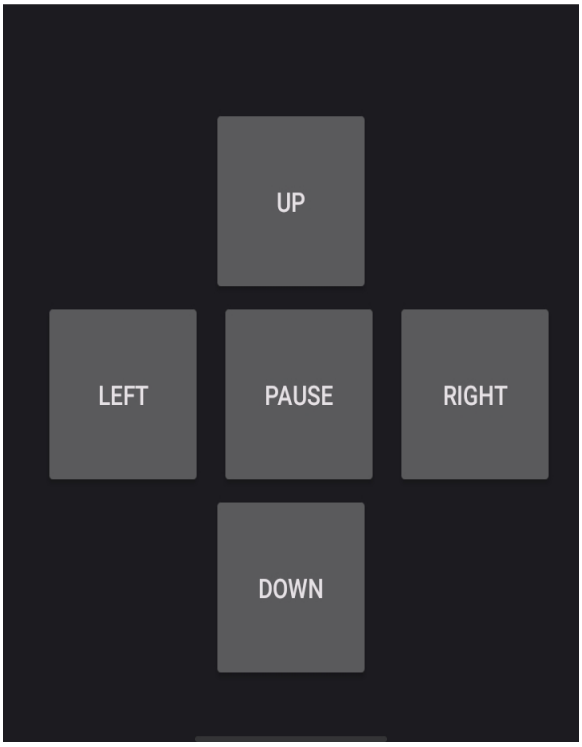
```
<vector android:height="24dp" android:tint="#00FF1E"
   android:viewportHeight="24" android:viewportWidth="24"
   android:width="24dp"
xmlns:android="http://schemas.android.com/apk/res/android">
 <pathandroid:fillColor="@android:color/white"
android:pathData="M15.53,17.46L12,21l-3.53,-3.54C9.37,16.56 10.62,16 12,16S14.63,16.56
15.53,17.46z"/>

</vector>
```

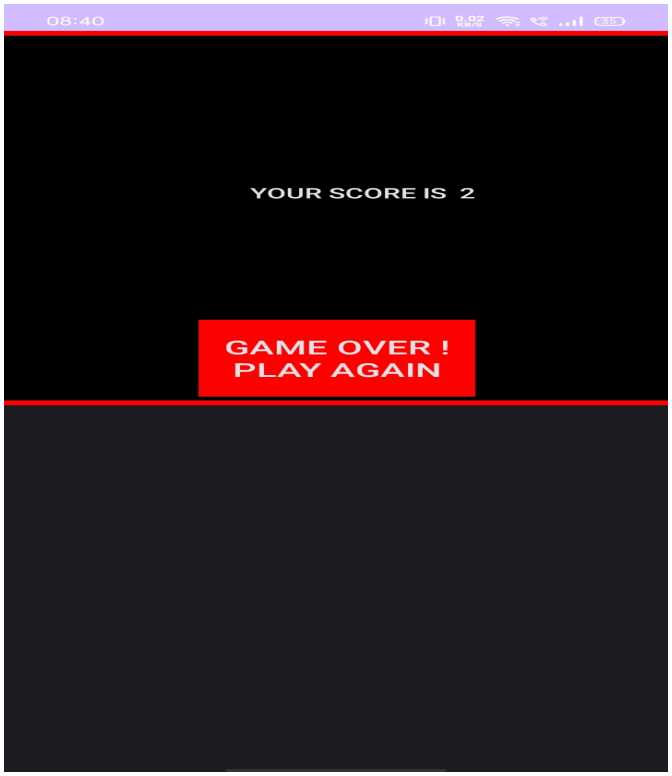
OUTPUT:

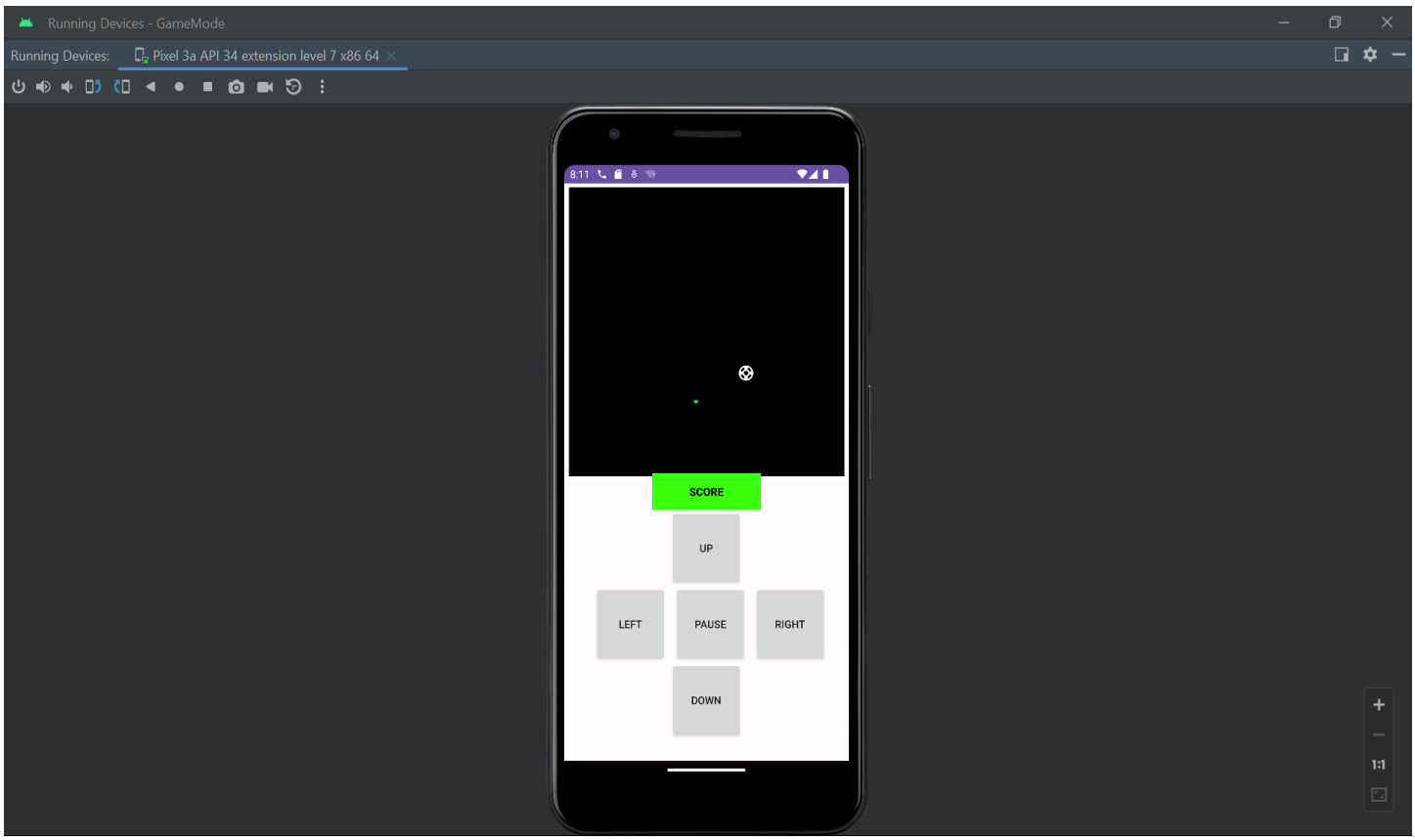

08:40

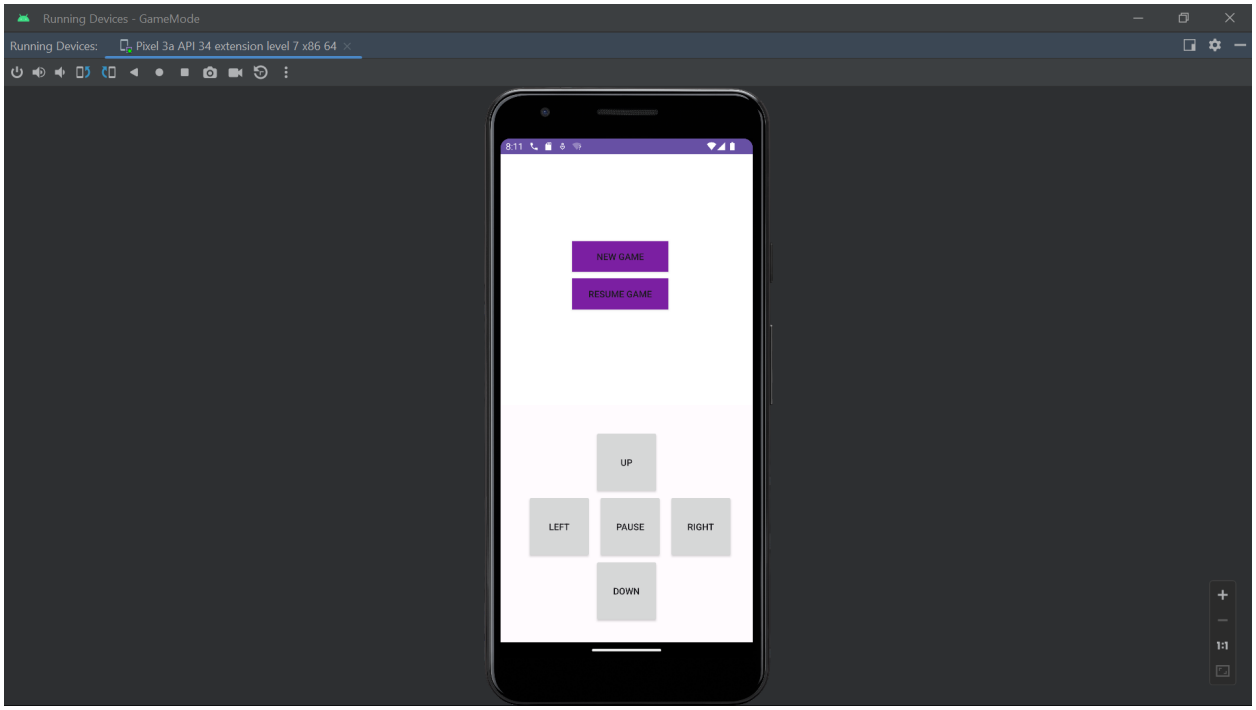

08:39
NEW GAME
RESUME GAME

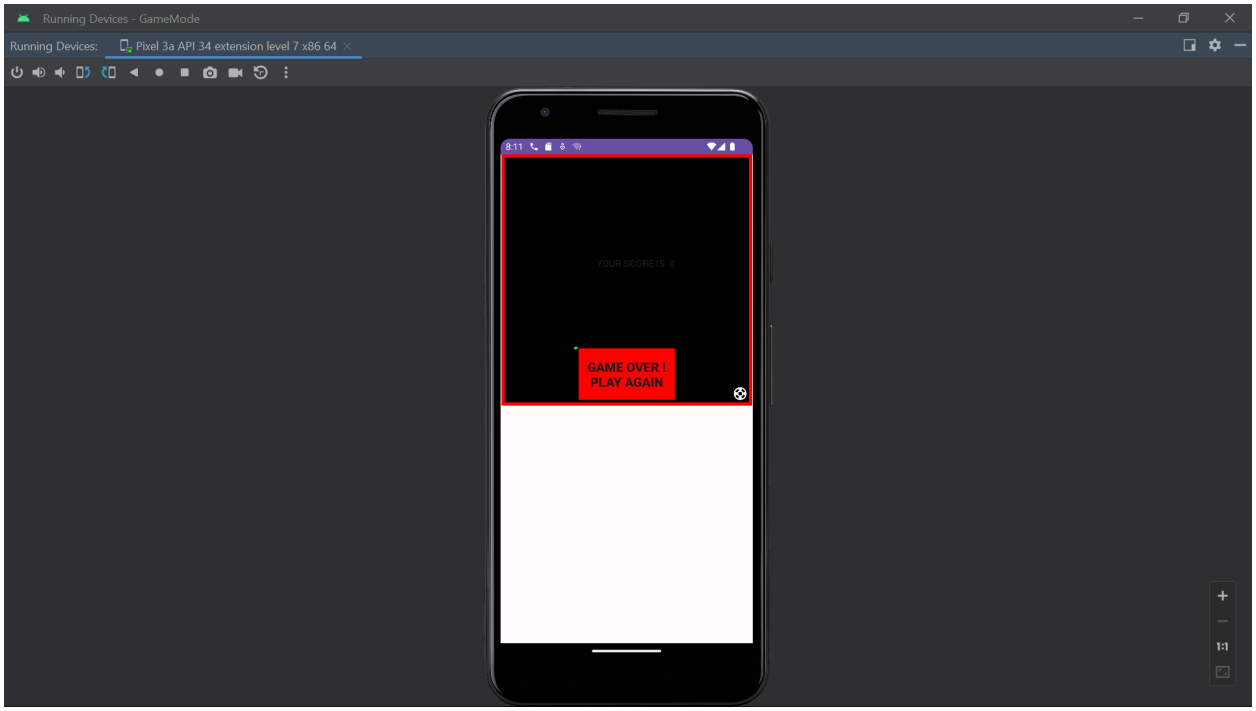

SCORE : 2
UP
LEFT
PAUSE
RIGHT
DOWN


UP
LEFT
PAUSE
RIGHT
DOWN


08:40
YOUR SCORE IS 2
GAME OVER !
PLAY AGAIN


Running Devices - GameMode
Running Devices:
Pixel 3a API 34 extension level 7 x86 64
8:11
SCORE
UP
LEFT
PAUSE
RIGHT
DOWN
1:1





RESULTS: Hence , the snake game was developed and launched successfully.