Національний технічний університет України
«Київський політехнічний інститут імені Ігоря
Сікорського»

Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №6
з дисципліни «Розробка мобільних застосувань під Android»
Тема: «Створення ігрових застосунків»

Виконала:
студентка групи ІМ-23
Косенко Кароліна

Перевірив:
Орленко С. П.

Київ 2025

Мета роботи: дослідити розробку ігрового мобільного застосунку, на прикладі було обрано падаючі предмети та ловля їх у інший предмет з використанням різноманітних текстур для фону та об’єктів.
Лістинг:
MainActivity.kt

```
package com.example.android_6

import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.LinearLayout
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private lateinit var gameView: GameView
    private lateinit var menuLayout: LinearLayout
    private lateinit var playButton: Button
    private lateinit var pauseButton: Button

    private var isPaused = false

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

        gameView = findViewById(R.id.game_view)
        menuLayout = findViewById(R.id.menu_layout)
        playButton = findViewById(R.id.play_button)
        pauseButton = findViewById(R.id.pause_button)

        playButton.setOnClickListener {
            menuLayout.visibility = View.GONE
            pauseButton.visibility = View.VISIBLE
            gameView.resumeGame()
        }
```

```
        pauseButton.setOnClickListener {
            if (!isPaused) {
                gameView.pauseGame()
                isPaused = true
            } else {
                gameView.resumeGame()
                isPaused = false
            }
        }
    }

    override fun onResume() {
        super.onResume()
        if (menuLayout.visibility == View.GONE &&
!isPaused) {
            gameView.resumeGame()
        }
    }

    override fun onPause() {
        super.onPause()
        gameView.pauseGame()
    }
}
```

GameView.kt

```
package com.example.android_6

import android.content.Context
import android.graphics.*
import android.util.AttributeSet
import android.view.MotionEvent
import android.view.SurfaceHolder
import android.view.SurfaceView
import kotlin.random.Random

class GameView(context: Context, attrs:
AttributeSet? = null) :
```

```
    SurfaceView(context, attrs), Runnable {

    private var gameThread: Thread? = null
    @Volatile private var isPlaying = false
    private val paint = Paint()
    private val scorePaint = Paint().apply {
        color = Color.BLACK
        textSize = 64f
        typeface = Typeface.DEFAULT_BOLD
    }

    private val backgroundBitmap: Bitmap =
BitmapFactory.decodeResource(resources,
R.drawable.background)
    private val basketBitmap: Bitmap =
BitmapFactory.decodeResource(resources,
R.drawable.basket)
    private val goodItemBitmap: Bitmap =
BitmapFactory.decodeResource(resources,
R.drawable.good_item)
    private val badItemBitmap: Bitmap =
BitmapFactory.decodeResource(resources,
R.drawable.bad_item)

    private var basketX: Float = 0f
    private var basketY: Float = 0f

    private var score: Int = 0

    private val items = mutableListOf<Item>()

    private var lastItemSpawnTime =
System.currentTimeMillis()
    private val spawnInterval: Long = 2000

    data class Item(
        var x: Float,
        var y: Float,
```

```
        val bitmap: Bitmap,
        val speed: Float,
        val isGood: Boolean
    )

    init {
        holder.addCallback(object :
SurfaceHolder.Callback {
            override fun surfaceCreated(holder:
SurfaceHolder) {
                basketY = height - basketBitmap.height
- 20f
                basketX = (width / 2 -
basketBitmap.width / 2).toFloat()
            }

            override fun surfaceChanged(holder:
SurfaceHolder, format: Int, width: Int, height: Int)
{}
            override fun surfaceDestroyed(holder:
SurfaceHolder) {}
        })
    }

    override fun run() {
        while (isPlaying) {
            update()
            drawGame()
            control()
        }
    }

    private fun update() {
        val currentTime = System.currentTimeMillis()
        if (currentTime - lastItemSpawnTime >
spawnInterval) {
            spawnItem()
            lastItemSpawnTime = currentTime
```

```
        }

        val iterator = items.iterator()
        while (iterator.hasNext()) {
            val item = iterator.next()
            item.y += item.speed

            if (isCollision(
                    basketX, basketY,
basketBitmap.width, basketBitmap.height,
                    item.x, item.y,
item.bitmap.width, item.bitmap.height
                )
            ) {
                if (item.isGood) {
                    score += 10
                } else {
                    score -= 10
                }
                iterator.remove()
            } else if (item.y > height) {
                iterator.remove()
            }
        }
    }

    private fun drawGame() {
        if (holder.surface.isValid) {
            val canvas: Canvas = holder.lockCanvas()

            val destRect = Rect(0, 0, width, height)
            canvas.drawBitmap(backgroundBitmap, null,
destRect, paint)

            canvas.drawBitmap(basketBitmap, basketX,
basketY, paint)

            for (item in items) {
```

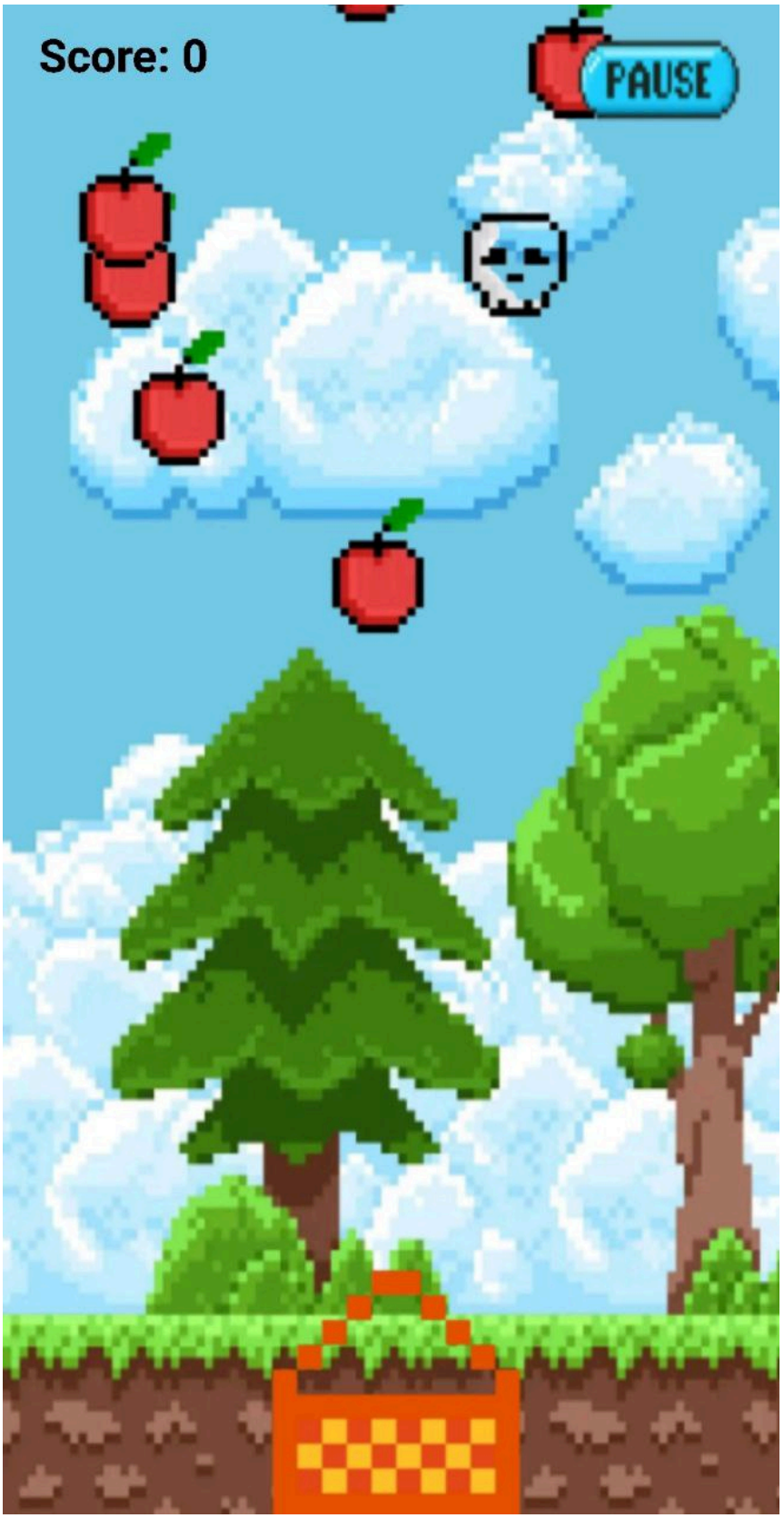
```
                canvas.drawBitmap(item.bitmap,
item.x, item.y, paint)
            }

            canvas.drawText("Score: $score", 50f,
100f, scorePaint)

            holder.unlockCanvasAndPost(canvas)
        }
    }

    private fun control() {
        try {
            Thread.sleep(17)
        } catch (e: InterruptedException) {
            e.printStackTrace()
        }
    }

    private fun spawnItem() {
        val isGood = Random.nextFloat() < 0.7f
        val bitmap = if (isGood) goodItemBitmap else
badItemBitmap
        val x = Random.nextInt(0, (width -
bitmap.width).coerceAtLeast(1)).toFloat()
        val y = -bitmap.height.toFloat()
        val speed = Random.nextInt(5, 15).toFloat()
        items.add(Item(x, y, bitmap, speed, isGood))
    }

    private fun isCollision(x1: Float, y1: Float, w1:
Int, h1: Int,
                            x2: Float, y2: Float, w2:
Int, h2: Int): Boolean {
        return (x1 < x2 + w2 && x1 + w1 > x2 && y1 <
y2 + h2 && y1 + h1 > y2)
    }
```

```
    override fun onTouchEvent(event: MotionEvent):
Boolean {
        when (event.action) {
            MotionEvent.ACTION_DOWN,
MotionEvent.ACTION_MOVE -> {
                basketX = event.x -
basketBitmap.width / 2
                if (basketX < 0) basketX = 0f
                if (basketX > (width -
basketBitmap.width))
                    basketX = (width -
basketBitmap.width).toFloat()
            }
        }
        return true
    }

    fun resumeGame() {
        if (!isPlaying) {
            isPlaying = true
            gameThread = Thread(this)
            gameThread?.start()
        }
    }

    fun pauseGame() {
        if (isPlaying) {
            isPlaying = false
            try {
                gameThread?.join()
            } catch (e: InterruptedException) {
                e.printStackTrace()
            }
        }
    }
}
```

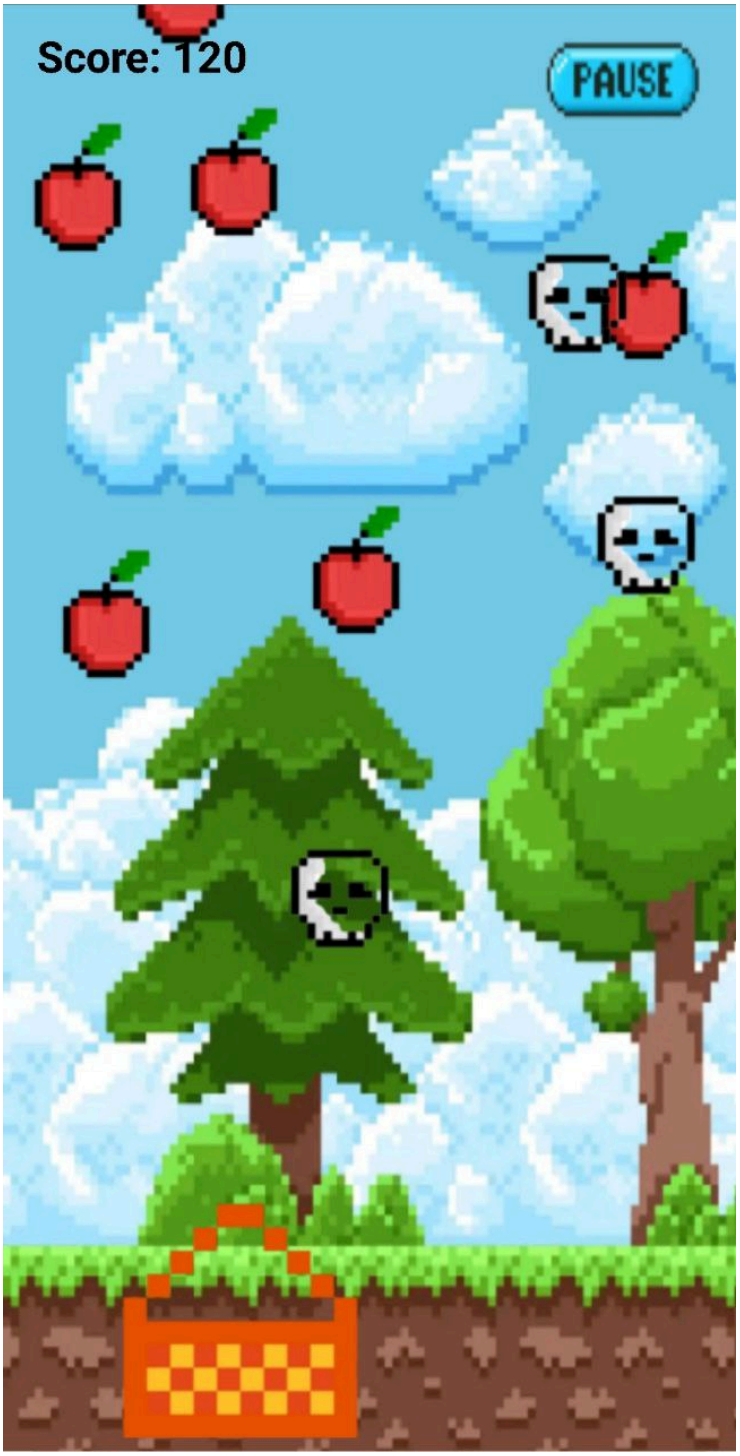
Скріншоти виконання програми:

Меню:

Початок та сама гра:



Після того як трішки назбирати яблук рахунок збільшується на 10, відповідно якщо збирати черепки зменшується на 10. Також присутня пауза, яка зупиняє падіння об’єктів і рух корзинки, нажавши паузу ще раз можна відновити гру.



Висновок:

Створення ігрових застосунків для платформи Android є процесом, що поєднує програмування, дизайн та роботу з мультимедійними ресурсами.

На прикладі гри «Ловля предметів» були розглянуті базові концепції розробки, серед яких реалізація ігрового циклу, обробка взаємодії з користувачем, логіка руху об'єктів та виявлення зіткнень.

Робота з текстурами полягає у завантаженні текстур у ресурси проекту (якщо є готові текстури, інакше потрібно малювати самому, як наприклад кошик намальований вручну, всі інші текстури взяті готові) та подальшому їхньому використанні за допомогою класу Bitmap та Canvas для малювання на екрані. При цьому важливо оптимізувати розміри та кількість текстур, адже це безпосередньо впливає на продуктивність та коректність гри.

Також було розглянуто, як можна створити інтерактивне меню з кнопками, використовуючи текстурні фони, що покращує візуальну привабливість гри.