Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Звіт
до лабораторної роботи №6 з дисципліни
«Розробка мобільних застосувань під Android»

Виконала:
Студентка групи ІА-22
Микитенко Ірина

Перевірив:
Орленко С. П.

Київ 2025

ЛАБОРАТОРНА РОБОТА №6
ВІЛЬНА ТЕМА

Мета роботи: змоделювати головну проблему наступного року навчання (у вигляді вибору теми дипломного проекту) шляхом самостійного формування завдання на 6 лабораторну роботу.

Завдання:

Написати програму під платформу Андроїд, яка буде заснована на темі, яка не розглядалась на попередніх лабораторних роботах (попередні навички теж можна використовувати, але «основою» має бути саме нова обрана тема).
Примітка: Очевидно, що конкретного варіанту не передбачено, студент сам формує завдання та вигляд програми. Наприклад, на попередніх лабах не була приділена увага web-у, створенню віджетів, роботі з зовнішнім обладнанням, створенню ігрових застосунків і т.д.

Хід роботи

1. Опис завдання

Створити мобільний застосунок-гру для Android:

- Користувач керує кошиком, ловить фрукти та уникає бомб.
- Відображаються поточний рахунок, рекорд та таймер гри.
- Збереження найкращого та останнього результатів за допомогою SharedPreferences.

2. Структура проєкту

Основні компоненти:

- MainActivity — головне меню гри
- GameActivity — активність самої гри
- GameView — ігрове поле
- GameOverActivity — екран завершення гри
- Fruit — клас для фруктів та бомб

3. Опис роботи кожного класу

MainActivity

Відображає рекорд та попередній результат.

Має кнопку для запуску гри.

```
class MainActivity : AppCompatActivity() {

    private lateinit var prefs: SharedPreferences

    @SuppressLint("SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        prefs = getSharedPreferences( name: "game", Context.MODE_PRIVATE)
        val highScore = prefs.getInt( key: "high_score",  defValue: 0)
        val lastScore = prefs.getInt( key: "last_score",  defValue: 0)

        val highScoreText = findViewById<TextView>(R.id.high_score_text)
        val lastScoreText = findViewById<TextView>(R.id.last_score_text)

        highScoreText.text = "🌟 Рекорд: $highScore 🌟"
        lastScoreText.text = "🎯 Попередній результат: $lastScore"
        highScoreText.setTextColor(Color.MAGENTA)
        highScoreText.textSize = 28f

        val startButton = findViewById<Button>(R.id.start_button)
        startButton.setOnClickListener {
            val intent = Intent( packageContext: this, GameActivity::class.java)
            startActivity(intent)
        }

    }
}
```

GameActivity

Запускає ігрове поле (GameView).

Керує паузою та відновленням гри.

```
class GameActivity : AppCompatActivity() {

    private lateinit var gameView: GameView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        gameView = GameView( context: this)
        setContentView(gameView)
    }
    override fun onPause() {
        super.onPause()
        gameView.pause()
    }

    override fun onResume() {
        super.onResume()
        gameView.resume()
    }
}
```

GameView

- Основна логіка гри:
    - Малювання кошика, фруктів, бомб, рахунку та таймера.
    - Обробка торкань для керування кошиком.
    - Зіткнення об'єктів.
    - Відтворення звуків.
    - Збереження рекорду.
- Використовуємо SurfaceView для малювання та окремий потік для оновлення екрану.

```
class GameView(context: Context) : SurfaceView(context), Runnable {

    @Volatile
    private var playing = true
    private var thread: Thread? = null
    private var canvas: Canvas? = null
    private val paint = Paint()

    private val originalBasket = BitmapFactory.decodeResource(resources, R.drawable.basket)
    private val basket = Bitmap.createScaledBitmap(originalBasket, dstWidth: 200, dstHeight: 200, filter: true)
    private var basketX = 500f
    private var basketY = 1700f
    private var basketSpeed = 0f

    private val fruits = mutableListOf<Fruit>()
    private val fruitBitmaps = listOf(
        R.drawable.fruit1,
        R.drawable.fruit2,
        R.drawable.fruit3,
        R.drawable.fruit4,
        R.drawable.fruit5
    ).map {
        val original = BitmapFactory.decodeResource(resources, it)
        Bitmap.createScaledBitmap(original, dstWidth: 100, dstHeight: 100, filter: true)
    }

    private val bombBitmap = Bitmap.createScaledBitmap(
        BitmapFactory.decodeResource(resources, R.drawable.bomb),
        dstWidth: 100, dstHeight: 100, filter: true
    )
```

```
    private val screenWidth = resources.displayMetrics.widthPixels
    private val screenHeight = resources.displayMetrics.heightPixels

    private val gameDuration = 30000L
    private var remainingTime = gameDuration
    private var gameStartTime = 0L
    private var pausedAt = 0L

    private var score = 0
    private var highScore = 0
```

```
private val prefs: SharedPreferences =
    context.getSharedPreferences( name: "game", Context.MODE_PRIVATE)

private val fruitSound = MediaPlayer.create(context, R.raw.fruit_catch)
private val bombSound = MediaPlayer.create(context, R.raw.bomb_hit)

init {
    highScore = prefs.getInt( key: "high_score",  defValue: 0)
    spawnFruit()
    gameStartTime = System.currentTimeMillis()
    thread = Thread( target: this)
    thread?.start()
}

override fun run() {
    while (playing) {
        update()
        draw()
        control()
    }
}
```

```
private fun spawnFruit() {
    fruits.add(
        Fruit(
            fruitBitmaps.random(),
            Random.nextInt( from: 0,  until: screenWidth - 200).toFloat(),
            y: -200f,
            isBomb: false
        )
    )
    if (Random.nextInt( until: 10) < 2) {
        fruits.add(
            Fruit(
                bombBitmap,
                Random.nextInt( from: 0,  until: screenWidth - 200).toFloat(),
                y: -200f,
                isBomb: true
            )
        )
    }
}
```

```
private fun update() {
    basketX += basketSpeed
    if (basketX < 0) basketX = 0f
    if (basketX > screenWidth - basket.width) basketX = (screenWidth - basket.width).toFloat()

    val iterator = fruits.iterator()
    while (iterator.hasNext()) {
        val fruit = iterator.next()
        fruit.y += 10f

        if (fruit.y > screenHeight) {
            iterator.remove()
            continue
        }

        if (abs( x: fruit.x - basketX) < 100 && abs( x: fruit.y - basketY) < 100) {
            if (fruit.isBomb) {
                score--
                bombSound.start()
            } else {
                score++
                fruitSound.start()
            }
            iterator.remove()
        }
    }
```

```
    if (Random.nextInt( until: 100) < 4) {
        spawnFruit()
    }

    if (score > highScore) {
        highScore = score
        prefs.edit().putInt("high_score", highScore).apply()
    }

    val elapsed = System.currentTimeMillis() - gameStartTime
    if (elapsed >= remainingTime) {
        playing = false
        (context as? Activity)?.runOnUiThread {
            val intent = Intent(context, GameOverActivity::class.java)
            intent.putExtra( name: "score", score)
            context.startActivity(intent)
            (context as Activity).finish()
        }
    }
}
```

```
private fun draw() {
    if (holder.surface.isValid) {
        canvas = holder.lockCanvas()
        canvas?.drawColor(Color.rgb( red: 255,  green: 242,  blue: 228))

        paint.textSize = 60f
        paint.color = Color.BLACK
        canvas?.drawText( text: "Score: $score",  x: 50f,  y: 100f, paint)
        canvas?.drawText( text: "Best: $highScore",  x: 50f,  y: 170f, paint)

        val elapsed = System.currentTimeMillis() - gameStartTime
        val timeLeft = (remainingTime - elapsed).coerceAtLeast( minimumValue: 0L)
        val progress = timeLeft.toFloat() / gameDuration

        val barLeft = 50f
        val barTop = 200f
        val barRight = screenWidth - 50f
        val barBottom = 240f
```

```
        paint.color = Color.LTGRAY
        canvas?.drawRect(barLeft, barTop, barRight, barBottom, paint)

        paint.color = Color.parseColor( colorString: "#FF5722")
        canvas?.drawRect(
            barLeft,
            barTop,
             right: barLeft + (barRight - barLeft) * progress,
            barBottom,
            paint
        )

        paint.color = Color.BLACK
        paint.style = Paint.Style.STROKE
        paint.strokeWidth = 4f
        canvas?.drawRect(barLeft, barTop, barRight, barBottom, paint)
        paint.style = Paint.Style.FILL

        canvas?.drawBitmap(basket, basketX, basketY, paint)
        val fruitsCopy: List<Fruit>
        synchronized( lock: this) {
            fruitsCopy = fruits.toList()
        }
        for (fruit in fruitsCopy) {
            canvas?.drawBitmap(fruit.bitmap, fruit.x, fruit.y, paint)
        }
```

```
         holder.unlockCanvasAndPost(canvas)
    }
}

private fun control() {
    Thread.sleep( millis: 17)
}
```

```
    override fun onTouchEvent(event: MotionEvent?): Boolean {
        event?.let {
            if (it.action == MotionEvent.ACTION_MOVE) {
                basketX = it.x - basket.width / 2
            }
        }
        return true
    }

    fun pause() {
        playing = false
        pausedAt = System.currentTimeMillis()
        thread?.join()
        val elapsed = pausedAt - gameStartTime
        remainingTime -= elapsed
    }

    fun resume() {
        playing = true
        gameStartTime = System.currentTimeMillis()
        thread = Thread( target: this)
        thread?.start()
    }
}
```

GameOverActivity

Відображає результат гри.

Кнопки для перезапуску гри або повернення до меню.

```
class GameOverActivity : AppCompatActivity() {
    @SuppressLint("SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_game_over)

        val score = intent.getIntExtra( name: "score",  defaultValue: 0)
        val prefs = getSharedPreferences( name: "game", Context.MODE_PRIVATE)
        prefs.edit().putInt("last_score", score).apply()

        val resultText = findViewById<TextView>(R.id.result_text)
        resultText.text = "Ваш результат: $score 🎯"

        val restartBtn = findViewById<Button>(R.id.restart_button)
        restartBtn.setOnClickListener {
            startActivity(Intent( packageContext: this, GameActivity::class.java))
            finish()
        }

        val backBtn = findViewById<Button>(R.id.back_button)
        backBtn.setOnClickListener {
            startActivity(Intent( packageContext: this, MainActivity::class.java))
            finish()
        }
    }
}
```

Fruit

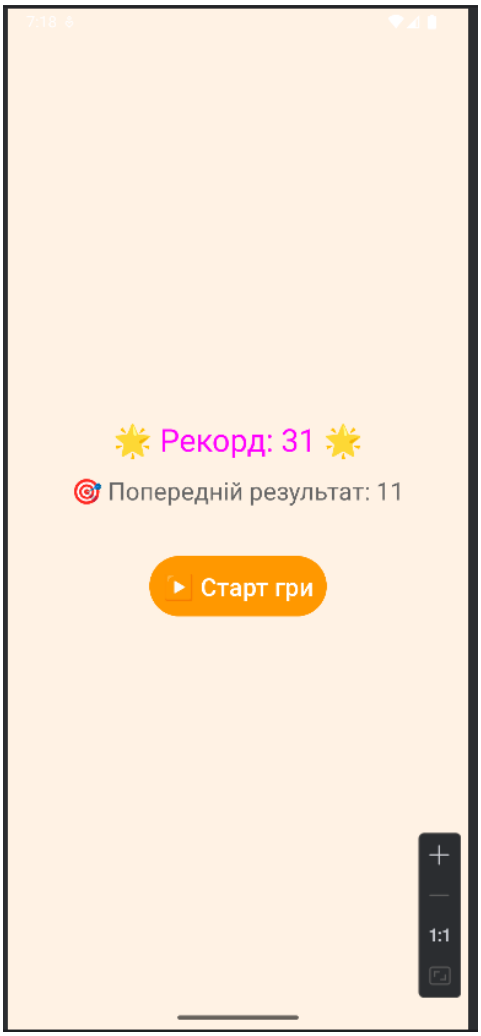
Модель фрукта або бомби.

Містить зображення, координати та прапорець, чи є це бомба.

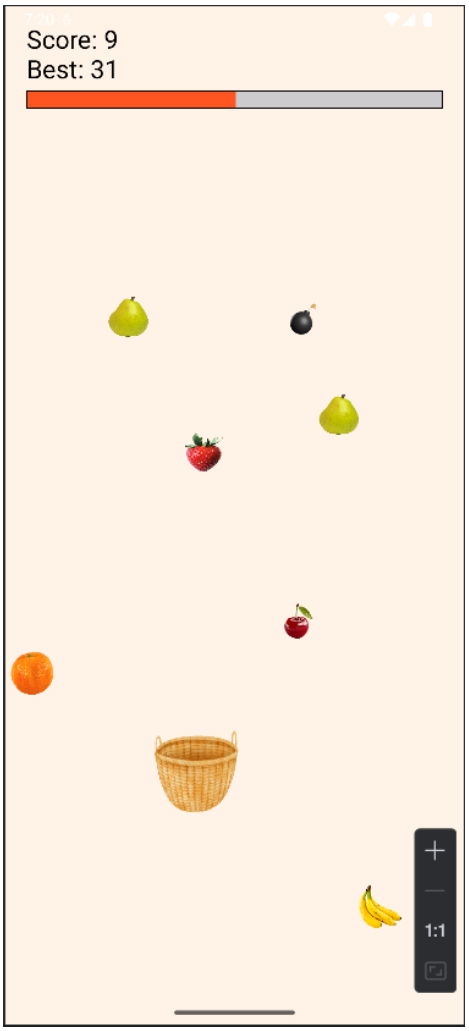
```
data class Fruit(
    val bitmap: Bitmap,
    var x: Float,
    var y: Float,
    val isBomb: Boolean
)
```

4. Скріншоти роботи застосунку

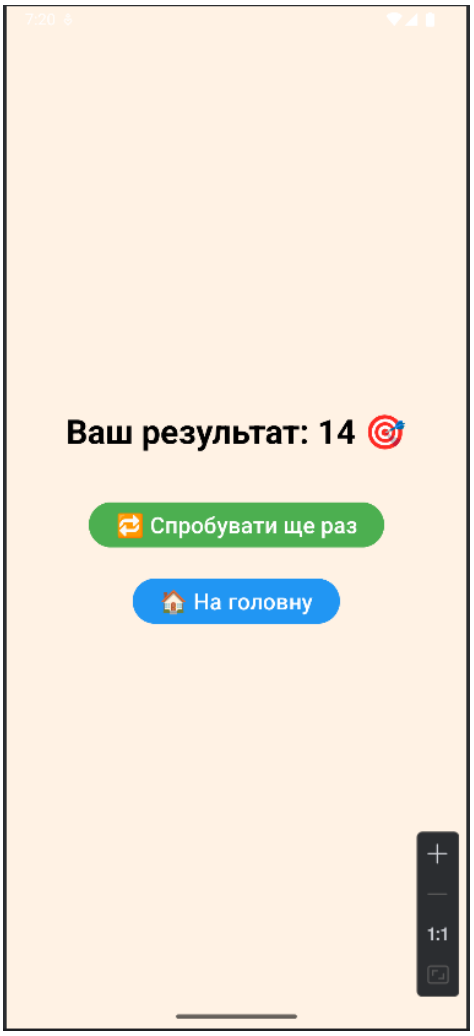
Головне меню:



Ігровий процес:



Екран завершення гри:



Висновок: в ході виконання лабораторної роботи створено інтерактивну гру для Android. Засвоєно основи роботи з SurfaceView, Canvas, обробкою торкань екрану та звуковими ефектами.