


Politechnika Bydgoska im. Jana i Jędrzeja Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki al. prof. S. Kaliskiego 7, 85-796 Bydgoszcz				 POLITECHNIKA BYDGOSKA Wydział Telekomunikacji, Informatyki i Elektrotechniki	
Przedmiot	Programowanie urządzeń mobilnych			Kierunek/Tryb	IS/ST
Nr laboratorium	6	Data wykonania	03.05.2025	Grupa	1
Ocena		Data oddania	03.05.2025	Imię Nazwisko	Cezary Tytko
Nazwa ćwiczenia	Interfejs gry, tworzenie i obsługa interfejsu, intencje				

2. Cel ćwiczenia

Celem tego ćwiczenia jest stworzenie gry na platformie Android, obejmującej pętlę gry, animacje, tło, obsługę zdarzeń oraz sterowanie postacią. Uczestnik będzie miał okazję zaznajomić się z różnymi aspektami programowania gier mobilnych.

Oczekiwane wyniki

Po ukończeniu ćwiczenia, uczestnik powinien być w stanie:

1. Utworzyć działającą pętlę gry, zapewniającą płynność działania.
2. Dodawać tło do sceny gry, nadając jej atrakcyjny wygląd.
3. Tworzyć animacje, wzbogacając interakcje w grze.
4. Dodawać i sterować dźwiękami.
5. Implementować interaktywne sterowanie postacią, co pozwoli graczowi aktywnie uczestniczyć w rozgrywce.

3. Opis projektu

Na podstawie przedstawionego poniżej szkieletu aplikacji implementującego pętlę gry należy uzupełnić rozgrywkę inspirowaną się klasyczną grą Arkanoid. Jest to to gra arkadowa, gdzie gracz steruje paletką, odbijając piłkę w kierunku bloków, próbując zniszczyć je wszystkie. Zadaniem gracza jest utrzymanie piłki w grze, unikając utraty życia poprzez przejście przez dolną krawędź ekranu, a gracz zdobywa punkty za każdy zniszczony blok. Gra oferuje różnorodne poziomy, bonusy i stanowi klasyczne wyzwanie zręcznościowe.

4. Implementacja

Kod został napisany w języku Kotlin w środowisku Android Studio.

Elementy ekranu:

- **Ball** – piłka.
- **Block** - bloki do zniszczenia
- **Paddle** –paletka.

MainActivity.kt:

```

1. package com.example.pumlab6
2.
3. import android.os.Bundle
4. import androidx.activity.enableEdgeToEdge
5. import androidx.appcompat.app.AppCompatActivity
6. import androidx.core.view.ViewCompat
7. import androidx.core.view.WindowInsetsCompat
8.
9. class MainActivity : AppCompatActivity() {
10.     private lateinit var gameView: GameView

```

```

11.
12.     override fun onCreate(savedInstanceState: Bundle?) {
13.         super.onCreate(savedInstanceState)
14.         gameView = GameView(this)
15.         setContentView(gameView)
16.         //         gameView.resume()
17.     }
18.
19.     override fun onPause() {
20.         super.onPause()
21.         gameView.pause()
22.     }
23.
24.     override fun onResume() {
25.         super.onResume()
26.         gameView.resume()
27.     }
28. }
29.

```

GameView.kt:

```

1. package com.example.pumlab6
2.
3. import android.content.Context
4. import android.graphics.Color
5. import android.graphics.Paint
6. import android.graphics.RectF
7. import android.util.Log
8. import android.view.MotionEvent
9. import android.view.SurfaceHolder
10. import android.view.SurfaceView
11.
12. class GameView(context: Context) : SurfaceView(context), Runnable {
13.     private var thread : Thread = Thread(this)
14.     private var playing = false
15.     private val paint = Paint()
16.     private val ball = Ball()
17.     private val paddle = Paddle()
18.     private val blocks = mutableListOf<Block>()
19.     private var isGameOver = false
20.
21.     init {
22.         holder.addCallback(object : SurfaceHolder.Callback {
23.             override fun surfaceCreated(holder: SurfaceHolder) {
24.                 resetBlocks()
25.                 //                 playing = true
26.                 //                 thread.start()
27.             }
28.             override fun surfaceChanged(holder: SurfaceHolder, format: Int, width: Int, height: Int) {}
29.             override fun surfaceDestroyed(holder: SurfaceHolder) { pause() }
30.         })
31.     }
32.
33.     private var score = 0
34.     private var level = 1
35.     private val textPaint = Paint().apply {
36.         color = Color.WHITE
37.         textSize = 60f
38.     }
39.
40.
41.     private fun resetBlocks() {
42.         blocks.clear()
43.         val rows = if (level == 1) 2 else 6
44.         val cols = if (level == 1) 3 else 7
45.         val blockWidth = width / cols
46.         val blockHeight = 60
47.
48.         for (i in 0 until rows) {
49.             for (j in 0 until cols) {
50.                 blocks.add(Block(j * blockWidth, i * blockHeight + 100, blockWidth - 10, blockHeight - 10))
51.             }
52.         }
53.     }
54.

```

```

55.     override fun run() {
56.         while (playing) {
57.             // Log.d("GAME", "Game loop running")
58.             update()
59.             draw()
60.             Thread.sleep(16)
61.         }
62.     }
63.
64.     private fun update() {
65.         ball.update()
66.         if (RectF.intersects(ball.rect, paddle.rect)) {
67.             ball.reverseY()
68.         }
69.
70.         val iterator = blocks.iterator()
71.         while (iterator.hasNext()) {
72.             val block = iterator.next()
73.             if (RectF.intersects(ball.rect, block.rect)) {
74.                 ball.reverseY()
75.                 iterator.remove()
76.                 score += 10
77.                 break
78.             }
79.         }
80.
81.         if (blocks.isEmpty()) {
82.             level++
83.             if (level > 2) {
84.                 playing = false // Gra wygrana
85.             } else {
86.                 resetBlocks()
87.                 ball.reset()
88.             }
89.         }
90.
91.         if (ball.rect.bottom > height) {
92.             resetGame() // koniec gry
93.         }
94.     }
95.
96.     private fun draw() {
97.         if (holder.surface.isValid) {
98.             // Log.d("GAME", "Drawing frame")
99.             val canvas = holder.lockCanvas()
100.            canvas.drawColor(Color.BLACK)
101.
102.            paint.color = Color.WHITE
103.            canvas.drawRect(paddle.rect, paint)
104.
105.            paint.color = Color.RED
106.            canvas.drawOval(ball.rect, paint)
107.
108.            paint.color = Color.BLUE
109.            blocks.forEach { canvas.drawRect(it.rect, paint) }
110.
111.            canvas.drawText("Wynik: $score", 50f, 80f, textPaint)
112.            canvas.drawText("Poziom: $level", width - 300f, 80f, textPaint)
113.
114.            holder.unlockCanvasAndPost(canvas)
115.        }
116.    }
117.
118.    override fun onTouchEvent(event: MotionEvent): Boolean {
119.        when (event.action) {
120.            MotionEvent.ACTION_MOVE -> paddle.moveTo(event.x)
121.        }
122.        return true
123.    }
124.
125.    fun pause() {
126.        playing = false
127.        try {
128.            if (thread.isAlive)
129.                thread.join()
130.        }
131.        catch (e: Exception){
132.            e.printStackTrace()

```

```

133.     }
134. }
135.
136. fun resume() {
137.     playing = true
138.     if (!thread.isAlive) {
139.         thread = Thread(this)
140.         thread.start()
141.     }
142. }
143.
144. private fun resetGame() {
145.     score = 0
146.     level = 1
147.     resetBlocks()
148.     ball.reset()
149. }
150.
151. }
152.

```

Ball.kt:

```

1. package com.example.pumlab6
2.
3. import android.graphics.RectF
4.
5. class Ball {
6.     var rect = RectF(1080f / 2f - 25, 600f, 1080f / 2f - 25 + 50, 650f)
7.     private var dx = 20f
8.     private var dy = 20f
9.
10.    fun update() {
11.        rect.offset(dx, dy)
12.        if (rect.left < 0 || rect.right > 1080) dx = -dx
13.        if (rect.top < 0) dy = -dy
14.    }
15.
16.    fun reverseY() {
17.        dy = -dy
18.    }
19.
20.    fun reset() {
21.        rect.set(1080f / 2f - 25, 600f, 1080f / 2f - 25 + 50, 650f)
22.        dx = 20f
23.        dy = 20f
24.    }
25. }
26.

```

Paddle.kt:

```

1. package com.example.pumlab6
2.
3. import android.graphics.RectF
4.
5. class Paddle {
6.     val rect = RectF(400f, 1700f, 680f, 1720f)
7.
8.     fun moveTo(x: Float) {
9.         val width = rect.width()
10.        rect.left = x - width / 2
11.        rect.right = x + width / 2
12.    }
13. }
14.

```

Block.kt:

```

1. package com.example.pumlab6
2.
3. import android.graphics.RectF
4.
5. class Block(x: Int, y: Int, width: Int, height: Int) {
6.     val rect = RectF(x.toFloat(), y.toFloat(), (x + width).toFloat(), (y + height).toFloat())
7. }
8.

```

Activity.main.xml:

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:id="@+id/main"
6.     android:layout_width="match_parent"
7.     android:layout_height="match_parent"
8.     tools:context=".MainActivity">
9.
10.    <TextView
11.        android:layout_width="wrap_content"
12.        android:layout_height="wrap_content"
13.        android:text="Hello World!"
14.        app:layout_constraintBottom_toBottomOf="parent"
15.        app:layout_constraintEnd_toEndOf="parent"
16.        app:layout_constraintStart_toStartOf="parent"
17.        app:layout_constraintTop_toTopOf="parent" />
18.
19. </androidx.constraintlayout.widget.ConstraintLayout>
20.
21.
```

5. Funkcje kluczowe

- Monitorowanie poziomu i wyniku rozgrywki
- Rozbijanie bloków
- Operowanie paletką
- Zwiększanie trudności z każdym poziomem

6. Testowanie

Testowanie gry obejmowało:

- Sprawdzenie, czy między paletką i piłką zachodzą poprawne interakcje
- Sprawdzenie, czy gra resetuje się po utraceniu piłki i jak resetuje się pozycja piłki.
- Sprawdzenie, czy program poprawnie reaguje na wygraną i zwiększa trudność poziomu.

7. Wyniki



8. Podsumowanie

Projekt realizuje klasyczną grę w Arkanoid w aplikacji mobilnej. Implementacja pozwoliła na zdobycie doświadczenia w tworzeniu aplikacji na Androida, obsłudze interakcji użytkownika oraz zarządzaniu stanem gry. Ćwiczenie umożliwiło lepsze zrozumienie obsługi zdarzeń i logiki gry.

9. Trudności i błędy

- Nie wystąpiły żadne trudności ani błędy.

10. Źródła i odniesienia

- Nie korzystano ze źródeł i odniesień innych niż ta instrukcja.

11. Dodatkowe materiały

- Nie korzystano z dodatkowych materiałów.