|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1Politechnika Bydgoska im. Jana i Jędrzeja Śniadeckich  Wydział Telekomunikacji, Informatyki i Elektrotechniki  al. prof. S. Kaliskiego 7, 85-796 Bydgoszcz | | | |  | |
| Przedmiot | **Programowanie urządzeń mobilnych** | | | Kierunek/Tryb | IS/ST |
| Nr laboratorium | 10 | Data wykonania | 17.06 | Grupa | 1 |
| Ocena |  | Data oddania | 17.06 | Imię Nazwisko | Cezary Tytko |
| Nazwa ćwiczenia | Pętla gry, animacja postaci, kolizje, obsługa dotyku | | |

**2. Cel ćwiczenia laboratoryjnego**

Celem tego ćwiczenia jest stworzenie gry inspirowanej na klasycznej grze Boulderdash, realizacja zadania obejmuje pętlę gry, animacje postaci i przeciwników, tło, dźwięki, obsługę zdarzeń oraz sterowanie postacią.

**3. Opis projektu**

Na podstawie przedstawionego poniżej szkieletu aplikacji implementującego pętle gry należy uzupełnić rozgrywkę inspirując się klasyczną grą Boulderdash (lista na końcu instrukcji).

**4. Implementacja**

**Kod został napisany w języku Kotlin w środowisku Android Studio.**

MainActivity.kt:

1. package com.example.pumlab1

2.

3. import android.R.attr.left

4. import android.R.attr.right

5. import android.content.res.Resources

6. import android.graphics.Bitmap

7. import android.graphics.BitmapFactory

8. import android.os.Bundle

9. import android.util.Log

10. import android.widget.Button

11. import android.widget.GridLayout

12. import android.widget.ImageView

13. import androidx.appcompat.app.AppCompatActivity

14.

15.

16. class MainActivity : AppCompatActivity() {

17.

18. private lateinit var spritesheetIdle: Bitmap

19. private lateinit var spritesheetLeft: Bitmap

20. private lateinit var spritesheetRight: Bitmap

21.

22. private enum class Direction { IDLE, LEFT, RIGHT }

23.

24. private var currentDirection = Direction.IDLE

25. private var frameIndex = 0

26. private val frameCount = 7

27. private val frameWidth = 32

28. private val frameHeight = 32

29.

30. private lateinit var rockfordImageView: ImageView

31.

32.

33. private lateinit var gridLayout: GridLayout

34. private val numRows = 10

35. private val numCols = 10

36. private lateinit var map: Array<CharArray>

37. private var rockfordX = 1

38. private var rockfordY = 1

39. private lateinit var spritesheet: Bitmap

40. private lateinit var rockford: Bitmap

41.

42. override fun onCreate(savedInstanceState: Bundle?) {

43. super.onCreate(savedInstanceState)

44. setContentView(R.layout.activity\_main)

45.

46. gridLayout = findViewById(R.id.boardGridLayout)

47. gridLayout.rowCount = numRows

48. gridLayout.columnCount = numCols

49.

50. // Inicjalizacja rockfordImageView:

51. val displayMetrics = Resources.getSystem().displayMetrics

52. val screenWidth = displayMetrics.widthPixels

53. rockfordImageView = ImageView(this).apply {

54. layoutParams = GridLayout.LayoutParams().apply {

55. width = screenWidth / numCols // jeszcze możesz to poprawić, np. po layout pass

56. height = screenWidth / numCols

57. }

58. // scaleType = ImageView.ScaleType.CENTER

59. }

60.

61. // gridLayout.addView(rockfordImageView) // Lub ustaw na odpowiedniej pozycji planszy

62. animationHandler.post(animationRunnable)

63.

64. initMap()

65. loadSprites()

66. drawMap()

67.

68. findViewById<Button>(R.id.btnUp).setOnClickListener { moveRockford(0, -1) }

69. findViewById<Button>(R.id.btnDown).setOnClickListener { moveRockford(0, 1) }

70. findViewById<Button>(R.id.btnLeft).setOnClickListener { moveRockford(-1, 0) }

71. findViewById<Button>(R.id.btnRight).setOnClickListener { moveRockford(1, 0) }

72. }

73.

74. private fun initMap() {

75. map = arrayOf(

76. "##########".toCharArray(),

77. "#R.......#".toCharArray(),

78. "#........#".toCharArray(),

79. "#....\*...#".toCharArray(),

80. "#........#".toCharArray(),

81. "#....O...#".toCharArray(),

82. "#........#".toCharArray(),

83. "#..\*.....#".toCharArray(),

84. "#........#".toCharArray(),

85. "##########".toCharArray()

86. )

87. }

88.

96. private fun loadSprites() {

97. val options = BitmapFactory.Options().apply { inScaled = false }

98. spritesheetIdle = BitmapFactory.decodeResource(resources, R.drawable.rockford\_idle\_32x32, options)

99. spritesheetLeft = BitmapFactory.decodeResource(resources, R.drawable.rockford\_left\_32x32, options)

100. spritesheetRight = BitmapFactory.decodeResource(resources, R.drawable.rockford\_right\_32x32, options)

101. }

102.

103.

104. private fun drawMap() {

105. gridLayout.removeAllViews()

106. val displayMetrics = Resources.getSystem().displayMetrics

107. val screenWidth = displayMetrics.widthPixels

108. val screenHeight = displayMetrics.heightPixels

109.

110. val tileWidth = screenWidth / numCols

111. val tileHeight = screenHeight / numRows

112.

123. for (i in 0 until numRows) {

124. for (j in 0 until numCols) {

125. if (map[i][j] == 'R' )

126. {

127. gridLayout.addView(rockfordImageView)

128. continue

129. }

130. val imageView = ImageView(this).apply {

131. layoutParams = GridLayout.LayoutParams().apply {

132. width = tileWidth

133. height = tileWidth

134. }

135. }

136.

137. when (map[i][j]) {

138. '#' -> imageView.setImageResource(R.drawable.wall\_32x32)

139. '.' -> imageView.setImageResource(R.drawable.ground\_32x32)

140. '\*' -> imageView.setImageResource(R.drawable.diamond\_32x32)

141. 'O' -> imageView.setImageResource(R.drawable.stone\_32x32)

142. ' ' -> imageView.setImageResource(R.drawable.empty\_32x32)

149. }

150.

151. gridLayout.addView(imageView)

152. }

153. }

154.

170. }

171.

172. private fun moveRockford(dx: Int, dy: Int) {

173. val newX = rockfordX + dx

174. val newY = rockfordY + dy

175.

176. if (map[newY][newX] != '#') {

177. // przesuń Rockforda

178. map[rockfordY][rockfordX] = ' ' // stare miejsce

179. map[newY][newX] = 'R' // nowe miejsce

180. rockfordX = newX

181. rockfordY = newY

182. drawMap()

183. }

184.

185. if (dx < 0) currentDirection = Direction.LEFT

186. else if (dx > 0) currentDirection = Direction.RIGHT

187. else currentDirection = Direction.IDLE

188. frameIndex = 0

189.

190. }

191.

192. private fun getFrameBitmap(direction: Direction, frame: Int): Bitmap {

193. val spritesheet = when(direction) {

194. Direction.IDLE -> spritesheetIdle

195. Direction.LEFT -> spritesheetLeft

196. Direction.RIGHT -> spritesheetRight

197. }

198. return Bitmap.createBitmap(spritesheet, frame \* frameWidth, 0, frameWidth, frameHeight)

199. }

200.

201. private val animationHandler = android.os.Handler()

202. private val animationRunnable = object : Runnable {

203. override fun run() {

204. Log.d("TAG", frameIndex.toString())

205. val frameBitmap = getFrameBitmap(currentDirection, frameIndex)

206. rockfordImageView.setImageBitmap(frameBitmap)

207. frameIndex = (frameIndex + 1) % frameCount

208. animationHandler.postDelayed(this, 100) // co 100ms

209. }

210. }

252. }

253.

254.

Activity\_main.xml:

1. <?xml version="1.0" encoding="utf-8"?>

2. <LinearLayout 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/mainLayout"

6. android:layout\_width="match\_parent"

7. android:layout\_height="match\_parent"

8. android:orientation="vertical"

9. tools:context=".MainActivity">

10.

11. <!-- Plansza gry -->

12. <FrameLayout

13. android:layout\_width="match\_parent"

14. android:layout\_height="0dp"

15. android:layout\_weight="1">

16.

17. <GridLayout

18. android:id="@+id/boardGridLayout"

19. android:layout\_width="match\_parent"

20. android:layout\_height="match\_parent"

21. android:padding="0dp"

22. android:visibility="visible"

23. tools:visibility="visible" />

24.

32. </FrameLayout>

33.

34. <!-- Sterowanie -->

35. <LinearLayout

36. android:id="@+id/controls"

37. android:layout\_width="match\_parent"

38. android:layout\_height="wrap\_content"

39. android:orientation="vertical"

40. android:gravity="center"

41. android:padding="8dp">

42.

43. <Button

44. android:id="@+id/btnUp"

45. android:layout\_width="wrap\_content"

46. android:layout\_height="wrap\_content"

47. android:text="↑" />

48.

49. <LinearLayout

50. android:layout\_width="wrap\_content"

51. android:layout\_height="wrap\_content"

52. android:orientation="horizontal">

53.

54. <Button

55. android:id="@+id/btnLeft"

56. android:layout\_width="wrap\_content"

57. android:layout\_height="wrap\_content"

58. android:text="←" />

59.

60. <Button

61. android:id="@+id/btnRight"

62. android:layout\_width="wrap\_content"

63. android:layout\_height="wrap\_content"

64. android:text="→" />

65. </LinearLayout>

66.

67. <Button

68. android:id="@+id/btnDown"

69. android:layout\_width="wrap\_content"

70. android:layout\_height="wrap\_content"

71. android:text="↓" />

72. </LinearLayout>

73. </LinearLayout>

74.

**5. Funkcje kluczowe**

**Wyświetlanie planszy gry w siatce (GridLayout)**

* **Graficzna reprezentacja planszy 10x10 z różnymi obiektami (ściany, ziemia, diamenty, kamienie, puste pola).**

**Sterowanie postacią Rockford za pomocą przycisków kierunkowych**

* **Możliwość poruszania się w górę, dół, lewo i prawo z aktualizacją pozycji na planszy.**

**Wczytywanie i animowanie sprite'ów Rockforda**

* **Wczytywanie trzech różnych arkuszy sprite’ów (idle, lewo, prawo) i animowanie ich co 100 ms w zależności od kierunku ruchu.**

**Dynamiczne odświeżanie planszy po ruchu**

* **Każdy ruch Rockforda powoduje przerysowanie całej planszy, z uwzględnieniem nowej pozycji.**

**6. Testowanie**

**Test ruchu i kolizji:**

* **Sprawdzono, czy postać Rockforda nie przechodzi przez ściany (#) oraz czy poprawnie przemieszcza się po dostępnych polach (., ).**
* **Przetestowano zachowanie przy próbie wyjścia poza granice planszy.**

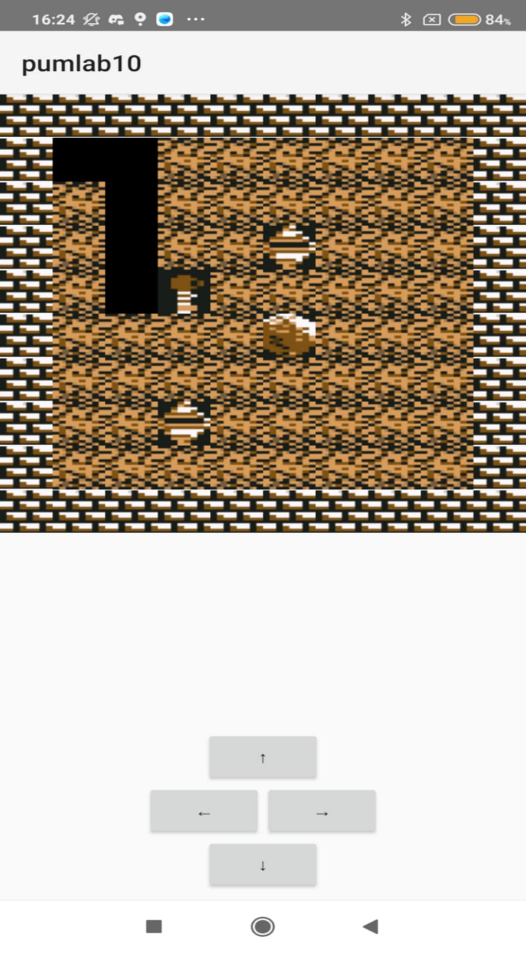
**Test animacji postaci:**

* **Zweryfikowano, czy animacje są zgodne z kierunkiem ruchu (idle, lewo, prawo).**
* **Upewniono się, że klatki animacji zmieniają się płynnie i w równych odstępach czasu.**

**Test interfejsu i reakcji na przyciski:**

* **Przetestowano działanie przycisków kierunkowych – sprawdzono, czy każdy klik powoduje odpowiedni ruch postaci.**
* **Oceniono responsywność interfejsu przy szybkim i wielokrotnym naciskaniu przycisków.**

**7. Wyniki**



**8. Podsumowanie**

Projekt realizuje przedstawienie gry inspirowanej na klasycznej grze Boulderdash. Do zbudowania planszy i stworzenia animacji wykorzystano dostarczone w instrukcji zasoby graficzne.

**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.