


<p>Politechnika Bydgoska im. Jana i Jędrzeja Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki al. prof. S. Kaliskiego 7, 85-796 Bydgoszcz</p>				 <p><b>POLITECHNIKA BYDGOSKA</b> Wydział Telekomunikacji, Informatyki i Elektrotechniki</p>	
Przedmiot	Programowanie urządzeń mobilnych			Kierunek/Tryb	IS/ST
Nr laboratorium	5	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 ćwiczenia jest zapoznanie studentów z tworzeniem prostej gry mobilnej na platformie Android przy użyciu interfejsu użytkownika. Studenci mają nauczyć się:

1. Projektować interfejs użytkownika, który obejmuje zarządzanie grą.
2. Programować obsługę zdarzeń związaną z rozgrywką i obsługą interfejsu.
3. Wykonywać proste operacje na danych wejściowych.
4. Wyświetlać wynik, obrazy i komunikaty na ekranie w czytelny sposób.
5. Obsługiwać odtwarzanie dźwięków w aplikacji.

## 3. Opis projektu

Na podstawie przedstawionego poniżej szkieletu aplikacji implementującego interfejs gry uzupełnić ekran rozgrywki grą w „piętnastkę” (lub inaczej "Gra w 15"). Jest to logiczna łamigłówka polegająca na układaniu kawałków obrazka w odpowiedniej kolejności, aby uzyskać poprawny obrazek. Gra ta jest często rozgrywana na planszy o wymiarach 4x4 (czyli 15 kawałków plus jedno puste pole), stąd jej nazwa. Zasady gry w piętnastkę

Celem gry jest ułożenie kawałków obrazka w odpowiedniej kolejności, aby uzyskać poprawny obrazek (zazwyczaj numerowany od 1 do 15 lub zawierający obrazek). Plansza do gry składa się z 4 rzędów i 4 kolumn (łącznie 16 pól). Na planszy znajdują się kawałki obrazka o numerach od 1 do 15 oraz jedno puste pole. Gracz może przemieszczać kawałki obrazka na planszy, przesuwając je na puste pole. Można to zrobić w jednym z czterech kierunków: w górę, w dół, w lewo lub w prawo. Kawałek jest przesuwany na puste pole, które staje się teraz zajęte przez ten kawałek. Gracz wygrywa, gdy uda mu się ułożyć kawałki w odpowiedniej kolejności, na przykład od 1 do 15, tworząc pełny obrazek. Gra w piętnastkę jest znana z tego, że może być trudna, ponieważ przemieszczanie kawałków na planszy wymaga rozwiązywania układanki logicznej. Niektóre konfiguracje planszy mogą być nierozwiązywalne. Implementację gry i rozgrywki należy uzupełnić o licznik czasu, licznik ruchów, warunek zwycięstwa lub porażki, liczbę przesuwanych puzzli w trakcie mieszania, dźwięki przesuwania i mieszania, dźwięki menu.

## 4. Implementacja

Kod został napisany w języku Kotlin w środowisku Android Studio.

Wykorzystane widoki:

- Ekran menu – umożliwia rozpoczęcie rozgrywki lub wyświetlenie nazwiska autorów.
- Ekran gry – rozpoczęcie rozgrywki, test wygranej lub mieszanie kawałków.

Elementy: ekranu:

- GameBoard – plansza.
- Tile – kawałek obrazka.

### MainActivity.kt:

```
1. package com.example.pumlab5
2. import android.content.Intent
3. import android.os.Bundle
4. Cezary Tytko
5. import android.widget.Button
6. import androidx.activity.enableEdgeToEdge
7. import androidx.appcompat.app.AppCompatActivity
8. class MainActivity : AppCompatActivity() {
9.     override fun onCreate(savedInstanceState: Bundle?) {
10.         super.onCreate(savedInstanceState)
11.         enableEdgeToEdge()
12.         setContentView(R.layout.activity_main)
13.         findViewById<Button>(R.id.startButton).setOnClickListener {
14.             startActivity(Intent(this, GameActivity::class.java))
15.         }
16.         findViewById<Button>(R.id.authorsButton).setOnClickListener {
17.             startActivity(Intent(this, AuthorsActivity::class.java))
18.         }
19.     }
20. }
21.
```

### GameActivity.kt:

```
1. package com.example.pumlab5
2. import android.content.Intent
3. import android.os.Bundle
4. import android.os.CountDownTimer
5. import android.os.Handler
6. import android.os.Looper
7. import android.widget.Button
8. import android.widget.GridLayout
9. import android.widget.TextView
10. import androidx.appcompat.app.AppCompatActivity
11. import com.example.pumlab5.model.GameBoard
12. import com.example.pumlab5.utils.SoundManager
13. class GameActivity : AppCompatActivity() {
14.     private lateinit var board: GameBoard
15.     private lateinit var timer: CountDownTimer
16.     private lateinit var grid: GridLayout
17.     private lateinit var testWinBtn: Button
18.     private var moveCount = 0
19.     override fun onCreate(savedInstanceState: Bundle?) {
20.         super.onCreate(savedInstanceState)
21.         setContentView(R.layout.activity_game)
22.         val backBtn = findViewById<Button>(R.id.backToMenuButton)
23.         val shuffleBtn = findViewById<Button>(R.id.shuffleButton)
24.         testWinBtn = findViewById<Button>(R.id.testWinButton)
25.         backBtn.setOnClickListener {
26.             finish() // wraca do menu głównego
27.         }
28.         shuffleBtn.setOnClickListener {
29.             start()
30.         }
31.         start()
32.         handler.postDelayed(timerRunnable, 1000)
33.     }
34.     private var secondsElapsed = 0
```

```

35. private val handler = Handler(Looper.getMainLooper())
36. private val timerRunnable = object : Runnable {
37. override fun run() {
38. secondsElapsed++
39. findViewById<TextView>(R.id.timerView).text = "Czas:
40. $secondsElapsed"
41. handler.postDelayed(this, 1000)
42. }
43. }
44. private fun start() {
45. grid = findViewById(R.id.gridLayout)
46. board = GameBoard(this, grid, ::onTileMoved, ::onGameWin)
47. board.shuffle()
48. testWinBtn.setOnClickListener {
49. board.setTestWinState()
50. }
51. resetCounters()
52. // startTimer()
53. }
54. private fun onTileMoved() {
55. moveCount++
56. findViewById<TextView>(R.id.moveCounter).text = "Ruchy: $moveCount"
57. SoundManager.playMove(this)
58. }
59. private fun onGameWin() {
60. // timer.cancel()
61. SoundManager.playWin(this)
62. val intent = Intent(this, EndGameActivity::class.java)
63. intent.putExtra("moves", moveCount)
64. startActivity(intent)
65. finish()
66. }
67. private fun resetCounters() {
68. moveCount = 0
69. findViewById<TextView>(R.id.moveCounter).text = "Ruchy: $moveCount"
70. secondsElapsed = 0
71. findViewById<TextView>(R.id.timerView).text = "Czas: 0"
72. }
73. override fun onDestroy() {
74. super.onDestroy()
75. handler.removeCallbacks(timerRunnable)
76. }
77. }
78. AuthorsActivity.kt:
79. package com.example.pumlab5
80. import android.os.Bundle
81. import android.widget.Button
82. import androidx.appcompat.app.AppCompatActivity
83. class AuthorsActivity : AppCompatActivity() {
84. override fun onCreate(savedInstanceState: Bundle?) {
85. super.onCreate(savedInstanceState)
86. setContentView(R.layout.activity_authors)
87. val backButton = findViewById<Button>(R.id.backButton)
88. backButton.setOnClickListener {
89. finish() // wraca do poprzedniego ekranu
90. }
91. }
92. }
93.

```

## EndGameActivity.kt:

```

1. package com.example.pumlab5
2. import android.content.Intent
3. import android.os.Bundle
4. import android.widget.Button
5. import android.widget.TextView
6. import androidx.appcompat.app.AppCompatActivity
7. class EndGameActivity : AppCompatActivity() {
8. override fun onCreate(savedInstanceState: Bundle?) {
9. super.onCreate(savedInstanceState)
10. setContentView(R.layout.activity_end_game)
11. val moves = intent.getIntExtra("moves", 0)
12. findViewById<TextView>(R.id.scoreText).text = "Wygrałeś! Ruchy: $moves"
13. findViewById<Button>(R.id.menuButton).setOnClickListener {
14. startActivity(Intent(this, MainActivity::class.java))
15. }
16. }

```

```
17. }
18.
```

## GameBoard.kt:

```
1. package com.example.pumlab5.model
2. import android.content.Context
3. import android.widget.GridLayout
4. import com.example.pumlab5.utils.SoundManager
5. import java.util.Collections
6. import kotlin.math.abs
7. class GameBoard(
8.     private val context: Context,
9.     private val gridLayout: GridLayout,
10.    private val onMove: () -> Unit,
11.    private val onWin: () -> Unit
12. ) { private val tiles = mutableListOf<Tile>()
13. fun shuffle() {
14.     tiles.clear()
15.     // Utwórz poprawnie ułożoną planszę: 1..15 i jedno puste pole (null)
16.     val numbers :MutableList<Int?> = (1..15).map { it }.toMutableList()
17.     numbers.add(null)
18.     numbers.forEach { number ->
19.         val tile = Tile(context, number)
20.         tile.setOnClickListener { tryMove(tile) }
21.         tiles.add(tile)
22.     }
23.     // Wykonaj 1000 losowych, poprawnych ruchów
24.     repeat(1000) { performRandomMove() }
25.     updateGrid()
26.     SoundManager.playShuffle(context)
27. }
28. private fun performRandomMove() {
29.     val emptyIndex = tiles.indexOfFirst { it.number == null }
30.     val row = emptyIndex / 4
31.     val col = emptyIndex % 4
32.     val possibleMoves = listOfNotNull(
33.         if (row > 0) emptyIndex - 4 else null, // góra
34.         if (row < 3) emptyIndex + 4 else null, // dół
35.         if (col > 0) emptyIndex - 1 else null, // lewo
36.         if (col < 3) emptyIndex + 1 else null // prawo
37.     )
38.     val moveIndex = possibleMoves.random()
39.     Collections.swap(tiles, emptyIndex, moveIndex)
40. }
41. private fun tryMove(tile: Tile) {
42.     val index = tiles.indexOf(tile)
43.     val emptyIndex = tiles.indexOfFirst { it.number == null }
44.     if (canSwap(index, emptyIndex)) {
45.         Collections.swap(tiles, index, emptyIndex)
46.         updateGrid()
47.         onMove()
48.         if (isWinning()) onWin()
49.     }
50. }
51. private fun canSwap(a: Int, b: Int): Boolean {
52.     val rowA = a / 4; val colA = a % 4
53.     val rowB = b / 4; val colB = b % 4
54.     return (rowA == rowB && abs(colA - colB) == 1) || (colA == colB &&
55.         abs(rowA - rowB) == 1)
56. }
57. private fun isWinning(): Boolean {
58.     return tiles.dropLast(1).withIndex().all { it.value.number == it.index +
59.         1 } }
60. private fun updateGrid() {
61.     gridLayout.removeAllViews()
62.     tiles.forEach { gridLayout.addView(it) }
63. }
64. fun setTestWinState() {
65.     tiles.clear()
66.     val numbers = (1..14).toMutableList() + listOf(null, 15) // tylko jeden
67.     ruch do wygranej
68.     numbers.forEach { number ->
69.         val tile = Tile(context, number)
70.         tile.setOnClickListener { tryMove(tile) }
71.         tiles.add(tile)
72.     }
73.     updateGrid()
}
```

```
74. }
75. }
76.
```

## Tile.kt:

```
1. package com.example.pumlab5.model
2. import android.view.Gravity
3. import android.content.Context
4. import android.view.ViewGroup
5. import androidx.appcompat.widget.AppCompatButton
6. class Tile(context: Context, val number: Int?) : AppCompatButton(context) {
7.     init {
8.         layoutParams = ViewGroup.LayoutParams(200, 200)
9.         text = number?.toString() ?: ""
10.        gravity = Gravity.CENTER
11.        textSize = 24f
12.    }
13. }
14.
```

## Activity.main.xml:

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:orientation="vertical"
6.     android:gravity="center"
7.     android:padding="24dp">
8.     <Button
9.         android:id="@+id/startButton"
10.        android:layout_width="wrap_content"
11.        android:layout_height="wrap_content"
12.        android:text="Start gry" />
13.     <Button
14.         android:id="@+id/authorsButton"
15.         android:layout_width="wrap_content"
16.         android:layout_height="wrap_content"
17.         android:text="Autorzy"
18.         android:layout_marginTop="16dp" />
19. </LinearLayout>
20. Activity.game.xml:
21. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
22.     android:layout_width="match_parent"
23.     android:layout_height="match_parent"
24.     android:orientation="vertical"
25.     android:padding="16dp">
26.     <LinearLayout
27.         android:layout_width="match_parent"
28.         android:layout_height="wrap_content"
29.         android:orientation="horizontal">
30.         <TextView
31.             android:id="@+id/timerView"
32.             android:layout_width="0dp"
33.             android:layout_height="wrap_content"
34.             android:text="Czas: 0"
35.             android:textSize="18sp"
36.             android:layout_weight="1" />
37.         <TextView
38.             android:id="@+id/moveCounter"
39.             android:layout_width="0dp"
40.             android:layout_height="wrap_content"
41.             android:text="Ruchy: 0"
42.             android:textSize="18sp"
43.             android:gravity="end"
44.             android:layout_weight="1" />
45.     </LinearLayout>
46.     <GridLayout
47.         android:id="@+id/gridLayout"
48.         android:layout_width="match_parent"
49.         android:layout_height="wrap_content"
50.         android:columnCount="4"
51.         android:rowCount="4"
52.         android:layout_marginTop="24dp"
53.         android:useDefaultMargins="true" />
54.     <LinearLayout
55.         android:layout_width="match_parent"
56.         android:layout_height="wrap_content">
```

```

57. android:orientation="horizontal"
58. android:gravity="center"
59. android:layout_marginTop="24dp"
60. android:weightSum="3">
61. <Button
62. android:id="@+id/backToMenuButton"
63. android:layout_width="0dp"
64. android:layout_height="wrap_content"
65. android:layout_weight="1"
66. android:text="Powrót" />
67. <Button
68. android:id="@+id/shuffleButton"
69. android:layout_width="0dp"
70. android:layout_height="wrap_content"
71. android:layout_weight="1"
72. android:text="Mieszaj" />
73. <Button
74. android:id="@+id/testWinButton"
75. android:layout_width="0dp"
76. android:layout_height="wrap_content"
77. android:layout_weight="1"
78. android:text="Test wygranej" />
79. </LinearLayout>
80. </LinearLayout>
81. Activity_end_game.xml:
82. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
83. android:layout_width="match_parent"
84. android:layout_height="match_parent"
85. android:orientation="vertical"
86. android:gravity="center"
87. android:padding="24dp">
88. <TextView
89. android:id="@+id/scoreText"
90. android:layout_width="wrap_content"
91. android:layout_height="wrap_content"
92. android:text="Wygrałeś!"
93. android:textSize="24sp"
94. android:textStyle="bold"
95. android:layout_marginBottom="32dp" />
96. <Button
97. android:id="@+id/menuButton"
98. android:layout_width="wrap_content"
99. android:layout_height="wrap_content"
100. android:text="Powrót do menu" />
101. </LinearLayout>
102.

```

## 5. Funkcje kluczowe

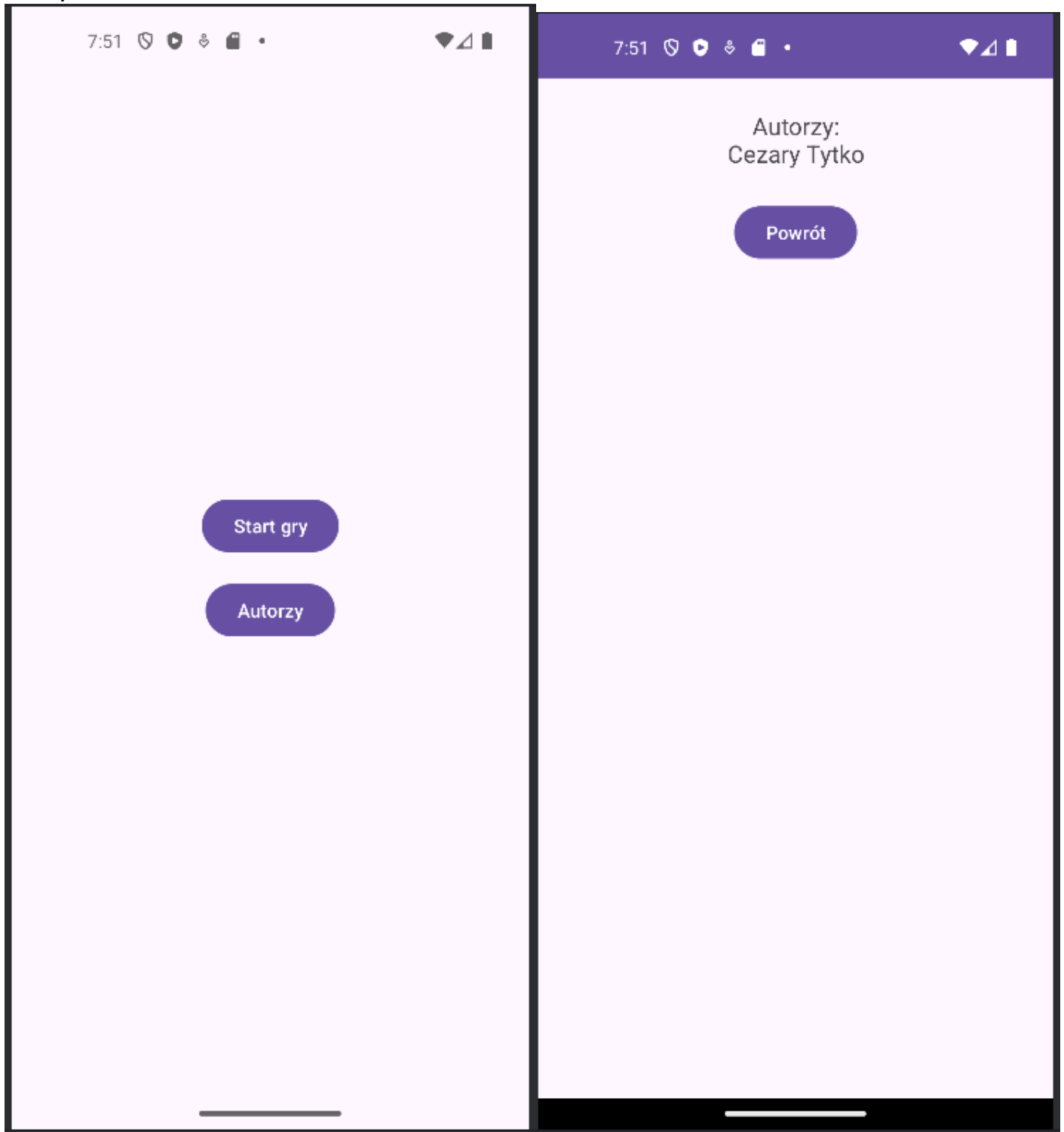
- Test wygranej – gra automatycznie ustawia wszystkie kawałki oprócz jednego, umożliwiając sprawdzenie, czy program odpowiednio reaguje na spełnienie warunku wygranej.
- Obsługa menu i przejść między widokami.
- Mieszanie kawałków – utworzenie nowej kombinacji.
- Wyświetlanie komunikatów – informowanie użytkownika o wygranej lub przegranej.
- Resetowanie gry – użytkownik może rozpocząć nową rozgrywkę.

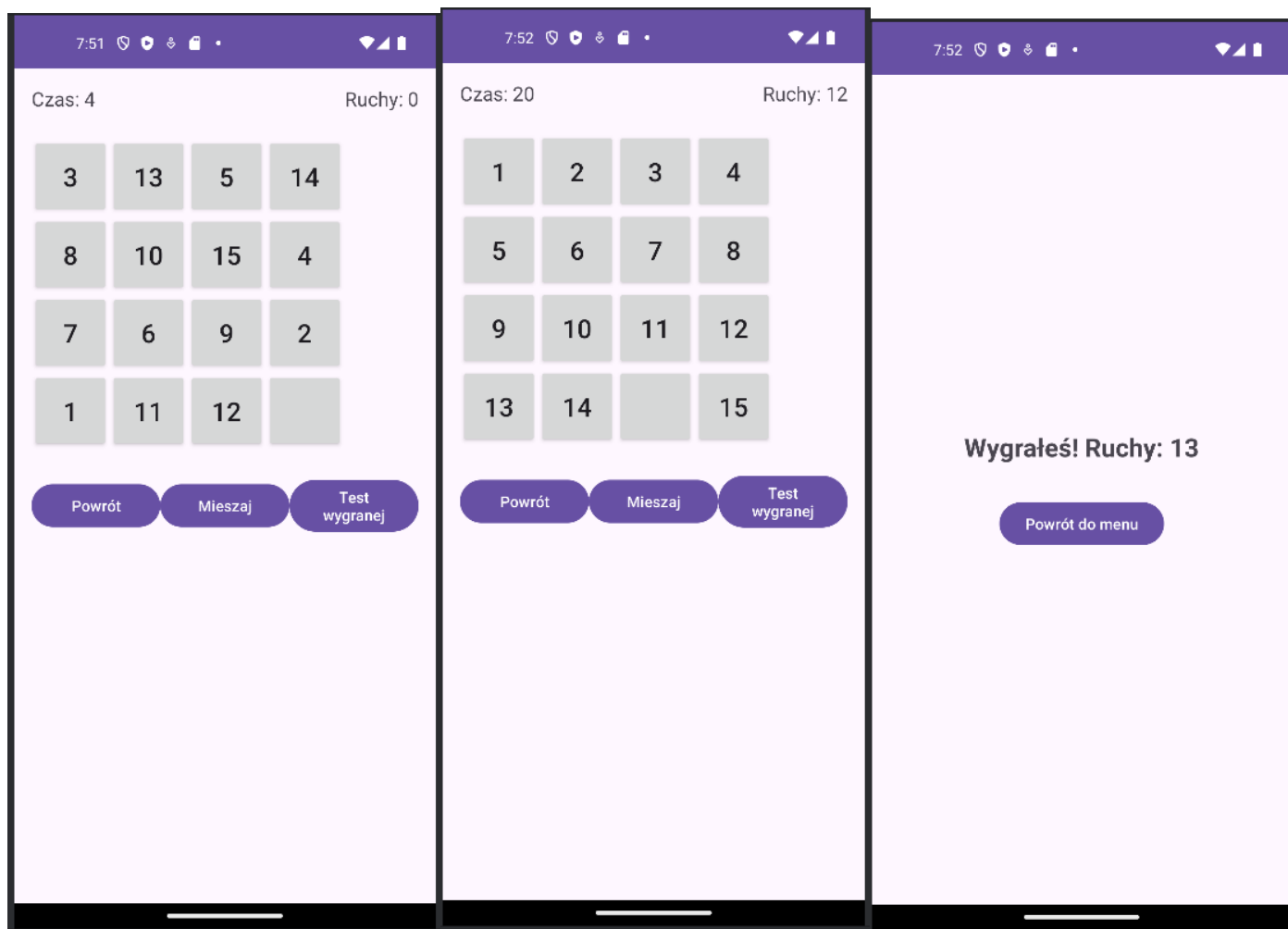
## 6. Testowanie

Testowanie gry obejmowało:

- Sprawdzenie czy gra reaguje na spełnienie warunku wygranej.
- Sprawdzenie losowości funkcji mieszania.
- Sprawdzenie, czy licznik czasu faktycznie działa.
- Możliwość przesuwania kawałków we wszystkie cztery strony.

## 7. Wyniki





## 8. Podsumowanie

Projekt realizuje klasyczną grę "w 15" 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.