Android Studio - Gra w kości

© Copyright by 3bird Projects 2024, http://edukacja.3bird.pl

Ogólne

Aplikacja napisana w języku *Kotlin + JetPack Compose*. Użytkownik rzuca kośćmi 5 razy i za każdym razem sumuje ilość zdobytych oczek (punktów). Może w ten sposób rywalizować z innym użytkownikiem aplikacji.

Do projektu należy skopiować pliki graficzne (kostki z oczkami) oraz - w przypadku wersji rozbudowanej - także plik dźwiękowy mp3.

Pliki graficzne: *folderProjektu / app / src / main / res / drawable / **

Pliki dźwiękowe: *folderProjektu / app / src / main / res / raw / **

Uwaga: Nigdy nie wolno kopiować kodu z PDF-a, gdyż zawiera on niewidoczne znaki końca linii, twarde odstępy, odmienne cudzysłowy, odmienne apostrofy, odmienne myślniki. To godziny dodatkowej pracy na wykrywanie błędów i ich poprawianie. Kod należy przepisać ze zrozumieniem.

Kod - wersja podstawowa

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            nazwaAplikacjiTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    WszystkieElementy()
                }
            }
        }
    }
}

@Composable
fun WszystkieElementy() {
    var oczka by remember { mutableStateOf(listOf<Int>()) }
    var wynikRzutu by remember { mutableIntStateOf(0) }
    var sumaOgolna by remember { mutableIntStateOf(0) }
    var iloscProb by remember { mutableIntStateOf(0) }
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Top,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Spacer(modifier = Modifier.height(40.dp))
```

```
Row(
    horizontalArrangement = Arrangement.Center,
    verticalAlignment = Alignment.CenterVertically,
    modifier = Modifier.fillMaxWidth()
) {
    // Jeśli liczby nie zostały jeszcze wylosowane (aplikację dopiero uruchomiono),
    // wtedy wyświetl znaki zapytania:
    if (oczka.isEmpty()) {
        oczka = listOf(0, 0, 0, 0, 0)
        repeat(5) {
            Image(
                painter = painterResource(id = R.drawable.znak_zapytania),
                contentDescription = "Znaki zapytania",
                modifier = Modifier
                    .size(64.dp)
                    .padding(4.dp)
            )
        }
    } else {
        oczka.forEach { liczba ->
            Image(
                painter = painterResource(id = wczytajObrazek(liczba)),
                contentDescription = "Wartość oczek: $liczba",
                modifier = Modifier
                    .size(64.dp)
                    .padding(4.dp)
            )
        }
    }
}

Spacer(modifier = Modifier.height(40.dp))
val tekstWylosowaneOczka = oczka.joinToString(", ", "", ".")
Text("Wylosowane oczka: $tekstWylosowaneOczka", fontSize = 20.sp)

Spacer(modifier = Modifier.height(16.dp))
val tekstSumaOczek = "Suma oczek: $wynikRzutu."
Text(tekstSumaOczek, fontSize = 20.sp)

Spacer(modifier = Modifier.height(16.dp))
val tekstIloscRzutow = "Ilość prób: $iloscProb."
Text(tekstIloscRzutow, fontSize = 20.sp)

Spacer(modifier = Modifier.height(16.dp))
val tekstSumaOgolna = "Suma ogólna: $sumaOgolna."
Text(tekstSumaOgolna, fontSize = 20.sp)

Spacer(modifier = Modifier.height(16.dp))
```

```
        Row {
            Button(onClick = {
                if (iloscProb >= 5) {
                    iloscProb =  0
                    wynikRzutu = 0
                    sumaOgolna = 0
                }
                oczka = List(5) { Random.nextInt(1, 7) }
                wynikRzutu = oczka.sum()
                iloscProb += 1
                sumaOgolna += wynikRzutu
            }, modifier = Modifier.padding(10.dp)) {
                Text("Losuj")
            }
        }
    }
}

fun wczytajObrazek(liczba: Int): Int {
    return when (liczba) {
        1 -> R.drawable.kostka1
        2 -> R.drawable.kostka2
        3 -> R.drawable.kostka3
        4 -> R.drawable.kostka4
        5 -> R.drawable.kostka5
        6 -> R.drawable.kostka6
        else -> R.drawable.znak_zapytania
    }
}
```

Kod - wersja rozbudowana

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            GraWKosciTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    WszystkieElementy()
                }
            }
        }
    }
}
```

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun WszystkieElementy() {
    val kontekstDlaWszystkichElementow = LocalContext.current
    // Dzięki "remember", stan poniższych zmiennych jest zachowywany
    // w komponencie pomiędzy rekonstrukcjami interfejsu:
    var oczka by remember { mutableStateOf(listOf<Int>()) }
    var wynikRzutu by remember { mutableIntStateOf(0) }
    var sumaOgolna by remember { mutableIntStateOf(0) }
    var iloscProb by remember { mutableIntStateOf(0) }
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Top,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        CenterAlignedTopAppBar(
            title = { Text("Gra w kości", fontWeight = FontWeight.ExtraBold, fontSize = 34.sp) },
            colors = TopAppBarDefaults.topAppBarColors(
                containerColor = Color(47, 79, 79, 255),
                titleContentColor = Color.White
            )
        )
        Spacer(modifier = Modifier.height(40.dp))
        Row(
            horizontalArrangement = Arrangement.Center,
            verticalAlignment = Alignment.CenterVertically,
            modifier = Modifier.fillMaxWidth()
        ) {
            // Jeśli liczby nie zostały jeszcze wylosowane (aplikację dopiero uruchomiono),
            // wtedy wyświetl znaki zapytania:
            if (oczka.isEmpty()) {
                oczka = listOf(0, 0, 0, 0, 0)
                repeat(5) {
                    Image(
                        painter = painterResource(id = R.drawable.znak_zapytania),
                        contentDescription = "Znaki zapytania",
                        modifier = Modifier
                            .size(64.dp)
                            .padding(4.dp)
                    )
                }
                // W przeciwnym przypadku, gdy liczby zostały już wylosowane,
                // wyświetl kostki z odpowiednią ilością oczek:
            } else {
                // W pętli uruchamiana jest funkcja "wczytajObrazek()",
                // która przyjmuje jako parametr kolejne wylosowane liczby:
                oczka.forEach { liczba ->
                    Image(
```

```
                painter = painterResource(id = wczytajObrazek(liczba)),
                contentDescription = "Wartość oczek: $liczba",
                modifier = Modifier
                    .size(64.dp)
                    .padding(4.dp)
            )
        }
    }
}

// ================= Napis "Wylosowane oczka" ==================
Spacer(modifier = Modifier.height(40.dp))
// Można to zrobić tak jak poniżej, ale wtedy nie będziemy mogli pokolorować przecinków:
// val tekstWylosowaneOczka = oczka.joinToString(", ", "", ".")
// Dlatego jednak wydobywamy liczby z listy i za każdym razem dodajemy szary przecinek:
val tekstWylosowaneOczka = buildAnnotatedString {
    withStyle(style = SpanStyle(color = Color.Gray)) {
        append("Wylosowane oczka: ")
    }
    // Zmienna "oczka" jest listą, więc możemy każdy element na podstawie indeksu
    // przyporządkować do zmiennej "liczba". Całość robimy w pętli:
    oczka.forEachIndexed { index, liczba ->
        append("$liczba")
        // Sprawdzamy, czy bieżący element jest ostatnim elementem na liście.
        // Jeśli jest ostatnim elementem, nie postawi przecinka (na końcu).
        // Mniejszy niż "-1", gdyż indeksy liczone są od zera.
        if (index < (oczka.size - 1)) {
            withStyle(style = SpanStyle(color = Color.Gray)) {
                append(", ")
            }
        }
    }
    withStyle(style = SpanStyle(color = Color.Gray)) {
        append(".")
    }
}
Text(tekstWylosowaneOczka, fontSize = 20.sp)

// ============== Napis "Suma oczek" ================
Spacer(modifier = Modifier.height(16.dp))
val tekstSumaOczek = buildAnnotatedString {
    withStyle(style = SpanStyle(color = Color.Gray)) {
        append("Suma oczek: ")
    }
    append("$wynikRzutu") // Bez kolorowania
    withStyle(style = SpanStyle(color = Color.Gray)) {
        append(".")
    }
```

```
}
Text(tekstSumaOczek, fontSize = 20.sp)

// ============== Napis "Ilość rzutów" ==============
Spacer(modifier = Modifier.height(16.dp))
val tekstIloscRzutow = buildAnnotatedString {
    withStyle(style = SpanStyle(color = Color.Gray)) {
        append("Ilość prób: ")
    }
    append("$iloscProb") // Bez kolorowania
    withStyle(style = SpanStyle(color = Color.Gray)) {
        append(".")
    }
}
Text(tekstIloscRzutow, fontSize = 20.sp)

// ============== Napis "Suma ogólna" ===============
Spacer(modifier = Modifier.height(16.dp))
val tekstSumaOgolna = buildAnnotatedString {
    withStyle(style = SpanStyle(color = Color.Gray)) {
        append("Suma ogólna: ")
    }
    withStyle(
        style = SpanStyle(
            color = Color.Red,
            fontSize = 24.sp,
            fontWeight = FontWeight.Bold
        )
    ) {
        append("$sumaOgolna")
    }
    withStyle(style = SpanStyle(color = Color.Gray)) {
        append(".")
    }
}
Text(tekstSumaOgolna, fontSize = 20.sp)

// ============== Przycisk LOSUJ ================
Spacer(modifier = Modifier.height(16.dp))
Row {
    // Funkcja Button() aktualizuje zmienne zapisane w "remember":
    Button(onClick = {
        // Jeśli ilość prób osiągnęła liczbę 5, zresetuj wyniki:
        if (iloscProb >= 5) {
            iloscProb = 0
            wynikRzutu = 0
            sumaOgolna = 0
        }
```

```
                // Bez względu na powyższe, wykonaj poniższe:
                oczka = List(5) { Random.nextInt(1, 7) }
                wynikRzutu = oczka.sum()
                iloscProb += 1
                sumaOgolna += wynikRzutu
                if (iloscProb <= 4) {
                    playBeep()
                } else {
                    // Jeśli to jest piąta próba:
                    playSound(kontekstDlaWszystkichElementow)
                    Toast.makeText(
                        kontekstDlaWszystkichElementow,
                        "Koniec gry. Zdobyto $sumaOgolna punktów.",
                        Toast.LENGTH_LONG
                    ).show()
                }
            }, modifier = Modifier.padding(10.dp)) {
                Text("Losuj")
            }
            // Przycisk resetu:
            Button(onClick = {
                oczka = emptyList()
                wynikRzutu = 0
                iloscProb = 0
                sumaOgolna = 0
            }, modifier = Modifier.padding(10.dp)) {
                Text("Reset")
            }
        }
    }
}

fun wczytajObrazek(liczba: Int): Int {
    // Zwróć, jeśli wylosowane zostały liczby:
    return when (liczba) {
        // Uwaga: Nazwy plików graficznych nie mogą zawierać myślnika / łącznika:
        1 -> R.drawable.kostka1
        2 -> R.drawable.kostka2
        3 -> R.drawable.kostka3
        4 -> R.drawable.kostka4
        5 -> R.drawable.kostka5
        6 -> R.drawable.kostka6
        else -> R.drawable.znak_zapytania // Domyślna wartość
    }
}

// Odtwarza "beep":
private fun playBeep() {
```

```
    val generatorTonow = ToneGenerator(AudioManager.STREAM_MUSIC, 80)
    generatorTonow.startTone(ToneGenerator.TONE_PROP_BEEP, 200)
}

// Odtwarza dźwięk *.mp3 (fanfary):
private fun playSound(context: Context) {
    val odtwarzaczMediow = MediaPlayer.create(context, R.raw.goodresult)
    if (odtwarzaczMediow.isPlaying) {
        odtwarzaczMediow.stop()
        odtwarzaczMediow.reset()
    }
    odtwarzaczMediow.setVolume(0.1f, 0.1f)
    odtwarzaczMediow.start()
}


@Preview(showBackground = true)
@Composable
fun Podglad() {
    GraWKosciTheme {
        WszystkieElementy()
    }
}
```

Uwaga: W wersji rozbudowanej należy także do aplikacji przypisać swoją własną ikonę.

Ostatnia aktualizacja: 30 czerwca 2024.