|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Politechnika 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 |  |
| Nr laboratorium | 4 | Data wykonania |  | Grupa |  |
| Ocena |  | Data oddania |  | Imię Nazwisko |  |
| Nazwa ćwiczenia | Interfejs użytkownika, wyświetlanie obrazów, dźwięki | | |

**Cel ćwiczenia laboratoryjnego**

Celem ćwiczenia jest zapoznanie studentów z tworzeniem prostej gry mobilnej na platformie Android przy użyciu interfejsu użytkownika **EditText,** **Button,** **TextView, EditText, ImageView, MediaPlayer**. Studenci mają nauczyć się:

1. Projektować interfejs użytkownika, który obejmuje wprowadzanie danych, przycisk do sprawdzania odpowiedzi oraz wyświetlanie wyników i komunikatów.
2. Programować obsługę zdarzeń związaną z rozgrywką.
3. Wykonywać proste operacje na danych wejściowych.
4. Wyświetlać wynik, obrazy i komunikaty na ekranie w czytelny sposób.
5. Obsługiwać odtwarzanie dźwięków w aplikacji.

**Zadanie**

Studenci zostaną poproszeni o rozbudowanie aplikacji, którą utworzyli na poprzednich zajęciach. Program ma wyświetlać obrazy ilustrujące rozgrywkę. Dodatkowo, w zależności od stanu gry odtwarzane będą odpowiednie dźwięki.

**Opcjonalne rozszerzenia**

* Możesz dodać bardziej rozbudowany interfejs użytkownika, który pozwoli graczowi wybierać słowa lub odczytywać je z pliku.
* Możesz rozszerzyć listę słów, które gracz może odgadywać, i losowo wybierać jedno z tych słów.
* Możesz dodać zapis wyników gry i wyników graczy.

**Oczekiwane wyniki**

Po zakończeniu tego ćwiczenia studenci powinni być w stanie:

1. Tworzyć interfejsy użytkownika w aplikacji mobilnej.
2. Programować obsługę zdarzeń związaną z rozgrywką.
3. Przeprowadzać logikę gry w wisielca.
4. Wyświetlać stan gry na ekranie w formie tekstu.
5. Zrozumieć i obsługiwać błędy w grze.
6. Testować i debugować grę.
7. Projektować interaktywne rozwiązania gier.
8. Rozwiązywać problemy związane z logiką gry.
9. Organizować kod i dokumentować go.
10. Tworzyć aplikacje mobilne w Android Studio.

**Zadanie**

Należy rozbudować aplikację wisielca. Program ma wyświetlać obrazek związany z rozgrywką gry oraz informować o stanie gry za pomocą dźwięków.

Obraz zawierający tekst, zrzut ekranu, linia, design

Opis wygenerowany automatycznie

**Sprawozdanie**

Sprawozdanie z ćwiczenia w ramach nauki programowania aplikacji mobilnych powinno zawierać istotne informacje i dokumentację dotyczącą zadania oraz jego realizacji. Oto kilka kluczowych elementów, które powinny być uwzględnione w sprawozdaniu:

**1. Tytuł i informacje ogólne**

* Tytuł ćwiczenia.
* Imię i nazwisko studenta.
* Data realizacji ćwiczenia.

**2. Cel ćwiczenia:**

* Krótka informacja o celu i znaczeniu ćwiczenia, jak również o tym, czego studenci mieli się nauczyć.

**3. Opis projektu:**

* Opis funkcjonalności gry Wisielec w ramach projektu.
* Charakteryzacja interfejsu użytkownika, zawierająca informacje o wykorzystanych widokach i elementach interfejsu.

**4. Implementacja:**

* Opis procesu tworzenia projektu, wraz z krokami realizacji gry Wisielec.
* Omówienie wykorzystanych narzędzi i technologii, w tym Android Studio, język Java/Kotlin.
* Przedstawienie kodu źródłowego aplikacji, zarówno XML (layout) jak i kodu Java/Kotlin.

**5. Funkcje kluczowe:**

* Omówienie kluczowych funkcji aplikacji, takich jak obsługa zgadywania liter, zarządzanie stanem gry oraz sposób prezentacji wyników.

**6. Testowanie:**

* Opis testowania gry Wisielec, włączając w to przykłady testów przeprowadzonych w trakcie implementacji.
* Przykładowe przypadki testowe i raport z wynikami testów.

**7. Wyniki:**

* Przedstawienie wyników działania aplikacji, w tym zrzuty ekranu demonstrujące działanie gry.

**8. Podsumowanie:**

* Krótka ocena projektu i osiągnięć w kontekście zrealizowanego zadania.
* Wnioski wynikające z ćwiczenia, jakie umiejętności i doświadczenie zdobyli studenci.

**9. Trudności i błędy:**

* Informacje na temat ewentualnych problemów napotkanych podczas implementacji gry i jak zostały one rozwiązane.

**10. Źródła i odniesienia:**

* Jeśli korzystano z materiałów lub źródeł zewnętrznych, uwzględnij je w tekście.

**11. Dodatkowe materiały:**

* Ewentualne dodatkowe materiały, takie jak kody źródłowe, zrzuty ekranu lub inne dokumentacje, które uzupełniają sprawozdanie.

**Zalecenia ogólne:**

* Sprawozdanie powinno być czytelne i przejrzyste, z odpowiednimi nagłówkami i numeracją stron.
* Projektowanie i implementacja powinny być opisane w sposób logiczny i zrozumiały.
* Sprawozdanie powinno być dostatecznie szczegółowe, aby inny programista mógł zrozumieć projekt i ewentualnie go udoskonalić lub wykorzystać w przyszłości.

**Przykład do wykorzystania w ćwiczeniu i opis propozycji rozwiązania zadania**

Oto przykład, który pomoże Ci utworzyć prostą aplikację w Android Studio, która będzie wyświetlać obraz w formacie PNG. Aplikacja będzie zawierać jedną aktywność (Activity) i będzie korzystać z elementu **ImageView** do wyświetlania obrazu.

Uruchom Android Studio i utwórz nowy projekt.

Znajdź lub przygotuj obraz w formacie PNG, który chcesz wyświetlić w aplikacji.

Umieść ten obraz w folderze **res/drawable** w katalogu projektu. Jeśli folder **drawable** nie istnieje, utwórz go ręcznie. Nazwa pliku obrazu musi być z małymi literami i nie zawierać znaków specjalnych, np. **sample\_image.png**.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Otwórz plik **activity\_main.xml** w folderze **res/layout**. To jest plik układu dla głównej aktywności.

Znajdź paletę widoków (**Palette**) i przeciągnij element **ImageView** na obszar projektowania.

W właściwościach **ImageView** zmień atrybut **src** na **@drawable/sample\_image**, gdzie **sample\_image** to nazwa pliku obrazu, który dodałeś w poprzednim kroku.

Otwórz plik **MainActivity.java**. Możesz znaleźć go w folderze **java** w pakiecie Twojej aplikacji.

W metodzie **onCreate**, utwórz **ImageView**, który stworzyłeś w pliku XML, używając **findViewById**.

**ImageView imageView = findViewById(R.id.imageView);**

Ustaw obraz na **ImageView**, używając metody **setImageResource** i podając identyfikator zasobu obrazu:

**imageView.setImageResource(R.drawable.sample\_image);**

Uruchom aplikację na emulatorze lub podłączonym urządzeniu Android.

Teraz Twoja aplikacja powinna wyświetlać obraz PNG na ekranie głównej aktywności.

Obraz zawierający zrzut ekranu, tekst, Grafika, logo

Opis wygenerowany automatycznie

Poniżej przedstawiona jest implementacja przykładu wyświetlającego obraz w aktywności:

package com.example.zadanie7a;  
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
import android.widget.ImageView;  
  
public class MainActivity extends AppCompatActivity {  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);

// Znajdź ImageView w układzie  
 ImageView imageView = findViewById(R.id.*imageView*);

// Ustaw obraz na ImageView  
 imageView.setImageResource(R.drawable.*android\_studio*);  
 }  
}

Dodaj do projektu z poprzedniego laboratorium rysunki z archiwum dołączone do tej instrukcji, tak aby w gałęzi **drawable** dostępne były wszystkie potrzebne obrazy:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Teraz dodaj do układu element **ImageView**:

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

W oknie Pick a Resource wybierz obraz image0.png

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

W ten sposób do naszego projektu dodaliśmy miejsce, w którym można wyświetlać obrazki. Teraz np. za pomocą instrukcji:

**obrazekimageView.setImageResource(R.drawable.*image\_0*);**

teraz możemy w naszej aplikacji zmieniać zawartość naszego elementu **ImageView**. Uzupełnij kod aplikacji, tak aby w zależności od liczby pozostałych prób wyświetlała się odpowiednia grafika:

Obraz zawierający tekst, zrzut ekranu, linia, design

Opis wygenerowany automatycznie

W dalszej części dowiesz się jak dodać efekty dźwiękowe. W tym celu, dla przećwiczenia nowej umiejętności otwórz nowy projekt, w którym bez obawy można eksperymentować. Poniżej znajduje się przykład prostej aplikacji Android, która odtwarza dźwięki. W tym przykładzie użyjemy elementu **MediaPlayer** do odtwarzania dźwięków z plików przechowywanych w folderze **raw** w zasobach projektu.

Obraz zawierający tekst, oprogramowanie, numer, Ikona komputerowa

Opis wygenerowany automatycznie

Przygotowanie plików dźwiękowych (pliki są w archiwum dołączonym do instrukcji).

Proponuję pobrać plik dźwiękowe z serwisu pixbay <https://pixabay.com/pl/sound-effects/>

cload

<https://pixabay.com/pl/sound-effects/atari-800xl-cload-command-buzzer-47923/>

windows

<https://pixabay.com/pl/sound-effects/windows-error-sound-effect-35894/>

Dla ułatwienia zmieniamy nazwy oryginalnych plików:

cload.mp3

windows.mp3

Umieść te pliki dźwiękowe (cload.mp3 i windows.mp3) w folderze **res/raw** w katalogu projektu. Jeśli folder **raw** nie istnieje, utwórz go.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Projektowanie interfejsu użytkownika

Otwórz plik **activity\_main.xml** i dodaj dwa przyciski:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:gravity="center"  
 tools:context=".MainActivity">  
  
 <Button  
 android:id="@+id/buttonSound1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Sound 1" />  
  
 <Button  
 android:id="@+id/buttonSound2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Sound 2" />  
  
</LinearLayout>

Otwórz plik MainActivity.java i dodaj kod do obsługi odtwarzania dźwięków:

package com.example.zadanie7b;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.media.MediaPlayer;  
import android.os.Bundle;  
import android.provider.MediaStore;  
import android.view.View;  
import android.widget.Button;  
  
public class MainActivity extends AppCompatActivity {  
  
  
 private MediaPlayer mediaPlayer;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 *// inicjalizacja przycisków* Button buttonSound1 = findViewById(R.id.*buttonSound1*);  
 Button buttonSound2 = findViewById(R.id.*buttonSound2*);  
  
 *// utwórz obiekt MediaPlayer* mediaPlayer = new MediaPlayer();  
  
 *// obsługa przycisku Sound 1* buttonSound1.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View view) {  
 playSound(R.raw.*cload*);  
 }  
 });  
  
 *// obsługa przycisku Sound 2* buttonSound2.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View view){  
 playSound(R.raw.*windows*);  
 }  
 });  
  
 }  
  
 private void playSound(int soundResource) {  
 try {  
 mediaPlayer.reset();  
 mediaPlayer = MediaPlayer.*create*(this, soundResource);  
 mediaPlayer.start();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 protected void onDestroy() {  
 if (mediaPlayer!=null){  
 mediaPlayer.release();  
 mediaPlayer=null;  
 }  
 super.onDestroy();  
 }  
}

Uruchom aplikację na emulatorze lub urządzeniu Android. Po kliknięciu przycisków "**Sound 1**" i "**Sound 2**" odpowiednie dźwięki powinny być odtwarzane. Upewnij się, że dźwięki są poprawnie umieszczone w folderze raw. Sprawdź ustawienia emulatora AVD i w przypadku konieczności zmiany głośności urządzenia dokonaj odpowiedniej regulacji.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

**buttonSound1.setOnClickListener** Ustawia nasłuchiwanie kliknięcia na przycisk "Sound 1" i wywołuje playSound(R.raw.sound1) po kliknięciu.

**buttonSound2.setOnClickListener** Ustawia nasłuchiwanie kliknięcia na przycisk "Sound 2" i wywołuje playSound(R.raw.sound2) po kliknięciu.

**private void playSound(int soundResource)** Jest to prywatna metoda, która przyjmuje jako argument zasób dźwięku soundResource.

**mediaPlayer.reset()** Resetuje stan odtwarzacza, aby umożliwić odtworzenie nowego dźwięku.

**mediaPlayer = MediaPlayer.create(this, soundResource)** Tworzy nowy obiekt MediaPlayer i ustawia go na podstawie zasobu dźwięku soundResource.

**mediaPlayer.start()** Rozpoczyna odtwarzanie dźwięku.

**onDestroy()** Jest to metoda cyklu życia aktywności, która jest wywoływana, gdy aktywność jest niszczona.

**mediaPlayer.release()** Zwalnia zasoby odtwarzacza dźwięku.

**super.onDestroy()** Wywołuje metodę bazową.

Po przetestowaniu i eksperymentowaniu z nowymi funkcjami aplikacji możemy wrócić do naszego wisielca i dodać dźwięki do gry. Proponuję dodać plik pobrane z wspomnianego wcześniej serwisu (plik są również dołączone do instrukcji ćwiczenia).

Start - Bang

<https://pixabay.com/pl/sound-effects/bang-140381/>

Niepoprawna odpowiedz - Failure 4 With Gong

<https://pixabay.com/pl/sound-effects/failure-4-with-gong-69906/>

Poprawna odpowiedz - Short Crowd Cheer

<https://pixabay.com/pl/sound-effects/short-crowd-cheer-6713/>

Wygrana - Congratulations (Deep Voice)

<https://pixabay.com/pl/sound-effects/congratulations-deep-voice-172193/>

Przegrana - Sad Piano Chord Progression

<https://pixabay.com/pl/sound-effects/sad-piano-chord-progression-27296/>

Dla ułatwienia zmieniamy nazwy oryginalnych plików:

start.mp3

niepoprawna.mp3

poprawna.mp3

wygrana.mp3

przegrana.mp3

Dodajemy do zasobów aplikacji do katalogu **res/raw**:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Teraz za pomocą instrukcji:

**playSound(R.raw.*start*);**

**playSound(R.raw.wygrana);**

**playSound(R.raw.przegrana);**

**playSound(R.raw.*poprawna*);**

**playSound(R.raw.niepoprawna);**

Można w odpowiednich miejscach „odgrywać” dźwięki, które będą towarzyszyć rozgrywce. Przykładowe rozwiązanie dołączone jest do instrukcji w formacie wideo.