**Instytut Informatyki  
Kolegium Nauk Przyrodniczych  
Uniwersytet Rzeszowski**

**Przedmiot:**

**Programowanie urządzeń mobilnych**

**Dokumentacja techniczna projektu:**

***Bouncy Bird***

**Wykonał:**

**Michał Zakielarz**

**Dominik Moskal**

**Prowadzący: mgr inż. Adam Szczur**

**Rzeszów 2023**

Spis treści

[1. Temat projektu i nazwa aplikacji 3](#_Toc157367000)

[2. Cel projektu 3](#_Toc157367001)

[2.1. Cele szczegółowe 3](#_Toc157367002)

[3. Funkcjonalności aplikacji 3](#_Toc157367003)

[4. Technologie 4](#_Toc157367004)

[5. Interesariusze aplikacji 4](#_Toc157367005)

[6. Harmonogram realizacji projektu 4](#_Toc157367006)

[6.1. Wykres Gannta 4](#_Toc157367007)

[7. Projekt GUI 5](#_Toc157367008)

[8. Struktura programu 11](#_Toc157367009)

[8.1. Dane wykorzystywane przez program 11](#_Toc157367010)

[8.2. Opis plików zewnętrznych 11](#_Toc157367011)

[8.3. Podział na moduły, komunikacja między modułami 11](#_Toc157367012)

[9. Diagramy UML 13](#_Toc157367013)

[9.1. Diagram przypadków użycia 13](#_Toc157367014)

[9.2. Diagram czynności / aktywności 13](#_Toc157367015)

[10. Literatura 14](#_Toc157367016)

# Temat projektu i nazwa aplikacji

Gra mobilna typu side-scroller.

Nazwa – Bouncy Bird

# Cel projektu

Zaprojektowanie i wykonanie oprogramowania umożliwiającego granie w grę mobilną w trybie dla pojedynczego gracza z typu side-scroller. Zapoznanie się z podstawami programowania gier na Androidzie, takimi jak silnik graficzny, logika gry, interfejs użytkownika, obsługa zdarzeń, dźwięki   
i efekty.

# Cele szczegółowe

* zaprojektowanie i implementacja logiki, grafiki, interfejsu oraz dźwięku gry
* Optymalizacja wydajności:
  + Głównym celem jest zapewnienie płynnej rozgrywki i minimalizacja opóźnień.
* Intuicyjna obsługa dotykowa:
  + Gracze powinni łatwo kontrolować swoją postacią
* Śledzenie postępów i osiągnięć:
  + Zachęcanie graczy do kontynuowania rozgrywki poprzez system osiągnięć, nagród   
    i monitorowania postępów
* wykorzystanie różnych sensorów i funkcji smartfona do urozmaicenia rozgrywki, takich jak:
  + odtwarzanie dźwięku gry
  + latarka (efekt specjalny podczas rozgrywki)
  + czujnik zbliżeniowy - blokowanie ekranu

# Funkcjonalności aplikacji

* Latarka
* Odtwarzanie dźwięku
* Wibracje

# Technologie

Język: C#  
IDE: Unity

# Interesariusze aplikacji

Gracze mobilni w każdym wieku posługujący się urządzeniem z systemem Android.

# Harmonogram realizacji projektu

* Planowanie logiki aplikacji
* Rozpisanie zadań
* Wprowadzenie zaplanowanych funkcjonalności
* Rozpisywanie poprawek
* Implementacja poprawek
* Testowanie aplikacji
* Przygotowanie dokumentacji

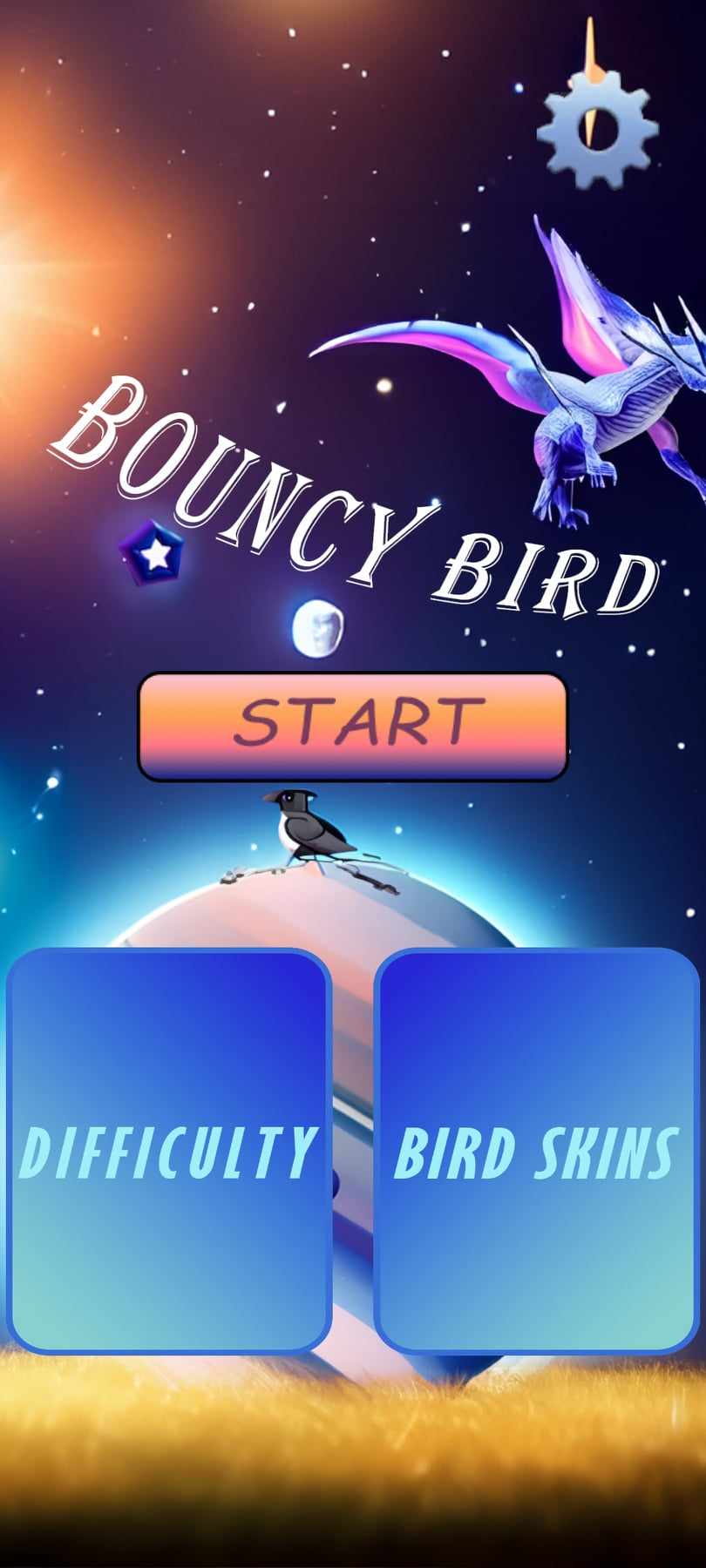
# Wykres Gannta

Obraz zawierający tekst, linia, Czcionka, numer

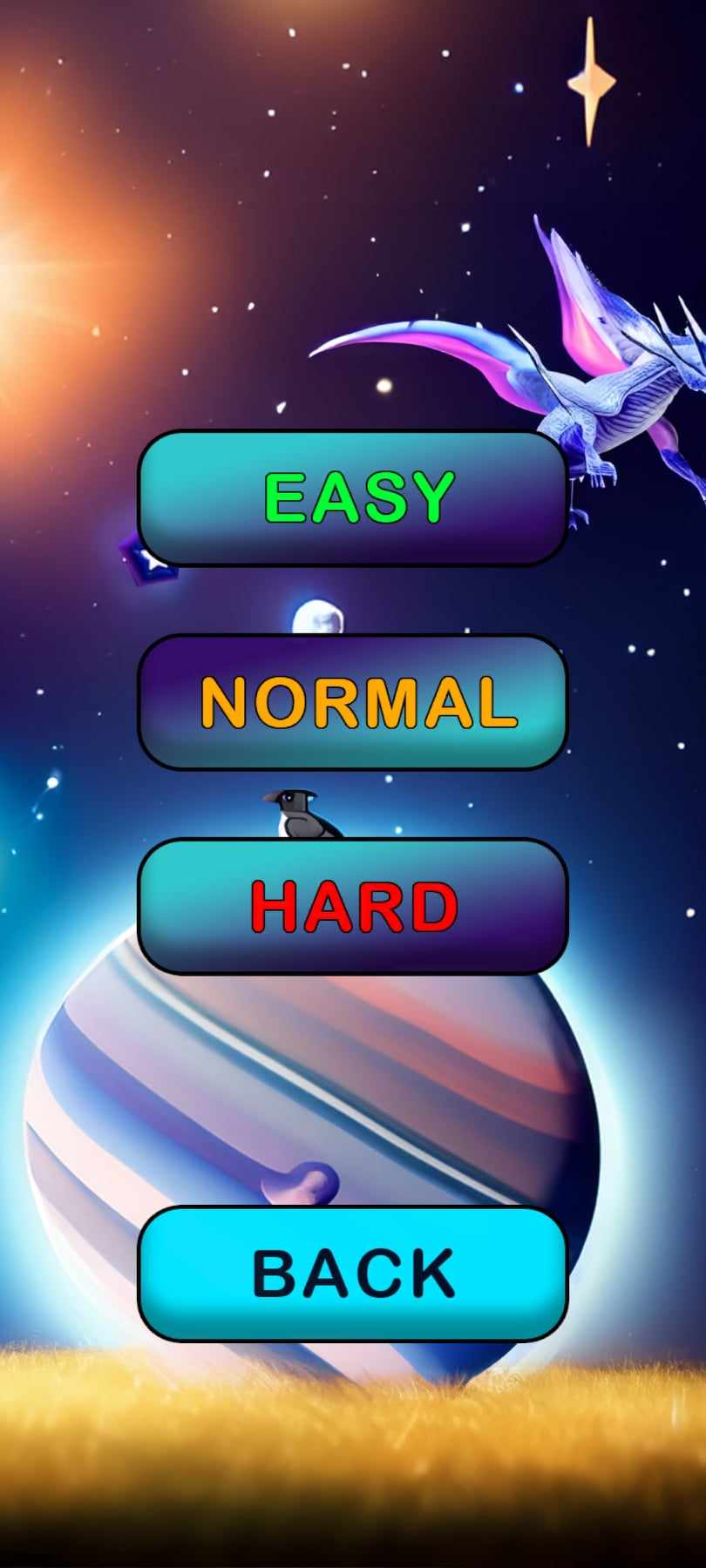
Opis wygenerowany automatycznie

Rysunek 1

# Projekt GUI



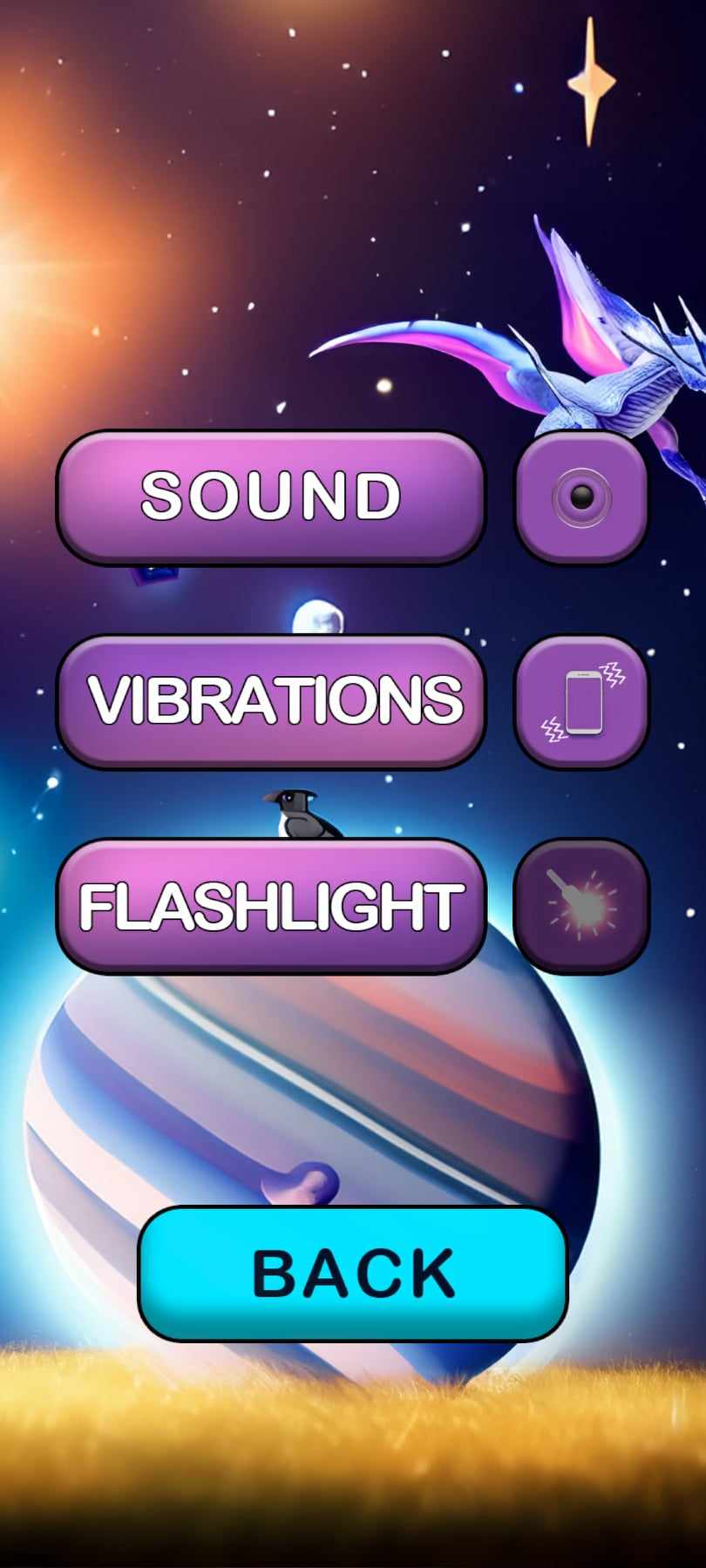
Rysunek 2 Widok menu głównego



Rysunek 3 Widok wyboru poziomu trudności



Rysunek 4 Widok wyboru wyglądu ptaka



Rysunek 5 Widok ustawień

Obraz zawierający zrzut ekranu, projekt graficzny, kreskówka

Opis wygenerowany automatycznie

Rysunek 6 Przykładowy widok rozgrywki

Obraz zawierający tekst, projekt graficzny, zrzut ekranu, Grafika

Opis wygenerowany automatycznie

Rysunek 7 Widok komunikatu końca gry

# Struktura programu

Programy w Unity C# składają się z co najmniej jednego pliku. Każdy plik zawiera zero lub więcej przestrzeni nazw. Przestrzeń nazw zawiera typy, takie jak klasy, struktury, interfejsy, wyliczenia i delegaty lub inne przestrzenie nazw. W programie, skrypty są klasami, które dziedziczą po MonoBehaviour, co pozwala na korzystanie z wielu wbudowanych funkcji Unity. MonoBehaviour to podstawowa klasa, od której dziedziczą wszystkie skrypty Unity. MonoBehaviour oferuje funkcje cyklu życia, które ułatwiają rozwijanie z Unity. MonoBehaviour zawsze istnieje jako komponent GameObject i może być instancjonowany za pomocą GameObject.AddComponent. Obiekty, które muszą istnieć niezależnie od GameObject, powinny dziedziczyć z innej klasy.

# Dane wykorzystywane przez program

Dane, które przyjmuje aplikacja :

* Stan ekranu
* Pozycja gracza
* Pozycja przeszkody
* Stan gracza

Dane wyjściowe:

* Pozycja ptaka
* Wynik
* Stan gry

# Opis plików zewnętrznych

W projekcie nie użyto plików zewnętrznych.

# Podział na moduły, komunikacja między modułami

W konwencji Unity program podzielony jest na sceny w których wywoływane są skrypty oddziałujące na poszczególne obiekty i klasy w aplikacji. Każda scena zawiera pewne zasoby   
i elementy które zawierają odwołania do innych scen i wzajemnie na siebie. Komunikacja między scenami w Unity jest możliwa dzięki klasie SceneManager. Można przełączać się między scenami, wywołując metody. Pierwszym argumentem jest nazwa sceny, na którą należy się przełączyć. Jeśli jako drugi argument zostanie określony LoadSceneMode.Single, można otworzyć tylko określoną scenę podczas zamykania bieżącej sceny.

Przykład wyżej opisanego mechanizmu:

Obraz zawierający tekst, zrzut ekranu, Czcionka, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Rysunek 8 Pasek wyboru sceny w środowisku Unity

Obraz zawierający tekst, zrzut ekranu, Oprogramowanie multimedialne, oprogramowanie

Opis wygenerowany automatycznie

Rysunek 9 Widok projektu sceny

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Rysunek 10 Skrypt z klasą SceneManager odpowiedzialny za komunikację między scenami

# Diagramy UML

# Diagram przypadków użycia

Obraz zawierający tekst, zrzut ekranu, krąg, diagram

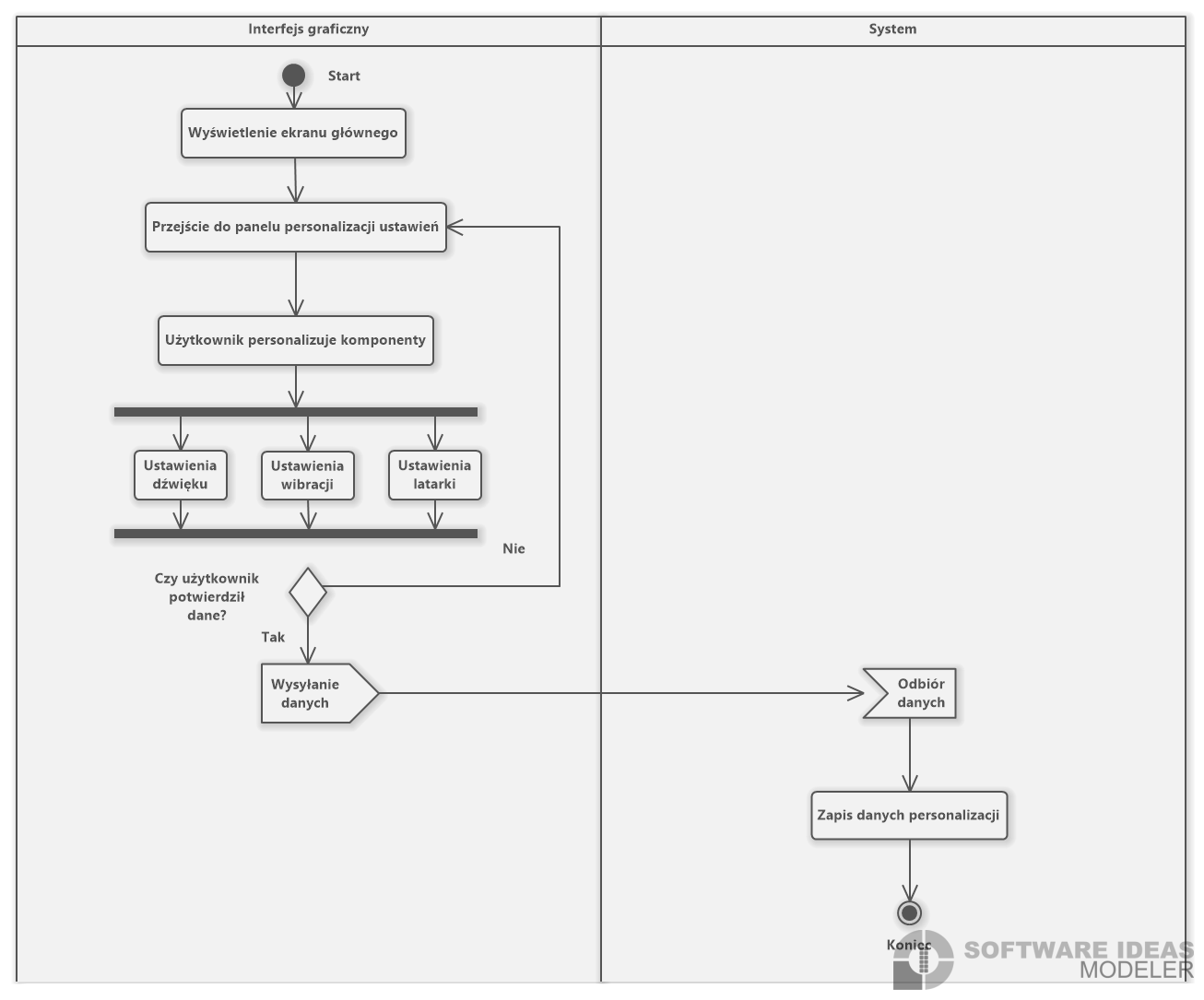
Opis wygenerowany automatycznie

Rysunek 11 diagram przypadków użycia

# Diagram czynności / aktywności

Zgodnie z powyższym opisem aplikacji można stworzyć następujący diagram aktywności gry:

**PU Personalizacja ustawień**



Rysunek 12 diagram aktywności PU Personalizacja ustawień

Zgodnie z opisem aplikacji można utworzyć powyższy diagram aktywności dla PU Personalizacja ustawień.

# Literatura

* https://docs.unity.com
* https://stackoverflow.com