Infinite Runner

Dorota Wojciechowicz

*1.Zakres projektu*

Zadanie projektowe polegało na stworzeniu gry mobilnej w typie „endless runner” na platformę Android. Dodatkowymi wymaganiami było wykonanie prostego trybu gry „debug mode” i wprowadzenie do projektu warstwy trwałości w postaci funkcji save game i load game.

*2.Instalacja*

W celu zainstalowania gry należy na urządzeniu mobilnym z systemem Android pobrać plik znajdujący się pod linkiem: <https://github.com/dwojciechowicz/Game/blob/master/Builds/InfiniteRunner.apk>. Następnie należy kliknąć na pobrany plik i kliknąć opcję „Zainstaluj”. W ekranie aplikacji pojawi się ikona zainstalowanej aplikacji:



*3.Instrukcja użytkowania*

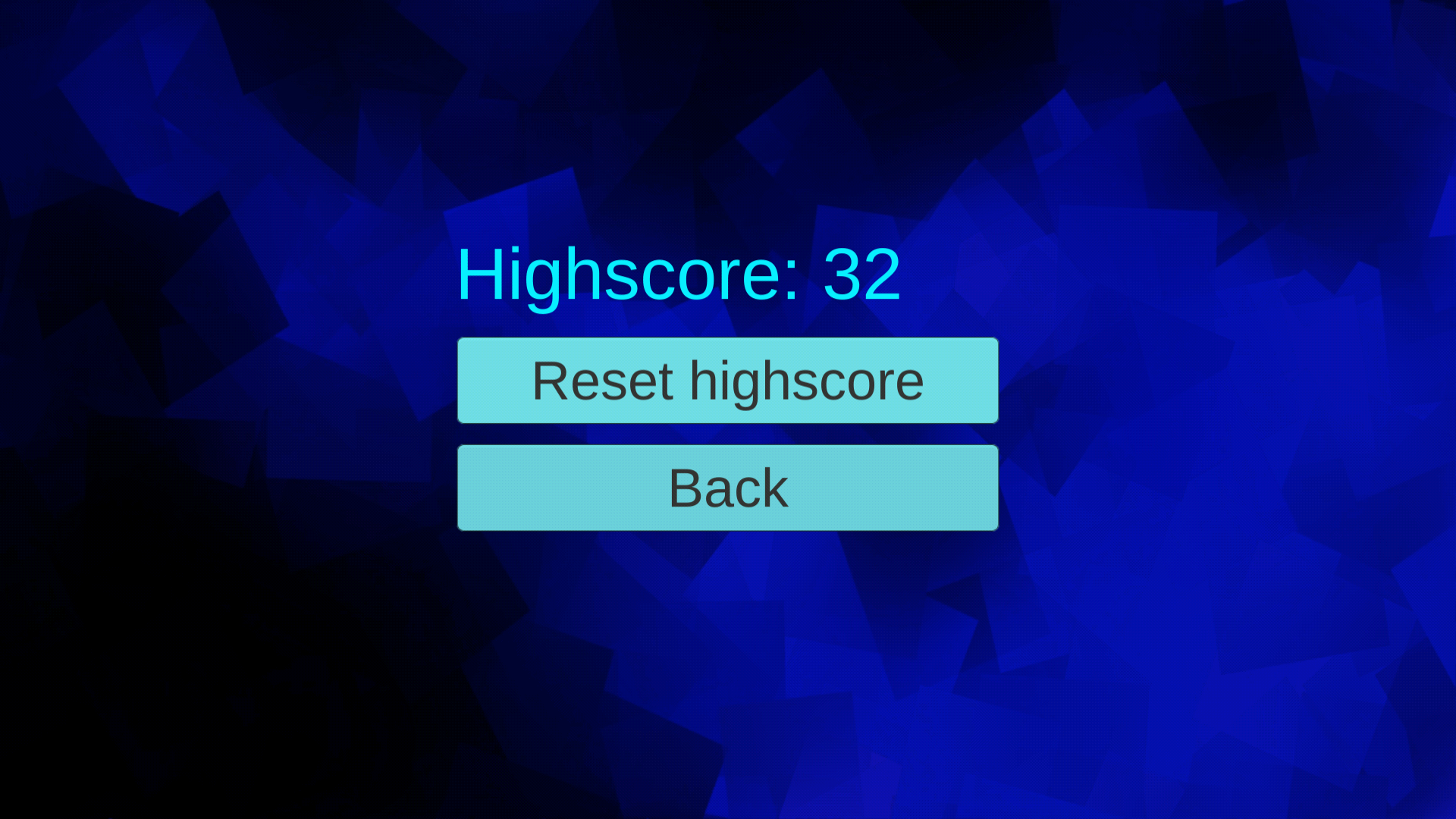
Menu:

Pierwszym ekranem po włączeniu gry jest ekran „Menu”. Posiada on trzy główne przyciski: „Play”, „Highscore”, „Exit”. Kliknięcie pierwszego z nich powoduje przejście do rozgrywki. Kliknięcie przycisku „Highscore” przekierowuje do ekranu „Highscore”. Opcja „Exit” powoduje wyjście z gry. W lewym górnym rogu znajduje się mały przycisk „Debug mode” typu toggle, który może być zanaczony lub odznaczony. Domyślnie jest on wyłączony. Zaznaczenie tej opcji powoduje zniknięcie wszystkich przeszkód w czasie rozgrywki. Po lewej stronie ekranu znajduje się też mały przycisk z etykietą „i”, który pozwala na podgląd instrukcji gry.



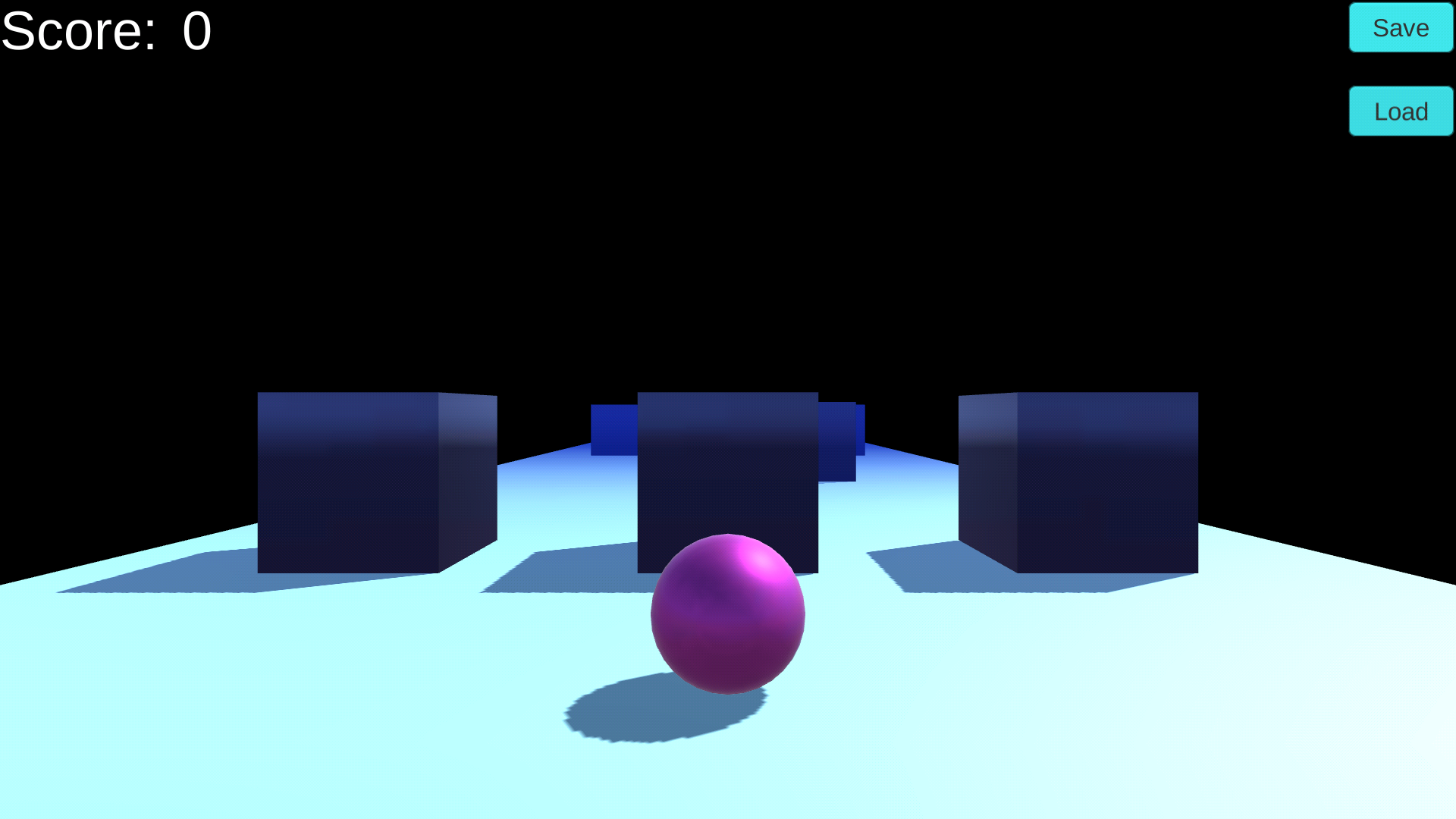
Highscore:

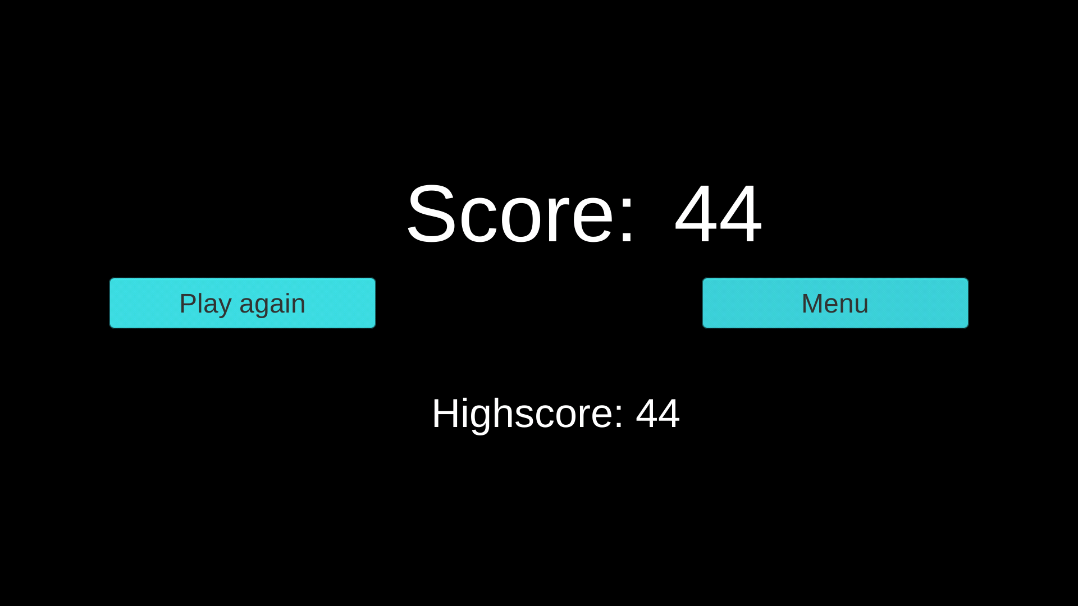
Ekran „Highscore” składa się z wyświetlanego najlepszego dotychczas uzyskanego wyniku i dwóch przycisków: „Reset highscore” i „Back”. Pierwszy z nich powoduje wyzerowanie highscore, a drugi przejście z powrotem do ekranu „Menu”.



Game:

Gra polega na poruszaniu się różową kulką i pokonaniu jak najdłuższej odległości. Aby sterować kulką należy klikać w lewą połowę ekranu by skręcić w lewo, lub w prawą by skręcić w prawo. Przytrzymanie palca powoduje większy skręt. Rozgrywka kończy się w momencie wypadnięcia z toru, lub uderzenia w przeszkodę (granatowe sześciany). W lewym górnym rogu wyświetlany jest aktualny wynik. W prawym górnym rogu znajdują się dwa przyciski: „Save” i „Load”. Przycisk „Save” odpowiedzialny jest za zapis aktualnego stanu gry, a „Load” za przywrócenie zapisanego stanu. W momencie gdy wcześniej nie zostanie zapisany stan gry, kliknięcie przycisku „Load” nie spowoduje reakcji. Po zakończeniu gry pojawi się ekran wyświetlający uzyskany wynik i highscore. Pojawią się także dwa przyciski: „Play again” i „Menu”. Kliknięcie pierwszego z nich spowoduje rozpoczęcie gry od nowa, a drugiego przejście do Menu gry. W ekranie gry puszczana jest inna muzyka niż w pozostałych ekranach, która nie zanika nawet po zakończeniu rozgrywki.





*4.Środowisko*

Do stworzenia tego projektu wykorzystałam silnik do gier Unity. Wszystkie skrypty pisane były w języku C#, a do testów wykorzystałam NUnit.

Silnik Unity to zintegrowane środowisko pozwalające na tworzenie zarówno dwuwymiarowych jak i trójwymiarowych gier. W swoim projekcie zdecydowałam się na stworzenie gry 3D.

* Unity pozwala na podgląd widoku użytkownika i jednoczesne wprowadzanie zmian wartości poszczególnych komponentów, co znacznie ułatwia pracę przy dobieraniu współczynników liczbowych.
* Kolejną zaletą tego środowiska jest obsługa zaawansowanej fizyki. Bardzo zależało mi na tym by w mojej aplikacji ruch kulki odbywał się płynnie, a także by wraz z pokonywaniem odległości jej prędkość stopniowo rosła. Problem ten rozwiązałam poprzez wprowadzenie tarcia pomiędzy kulką, a platformą po której się porusza i zrealizowanie ruchu poprzez przyłożenie określonej siły w punkt znajdujący się nad środkiem ciężkości kulki. Powoduje to, że kulka nie ślizga się a toczy.
* Innym plusem Unity jest możliwość zastosowania prefabrykatów. W moim projekcie przydały się one do stworzenia platform, po których porusza się kulka. Do stworzenia platformy potrzebne są dwa elementy: przeszkoda i płyta. Każda platforma różni się od siebie ilością i rozmieszczeniem przeszkód. Dzięki zastosowaniu prefabrykatów możliwe jest generowanie kolejnych platform w grze.
* Kolejnym elementem gry przy którym właściwości silnika Unity okazały się pomocne jest ruch kamery. W mojej aplikacji zrealizowałam go w ten sposób, by kamera poruszała się wraz z graczem w stałej odległości od niego.
* Unity oferuje dwie możliwości przeprowadzania testów: w trybie gry i trybie edycji. Jako, że w mojej grze nie ma logiki, która jest niezależna od obiektu, zdecydowałam się pisać testy w trybie gry.
* Budowanie projektu na platformę Android odbywa się przy pomocy Gradle.

*5.Problemy*

Pierwszym problem, który napotkałam było nietypowe zachowanie kulki na złączeniach platform – kulka zaczynała podskakiwać. Udało mi się tą kwestię rozwiązać za pomocą wyzwalacza włączanego tuż przed przejazdem kulki nad złączeniem, który „zamraża” położenie kulki na osi wertykalnej.

Kolejny problem napotkałam przy tworzeniu testów, ponieważ większość klas dziedziczy po klasie MonoBehaviour co sprawia, że nie można utworzyć obiektu takiej klasy przy wykorzystaniu operatora new. Rozwiązaniem tego problemu jest utworzenie osobnych klas, w których będzie znajdowała się logika, którą chcemy przetestować. Ponadto kolejnym problemem, który napotkałam była niemożność odwoływania się do konkretnych elementów gry podczas tworzenia testów.

*6.Klasy*

Większość klas w Unity3D dziedziczy po klasie MonoBehaviour. Dzięki niej możemy decydować kiedy powinny wykonywać się kolejne instrukcje. Udostępnia takie metody jak Start(), Update(), OnTriggerEnter() itp.

Każdy z ekranów gry posiada swojego własnego managera, w którym znajdują się metody potrzebne do jego obsługi.

Ruch kamery obsługiwany jest przez klasę FollowPlayer.

Klasa TileSpawning odpowiedzialna jest za generowanie kolejnych platform. Metoda Start() dba o to by pierwsze dwie platformy nie posiadały przeszkód, dzięki czemu nie ma możliwości, aby gra zakończyła się tuż po starcie.

Realizacja ruchu kulki znajduje się w klasie PlayerMotion. W tej klasie zrezygnowałam z metody Update() na rzecz FixedUpdate(). Powodem tego jest uwzględnienie fizyki w ruchu kulki. Ilość wywołań metody FixedUpdate() w przeciwieństwie do Update() nie zależy od aktualnej liczby klatek na sekundę, lecz wywoływana jest w stałych odstępach czasowych.

Za obsługę kolizji odpowiedzialna jest klasa PlayerCollision.

Opcja zapisywania i wczytywania stanu gry została zrealizowana poprzez klasę SaveGame. Klasa ta odpowiada za serializację i deserializację binarną danych znajdujących się w specjalnie stworzonej do tych celów klasie - PlayerData. Klasa ta przechowuje informacje dotyczące położenia gracza. Metody klasy SaveGame wywoływane są odpowiednio przy kliknięciu przycisku „Save” (metoda SaveData) lub przycisku „Load” (metoda LoadData) na ekranie gry. Wszystkie potrzebne dane do realizacji funkcji Save/Load zapisywane są do pliku data.txt.

*7.Podsumowanie*

Realizacja tego projektu dała mi okazję zapoznać się ze środowiskiem Unity i językiem C#. Była to moja pierwsza styczność z tworzeniem gier. Realizacja tego zadania projektowego kosztowała mnie dużo czasu i pracy, ale dała mi też bardzo dużo satysfakcji. Pomimo wielu godzin nauki nadal nie udało mi się poznać wszystkich możliwości jakie oferuje ten silnik gier, ale mam zamiar dalej pogłębiać swoją wiedzę i rozwijać się w kierunku tworzenia grafiki i gier komputerowych. W najbliższych miesiącach mam zamiar dalej udoskonalać swoją grę np. dodać opcję wyboru poziomu trudności.