POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

KIERUNEK INFORMATYKA

Projekt inżynierski

Aplikacja mobilna implementująca grę Arkanoid

Autor: Alex Porębski

Kierujący pracą: dr inż. Jacek Widuch

Gliwice, styczeń 2019

Załącznik Nr 2 do Zarz. Nr 97/08/09

Oświadczenie

Wyrażam zgodę/nie wyrażam* zgody na udostępnienie mojej pracy dyplomowej/rozprawydoktorskiej*

……..……., dnia……

……..……..……
(podpis)

……
(poświadczenie wiarygodności podpisu przez Dziekanat)

** właściwe podkreślić*

Oświadczenie promotora

Oświadczam, że praca „Aplikacja mobilna implementująca grę Arkanoid” spełnia wymagania formalne pracy dyplomowej inżynierskiej.

Gliwice, dnia …… ……..……..……

(podpis)

Spis treści

1. Wstęp

Od pojawienia się pierwszych telefonów komórkowych na początku lat '90 XX wieku urządzenia mobile stały się nieodłącznym elementem codziennego funkcjonowania ludzi na całym świecie. Wraz z wzrostem liczby osób posiadających smartfony, gry na urządzenia mobilne zyskiwały coraz to większą popularność. Zgodnie z statystykami podanymi przez Newzoo w 2018 roku 2.3 miliarda graczy na całym świecie wydało ponad 60 miliardów dolarów na gry mobile [7].

Celem niniejszej pracy jest napisanie aplikacji mobilnej implementującej klasyczną grę Arkanoid. Aplikacja będzie przeznaczona na urządzenia z systemem operacyjnym Android, który jest teraz najpopularniejszym systemem na smartfony. Gracz będzie się mógł zmierzyć z kilkunastoma etapami o rosnącym poziomie trudności w których jego celem będzie zniszczenie wszystkich punktowanych bloczków.

Zakres prac nad projektem będzie obejmował analizę problemu oraz określenie wymagań i funkcjonalności programu. Następnie zostaną wybrane narzędzia najlepiej nadające siędo osiągnięcia zamierzonego efektu. Po fazie tworzenia gry zostanie ona poddana testowaniu, w trakcie którego zostaną zebrane opinie graczy odnoście gry oraz ewentualnych poprawek i nowych funkcji w przyszłości.

Praca składa się z 7 rozdziałów. W rozdziale drugim zostanie przedstawiona krótka historia rozwoju gier mobilnych oraz główne założenia odnośnie aplikacji, jej elementów oraz dodatkowych funkcji aplikacji. Trzeci rozdział będzie poświęcony wymaganiom sprzętowym i narzędziom użytym do napisania programu. Rozdziały czwarty i piąty będą zawierały kolejno specyfikację zewnętrzną i wewnętrzną aplikacji. Rozdział szósty przedstawi

wyniki testowania gry na grupie graczy o różnych poziomach doświadczenia z grami mobilnymi, ich opinie na temat aplikacji i rezultaty wprowadzonych poprawek. Rozdział siódmy zawiera wnioski, możliwości rozwoju aplikacji oraz przegląd problemów napotkanych w trakcie prac.

2. Analiza tematu

W niniejszym rozdziale zostanie przybliżona historia gier na urządzenia mobilne, przedstawiony zostanie cel pracy, zostanie przeprowadzona analiza już istniejących rozwiązań i narzędzi do pisania gier na urządzenia z systemem operacyjnym Android.

2.1. Gry Mobilne

Jeden z pierwszych szeroko dostępnych modeli telefonów komórkowych Nokia 6610 (wprowadzony na rynek w 1997r.) posiadał zainstalowaną standardowo grę Snake, która bardzo szybko stała się klasykiem swojego gatunku. Dzięki prostym zasadom i potencjalnie niekończącej się rozgrywce idealnie nadawała się na platformę jaką jest telefon komórkowy.

Kolejnym krokiem w rozwoju gier na urządzenia mobilne było pojawienie się możliwości połączenia telefonów z Internetem. Dzięki temu producenci mogli o wiele łatwiej dotrzeć do klientów chętnych na zakup ich gier. Lata 2000-2008 cechują się szeroką różnorodnością rodzajów gier dostępnych na telefony komórkowe, poczynając od wyścigów 3D, a kończąc na grach logicznych.

Przełomem na rynku rozrywki mobilnej było udostępnienie przez firmę Apple usługi AppStore, co umożliwiło wielu developerom na udostępnienie swoich aplikacji użytkownikom iPhone'a. Wiązało się to jednak z dodatkowymi wyzwaniami, ponieważ pojawienie się iPhone'a zapoczątkowało wyparcie klawiatur na rzecz ekranów dotykowych.

Wymagało to od programistów zaprojektowania gier wykorzystujących ten nowy rodzaj interfejsu. Jednymi z pierwszych aplikacji które zdobyły popularność były gry AngryBirds i DoodleJump.

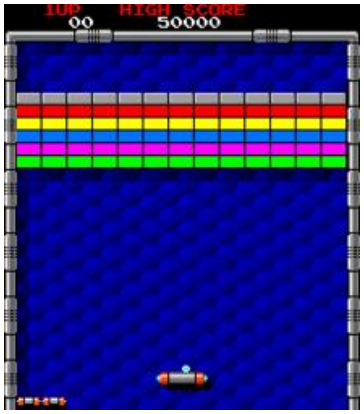
Obecnie rynek gier mobilnych stanowi ponad 50% całego rynku gier komputerowych. Dzięki coraz to lepszym smartfonom oraz narzędziom ułatwiającym tworzenie programów na platformy mobilne różnorodność typów gier jakie mogą zostać zaimplementowane jest olbrzymia, co pozwala na eksperymentowanie z nowymi pomysłami lub interpretacje starych na nowej platformie jaką są telefony komórkowe[8].

2.2. Cel pracy

Celem pracy jest zaprojektowanie i zaimplementowanie gry typu Arkanoid dla mobilnego systemu operacyjnego Android.

2.2.1. Arkanoid

Oryginalny Arkanoid był grą na automaty do gier wprowadzony na rynek przez japońskie studio Taito w 1986r. Rozbudował on rozgrywkę znaną z gry Breakout dodając power-up'y, różne rodzaje bloczków oraz nowe etapy.Rys.2.1 przedstawia pierwszy poziom Arkanoida.



Rys. 2.1 *Arkanoid rok 1986*

Gracz sterował statkiem kosmicznym "Vaus", który służył za platformę od której gracz miał odbijać piłkę nie pozwalając jej spaść poza pole gry. Uderzenie piłki w bloczek powodowało zniszczenie bloczka i naliczenie punktów graczowi. Po zniszczeniu wszystkich bloczków odblokowywała się możliwość przejścia do następnego etapu. W trakcie rozgrywki można było ulepszyć Vaus zbierając ulepszenia (możliwość strzelania rakietami, dodatkowe piłeczki, życia, poszerzenie platformy, itp.) spadające z góry.

Gra od samego początku cieszyła się popularnością. W recenzji Clare Edgeley dla grudniowego wydania czasopisma "Computer and Video Games" porównała go do Pong'a i Space Invaders z powodu prostoty i uzależniającej rozgrywki Arkanoida [3].

2.2.2. Założenia i funkcje

Biorąc pod uwagę gwałtowny wzrost popytu na gry mobilne oraz popularność gier zręcznościowych na urządzania przenośne Arkanoid jest doskonałym wyborem na grę na telefon. Implementacja Arkanoid'a będąca celem tej pracy będzie pozwalała graczowi, tak jak w oryginale, na sterowanie platformą odbijającą piłeczkę do zbijania bloczków. Dodatkowo gracz będzie mógł zbierać ulepszenia, które pomogą mu w przejściu poziomów. Ponadto do gry zostanie dodany system "Combo", dzięki któremu zniszczenie kilku

bloczków bez dotknięcia piłeczki przez platformę spowoduje naliczenie większej liczby punktów.

Ekran główny gry pozwoli graczowi na przejście do ekranu wyboru poziomu lub do tablicy z wynikami. Przy zwycięstwie lub porażce będą wyświetlały się panele informujące użytkownika o jego sukcesie lub przegranej i dające mu opcje do przejścia do następnego etapu, restartu poziomu, powrotu do ekranu startowego lub wyjścia z gry. W trakcie rozgrywki będzie istniała również możliwość spauzowania gry przy pomocy przycisku pauzy u góry ekranu. Wyniki po przejściu każdego etapu będą zapisywane i będzie je można wyświetlić na ekranie "Highscores". Ponadto będzie istniała możliwość wykasowania wszystkich wyników z pamięci gry.

2.3. Istniejące rozwiązania

Sklep Google Play jest domyślną usługą wykorzystywaną przez większość użytkowników Androida w celu odkrywania, porównywania i instalacji aplikacji na urządzenia z tym systemem. Po wpisaniu w pole wyszukiwania "arkanoid" zostają wyświetlone aplikacje, które według wyszukiwarki najlepiej spełniają podane kryterium. Na wyniki wyszukiwania mają wpływ popularność aplikacji, ocena użytkowników, czy aplikacja jest darmowa, itp.

Poniżej zostaną krótko opisane 3 przykładowe implementacje gry typu Arkanoid znalezione w sklepie Google Play. Wszystkie gry były dostępne do pobrania dnia 8.12.2018.

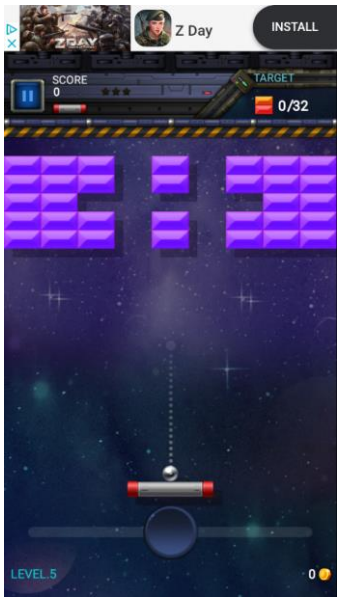
2.3.1. BrickBreaker Star: Space King

BrickBreaker Star: Space King jest grą koreańskiego studia Springcomes wydaną w listopadzie 2016 roku. Aktualnie jest ona pierwszą propozycją gry

po wyszukaniu w sklepie Play frazy "arkanoid" i została pobrana ponad 10 milionów razy.

Po załadowaniu aplikacji od razu wyświetla się ekran wyboru poziomu z dodatkowymi panelami informującymi gracza o możliwości zakupu ulepszenia do platformy za 20.99zł i reklamującymi inne gry tych twórców. W prawym górnym rogu ekranu mrugają dwie ikony z wizerunkiem monety próbujące przyciągnąć uwagę użytkownika.

Wybranie poziomu przenosi nas do ekranu zwierającego informację o rozstawieniu bloczków na etapie, liczbie zdobytych gwiazdek symbolizujących poziom ukończenia danego etapu. Gracz posiada również możliwość poprawienia efektywności power-up'ów dostępnych w trakcie rozgrywki za pomocą monet zebranych w trakcie przechodzenia wcześniejszych poziomów. Sporą cześć górnej części ekranu zajmuje baner z reklamą losowej aplikacji z sklepu Google Play (Rys.2.2).



Rys. 2.2 *BrickBreaker Star: Space King 2016*

Gra posiada bardzo płynną rozgrywkę i szybko reaguje na komendy gracza. Power-up'y dostępne do zebrania pokrywają się z tymi z oryginalnego

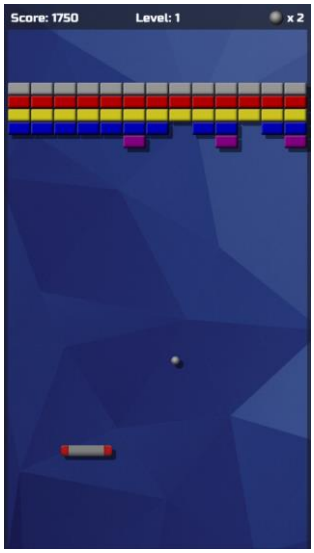
Arkanoida, ale występują też nowe ulepszenia takie jak np. powiększenie piłki. Informacje takie jak zdobyte punkty pozostałe życia i liczby zbitych bloczków wyświetlane na górze ekranu, zaraz pod banerem z reklamą. Prawdopodobnie jest to świadome rozmieszczenie elementów GUI, które ma zmusić gracza na zwrócenie uwagi na reklamę. Po wyjściu z etapu lub przejściu dwóch etapów wyświetla się kolejna reklama którą możemy wyłączyć przyciskiem w lewym górnym rogu.

2.3.2. Arkanoid Collection Free

Arkanoid Collection Free jest grą wyprodukowaną przez AlesApps i udostępnioną w sklepie Google Play w marcu 2018 roku, która na chwilę obecną została pobrana ponad 5000 razy.

Oprócz standardowego przycisku "Play" i "Exit" ekran główny posiada również możliwość zmiany ustawień gry (m. in. zmianę czułości ekranu, szybkości gry, głośności muzyki i efektów oraz załadowanie zapisanych postępów z pliku).

Sama rozgrywka w dużym stopniu przypomina pierwszego Arkanoida, głównie z powodu podobnego rozmieszczenia bloczków na poziomach. Dodatkiem do znanej rozgrywki jest występowanie negatywnych power-up'ów utrudniających ukończenie etapu. Ponadto gra posiada 500 różnych poziomów, co znacznie wydłuża czas jaki gracz jest w stanie spędzić przy niej bez uczucia nudy.

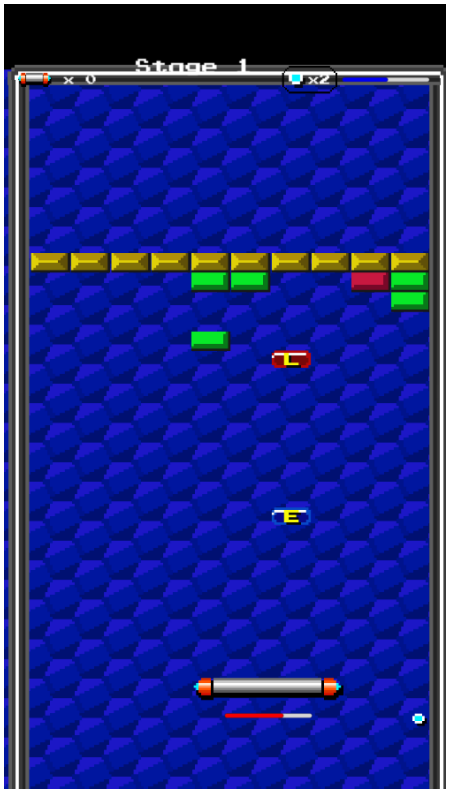


Rys. 2.3 *Arkanoid Collection Free*

W porównaniu do BrickBreakera wypełnionego po brzegi przeszkadzającymi reklamami, Arkanoid Collection posiada tylko jeden mały baner z reklamą na ekranie startowym, który znika po przejściu do rozgrywki pozostawiając na ekranie więcej miejsca na GUI i bloczki (Rys.2.3). Jeżeli użytkownik chce się mimo to pozbyć tej reklamy istnieje możliwość zakupu wersji bez reklam za 3USD(ok. 11 zł).

2.3.3. arkanoid-classic

Ostatnią z badanych w niniejszym podrozdziale gier jest arkanoid-classic od Rolling Eggs. Jest to gra próbująca skopiować uczucie jakie towarzyszyło graniu w pierwszego Arkanoida poprzez wykorzystanie tekstur i tła z oryginału oraz dźwięków przypominających te z klasycznych automatów do gier (Rys.2.4). Gra posiada 25 etapów wzorowanych na grze Arkanoid z 1986 roku. Frustrującym dla użytkownika może okazać się fakt że gra nie posiada możliwości zatrzymania lub wyjścia z gry oraz wyświetlające się co jakiś czas cało ekranowe reklamy popularnych gier z sklepu Google Play.



Rys. 2.4 *arcanoid-classic*

2.4. Przegląd dostępnych narzędzi programistycznych do tworzenia aplikacji mobilnych

Pisanie aplikacji na urządzenia mobilne z systemem Android wiąże się z wieloma wyzwaniami i problemami na jakie projektant tych programów musi zwrócić uwagę. Dzięki dostępowi do Internetu w dzisiejszych czasach z łatwością można znaleźć materiały pomagające początkującym twórcą zrozumieć powyższe zagadnienia. Dodatkowo z powodu popularności platformy Androida z roku na rok pojawia się coraz więcej profesjonalnej literatury na temat pisania aplikacji i gier.

Platforma Android jest specjalną platformą Javy udostępnioną przez firmę Google. Składa się ona głównie z podstawowych bibliotek języka Java oraz maszyny wirtualnej Dalvik. To wszystko jest uruchamiane na zmodyfikowanym jądrze Linuxa [4].

Według [2] smartfony to "przenośne urządzenia z kolorowym, wielodotykowym ekranem, które łączą w sobie funkcje telefonu komórkowego, odtwarzacza MP3, konsoli do gier, aparatu cyfrowego, a także wielu innych urządzeń pozwalając np. na uzyskanie połączenia z Internetem". Ekran dotykowy jest przyczyną dlaczego urządzenia mobilne zyskały w ostatnich latach tak ogromną popularność, jednakże jego wykorzystanie może być kłopotliwe, ponieważ gesty jakie użytkownik może wykonać na ekranie posiadają wiele właściwości i parametrów takich jak liczba dotknięć, długość dotknięć, ruch palców po ekranie, itp. Jest to szczególnie ważne dla gier, które będą przeznaczone na urządzenia z ekranami wielodotykowymi. Implementacja bardziej skomplikowanych gestów zapewni więcej możliwości interakcji użytkownika co sprawi że rozgrywka będzie bardziej wciągająca. Należy jednak pamiętać że niepotrzebne skomplikowanie sterowania może doprowadzić do sytuacji, w których gracz traci kontrolę nad grą, ponieważ niechcący wykonał gest zinterpretowany przez grę jako komendę akcji, której nie chciał wykonać.

Podstawowym narzędziem do obsługi grafiki na platformie Android jest biblioteka OpenGL ES (ang. Open Graphics Library for Embedded Systems) [9]. Pozwala ona na wyświetlanie, przekształcanie i edycje grafiki 2D i 3D zapewniając jednocześnie programiście dostęp do wszystkich parametrów związanych z renderowaniem obrazu, jednakże wymaga ona od niego posiadania szerokiej wiedzy z dziedziny grafiki komputerowej. Ponadto implementacja gry przy użyciu OpenGL wymaga napisania silnika gry od zera.

Innym narzędziem, które może zostać wykorzystane do tworzenia gier na urządzenia mobilne jest Corona SDK (ang. software development kit). Jest to szablon projektowy (ang. *framework*) stosowany do pisania aplikacji i gier na

różne platformy [10]. Używa on języka skryptowego Lua, który został użyty przy tworzeniu gier takich jak Warcraft, AngryBirds lub Civilization. Ponadto Corona SDK posiada wbudowany silnik do symulacji fizyki co znacznie przyśpiesza proces implementacji gier.

Bardzo popularnym narzędziem do tworzenia gier na komputery osobiste i urządzenia mobilne jest silnik Unity firmy Unity Technologies. Według [1] Unity jest popularnym darmowy silnikiem pozwalający na budowanie gier na wiele platform jednocześnie m.in.: Windows, macOS, Android, iOS. Dzięki przejrzystemu interfejsowi i możliwości tworzenia profesjonalnych efektów audiowizualnych Unity jest doskonałym wyborem narzędzia do tworzenia gier na dowolną platformę. Ponadto w sieci można znaleźć wiele materiałów mających na celu pomoc przy tworzeniu gier za pomocą Unity, a oficjalne fora silnika są nadzwyczaj aktywne i wypełnione ludźmi chętnymi odpowiedzieć na pytania. Dodatkowo większość podstawowych operacji (rozmieszczenie i edycja obiektów na scenie, obsługa fizyki, zarządzanie zasobami) użytkownik jest w stanie wykonać przy pomocy rozbudowanego edytora. Bardziej skomplikowane interakcje obiektów na scenie można zdefiniować przy użyciu języka skryptowego opartego o język C#. Biorąc pod uwagę powyższą analizę do dalszej pracy nad projektem został wybrany silnik Unity.

3. Wymagania dotyczące aplikacji mobilnej i narzędzia programistyczne

W niniejszym rozdziale zostaną przedstawione wymagania funkcjonalne i nie funkcjonalne aplikacji, opisane zostaną funkcje i możliwości edytora Unity. Na końcu rozdziału zostanie przybliżona metodyka pracy nad projektem.

3.1. Wymagania funkcjonalne i nie funkcjonalne

Aplikacja będąca celem niniejszej pracy będzie spełniała następujące wymagania funkcjonalne:

- gracz będzie posiadał możliwość sterowania platformą naciskając na stronę lewą lub prawą połowę ekranu dotykowego urządzenia powodując ruch platformy w lewo lub prawo,
- rozgrywka rozpocznie się po dwukrotnym naciśnięciu ekranu dotykowego w dowolnym miejscu,
- w trakcie rozgrywki użytkownik będzie posiadał możliwość zatrzymania gry poprzez naciśnięcie przycisku pauzy lub przyciśnięcie przycisku Wstecz na urządzeniu,
- aplikacja będzie automatycznie przechodzić w tryb pauzy przy zminimalizowaniu aplikacji bez jej wyłączenia,

- przy zatrzymanej grze gracz będzie miał możliwość powrotu do ekranu startowego lub wyłączenia gry,
- jeżeli urządzenie posiada akcelerometr użytkownik będzie mógł zmienić kierunek ruch piłki potrząsając urządzeniem w stałym odstępie czasu, możliwość użycia akcelerometru będzie sygnalizowana,
- po przejściu etapu liczba uzyskanych punktów będzie zapisywana jeżeli była najlepszym uzyskanym wynikiem przez gracza,
- możliwość wybrania dowolnego etapu,
- będzie można wyświetlić najlepszy wynik dla każdego poziomu oraz usunąć wszystkie zapisane wyniki.

Program powinien również spełniać poniższe wymagania niefunkcjonalne:

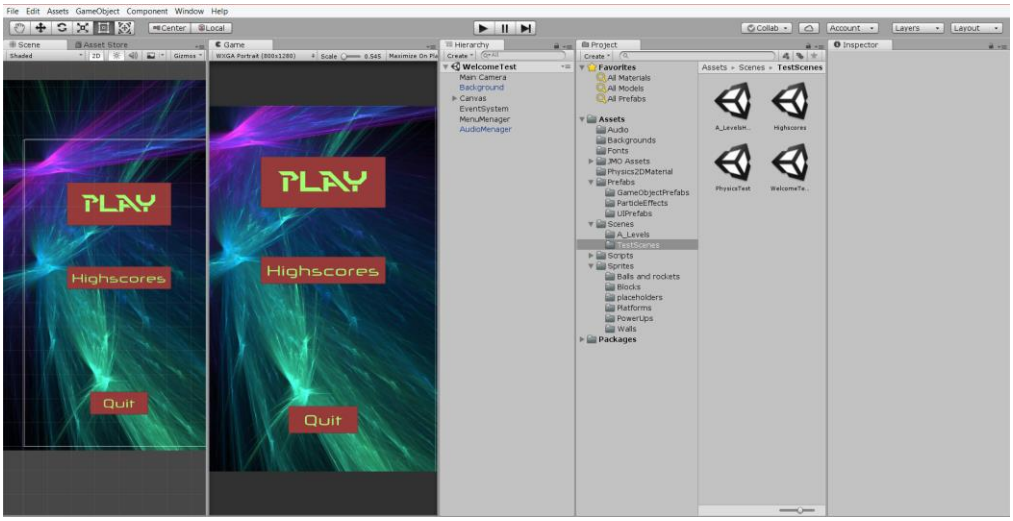
- aplikację będzie można zainstalować na urządzeniach posiadających system Android wersji 4.1 lub nowszej,
- GUI oraz wielkość etapów powinny być skalowane do wielkości ekranu urządzenia,
- program powinien zostać zaprojektowany w sposób pozwalający na szybkie dodanie w przyszłości nowych etapów oraz implementację dodatkowych power-up'ów.

3.2. Unity

Unity firmy Unity Technologies jest popularnym narzędziem do tworzenia gier za którego pomocą zostały stworzone gry takie jak Hearthstone: Heroes of Warcraft, Kerbal Space Program i Pokemon Go. Pozwala ono na budowanie aplikacji na większość platform mobilnych i

stacjonarnych oraz większość funkcjonalności edytora jest dostępna w darmowej wersji. Aby móc skorzystać z zaawansowanych narzędzi analitycznych oraz funkcji pomagających w dodaniu możliwości wydania pieniędzy w grze należy zakupić wersję płatną.

Przy tworzeniu nowego projektu użytkownik może wybrać między dwoma rodzajami gry: 2D lub 3D. Wybór rodzaju projektu ma wpływ na to jak silnik będzie generował obiekty oraz tryb pracy kamery w grze. Po stworzeniu nowego projektu zostanie otworzone okno edytora podzielone na kilka sekcji (Rys.3.1).Poniżej zostaną opisane najważniejsze elementy interfejsu edytora Unity [5]. Należy dodać że użytkownik może każdą z tych sekcji przesunąć lub zmienić jej rozmiar, aby dostosować interfejs edytora do swoich preferencji.



Rys. 3.1 *Okno edytora Unity*

Scena

W tym oknie wyświetlana będzie w tej chwili edytowana scena. Programista ma możliwość na przesunięcie widoku we wszystkie strony, przybliżanie, oddalanie oraz zaznaczanie i przesuwanie obiektów na scenie.

Inspektor

Inspektor pozwala na dodawanie, usuwanie i zmianę wartości pól komponentów umieszczonych na zaznaczonym na scenie obiekcie. Unity posiada gotową listę komponentów pozwalających na symulację fizyki, odtwarzanie dźwięków, dodanie hitbox'a otaczającego obiekt, itp.

Projekt

To okno pozwala na zarządzanie zasobami projektu. Użytkownik jest w stanie tworzyć nowe foldery, dodawać zasoby (tekstury, modele, pliki audio, skrypty, itd.) do projektu przeciągając pliki z eksploratora bezpośrednio do edytora oraz umieszczać obiekty na scenie.

Gra

Okno "Gra" jest wykorzystywane przy testowaniu gry w edytorze. W dowolnym momencie programista może włączyć symulację sceny, aby przetestować wprowadzone zmiany. Jest to szczególnie przydatne przy sprawdzaniu niewielkich zmian, ponieważ nie wymaga budowania i uruchamiania gry na nowo. Niestety nie ma możliwości na symulacje zachowania ekranu dotykowego, przez co przy tworzeniu gier na urządzenia gdzie głównym interfejsem, przez który gracz będzie komunikował się z aplikacją, jest ekran dotykowy programista jest zmuszony do budowania programu przy każdej zmianie lub napisanie osobnej obsługi dla klawiatury lub myszy.

3.2.1. Skrypty

Do zdefiniowania niestandardowych zachowań obiektów, nie obsługiwanych przez komponenty udostępnione w edytorze, używane są skrypty. Przy ich pomocy programista ma możliwość, wykonywania pewnych akcji przy tworzeniu obiektu lub co klatkę, reagować na kolizję lub zatrzymanie aplikacji. Silnik Unity posiada gotowe definicje obiektów reprezentujących komponenty z edytora i funkcje pozwalające na znajdowanie obiektów na scenie, pobranie ich komponentów, niszczenie lub tworzenie obiektów o podanych parametrach oraz ładowanie kolejnych scen.

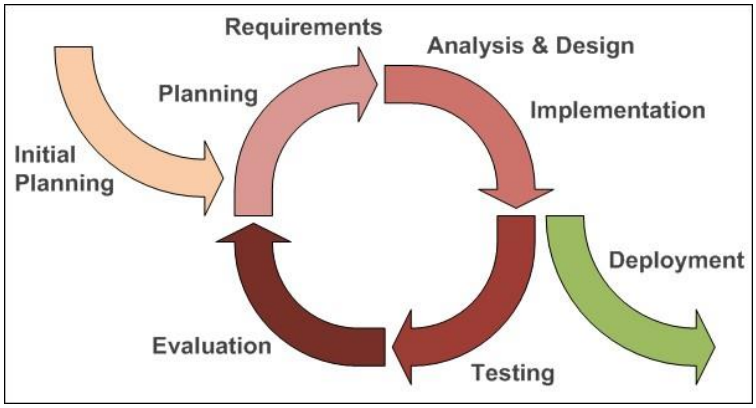
Cały proces tworzenia skryptu jest realizowany w Visual Studio, które zostało zintegrowane z edytorem Unity. Pozwala to na debugowanie kodu z poziomu VS w trakcie testowania gry w edytorze.

3.2.2. Budowanie aplikacji

Jak już zostało wspomniane wcześniej, Unity umożliwia budowanie gry na wiele platform. W darmowej wersji edytora istnieje możliwość wybrania następujących platform: Windows, Mac, Linux, iOS, tvOS, XBoxOne i Android. Można również, przy budowaniu na platformę Android, wybrać algorytm kompresji tekstur, system za pomocą którego gra zostanie zbudowana, czy ma to być wersja deweloperska (możliwość debugowania gry i analizy wydajności) i algorytm kompresji samej aplikacji.

3.3. Metodyka pracy nad projektem

Projekt został zrealizowany stosując przyrostowy model (ang. iterative development model) pisania oprogramowania. Model ten polega na projektowaniu i implementowaniu kilku z funkcjonalności na raz, przetestowaniu ich i przesłaniu projektu po każdej inkrementacji do klienta w celu uzyskania jego opinii. Ten proces jest powtarzany do momentu w którym program spełnia wszystkie przewidziane wymagania [6]. Sposób działania modelu przedstawiony został na Rys.3.2.



Rys. 3.2 *Model przyrostowy*[1]

Na początku zostały określone funkcjonalności i wymagania, które powinna spełniać aplikacja będąca tematem niniejszej pracy. W pierwszej iteracji zostały zaprojektowane podstawowe elementy rozgrywki m.in.: ruch piłki, niszczenie bloczków, ruch platformą oraz obsługa zakończenia etapu. Pierwsza wersja gry posiadająca zaimplementowane powyższe elementy rozgrywki została wysłana do testerów w celu sprawdzenia czy sposób sterowania jest wygodny i intuicyjny. W kolejnych iteracjach aplikacja była rozbudowywana o nowe funkcje oraz poprawiane były wcześniej prowadzone elementy. Ostatnie iteracje były skupione głównie na poprawie błędów oraz implementacji funkcji zaproponowanych przez gracza.

[1]https://www.researchgate.net/profile/A_B_M_Moniruzzaman/publication/249011841/figure/fig2/AS:341184099700745@1458356036923/Iterative-and-incremental-agile-development-process-source.png

4. Specyfikacja zewnętrzna

W niniejszym rozdziale zostaną przedstawione wymagania sprzętowe aplikacji, sposób instalacji oraz instrukcja rozgrywki.

4.1. Wymagania sprzętowe i instalacja

Według [11] zalecane wymagania sprzętowe dla gry stworzonej przy pomocy Unity wersji 2018.3 to posiadanie urządzenia z systemem Android 4.1 lub nowszym, procesorem ARMv7 z wsparciem NEON lub Atom CPU oraz OpenGL ES 2.0 lub nowszy. Wydajność gry może spaść w zależności od sprzętu na którym jest uruchamiana oraz od optymalizacji samej gry.

Aby zainstalować aplikację na urządzeniu należy umieścić plik .apk dołączony do pracy na urządzeniu, a następnie go na nim uruchomić. System operacyjny zapyta się czy użytkownik wyraża zgodę na instalację programu z nieznanego źródła i udzieleniu zgody gra zostanie zainstalowana na urządzeniu. Od tego momentu można ją uruchomić przy pomocy ikony na ekranie domowym urządzenia.

4.2. Instrukcja obsługi

Po uruchomieniu gry na ekranie zostanie wyświetlony ekran głównego menu (Rys.4.1). Znajdują się na nim trzy przyciski:

- PLAY: przenosi użytkownika do ekranu wyboru poziomu,
- Highscores: wyświetla tabele z najlepszymi wynikami uzyskanymi na każdym z poziomów,
- Quit: powoduje wyjście z aplikacji.

Rys. 4.1 *Główne menu gry*

Po naciśnięciu przycisku "PLAY" na ekranie zostanie wyświetlona lista dostępnych w grze poziomów (Rys.4.2). Gracz posiada możliwość przewijania tej listy oraz powrotu do menu głównego używając przycisku "Back". Aby rozpocząć rozgrywkę wymagane jest wybranie etapu klikając na jeden z przycisków w liście.

Rys. 4.2 *List dostępnych w grze poziomów*

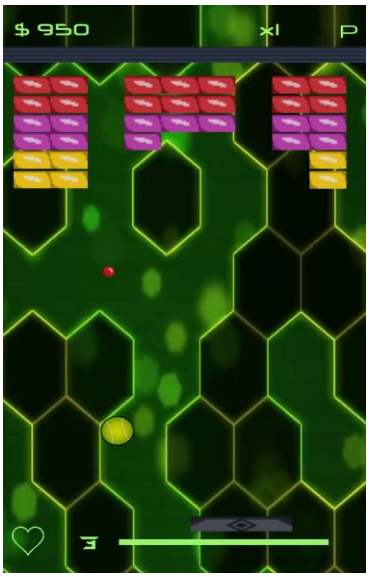
Przycisk "Highscores" odsyła użytkownika do ekranu wyświetlającego najlepsze osiągnięte wyniki dla każdego z etapów w grze (Rys.4.3). Przy pomocy przycisku "Reset" można usunąć wszystkie zapisane wyniki. Przycisk "Back" odsyła gracza z powrotem do menu głównego.

Rys. 4.3 *Lista najlepszych wyników*

4.2.1. Rozgrywka

Wybranie poziomu przenosi użytkownika do ekranu rozgrywki (Rys.4.4). Celem gracza jest zniszczenie wszystkich kolorowych bloczków przy pomocy czerwonej piłeczki i ulepszeń. Dodatkowo gracz nie może pozwolić piłeczce spaść poniżej platformy. Każde upuszczenie piłeczki powoduje utratę życia, a kiedy życie spadnie do zera etap kończy się przegraną.

Platformą steruje się naciskając lewą lub prawą stronę ekranu co powoduje ruch platformy w lewo lub prawo. Platforma będzie się poruszać tak długo jak długo ekran jest przyciśnięty lub do momentu kolizji z ścianą boczną. Ruch piłki rozpoczyna podwójne kliknięcie ekranu w dowolnym miejscu (oprócz przycisku pauzy znajdującego się w prawym górnym rogu ekranu). W trakcie rozgrywki gracz posiada możliwość użycia akcelerometru do zmiany kierunku ruchu piłki. Program sprawdza czy urządzenie posiada zainstalowany akcelerometr i po rozpoczęciu rozgrywki zaczyna odliczanie czasu. Upływ czasu jest reprezentowany przez pasek postępu znajdujący się pod platformą (Rys.4.4). Kiedy pasek postępu zmieni kolor z czerwonego na zielony można zmienić kierunek ruch czerwonej piłki potrząsając urządzeniem. Nowy kierunek piłki będzie zgodny z kierunkiem przyśpieszenia zarejestrowanego przez akcelerometr. Upuszczenie piłki powoduje reset paska postępu. Pozostałe graczowi życia wyświetlane są w lewym dolnym rogu obok paska postępu akcelerometru.



Rys. 4.4 *Poziom A4*

W trakcie rozrywki będą się pojawiać spadające ulepszenia, które można zebrać przy pomocy platformy (Rys.4.5). Zebranie pomaga graczowi w przejściu etapu. Istnieją cztery rodzaje ulepszeń:

- Piłka tenisowa: tworzy zieloną piłkę, która można wprawić w ruch tak samo czerwoną piłkę, nie reaguje ona na użycie akcelerometru i upuszczenie jej nie powoduje utraty życia przez gracza.
- Serce: dodaje dodatkowe życie.
- Gwiazda: poszerza platformę na 5 sekund.
- Rakieta: tworzy dwie mniejsze rakiety lecące w górę i niszczące pierwszy napotkany bloczek.

Rys. 4.5 *Możliwe do zebrania ulepszenia*

Każde zniszczenie bloczka dodaje punkty do wyniku uzyskanego na koniec poziomu. Dodane punkty obliczane są poprzez pomnożenie wartości

punktów przypisanej do bloczka i pomnożenie jej przez wartość combo wyświetlaną obok uzyskanej liczby punktów (Rys.4.6). Combo zaczyna z wartością 1 i jest inkrementowane za każdym razem gdy bloczek zostanie zniszczony. Odbicie czerwonej piłki od platformy resetuje wartość combo do wartości 1.

Rys. 4.6 *GUI wyświetlające liczbę punktów i wartość combo*

5. Specyfikacja wewnętrzna

W niniejszym rozdziale zostanie opisana architektura budowy poziomu gry oraz zaprezentowane ważniejsze fragmenty napisanych skryptów obsługujących interakcję obiektów na scenie.

5.1. Architektura poziomu gry

Poziomy w grze typu Arkanoid, mimo że drastycznie różnią się od siebie rozmieszczeniem obiektów na scenie, posiadają bardzo prostą strukturę. Kluczowymi elementami które muszą znajdować się na każdym etapie to:

- platforma którą steruje gracz,
- piłka którą gracz niszczy bloczki,
- bloczki, których niszczenie przyznaje punkty,
- ściany ograniczające ruch piłki.

Oprócz wyżej wymienionych elementów na każdej scenie znajdują się obiekty wyświetlające informację o liczbie punktów i żyć, odtwarzające muzykę i losowo generujące ulepszenia.

Wszystkie interakcję między tymi obiektami zostały zdefiniowane w skryptach dodanych do nich jako komponenty. Każdy skrypt definiujący zachowanie obiektu gry musi dziedziczyć po klasie `MonoBehaviour`. Dzięki temu programista ma dostęp do dwóch kluczowych metod: `Start()`i `Update()`.`Start()` jest metodą wywoływaną po stworzeniu obiektu (dla

większości przypadków będzie to moment załadowania sceny), a `Update()` jest wykonywany co klatkę w trakcie trwania gry. Ponadto silnik Unity jest w stanie wywoływać szereg różnych metod obsługujących zdarzenia związane z obiektem do którego dodany jest skrypt czy samym stanem gry. Jedną z takich metod jest metoda `OnCollisionEnter2D()` wywoływana przy kolizji dwóch obiektów 2D z hitbox'ami. Argumentem tej metody jest obiekt `Collision2D` zawierający wszystkie informacje o kolizji, co pozwala na jej wygodną obsługę.

W punktach5.1.1-5.1.4 zostaną opisane ważniejsze elementy tworzące etap gry i zależności między nimi.

5.1.1. Platforma

W grze typu Arkanoid platforma, jako obiekt nad którym gracz ma kontrolę i przez niego oddziałuje na inne obiekty na poziomie, jest kluczowym elementem każdego etapu. Platforma została zamodelowana przy pomocy czterech komponentów: Transform, SpriteRenderer, Rigidbody2D i BoxCollider2D. Transform przechowuje informację o pozycji obiektu na scenie, jego rotacji i skali. SpriteRenderer odpowiada za wyświetlanie tekstury przypisanej do platformy. Rigidbody 2D pozwala na symulację oddziaływania fizyki na obiekt oraz zmianę właściwości obiektu m.in.: jego masy, skali oddziaływania grawitacyjnego, zablokowanie pozycji i rotacji obiektu, zmianę trybu wykrywania kolizji, itp. Dzięki komponentowi BoxCollider2D silnik Unity jest w stanie wykryć kolizję między platformą a innymi obiektami.

Do platformy zostały również dodane trzy skrypty: `PlatformController`, `LevelMenager` i `PowerUpMenager`.

`PlatformController` jest skryptem umożliwiającym sterowanie platformą za pomocą ekranu dotykowego. Pobiera on listę wszystkich gestów wykonanych na ekranie w ostatniej klatce, sprawdza czy któryś z nich spełnia określone warunki i jeżeli tak przesuwa platformę.

`LevelMenager` odpowiada za przechowywanie informacji o stanie rozgrywki oraz reagowanie na zmianę tego stanu. Po załadowaniu sceny ustawia skalę upływu czasu silnika na 1, wywołuje metodę w obiekcie `AudioMenager` odpowiedzialną za odtworzenie muzyki i ustawia początkową liczbę żyć gracza. W trakcie gry w metodzie `Update()` jest sprawdzane czy nie zostały zniszczone wszystkie bloczki i czy aplikacja nie została zatrzymana. Jeżeli wszystkie bloczki zostały zniszczone zostaje wywołana metoda `Victory()` (*Listing 5.1*), która zapisuje wynik uzyskany przez gracza jeżeli jest najlepszy uzyskany wynik i wyświetla panel informujący gracza o zakończeniu poziomu. Wyniki zapisywane są przy użyciu klasy `PlayerPrefs` będącej częścią silnika Unity.

Listing 5.1 Metoda `Victory()` klasy `LevelMenager`

```
1  private void Victory()
2      {
3  victoryPauseMenu.Victory();
4  intscore = scoreGUI.GetScore();
5  if (score >PlayerPrefs.GetInt(scenName+"HighScore"))
6  {
7  PlayerPrefs.SetInt(scenName + "HighScore", score);
8          }
9      }
```

Metoda `BottomWallHit()` (*Listing 5.2*) jest wywoływana po uderzeniu czerwonej piłki w dolną ścianę. Na początku sprawdzane jest czy gracz nie posiada niewrażliwości. Następnie odejmowane jest jedno życie i sprawdzane jest czy liczba żyć nie jest równa lub mniejsza od zera. Jeżeli tak poziom kończy się porażką i wyświetlony zostaje panel informujący o porażce. Jeżeli nie piłka jest zatrzymywana i umieszczana na platformie w miejscu reprezentowanym przez obiekt `Transform ballPoint`

Listing 5.2 Metoda `BottomWallHit()` klasy `LevelMenager`

```
1  public void BottomWallHit() {
2          if (!godMode) {
3              lives--;
4  if (lives <= 0)
5              {
6                  Defeat();
7              }
8  else {
9  ball.GetComponent<BallController>()
10                      .Stop(ballPoint);
11 SetHealthText();
12             }
13         }
14     }
```

Skrypt `PowerUpMenager` jest skryptem odpowiedzialnym za działanie ulepszeń zebranych przez platformę i zostanie omówiony przy opisywania sposobu losowej generacji ulepszeń dalej w niniejszym rozdziale.

5.1.2. Piłka

Tak samo jak platforma obiekt piłki posiada przypisane do siebie komponenty Rigidbody2D i SpriteRenderer, a hitbox piłki reprezentowany jest przez CircleCollider2D. Dodatkowo do obiektu jest dołączony skrypt `BallController`. Jego zadaniem jest rozpoczęcie ruchu piłki po wykryciu szybkiego podwójnego kliknięcia na ekranie, obsługa kolizji z innymi obiektami oraz obsługa użycia akcelerometru do zmiany kierunku ruchu piłki.

Obsługa użycia akcelerometru odbywa się w metodzie `Acelerate()` (*Listing 5.3*). Najpierw pobierane są wartości przyspieszenia z osi X i Y, które są potem użyte do obliczenia wartości przyspieszenia poziomego do ekranu dotykowego urządzenia. Następnie sprawdzane jest czy obliczone przyspieszenie jest większe od ustalonego przyspieszenia granicznego. Dla wszystkich poziomów gry ta wartość to 1,1 przyspieszenia ziemskiego. Jeżeli obliczone przyspieszenie przekracza tę wartość piłce zostaje nadany kierunek ruchu zgodny z zarejestrowanym przyspieszeniem.

Ponadto rozpoczyna się odliczanie czasu do kolejnej możliwości użycia akcelerometru, a pasek postępu reprezentujący ten postęp zostaje ustawiony na wartość 0.

Listing 5.3 Metoda `Acelerate()` klasy `BallController`

```
 1  private void Acelerate() {
 2
 3  float x = Input.acceleration.x;
 4  float y = Input.acceleration.y;
 5  float magnitude2D =Mathf.Sqrt( x * x + y * y);
 6  // używane przy testowaniu do wyświetlania wartości
 7     // przyspieszenia
 8  if (acelerationDisplay != null) {
 9  acelerationDisplay.text = magnitude2D.ToString();
10          }
11  if (Time.time<lastAcelerationTime+ acelerationCooldown) {
12  UpdateSlider();
13          }
14  else if (magnitude2D>minAcelerate) {
15              Vector2 aceleration = new Vector2(x, y);
16  aceleration.Normalize();
17  GetComponent<Rigidbody2D>().velocity
18                                  =aceleration* speed;
19  lastAcelerationTime = Time.time;
20  acelerationSlider.value = 0;
21  acelerationSliderImage.color = cdBar;
22          }
23      }
24
```

5.1.3. Power-Up

Poprawne działanie ulepszeń w grze i ich interakcja z innymi obiektami na scenie jest zapewniona dzięki użyciu trzech skryptów: `PowerUpGeneratorController`, `PowerUpController` i `PowerUpMenager`.

`PowerUpGeneratorController` zawiera w sobie tablice prefabrykatów obiektów ulepszeń z której jest losowany power-up do umieszczenia na scenie. Nowe prefabrykaty można dodać z poziomu edytora, co pozwala na szybkie rozbudowanie gry o nowe ulepszenia w przyszłości. Generator może stworzyć nowy obiekt ulepszenia tylko wtedy kiedy piłka jest w ruchu i

upłynął określony czas od ostatniego tworzenia. Ulepszenie jest tworzone poprzez wylosowanie prefabrykatu z tablicy, a następnie umieszczenie go w losowym miejscu pod górną ścianą na poziomie (*Listing 5.4*).

Listing 5.4 Fragment metody `Update()` klasy `PowerUpGeneratorController`

```
// losowanie indeksu ulepszenia do stworzenia
intindex =
  (int)Random.Range(0f, powerUpsPrefabs.Length - 0.5f);
GameObjectpowerUp = Instantiate(powerUpsPrefabs[index]);
// tworzenie wektora pozycji dla nowego ulepszenia
Vector3 tmp =new Vector3
  (transform.position.x +
Random.Range(-generationArea, generationArea),
transform.position.y, 0);
powerUp.transform.position = tmp;
// zapisanie czasu stworzenia ostatniego ulepszenia
lastGeneration = Time.time;
```

Listing 5.5 Fragment metody `Start()` klasy `PowerUpController`

```
GameObject[] balls=
GameObject.FindGameObjectsWithTag(Tags.BALL);
GameObject[] blocks =
GameObject.FindGameObjectsWithTag(Tags.BLOCK);
GameObject[] hardblocks=
GameObject.FindGameObjectsWithTag(Tags.HARD_BLOCK);
GameObject[] rockets =
GameObject.FindGameObjectsWithTag(Tags.ROCKET);
Collider2D collider = GetComponent<Collider2D>();
if (collider != null)
{
foreach (GameObject ball in balls)
 {
  Physics2D.IgnoreCollision(collider,
ball.GetComponent<Collider2D>());
 }
foreach (GameObject block in blocks)
 {
  Physics2D.IgnoreCollision(collider,
block.GetComponent<Collider2D>());
 }
foreach (GameObject block in hardblocks)
 {
  Physics2D.IgnoreCollision(collider,
block.GetComponent<Collider2D>());
 }
foreach (GameObject rocket in rockets)
 {
  Physics2D.IgnoreCollision(collider,
rocket.GetComponent<Collider2D>());}}
```

`PowerUpController` jest komponentem każdego nowostworzonego ulepszenia. Jego głównym zadaniem jest wyłączenie kolizji między ulepszeniem, a obiektami innymi niż platforma i ściany (piłki, bloczki, inne ulepszenia) (*Listing 5.5*). Wyłączenie kolizji zapewnia swobodny ruch obiektu do momentu uderzenia w platformę lub dolną ścianę. Realizowanie jest poprzez pobranie list wszystkich obiektów z określonymi tagami i iterowanie po nich wyłączając kolizje między obiektami.

Po kolizji ulepszenia z platformą klasa `PowerUpMenager` będąca komponentem platformy rozpoznaje typ ulepszenia i obsługuje jego działanie m.in.: poszerza platformę, tworzy nowe piłki, itd. Parametry działania ulepszeń można zmieniać z poziomu edytora Unity.

5.1.4. AudioMenager

Odtwarzanie muzyki i dźwięków w grze jest realizowane przy pomocy obiektu singleton AudioMenager. Jest on tworzony po uruchomieniu aplikacji i nie jest niszczony przy ładowaniu kolejnych scen. Posiada on tablice obiektów klasy Sound (*Listing 5.6*) przechowujących informację o plikach muzycznych dostępnych do odtworzenia w trakcie rozgrywki. Dzięki zastosowaniu adnotacji [System.Serializable] programista posiada dostęp do pól obiektów Sound z poziomu edytora Unity, co pozwala na dostosowywanie parametrów klipu muzycznego(głośność, wysokość, czy ma być odtwarzany w pętli).

Listing 5.6 Implementacja klasy Sound

```
1  [System.Serializable]
2  public class Sound  {
3
4      public string name;
5
6      public AudioClip clip;
7
8  [Range(0f, 1f)]
9      public float volume=1f;
10
11 [Range(0.1f,3f)]
12     public float pitch=1f;
13
14     public bool loop = false;
15
16 [HideInInspector]
17     public AudioSource source;
18 }
```

Dzięki zastosowaniu wzorca projektowego singleton obiekt AudioMenager może być wywołany przez każdy obiekt na scenie w celu odtworzenia dowolnego dźwięku w nim przechowywanego. Przykładowe wywołanie AudioMenager'a przedstawia *Listing5.7*.

Listing 5.7 Przykładowe wywołanie obiektu AudioMenager

```
1  public string themeToPlay = "Theme";
2  FindObjectOfType<AudioMenager>().PlayTheme(themeToPlay);
```

5.2. GUI

Wszystkie elementy GUI w grze zostały wykonane przy pomocy gotowych obiektów dostarczonych przez silnik Unity. Do ich działania niezbędne jest umieszczenie na scenie obiektu Canvas i EventSystem. Canvas określa przestrzeń w której mogą zostać umieszczone elementy interfejsu, natomiast zadaniem obiektu EventSystem jest wywoływanie metod obsługi zdarzeń po użyciu obiektów GUI.

5.2.1. Highscores

Zapisywanie najlepszych wyników uzyskanych przez gracza na skończonych etapach zostało wykonane używając klasy `PlayerPrefabs` będącej częścią silnika Unity. Pozwala ona na zapisywanie wartości oraz nadawanie im aliasów, za pomocą których można je potem odczytać. Ponadto są one przechowywane w pamięci między kolejnymi uruchomieniami gry. Po zakończeniu poziomu wynik uzyskany przez gracza jest zapisywany z kluczem "Highscore" + nazwa_poziomu (np. dla poziomu A1 klucz to HighscoreA1). Następnie po załadowaniu sceny wyświetlającej najlepsze wyniki skrypt HighScoreController odczytuje wszystkie nawy poziomów z klasy Levels, dla każdego odczytanego poziomu tworzy panel do wyświetlania wyniku i zapisuje na nim odczytaną wartość punktów uzyskaną przez gracza (*Listing 5.8*). Jeżeli żaden wynik nie został zapisany dla danego poziomu wyświetlany jest ciąg znaków równy "-----".

Listing 5.8 Metoda `GetAllHighScores()` klasy `HighScoreController`

```
 1  void GetAllHighScores() {
 2  string allHighScores = "";
 3  string tmp;
 4  Levels levels = new Levels();
 5  foreach(string levelTypeinLevels.LEVEL_TYPES) {
 6  foreach (string levelNamein(string[])levels.GetType().
 7  GetField(levelType).GetValue(levels)) {
 8  GameObject panel = Instantiate(panelPrefab) as
 9  GameObject;
10  tmp = GetHighScoreByLevelName(levelName);
11  panel.transform.SetParent(this.transform, false);
12  panel.GetComponentInChildren<Text>().text =
13  levelName +HIGH_SCORE_TEXT + tmp;
14  highScoresPanels.Add(panel);
15              }
16          }
17      }
```

Odczytywanie wyniku z `PlayerPrefs` odbywa się w metodzie`GetHighScoreByLevelName(string)` (*Listing 5.9*).

Listing 5.9 Metoda `GetHighScoreByLevelName(string)` klasy *HighScoreController*

```
1  string GetHighScoreByLevelName(string levelName) {
2          string highScore = PlayerPrefs.GetInt(levelName +
3                                  HIGH_SCORE_KEY, 0).ToString();
4          if (highScore == "0") {
5  highScore = NO_HIGH_SCORE_TEXT;
6          }
7  return highScore;
8      }
```

6. Weryfikacja i walidacja

W niniejszym rozdziale zostanie opisany sposób testowania aplikacji, błędy znalezione przez testerów oraz w jaki sposób zostały poprawione.

6.1. Sposób testowania aplikacji

W trakcie pracy nad projektem został wykorzystany model przyrostowy. Zakłada on że po każdej wprowadzonej zmianie należy przeprowadzić testy. Dzięki temu programista jest w stanie szybko wykryć błędy i otrzymać opinie klienta o wprowadzonych zmianach. Jeżeli klient będzie niezadowolony działaniem nowych funkcji można je od razu poprawić w następnej iteracji.

Pisanie gry rozpoczęto od implementacji szkieletu rozgrywki, czyli ruchomej platformy, odbijającej się od niej piłki i bloczków których zniszczenie powodowało zakończenie etapu. Pozwoliło to na skupienie się na samym początku nad dwoma najważniejszymi aspektami rozgrywki: ruchem piłki i platformy.

W następnych iteracjach były dodawane nowe funkcję m.in.: system ulepszeń, GUI wyświetlające zdobyte punkty, ekrany wyboru poziomu i wyświetlające wyniki uzyskane przez gracza oraz użycie akcelerometru do zmiany kierunku ruchu piłki. Po implementacji każdej z powyższych funkcjonalności były one testowane w celu sprawdzenia czy działają w zamierzony sposób.

Kiedy gra posiadała już większość kluczowych funkcji, została ona przekazana grupie testerów w celu zebrania ich opinii na jej temat. Testerami

były osoby o sporym doświadczeniu z grami mobilnymi oraz takie które nie miały z nimi żadnej styczności. Ich zadaniem było przejście wszystkich dziesięciu etapów gry i informowanie o znalezionych błędach. Pozwoliło to na dogłębne przetestowanie aplikacji i uzyskanie opinii pochodzących od kilku rodzajów graczy.

6.2. Wykryte błędy i wprowadzone udoskonalenia

W trakcie testowania zostało wykrytych wiele błędów, z których większość została naprawiona. Jednym z pierwszych znalezionych problemów było długie uruchamianie aplikacji. W trakcie prac gra była regularnie testowana na urządzeniu mobilnym i jej uruchomienie trwało maksymalnie kilka sekund. Jednakże po dodaniu ostatnich funkcjonalności ten czas wydłużył się do ok. 20-30 sekund. Okazało się że powodem takiego zachowania było dodanie odtwarzania muzyki w trakcie gry. Unity posiada trzy tryby obsługi plików dźwiękowych. Dwa pierwsze ładują plik dźwiękowy do pamięci RAM skąd jest później odczytywany w trakcie działania aplikacji, jednakże wydłuża to czas uruchamiania i obciążenie pamięci RAM. Trzeci tryb pozwala na odczytywanie pliku strumieniowo z twardego dysku w trakcie działania programu i nie obciąża pamięci podręcznej. Po zmianie trybu ładowania gra ponownie uruchamiała się tylko przez kilka sekund.

Kolejnym znalezionym błędem były niedziałające przejścia między kolejnymi etapami. Wynikało to niepoprawnych nazw scen przekazywanych do metod ładujących kolejne sceny. Zmiana wartości na poprawne usunęła to niepożądane zachowanie.

Jak się okazało w trakcie testowania największe problemy w trakcie rozgrywki sprawiało graczom użycie akcelerometru. Jeden z testerów zwrócił uwagę na to że jeżeli urządzenie zostanie umieszczone w pozycji pionowej

prostopadle do podłogi piłka zaczyna spadać. Przyczyną takiego ruchu piłki była aktywacja akcelerometru pod wpływem przyspieszenia ziemskiego. Aby naprawić ten błąd została zwiększona wartość minimalnego przyspieszenia wymaganego do zmiany kierunku ruchu piłki.

Ostatnim poważnym błędem wykrytym przez testerów były szczeliny między niezniszczalnymi bloczkami na poziomach od A5 do A10, które pozwalały piłce przejść na drugą stronę niezniszczalnych bloczków w miejscach, w których to powinno być niemożliwe. Było to spowodowane użyciem źle skonstruowanego prefabrykatu. Po przebudowie prefabrykatu i zastosowaniu go do budowy niezniszczalnych bloczków problem nie został ponownie napotkany.

Jednym z drobnych problemów zaobserwowanych w trakcie testów było znikanie wartości uzyskanych punktów w trakcie przechodzenia poziomu. Okazało się że jest to spowodowane za dużą liczbą uzyskanych punktów i za małym obszarem przeznaczonym do ich wyświetlania. Z tego powodu słowo "Score" przed liczbą punktów zostało zastąpione symbolem dolara "$", a słowo "Combo" przed wartością combo zostało zamienione na "x".

Pod koniec pracy nad grą zostały wprowadzone drobne zmiany zaproponowane przez graczy. Jedną z nich było dodanie możliwości zatrzymania gry przy użyciu przycisku "wstecz" na urządzeniu oraz automatyczne zatrzymywanie gry po zablokowaniu telefonu. Dodatkowo do ekranu wyboru poziomu został dodany przycisk pozwalający na powrót do menu głównego. Oprócz tego zostały podmienione tekstury ulepszeń na takie, które lepiej symbolizują co dane ulepszenie robi.

7. Podsumowanie i wnioski

W trakcie prac nad aplikacją udało się zrealizować większość zamierzonych wymagań. Została przeprowadzona analiza dostępnych narzędzi do tworzenia gier na platformę Android i na jej podstawie zostało wybrane najlepsze narzędzie pozwalające na szybką i w miarę prostą implementację programu. Rozgrywka posiada większość elementów jakie posiadał oryginalny Arkanoid. Ponadto istnieje możliwość wykorzystania akcelerometru zainstalowanego w urządzeniu, co dodaje rozgrywce nowy poziom głębi, który nie był możliwy w erze automatów do gier, na które był przeznaczony pierwszy Arkanoid.

Aplikacja posiada prawie nieograniczony potencjał do rozwoju. Dzięki przygotowanym zasobom dodanie nowych poziomów do gry nie stanowi najmniejszego problemu, jedyne ograniczenia to wyobraźnia projektanta i pamięć urządzenia, na które gra jest przeznaczona. Tworząc nowe etapy można pozmieniać szybkość ruchu piłki i platformy, częstotliwość pojawiania się ulepszeń i czas oczekiwania na ponowne użycie akcelerometru. Dodatkowo można zaprojektować nowy rodzaj interaktywnych obiektów, które działałyby podobnie do ulepszeń ale zamiast pomagać graczowi, utrudniałyby mu rozgrywkę.

Większość problemów napotkanych podczas prac była związana z specyfiką działania silnika Unity. Jak zostało wspomniane w rozdziale 6, złe dobranie trybu dostępu do plików dźwiękowych spowodowało znaczne wydłużenie czasu uruchamiania aplikacji. Sama naprawa tego problemu nie wymagała dużego nakładu pracy, ale na zlokalizowanie przyczyny zmarnowano kilka godzin. Innym problemem napotkanym na samym

początku prac była obsługa ekranu dotykowego. Jako że edytor Unity nie posiada możliwości zasymulowania działania ekranu dotykowego podczas testowania gry w edytorze, każda zmiana w sposobie obsługi musiała być sprawdzana na telefonie komórkowym. Było to kłopotliwe podczas testowania użycia akcelerometru. Dodatkowo wymagane było napisanie osobnego sposobu sterowania platformą dla klawiatury i myszy aby można było testować grę w edytorze. Ostatnim problemem na który należy zwrócić uwagę było złe dobranie początkowych wymiarów kamery. Po dodaniu obiektu Canvas okazało się że nie można zmienić jego rozmiarów. Wymusiło to przeskalowanie wszystkich innych obiektów już zaimplementowanych w grze. Z tego powodu proporcje między obiektami, a elementami GUI na scenie mogą się różnić między urządzeniami o różnych rozmiarach ekranów.

Bibliografia

[1] Robert Ciesla. *MostlyCodeless Game Development: New School Game Engines.*Apress, Helsinki, 2017.

[2] Paul Deitel, Harvey Deitel, Alexander Wald. *Android 6. Techniki tworzenia aplikacji.* Helion, Gliwice,2016.

[3] Clare Edgeley. Arkade Action: Arkanoid. Computer and Video Games, 62(12), 142, 1986.

[4] Jeff Friesen . *Java. Przygotowanie do programowania na platformę Android*. Helion, Gliwice,2012.

[5] Adam Sinicki. *Learn Unity for Android Game Development: A Guide to Game Design, Development, and Marketing.* Apress, Guildford, 2017.

[6] Ian Spence, Kurt Bittne. *IBM.What is iterative development?* *https://www.ibm.com/developerworks/rational/library/may05/bittner-spence/*[2018-12-22]

[7] Tom Wijman. *Newzoo* . https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/ [2018-12-15].

[8] Mario Zechner, Robert Green. *Beginning Android 4 Games Development.* Apress, 2011.

[9] *Android Developer Manual: OpenGL*. https://developer.android.com/guide/topics/graphics/opengl[2018-12-15]

[10] *Corona SDK Manual.*https://docs.coronalabs.com/guide/programming/intro/index.html[2018-12-15]

[11] *Unity System Requirements.* https://unity3d.com/unity/system-requirements[2018-12-18]

Spis skrótów i symboli

OpenGL ES	ang. Open Graphics Library for Embedded Systems
SDK	ang. software development kit

Zawartość dołączonej płyty

Na płycie DVD dołączonej do dokumentacji znajdują się następujące materiały:

- praca w formacie pdf,
- źródła programu,
- plik archiwum .apk z aplikacją

Spis rysunków

Spis listingów