Politechnika Warszawska
WYDZIAŁ ELEKTRYCZNY

Sprawozdanie z projektu indywidualnego

Gra na platformę Android przy użyciu języka JAVA

Piotr Rzewnicki

Opiekun projektu
dr inż. Maciej Stodolski

Oświadczam, że niniejsza praca stanowiąca podstawę do uznania osiągnięcia efektów uczenia się została wykonana przeze mnie samodzielnie.

Spis treści

1. Wstęp

Celem projektu indywidualnego jest stworzenie gry dedykowanej dla urządzeń z systemem operacyjnym Android. Będzie to gra platformowa o tematyce piłkarskiej.

W dzisiejszych czasach smartfon jest nieodłącznym elementem większości ludzi. Ponad 90% osób w Polsce w wieku 16-24 lata posiada smartfon[4]. Biorąc pod uwagę również fakt, że przeciętny użytkownik telefonu z Androidem miesięcznie korzysta z 30 różnych aplikacji, których używa średnio przez miesiąc rocznie[2], możemy zauważyć jakim powszechnym narzędziem są programy na urządzenia mobilne.

Coraz więcej przedsiębiorstw dostrzega siłę jaka drzemie aplikacjach mobilnych. 11,3% firm dokonuje sprzedaży swoich produktów za pomocą strony internetowej lub aplikacji mobilnej[5]. Jest to wzrost w stosunku do lat poprzednich. W wyniku tego zapotrzebowanie na programistów mobilnych ciągle rośnie, co wiąże się z łatwiejszym znalezieniem zatrudnienia.

Wymienione dane wpłynęły na to, że chciałem nauczyć się programować aplikacje na urządzenia mobilne, więc dlatego jako cel projektu indywidualnego postanowiłem wybrać napisanie programu na Androida. Zdecydowałem, że będzie to gra, ponieważ w tej dziedzinie szczególnie łatwo jest zdobyć odbiorców aplikacji oraz odnieść sukces biznesowy. Kolejną zaletą tworzenia gier jest fakt, że jedynym ograniczeniem przy jej produkcji jest moja wyobraźnia.

1.1 O aplikacji

Aplikacja została nazwana „PeeW”. Została ona udostępniona w „Sklep Play” pod tą samą nazwą. Gra posiada dwie wersje językowe - polską i angielską, pomiędzy którymi można przełączać się w głównym menu (rysunek nr 1).
Użytkownik może również zapoznać się z zasadami gry, klikając w przycisk „Jak grać” („How to play” w wersji anglojęzycznej). Po naciśnięciu przycisku otworzy się poradnik z widokami okien gry i przyporządkowanym do nich opisem.

Aby wygrać grę, Użytkownik musi ukończyć każdy z 11 poziomów o rosnącym poziomie trudności.

Aplikacja jest obsługiwana dla wersji Android 4.1 lub nowszej, a więc 99,8% użytkowników systemu Android może z niej korzystać[1].

1.2 Zasoby aplikacji

Aplikacja posiada zasoby graficzne i multimedialne. Obraz piłki pochodzi z serwisu www.dryicons.com, natomiast pozostałe ilustracje oraz dźwięki są opracowaniem własnym autora.

1.3 Zasady gry

Celem każdego poziomu jest przetransportowane piłki do bramki. Jeśli na mapie znajduje się więcej niż jedna piłka, należy umieścić wszystkie piłki do bramki. Postać oraz piłki nie mogą dotknąć kolców. Jeśli to nastąpi, dokonuje się reset poziomu.

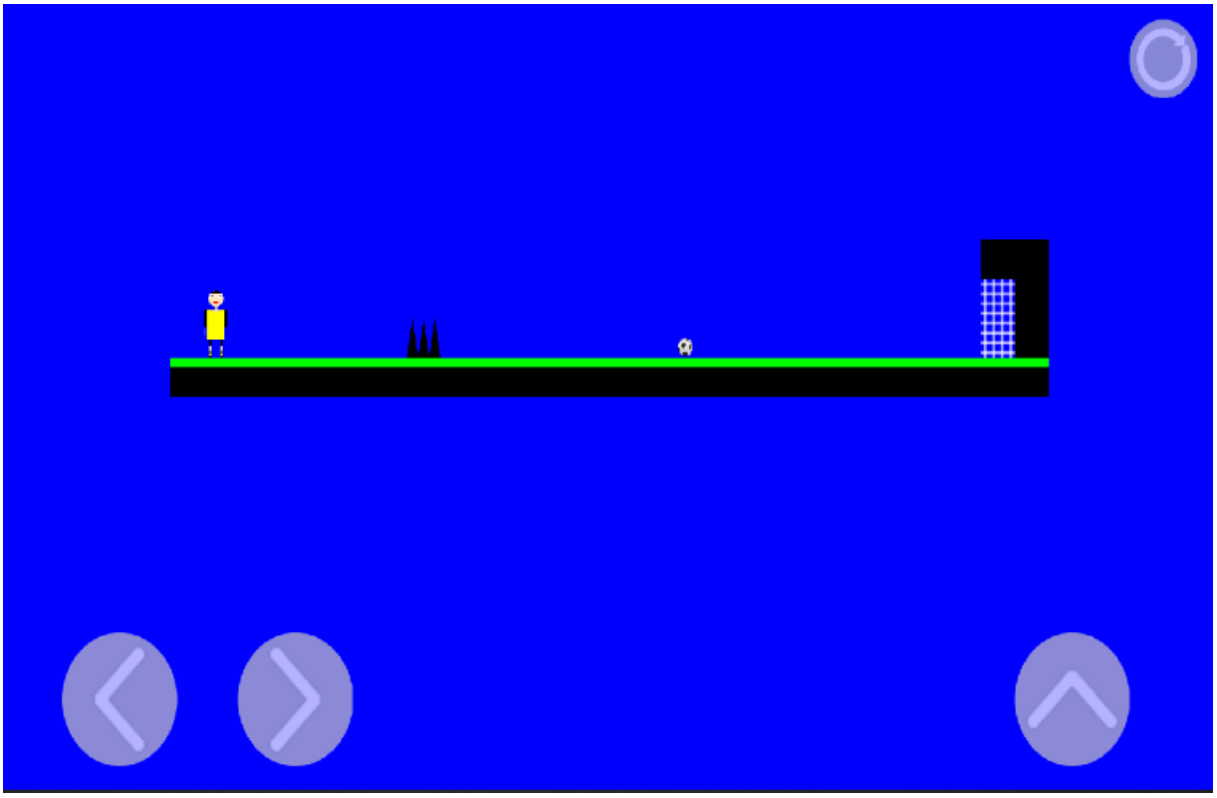
Użytkownik może sterować (poruszać się na boki oraz skakać) piłkarzem za pomocą przycisków. Jeśli użytkownik znajduje się blisko piłki, na ekranie pojawia się przycisk strzału. Po jego wciśnięciu zostaje wyświetlone okno strzału (rysunek nr 3).

Aby oddać strzał, należy przeciągnąć palcem po ekranie, a piłka poleci w przeciwnym kierunku do ruch palcem (analogicznie do robienia zamachu nogą przy kopaniu piłki). Długość przeciągnięcia wpływa na moc strzału. Po kopnięciu piłka leci we wskazanym przez Użytkownika kierunki, a następnie opada.

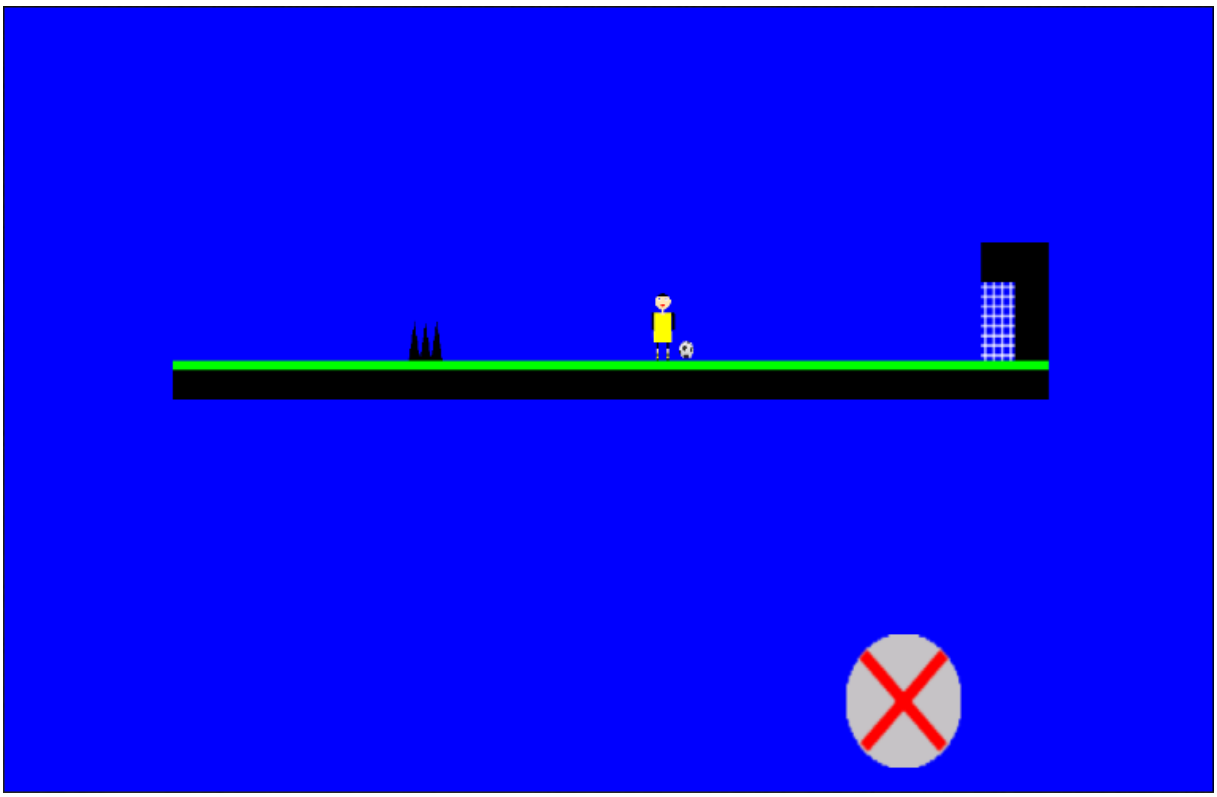
Użytkownik może powrócić do okna poruszania się (rysunek nr 2), klikając przycisk ze znakiem „X”.

Innym sposobem przemieszczania piłki jest jej prowadzenie. Polega ono na popychaniu piłki.

Rysunek 1: Widok głównego menu aplikacji



Rysunek 2: Okno poruszania się



Rysunek 3: Okno strzału

2. Budowa aplikacji

Aplikacja składa się z kilkunastu plików i katalogów. Omówię najważniejsze z nich:

● Katalog „build” przechowuje automatycznie generowane pliki aplikacji.
● W folderze „src” umieszczone są wszystkie kody źródłowe programu.
● AndroidManifest.xml (ścieżka dostępu src/main/AndroidManifest.xml) jest głównym plikiem konfiguracyjnym. Zawiera informację o możliwościach aplikacji, sposobie jej uruchomienia oraz o uprawnieniach jakich wymaga.
● W katalogu „res” (src/main/res) znajdują się wszystkie zasoby używane przez aplikację. W jego skład wchodzi kilka podkatalogów m.in. „drawable”, „layout” i „raw”. Pierwszy z nich odpowiada za przechowywanie zasobów graficznych. W folderze „layout” znajdują się pliki XML odpowiadające za określenie interfejsu użytkownika. Folder „raw” wykorzystywany jest do gromadzenia plików dźwiękowych. Innym ważnym podkatalogiem jest „values”, który za pomocą plików XML przechowuje informację o łańcuchach znaków i stylu aplikacji.
● Plik „build.gradle” służy do automatyzowania procesu budowania, uruchamiania, testowania i pakowania aplikacji na Androida[3].

2.1 Aktywności

Aplikacja składa się z 4 aktywności, czyli inaczej mówiąc zadań. Każda z aktywności realizuje swoje unikalne przeznaczenie. Ogólnie rzecz biorąc, przeznaczeniem aktywności jest obsługa konkretnego ekranu aplikacji[3].

Aktywności w mojej aplikacji:
- MenuActivity,
- TutorialActivity,
- GameActivity,
- WinGameActivity.

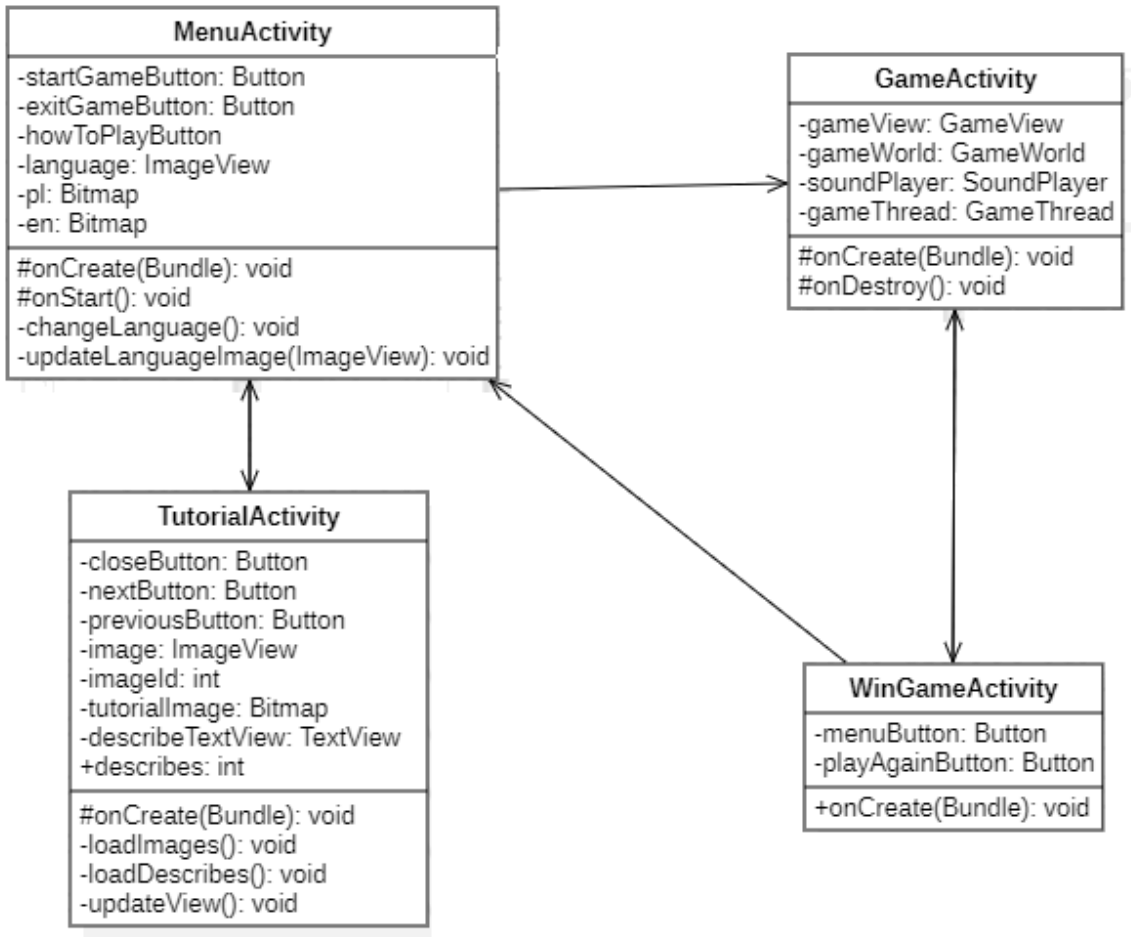
● MenuActivity odpowiada za wyświetlanie głównego menu aplikacji. Realizuje ona również obsługę przycisków i w zależności od wyboru Użytkownika przełącza się pomiędzy poszczególnymi aktywnościami.

● TutorialActivity odpowiada za ukazywanie na ekranie smartfonu poradnika dotyczącego gry oraz obsługę zdarzeń wywołanych kliknięciem w przycisk.

● GameActivity obsługuje grę. Znajdują się w niej klasy „GameWorld”, „GameThread” oraz „GameView”.

● WinGameActivity obsługuje okienko wyświetlane po wygraniu gry. Użytkownik ma możliwość ponownego zagrania oraz powrót do głównego Menu.

Zależności pomiędzy poszczególnymi aktywnościami przedstawia rysunek nr 4.



Rysunek 4: Diagram zależności między aktywnościami

2.2 Budowa gry

Jak wspomniałem we wcześniejszym podrozdziale za obsługę gry odpowiada aktywność „GameActivity”. Część główna aplikacji została stworzona przy użyciu 18 klas. Niektóre klasy są reprezentantami poszczególnych elementów gry (np. „Ball”, „Player”), a inne odpowiadają za logikę gry (np. „CollisionDetector”).

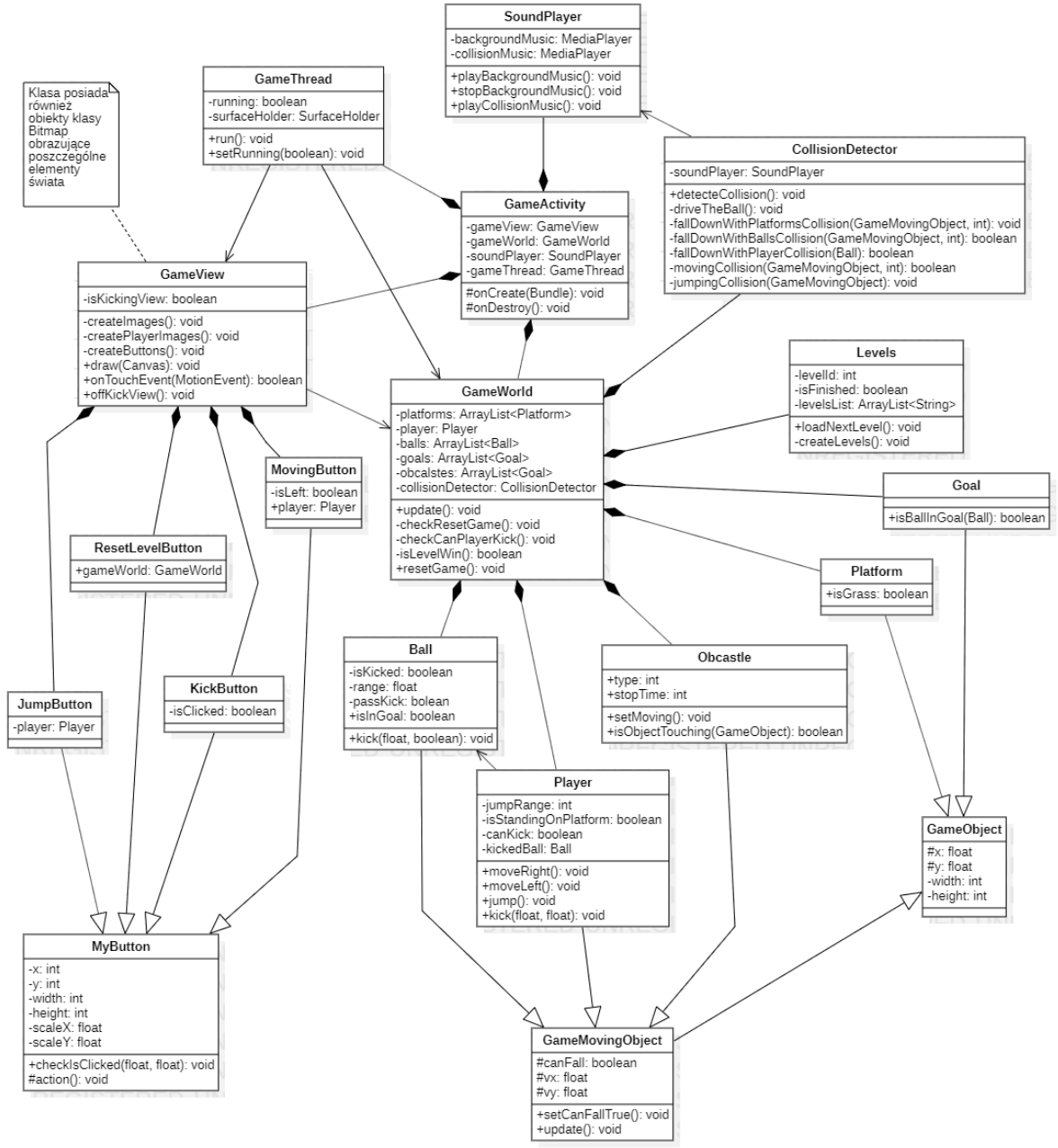
Główne klasy gry to:
- GameWorld,
- GameView,
- GameThread.

● GameWorld jest reprezentantem świata gry. Posiada obiekty tj. gracz, piłki, przeszkody, bramki itp. W tej klasie znajduje się również klasa odpowiedzialna za obsługę kolizji między obiektami.

● GameView jest to podklasa klasy „SurfaceView”, która odpowiada za wyświetlanie elementów świata. Implementuje „SurfaceHolder.Callback”, aby możliwe było rysowanie z innej klasy.

● GameThread jest to klasa rozszerzająca klasę Thread. Odpowiada za aktualizowanie świata gry oraz kolejnych jego wizualizacji. Uaktualnienie następuje co 20 milisekund, co daje 50 klatek na sekundę.

Rysunek nr 5 przedstawia budowę aplikacji zapisaną za pomocą diagramu klas. Grafika ukazuje wyłącznie główne atrybuty i metody danych klas. Pełny (zawierający wszystkie atrybuty i metody) diagram znajduje się w „Dodatku A”.



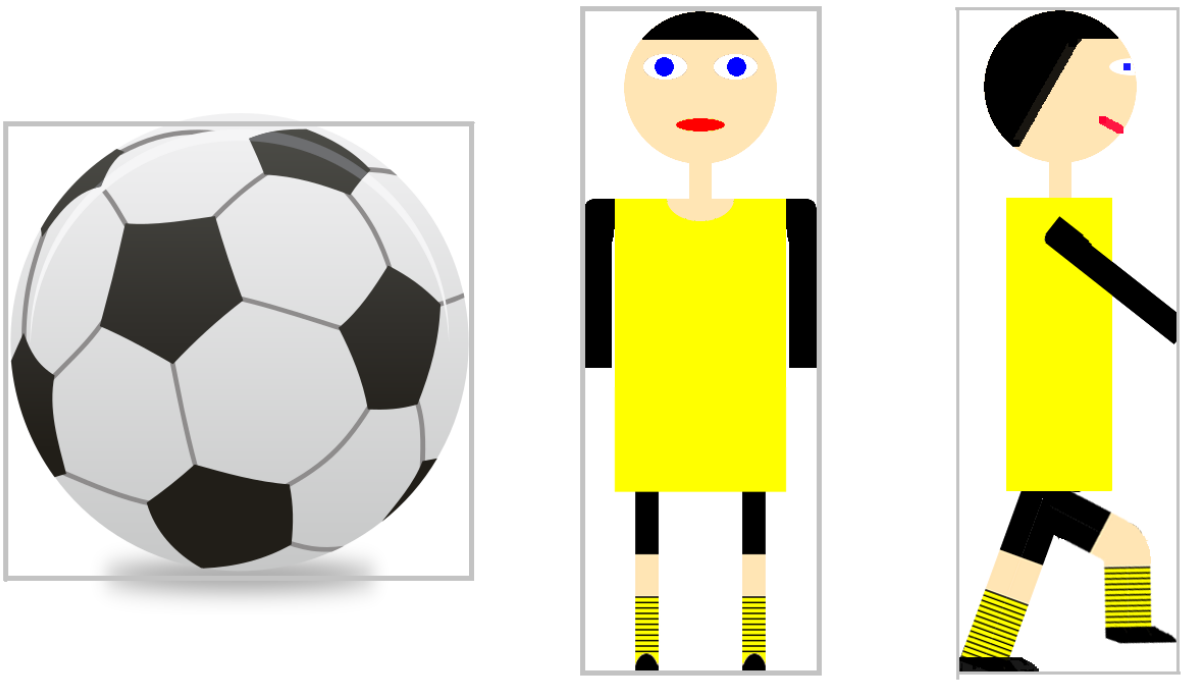
Rysunek 5: Diagram klas aplikacji

2.3 Wykrywanie kolizji

Ważnym elementem gry jest wykrywanie kolizji między obiektami. Idealną kolizją jest taka, gdy pojedyncze piksele jednego i drugiego obiektu nachodzą na siebie. Jest to zadanie zbyt czasochłonne obliczeniowo. Zastosowanie takiego rozwiązania mogłoby powodować brak płynnych animacji obrazu.

W moim projekcie wykorzystałem metodę prostokątów. Polega ona na zamknięciu każdego obiektu w jak najmniejszym prostokącie, a następnie wykrywane jest zdarzenie pomiędzy tymi figurami. W grze tylko piłka oraz gracz mają nieregularne kształty, co nie pozwala idealnie zamknąć ich w prostokątach. Wizualizację metody prostokątów dla tych dwóch obiektów przedstawia rysunek nr 6.

Użycie tego sposobu rozwiązywania problemów kolizji jest dość proste. Jednak poważną wadą jest możliwość wykrycia kolizji, gdy obiekt nie dotyka innego obiektu. Przykład takiego zjawiska obrazuje rysunek nr 7. Podczas sytuacji prezentowanej na grafice, kolizja zostanie wykryta, mimo że piłka w rzeczywistości nie dotyka innego obiektu.



Rysunek 6: Przedstawienie gracza i piłki za pomocą metody prostokątów

Rysunek 7: Obrazowanie błędów kolizji używając metody prostokątów

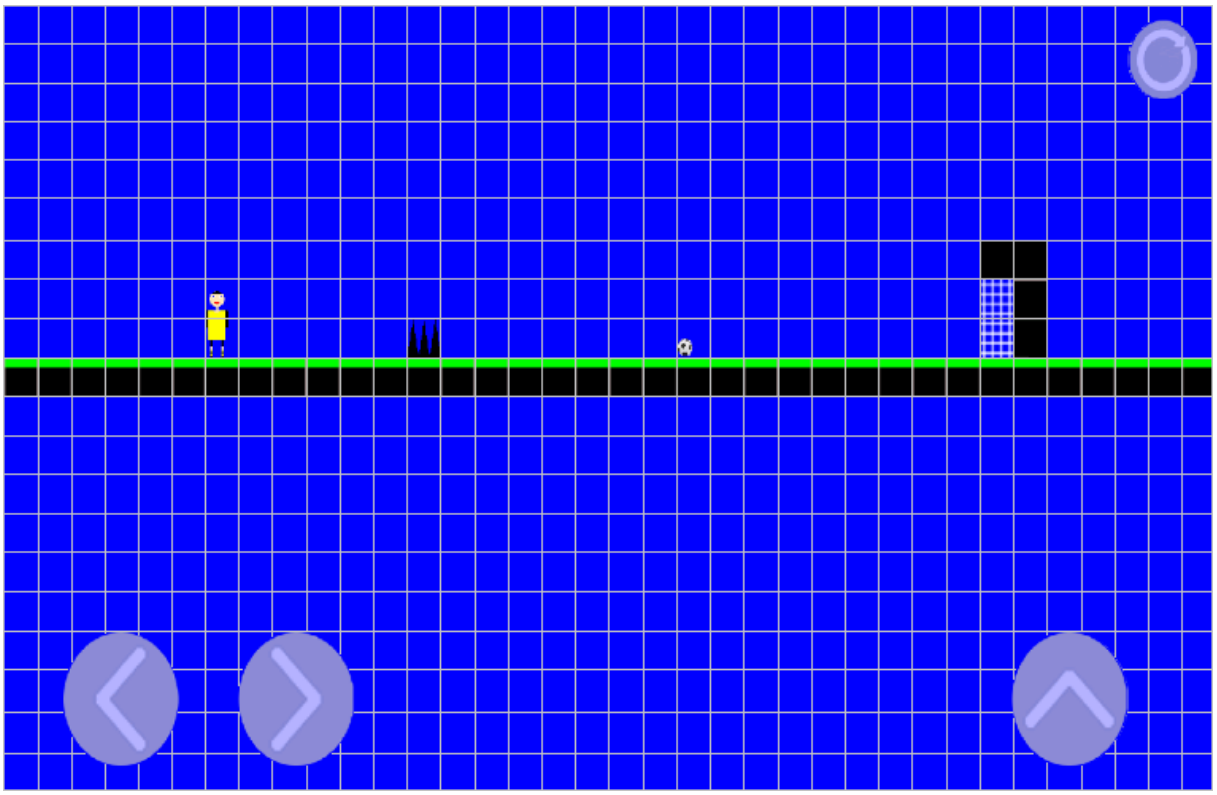
2.4 Wczytywanie i wyświetlanie poziomów

Każdy poziom zbudowany jest z siatki kwadratów o wymiarach 36x20, gdzie kwadrat ma wymiary 50x50 pikseli (rysunek nr 8). Następnie obraz jest skalowany do wymiaru ekranu telefonu.

Poziomy są wczytywane z łańcucha znaków o długości co najmniej 720 znaków (większa liczba znaków zależy od ilości przeszkód). Każdy kwadrat w siatce (zaczynając od górnego lewego rogu i poruszając się wierszami) jest kodowany za pomocą jednego lub większej ilości kolejnych znaków w wyrazie.

Poniżej umieszczam listę, co oznaczają poszczególne znaki w łańcuchu:
- „0” – pusta przestrzeń,
- „1” – czarna platforma,
- „2” – platforma z trawą,
- „3” – gracz,
- „4” – piłka,
- „5” – bramka,
- „6${K}${M}” – przeszkoda. K = {1,2,3,4} oznacza kierunek kolców, gdzie: 1 – góra, 2 – prawa, 3 – dół, 4 – lewa. W przypadku, gdy chcemy aby przeszkoda się poruszała dopisujemy współczynnik M=8.

Szczegółowa instrukcja tworzenia kolejnych poziomów opisana jest w „Dodatku B”.



Rysunek 8: Widok siatki kwadratów poziomu

3. Architektura techniczna

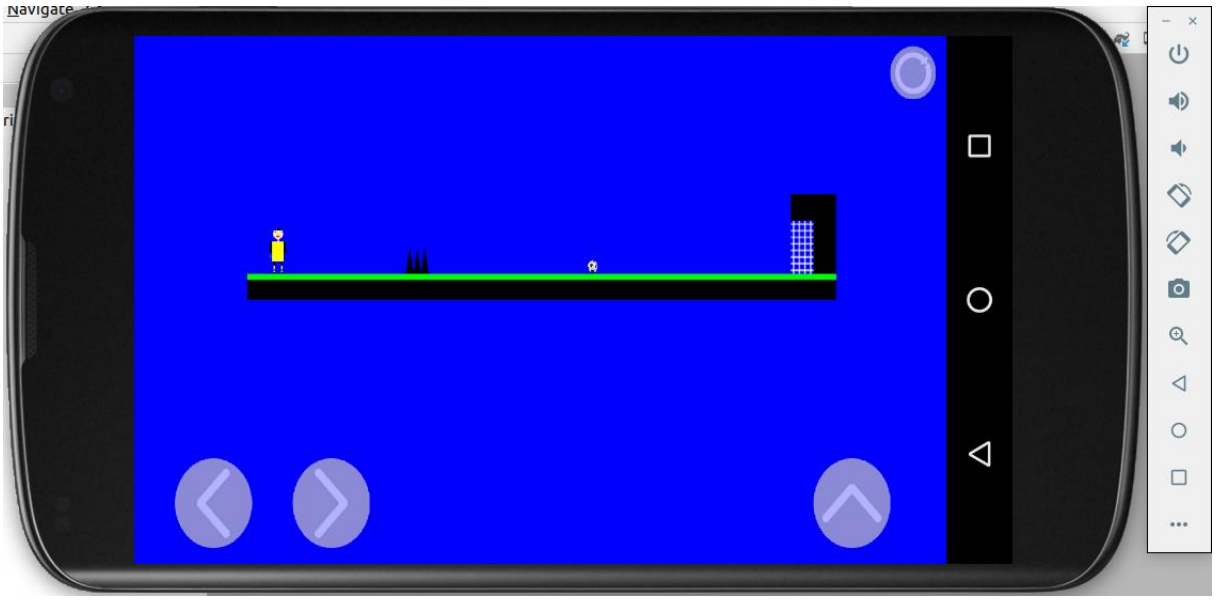
Niezbędną umiejętnością do tworzenia aplikacji mobilnych jest znajomość języka programowania oraz odpowiednich bibliotek przeznaczonych do tworzenia oprogramowania na urządzenia mobilne.

3.1 Android Studio

Pomocnym narzędziem do pisania aplikacji mobilnych jest środowisko programistyczne „Android Studio”. Jest to oficjalne zintegrowane środowisko programistyczne dla systemu operacyjnego Android firmy Google. Umożliwia ono programowanie w językach Java oraz Kotlin.

Zdecydowałem się wybrać te IDE ze względu na jego popularność. Większość kursów pisania aplikacji mobilnych jest tworzona w oparciu o ten program, co pozwala w łatwy sposób nauczyć się jego obsługi. Innym faktem, który przekonał mnie do wyboru tego środowiska jest prostota pisania kodu przy jego użyciu. Wynika to z faktu dobrej organizacji projektu (dzięki plikom i folderom opisanym w rozdziale 2. „Budowa aplikacji”).

Zaletą środowiska „Android Studio” jest łatwa obsługa emulatora (rysunek nr 9), czyli symulację rzeczywistego urządzenia, w którym może działać tworzona aplikacja.



Rysunek 9: Emulator Androida

3.2 GNU Image Manipulation Program

Wytwarzanie oprogramowania to nie tylko sam kod, ale także grafika umożliwiająca wyświetlanie operacji logicznych aplikacji.

Jako narzędzie do edycji obrazów zdecydowałem się użyć programu „GNU Image Manipulation Program” (GIMP). Jest to darmowe oprogramowanie do edycji grafiki rastrowej. Głównym powodem dla którego wybrałem GIMP był fakt, że umożliwia on tworzenie przezroczystego tła. Kolejną zaletą tego programu jest możliwość zaznaczania, a następnie kopiowania lub usuwania, skomplikowanych kształtów.

3.3 Audacity

Innym równie ważnym elementem gry są efekty dźwiękowe. Do ich edycji użyłem programu „Audacity”. Jest to darmowy edytor audio, który umożliwia m.in. wycinanie fragmentów pliku dźwiękowego. Zdecydowałem się na wybór tego programu ze względu na prostotę jego obsługi oraz możliwość edycji plików w formacie MP3.

4. Testy

Ważnym elementem tworzenia oprogramowania jest jego testowanie, które pozwala wyłapać błędy w kodzie. Testy odbywały się po dodaniu nowej funkcjonalności na emulatorze oraz rzeczywistym urządzeniu. Testy składały się z kilku kategorii, które zostaną omówione w kolejnych podrozdziałach.

Jako rzeczywiste urządzenie posłużył telefon Meizu M6 o następujących parametrach:
- system operacyjny: Android 7.0,
- pamięć RAM: 3 GB,
- procesor: MediaTek Helio P10 (MT6755),
- liczba rdzeni procesora: 8,
- taktowanie procesora: 1,5 GHz,
- rozdzielczość ekranu: 720 x 1280,
- przekątna ekranu: 5,2 cala,
- współczynnik odświeżania: 58 Hz.

Po wykryciu błędu był on poprawiany, a następnie ponownie dokonywano testów. W przypadku braku błędów tworzona była kolejna funkcjonalność.

4.1 Testowanie wizualnej atrakcyjności aplikacji oraz jej użyteczności

Prostota korzystania z aplikacji jest ważnym czynnikiem, który wpływa na jej odbiór. Działanie aplikacji było prezentowane najbliższym członkom rodziny, a ich uwagi dotyczące wyglądu, funkcjonalności oraz propozycje nowych funkcjonalności były brane pod uwagę podczas kolejnych etapów realizacji projektu. Dzięki tym testom została wprowadzona funkcjonalność ruchomych kolców, co zostało zasugerowane, jako ciekawy dodatek, podczas użytkowania aplikacji.

4.2 Testowanie uaktualnień aplikacji

Jak wspomniałem we wstępie rozdziału, każda nowa funkcjonalność była testowana na rzeczywistym urządzeniu. Aktualizacje aplikacji na urządzeniu odbywały się bez problemów.

Testy odbyły się również po opublikowaniu gry w „Sklepie Play”. Podczas pobierania nowej wersji programu nie napotkano na żadne problemy.

4.3 Testowanie instalacji

Testy polegały na wielokrotnym usuwaniu i ponownym instalowaniu aplikacji. Testy przeprowadzano z różnymi ustawieniami telefonu.

4.4 Testowanie wielojęzyczności

Testy składały się na uruchomieniu aplikacji z wielu różnych regionów świata. W tym celu został wykorzystany emulator, który umożliwia wybór lokalizacji urządzenia. Testy sprawdzały jaka wersja językowa aplikacji zostanie uruchomiona podczas jej startu.

4.5 Testowanie zgodności

Testy polegały na uruchomieniu aplikacji na różnych urządzeniach i sprawdzeniu poprawności jej działania w różnych środowiskach. Ze względu na ograniczenia finansowe testy zgodności zostały przeprowadzone tylko na dwóch rzeczywistych urządzeniach. Po stworzeniu gotowej aplikacji, testy zostały również przeprowadzone na telefonie Huawei Y7 2019.

Parametry telefonu Huawei Y7 2019:
- system operacyjny: Android 8.1,
- pamięć RAM: 3 GB,
- procesor: Qualcomm Snapdragon 450,
- liczba rdzeni procesora: 8,
- taktowanie procesora: 1,8 GHz,
- rozdzielczość ekranu: 720 x 1520,
- przekątna ekranu: 6,26 cala.

Testy zgodności przeprowadzono również za pomocą emulatora. Testowano możliwości dla urządzeń o różnej rozdzielczości, przekątnej ekranu oraz pamięci RAM.

Dla urządzeń o małej przekątnej ekranu (3,7 cala) zaobserwowano, że tekst niektórych okienek w poradniku nie mieści się w całości na ekranie. Błąd ten rozwiązano za pomocą dodania do pola tekstowego scrollu. Płynność rozgrywki jest bardzo dobra.

Dla smartfonów posiadających dużą przekątną ekranu (6,3 cala) zaobserwowano mniejszą płynność animacji w stosunku do telefonów o mniejszym rozmiarze. Jednak nie wpłynęło to negatywnie na możliwość korzystania z aplikacji.

5. Wnioski

Jestem zadowolony efektów mojej pracy. W programie występują elementy, które wymagają udoskonalenia, ale podstawowe mechanizmy działają bezbłędnie.

Nauka programowania aplikacji na urządzenia mobilne była ciekawym doświadczeniem, które (mam nadzieję) zaowocuje w przyszłości. Tworzenie gry pozwoliło mi spojrzeć na niektóre problemy z innej perspektywy. Jako przykład podam sposób wykrywania kolizji. Pozwoliło mi to rozważyć kształt m.in. postaci jako jedną figurę – prostokąt, a nie na jako kilka nieregularnych.

Największą trudnością w nauce kodowania programów na urządzenia mobilne było zrozumienie nowych pojęć tj. „aktywność”, które nie występowały wcześniej w programowaniu w Javie aplikacji na komputery. Kolejnym problemem było poznanie plików występujących w projekcie. Skomplikowane było zrozumienie pliku AndroidManifest.xml oraz plików definiujących układu elementów.

Programowanie gier jest bardzo czasochłonnym zadaniem. Wynika to głównie z konieczności poprawiania drobnych błędów, które są ciągle wyłapywane podczas testowania. Innym czynnikiem wpływającym na długość tworzenia aplikacji jest fakt, że trzeba brać pod uwagę każde możliwe zachowanie użytkownika oraz obsłużyć je w odpowiedni sposób, aby program nie przestał działać.

6. Dalsze możliwości rozwoju

6.1 Zwiększenie liczby poziomów

Podstawowym krokiem do dalszego rozwoju Aplikacji jest zwiększenie liczby poziomów. Po dokonaniu tej czynności należy również dodać przycisk umożliwiający kontynuowanie gry od poziomu na którym zakończyło się grę poprzednim razem, a nie jak ma to miejsce w tej chwili, gdy trzeba przechodzić wszystkie poziomy od początku.

6.2 Dodanie monet i nowych postaci

Następnym elementem rozwijającym rozgrywkę jest umieszczenie na mapie monet, które Użytkownik może zbierać trafiając w nie piłką. Za zebrane monety będzie możliwość kupienia nowej postaci, nowego stroju lub ulepszeń (szybkość, wyższe skoki itp.).

Nowi gracz będą różnić się także parametrami szybkościowymi. W przypadku popularności aplikacji można wprowadzić do gry oficjalne stroje klubów piłkarskich, co dodatkowo zachęci Użytkowników do zdobywania monet.

6.3 Dodanie mikropłatności w aplikacji

Oczywistym jest fakt, że ceny postaci, wyposażenia i bonusów będą na tyle wysokie, że zebranie odpowiedniej ilości monet podczas gry będzie bardzo czasochłonnym zadaniem. Dlatego należy wprowadzić również mikropłatności umożliwiające zakup monet, co będzie generowało zysk aplikacji. Jednak nie wszyscy Użytkownicy mają możliwość kupowania za pomocą karty kredytowej lub też nie chcą wydawać rzeczywistych pieniędzy. Do tej grupy należy dotrzeć poprzez dawanie im monet w zamian za obejrzenie reklamy, co będzie dodatkowym przychodem aplikacji.

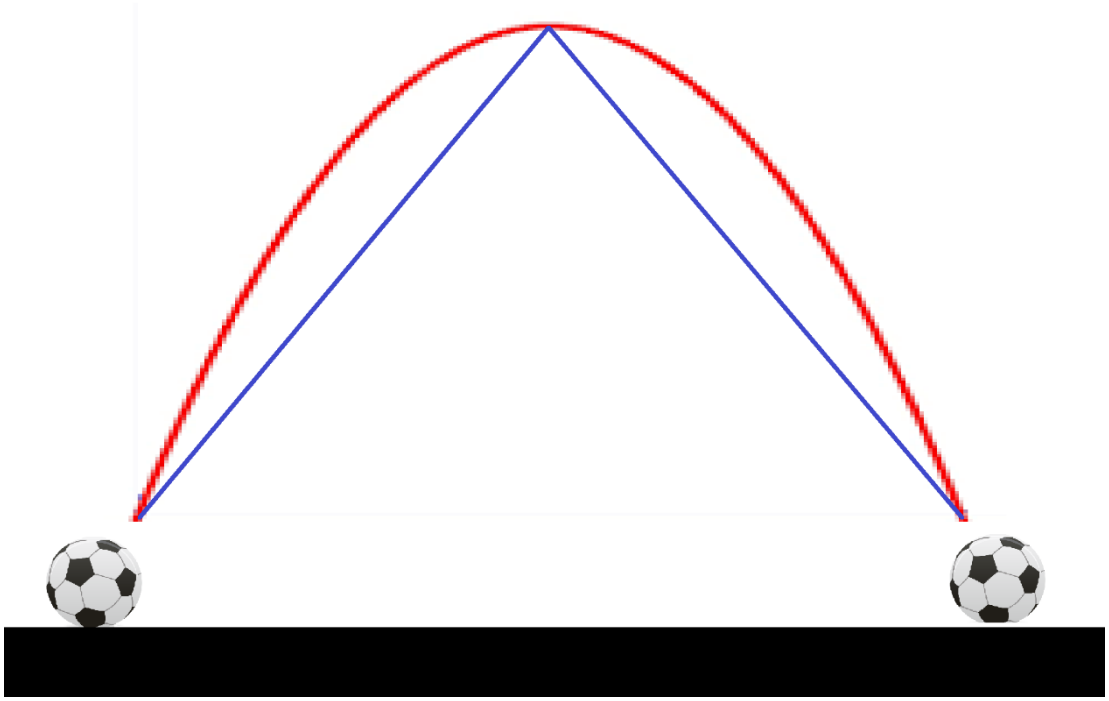
6.4 Dodanie nowych przeszkód

Elementem rozwoju aplikacji dotyczącym rozgrywki jest możliwość wprowadzenia nowego typu przeszkód – piłkarzy z drużyny przeciwnej. Poruszali by się oni w określonym obszarze, a trafienie w nich piłką miałoby analogiczny skutek jak trafienie w kolce. Można również dodać bramkarz który poruszałby się w okolicach bramki (przykładowo skakał), co utrudniałoby zdobycie bramki.

6.5 Udoskonalenie fizyki piłki

Kolejnym elementem wymagającym udoskonalenia jest lot kopniętej piłki. Aktualnie piłka porusza się po linii prostej do pewnego punktu (zależnego od siły strzału), a następnie opada również w linii prostej. Bardziej realistyczny tor uzyskamy stosując, znane z fizyki, wzory na współrzędne rzutu ukośnego. Rysunek nr 10 przedstawia aktualny oraz udoskonalony tor lotu piłki.

Należy również dodać kolizję między piłkami. Aktualnie kolizje są wykrywanie wyłącznie, gdy jedna piłka znajduje się na drugiej.



Rysunek 10: Tor lotu piłki. Kolor niebieski - tor aktualny, czerwony - rzut ukośny

6.6 Dodanie odbić piłki od ziemi i warunków atmosferycznych

Kolejnym elementem jest zaimplementowanie odbić piłki od ziemi. Można również wprowadzić różne warunki atmosferyczne np. śnieg, deszcz, mgłą.

Podczas śniegu piłka kopnięta po ziemi leciałaby dłużej, co ma symulować pokrycie lodem. W deszczu piłka nie odbijałaby się od murawy, natomiast podczas mgły widok niektórych elementów byłby zasłonięty przez mgłę, a więc Użytkownik musiałby strzelać do bramki nie widząc jej.

6.7 Dodanie funkcjonalności tworzenia poziomów przez Użytkownika

Ciekawym – z punktu widzenia Użytkownika – elementem aplikacji byłoby możliwość stworzenia własnego poziomu, który następnie należało przejść. Osoby korzystające z gry mogłyby tworzyć własne misje, co zwiększyłoby zadowolenie wśród odbiorców produktu.

Bibliografia

[1] Android Platform/API Version Distribution

[2] App Annie, “Spotlight on Consumer App Usage”

http://files.appannie.com.s3.amazonaws.com/reports/1705_Report_Consumer_App_Usage_EN.pdf

[3] Joseph Annuzzi Jr., Lauren Darcey, Shane Conder „Android. Wprowadzenie do programowania aplikacji. Wydanie V” Wydawnictwo „Helion” 2016

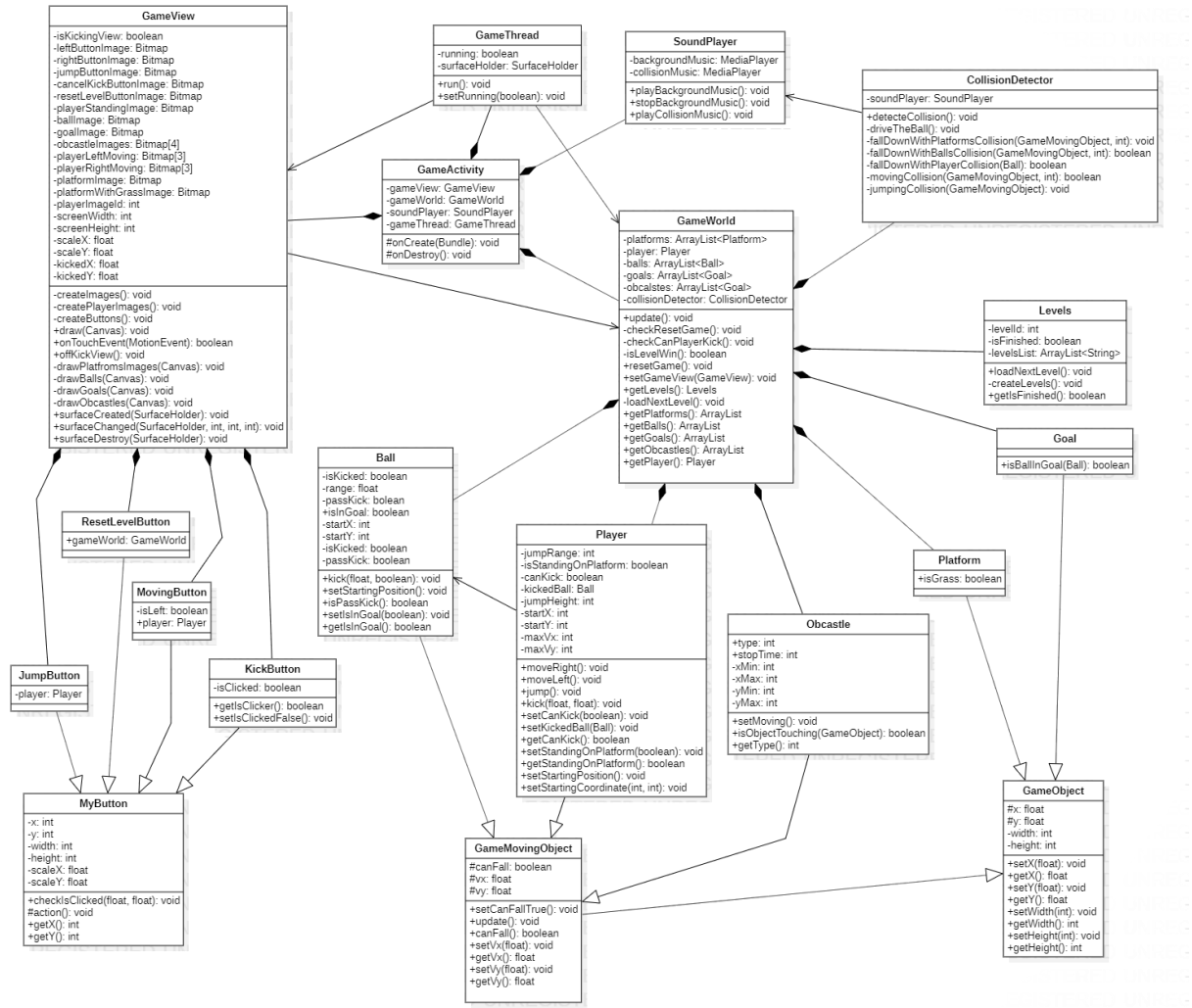
[4] Społeczeństwo informacyjne w Polsce w 2017 r.

https://stat.gov.pl/obszary-tematyczne/nauka-i-technika-spoleczenstwo-informacyjne/spoleczenstwo-informacyjne/spoleczenstwo-informacyjne-w-polsce-w-2017-roku,2,7.html

[5] Społeczeństwo informacyjne w Polsce w 2019 r.

https://stat.gov.pl/obszary-tematyczne/nauka-i-technika-spoleczenstwo-informacyjne/spoleczenstwo-informacyjne/spoleczenstwo-informacyjne-w-polsce-w-2019-roku,2,9.html

Dodatek A. Pełny diagram klas



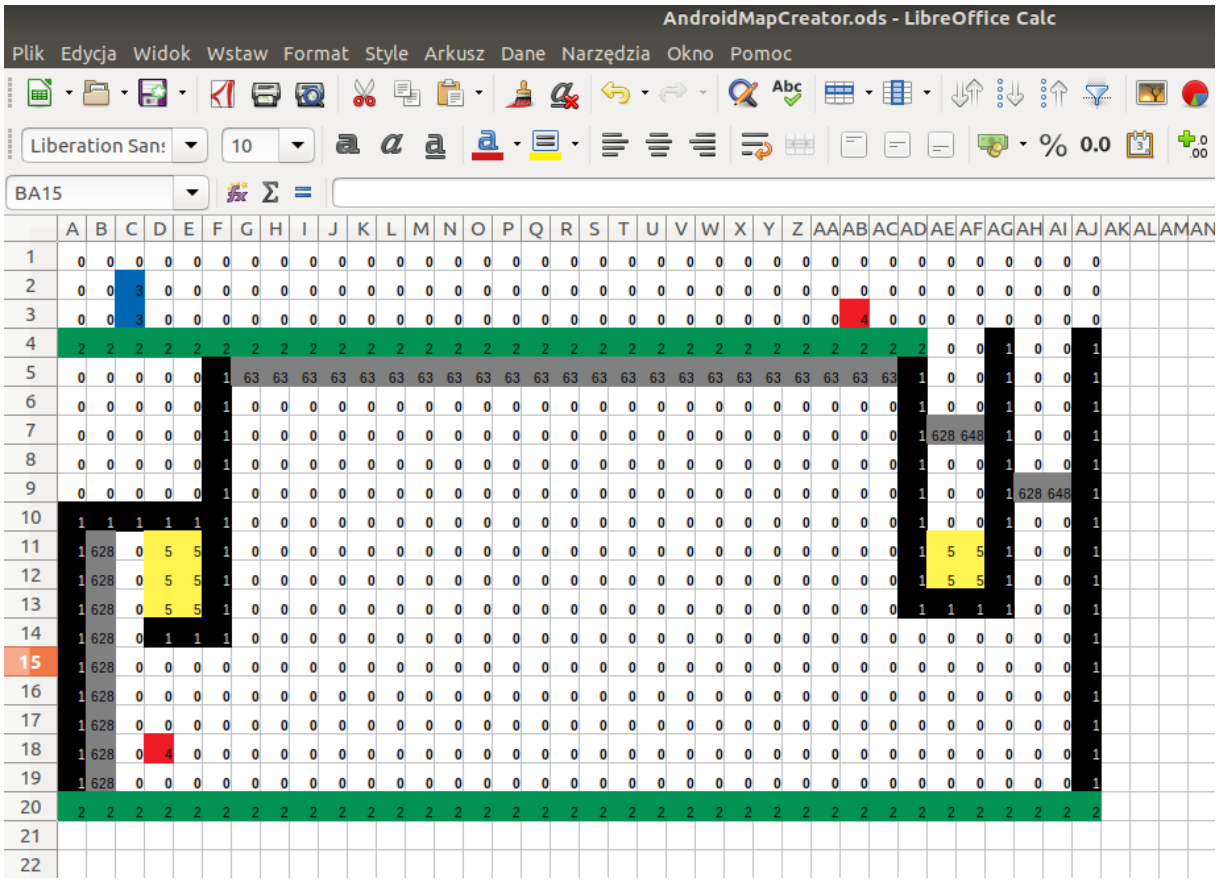
Rysunek 11: Pełny diagram klas

Dodatek B. Instrukcja tworzenia nowych poziomów

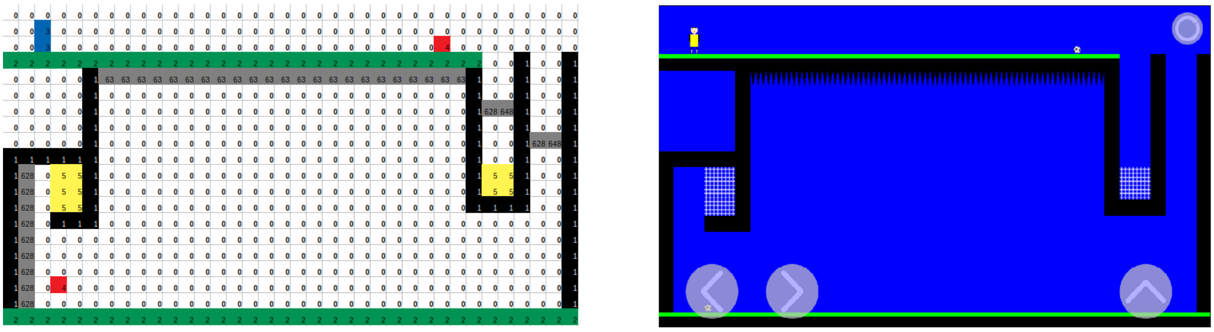
W tym dodatku opiszę sposób tworzenia nowych poziomów. Pomocy będzie arkusz kalkulacyjny „AndroidMapCreator” znajdujący się w repozytorium. Widok arkusza przedstawia rysunek nr 12. Pierwsze 36 kolumn i 20 wierszy jest wypełnione cyframi. Każda komórka arkusza przedstawia konkretny fragment siatki mapy (o której wspomniałem w rozdziale 2.4). Wpisując odpowiednią liczbę (zgodnie z zasadami z rozdziału 2.4), dana komórka zmieni swój kolor, co pomaga wyobrazić sobie budowę mapy. Porównanie świata w arkuszu kalkulacyjnym z tym w aplikacji przedstawia rysunek nr 13.

Następnie należy skopiować komórki używane do tworzenia świata (36 kolumn, 20 wierszy), a następnie ze skopiowanego tekstu usunąć białe znaki. Nie jest to problematyczne zadanie, ponieważ w Internecie można znaleźć wiele darmowych programów rozwiązujących tą trudność. Końcowym etapem jest dopisanie w klasie „Levels” linijki kodu znajdującej się w metodzie „createLevels()”. Powinna ona brzmieć: „levelslist.add(new String(„${text}));”, gdzie za zmienną „text” należy wkleić skopiowany tekst bez białych znaków (rysunek nr 14).

W dalszym rozwoju aplikacji planowana jest również zmiana sposobu wczytywania poziomów z plików tekstowych – bez ingerencji w kod.



Rysunek 12: Zrzut ekranu arkusza kalkulacyjnego



Rysunek 13: Porównanie świata w arkuszu kalkulacyjnym oraz w grze

```
private void createLevels() {


    levelsList.add(new String( original: "000000000000000000000000000000000000000000000000000000000000000000000
    levelsList.add(new String( original: "000000000000000000000000000000000000000000000000000000000000000000000
    levelsList.add(new String( original: "000000000000000000000000000000000000000000000000000000000000000000000
    levelsList.add(new String( original: "000000000000000000000000000006411110000000000000000000000000000000641
    levelsList.add(new String( original: "000000000000000000000000000000000000000000000000000000000000000000000
    levelsList.add(new String( original: "620300000000000000000000000000000001162030000000000000000000000004000
    levelsList.add(new String( original: "000000000000000000000000000000000000003000000000000000000000000000000
    levelsList.add(new String( original: "000000000000000000000000000000000000000000000000000000000000000000000
    levelsList.add(new String( original: "000000000000000000000000000000000000000000000000000000000000000000000
    levelsList.add(new String( original: "000000000000000000000000000000000000000000000000000000000000000000000
    levelsList.add(new String( original: "000000000000000000000000000000000000000000000000000000000000000000000


}
```

Rysunek 14: Fragment kodu przedstawiający metodę „createLevels()"