POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Dawid Pacholczyk
Nr albumu 6144

Tomasz Krajewski
Nr albumu 7175

Realizacja rzeczywistości rozszerzonej dla platform mobilnych

Praca Magisterska napisana
pod kierunkiem:

dr inż. Mariusz Trzaska

Warszawa, Lipiec, 2015

1 Spis treści

Streszczenie

Celem niniejszej pracy magisterskiej jest przybliżenie tematu rzeczywistości rozszerzonej, analiza możliwości jakie za sobą niesie, oraz adaptacji tej technologii w życiu codziennym dzięki urządzeniom przenośnym. W wyniku przeprowadzonej analizy, jako przykład praktycznego wykorzystania omawianej technologii została utworzona biblioteka ułatwiająca pracę z tym zagadnieniem. Następnie już przy wykorzystaniu tej biblioteki powstał prototyp aplikacji opartej o rzeczywistość rozszerzoną.

Prototyp składa się z dwóch części: gry mobilnej na platformę *Android* oraz zestawu narzędzi pozwalających na szybsze i prostsze tworzenie aplikacji z wykorzystaniem kluczowych – dla rzeczywistości rozszerzonej – elementów takich jak sensory czy kamera.

Projekt został napisany w języku JAVA na platformę Android z wykorzystaniem Google Play Service, Google Maps Android API v2, Support Library.

2 Wprowadzenie

Rzeczywistość rozszerzona to technologia, która nabiera coraz większego tempa i interesuje się nią coraz większa rzesza ludzi. Wynika to z faktu dynamicznego rozwoju technologii, którego jesteśmy świadkami. Ogólny i szeroki dostęp do urządzeń mobilnych typu *smartphone* oraz coraz lepsze parametry tych urządzeń sprawiają, że coś co jeszcze dekadę temu było dostępne tylko dla wybitnych specjalistów staje się narzędziem wykorzystywanym przez programistów aplikacji mobilnych na Nasze telefony.

Rzeczywistość rozszerzona powoli wkracza w coraz więcej aspektów Naszego życia. Można ją spotkać w medycynie, turystyce, edukacji, branży nieruchomości, rozrywce. Sposoby wykorzystania tej technologii wydają się niezliczone. Od telewizji, przez komputery, laptopy, tablety, telefony.

Technologia ta fascynuje rzeszę ludzi, jak wszystko co jeszcze niedawno kojarzyło się wyłącznie z filmami *science fiction.* Na pewno nie bez echa pozostaje fakt, że można dzięki niej znacząco usprawnić codzienne życia, z kolei specjalistom pomóc w ich trudnych zadaniach. Podczas przygotowywania się do pisania tej pracy magisterskiej, oraz przeglądania różnych projektów i zastosowań rzeczywistości rozszerzonej, zauważyliśmy, że wiele projektów dąży do uwolnienia rąk użytkowników. Realizacje te mają na celu na przykład wspomóc montaż na liniach produkcyjnych. System przedstawia pracownikowi na goglach poszczególne kroki postępowania. Minimalizuje to możliwość popełnienia błędu wynikającego z przeoczenia. Innym doskonałym przykładem jest projekt *Google Glass,* który miał zrewolucjonizować Nasze podejście do technologii. Miał przybliżyć Nam wszystko to z czego na co dzień korzystamy:

- Poczta
- Kalendarz spotkań
- Rozrywkę
- Poruszanie się po mieści

Projekt fascynujący, ale póki co jeszcze nie wkroczył na dobre w codzienność.

Przykład projektu *Google Glass* oraz innych podobnych pokazuje jak bardzo skomplikowane jest to przedsięwzięcie oraz ile problemów ze sobą niesie. Wraz z rozwojem technologii wkraczamy w zupełnie nowe rejony wiedzy, ale również etyki i moralności. Należy pamiętać jak wiele podniesiono sprzeciwów i protestów wobec tego i innych podobnych projektów. Ludzie obawiają się, że będą potajemnie nagrywani, fotografowani.

Postanowiliśmy, że w ramach Naszej pracy magisterskiej skupimy się na bezpiecznym obszarze, czyli tematem związanym z rozrywką skierowaną na urządzenia mobilne. Doszliśmy do wniosku, że na tym polu będziemy w stanie wykazać jak szerokie możliwości niesie ze sobą ta technologia, ale również

wskazać z jakimi komplikacjami i problemami mogą spotkać się twórcy oprogramowania wykorzystującego rzeczywistość rozszerzoną.

Obecnie aplikacje mobilne (a w szczególności gry) to jeden z najszybciej rozwijających się rynków związanych z oprogramowaniem. Zgodnie z raportem *Komisji Europejskiej* rynek aplikacji mobilnych do roku 2018 da zatrudnienie blisko 5 milionom osób, a jego wartość może osiągnąć nawet 63 miliardy euro [1]. Jest to kolejny obszar w jaki Nasz prototyp wpisuje się bardzo dobrze. Jednym z jego elementów będzie biblioteka, która ma na celu usprawnienie proces wytwarzania podobnego oprogramowania. Wyszliśmy z założenia, że skoro ta branża rozwija się w tak zawrotnym tempie to właśnie tu Nasz prototyp może uznać się – po stosownym rozwoju – z najszerszym przyjęciem i odzewem ludzi ze środowiska.

Oczywiście – jak już wspomnieliśmy – zastosować jest dużo więcej niż jedyni gry, a narzędzia, które będą wytworzone nie są skupione wyłącznie na tym. Dla przykładu renderowanie obiektu trójwymiarowego i przetwarzanie współrzędnych może być zarówno wykorzystane w grze jak w oprogramowaniu instytucji zajmującej się zabytkami, która chce przedstawić obiekty, które były, a już ich nie ma.

2.1 Charakter pracy

Niniejsza praca magisterska jest efektem pracy zespołu dwuosobowego. Stanowi opis wytworzonego prototypu oraz technologii rzeczywistości rozszerzonej. Staraliśmy się w niej oddać cały owoc Naszej pracy, zarówno wszystko to co udało się zrealizować w pełni, jak i również wszystkie problemy i ograniczenia na jakie natrafiliśmy. Zarówno jedno jak i drugie posiada bardzo dużą wartość merytoryczną. Wynika to z faktu, że technologia nie jest może niczym nowym, jednak nadal nie została w pełni zutylizowana, co otwiera furtkę dla przyszłych programistów i twórców oprogramowania.

Struktura pracy oddaje wszystkie najważniejsze aspekty naszej pracy. Zarówno te merytoryczne jak i czysto produkcyjne. Znajdziemy w niej:

- Niezbędne wprowadzenie i wyjaśnienie podstawowych zagadnień
- Analizę prototypu, który postanowiliśmy wytworzyć
- Przedstawienie najważniejszych elementów biblioteki
- Przedstawienie sposobu wykorzystania biblioteki
- Opis wykorzystanej technologii (składowych)

Załącznikiem do – zgodnie z Zarządzeniem Dziekana Wydziału PJATK – jest:

- Gotowa praca
- Kod źródłowy
- Praca w formie elektronicznej

2.2 Podział pracy – część implementacyjna

Przedstawiamy podział pracy w części implementacyjnej.

Tabela 1 Podział pracy - część implementacyjna. Źródło: Opracowanie własne.

Zakres/temat	Zespół
Zebranie wymagań	Dawid Pacholczyk
Analiza wymagań	Tomasz Krajewski
Projekt (biblioteka, gra)	Dawid Pacholczyk, Tomasz Krajewski
Obsługa sensorów	Tomasz Krajewski
Obsługa 3D i kamery	Dawid Pacholczyk
Implementacja założeń gry	Tomasz Krajewski
Implementacja logiki aplikacji	Dawid Pacholczyk
Analiza klatek z kamery na potrzeby aplikacji	Dawid Pacholczyk, Tomasz Krajewski
Praca nad poprawą wyników	Dawid Pacholczyk, Tomasz Krajewski
Testy	Dawid Pacholczyk, Tomasz Krajewski

2.3 Podział pracy – część pisemna

Przedstawiamy podział pracy w części pisemnej.

Tabela 2 Podział pracy - część pisemna. Źródło: Opracowanie własne.

Zakres/temat	Zespół
Wprowadzenie	Dawid Pacholczyk
Podstawowe definicje i pojęcia	Dawid Pacholczyk, Tomasz Krajewski
Wybrana technologia	Dawid Pacholczyk, Tomasz Krajewski
Analiza wymagań	Tomasz Krajewski
Największe trudności	Dawid Pacholczyk, Tomasz Krajewski
Proponowane rozwiązania	Dawid Pacholczyk, Tomasz Krajewski
Rozwiązania alternatywne	Dawid Pacholczyk
Składowe tworzonej biblioteki	Dawid Pacholczyk, Tomasz Krajewski
Wygładzenie odczytów	Tomasz Krajewski
Projekt aplikacji	Dawid Pacholczyk
Testy	Tomasz Krajewski
Podsumowanie i wnioski	Dawid Pacholczyk, Tomasz Krajewski

3 Podstawowe definicje i pojęcia

W niniejszej pracy wielokrotnie będziemy się odnosić do takich terminów jak „urządzenie mobilne", „smartphone", czy też „rzeczywistość rozszerzona". Dlatego też w tym rozdziale postaramy się po krótce przybliżyć, co oznaczają te terminy. Ponadto zostaną tu umieszczone informacje, dlaczego dane zagadnienia zainteresowały autorów tej pracy, oraz wreszcie przedstawimy przykłady praktycznego ich zastosowania.

3.1 Urządzenia mobilne

Pierwszym terminem, który przybliżymy w tym rozdziale jest pojęcie „urządzeń mobilnych". Wybór właśnie tego zagadnienia na początek nie jest przypadkowy, gdyż rozwiązania mobilne są podstawą, bez których to nie powstała by ta praca. Poniżej zawarliśmy krótką definicje takiego urządzenia z wyszczególnieniem smartphonów. Również zawarty został tu podrozdział z danymi statystycznymi przy pomocy, których postaramy się udowodnić, że rynek urządzeń mobilnych jest perspektywicznym i wartym zainteresowania.

3.1.1 Definicja urządzeń mobilnych

Można znaleźć wiele różnych definicji urządzeń mobilnych, lecz my zdecydowaliśmy się na wykorzystanie tej, która pojawia się najczęściej i w naszej opinii jest kompletna. Definicja ta została zaproponowana przez M. Macutkiewicza w pracy „*Wykorzystanie rozwiązań mobilnych w systemach klasy e-commerce"*. Definiuje ona urządzenia mobilne, jako:

„Urządzenie elektroniczne pozwalające na przetwarzanie, odbieranie oraz wysyłanie danych bez konieczności utrzymywania przewodowego połączenia z siecią. Urządzenie mobilne może być przenoszone przez użytkownika bez konieczności angażowania dodatkowych środków" [2]

Powyższa definicja ukazuje nam, jak szerokim terminem jest pojęcie urządzeń mobilnych. Dotyczy ona takich urządzeń jak smartfony, tablety, laptopy, czy też coraz popularniejszych rozwiązań zaliczanych do kategorii technologii ubieralnych (z ang. Wearables).

Oczywiście wymienione tutaj przykłady są jedynie małym wycinkiem wszystkich rozwiązań, które można zaliczyć do urządzeń przenośnych. Dla nas jednak najważniejszym przedstawicielem tej grupy jest smartphone, który to będzie docelową platformą tworzonego w tej pracy rozwiązania. Przyjrzyjmy się zatem jego definicji:

„Przenośne urządzenie telefoniczne (ang. smartphone) łączące w sobie funkcje telefonu komórkowego i komputera kieszonkowego (PDA – Personal Digital Assistant). Pierwsze smartfony powstały pod

koniec lat 90., a obecnie łączą funkcje telefonu komórkowego, poczty elektronicznej, przeglądarki sieciowej, pagera, GPS, jak również cyfrowego aparatu fotograficznego i kamery wideo.” [3]

Ponownie definicja pozwala nam zaklasyfikować wiele urządzeń, jako smartphone, a granica pomiędzy zwykłym telefonem komórkowym, posiadającym rozbudowane funkcje, a właśnie smartphone`em jest niezwykle cienka. Według powszechnie przyjmowanej definicji urządzenie powinno również charakteryzować się ekranową lub fizyczną klawiaturą typu QWERTY. Jednak na potrzeby tej pracy przyjmiemy, że urządzenie zwane smartphone`em musi charakteryzować się takimi cechami:

- Posiadać ekran min. 4 calowy, współpracujący z wielodotykiem
- Posiadać moduł GPS, oraz wspierać podstawowe sensory takie jak np. akcelerometr
- Działać pod kontrolą dowolnego nowoczesnego systemu mobilnego (w przypadku tej pracy systemu Android w wersji min. 4.4 KitKat)
- Udostępniać przestrzeń dyskową, umożliwiającą wgrywanie dodatkowego oprogramowania

3.1.2 Dane statystyczne

Rynek urządzeń mobilnych jest jednym z najszybciej rozwijających się rynków historii. Za dowód może służyć fakt, iż sprzedaż smartphone`ów już w pierwszym kwartale 2013 roku przegoniła sprzedaż standardowych telefonów komórkowych, Jednocześnie w tym samym roku przekroczyła granice miliarda sprzedanych nowych urządzeń, co stanowiło lepszy wynik o 38,4% w porównaniu do poprzedniego roku [4].

Tabela 3 Sprzedaż smartphone lata 2012-2013. Źródło: [5]

Producent	**2013 Sprzedaż w milionach sztuk**	**2013 Udział w rynku**	**2012 Sprzedaż w milionach sztuk**	**2012 Udział w rynku**	**Zmiana rok-do-roku**
Samsung	313.9	31.3%	219.7	30.3%	42.9%
Apple	153.4	15.3%	135.9	18.7%	12.9%
Huawei	48.8	4.9%	29.1	4,0%	67.5%
LG	47.7	4.8%	26.3	3.6%	81.1%
Lenovo	45.5	4.5%	23.7	3.3%	91.7%
Pozostałe	394.9	39.3%	290.5	40.1%	35.9%
Suma	**1004.2**	**100%**	**725.3**	**100%**	**38.4%**

Szukając pomysłu na temat pracy chcieliśmy, aby w jej wyniku powstało rozwiązanie trafiające na jedną z popularnych platform tak, aby jego wartość nie była jedynie czysto akademicka. Przeprowadzona analiza rynku, a w szczególności załączone powyżej wyniki sprzedażowe pozwalają stwierdzić, że tworząc oprogramowanie na smartphone`y jesteśmy w stanie dotrzeć do bardzo licznego grona odbiorców. Więcej na temat wyboru platformy docelowej została zawarta w rozdziale czwartym.

3.2 Rzeczywistość rozszerzona

Rzeczywistość rozszerzona (*Augmented Reality)* jest to technologia/system, które pozwalają na połączenie świata rzeczywistego ze światem wirtualnym w czasie rzeczywistym. Odbywać się to może na kilka sposobów:

- za pośrednictwem obrazu z kamery
- za pośrednictwem dźwięku
- z wykorzystaniem dodatkowych sensorów.

Istnieje bardzo wiele różnych kombinacji tych elementów, które pozwalają na szerokie i zróżnicowane wykorzystanie rzeczywistości rozszerzonej.

Co ciekawe, rzeczywistość rozszerzona to nie jest nowinka technologiczna. Elementy z nią związane towarzyszą Nam już dawna. Jednym z najlepszych przykładów są mecze piłki nożnej jakie możemy oglądać w telewizji. Widzimy tam obraz transmitowany na żywo (rzeczywistość), nadawca niejednokrotnie rozszerza to, co widzimy o cyfrowe informacje, na przykład linia spalonego, dzięki której widz widzi to czego normalnie mógłby nie zauważyć.

W dalszej części pracy przybliżone zostanie wykorzystanie tej technologii w różnych aspektach i dziedzinach życia takich jak wojsko czy medycyna.

3.2.1 Rzeczywistość rozszerzona vs Rzeczywistość wirtualna

Te dwie zbliżone do siebie technologie, mimo zasadniczych różnic są ze sobą mylone. W Naszej pracy skupiamy się wyłącznie na rzeczywistości rozszerzonej, czyli nałożeniu treści cyfrowych na faktyczny obraz świata w czasie rzeczywistym. Z kolei rzeczywistość wirtualna jest to przedstawienie świata cyfrowego – najczęściej za pośrednictwem specjalnego zestawu do założenia – i pozwolenie użytkownikowi na wejście z nim w interakcje [6].

3.2.1.1 Rzeczywistość wirtualna

Jedną z ogólnie przyjętych definicji rzeczywistości wirtualnej jest tzw. I^3 czyli:
Interakcja (**I**nteraction) | Zagłębienie (**I**mmersion) | Wyobraźnia (**I**magination) [7]

Ta definicja bardzo wyraźnie wskazuje najważniejsze elementy, które tworzą sedno tej technologii. Ze względu na fakt, że obraz/świat przedstawiony przez rzeczywistość wirtualną nie musi

mieć korelacji ze światem rzeczywistym (w przeciwieństwie do rzeczywistości rozszerzonej) to autorzy takiego oprogramowania nie są niczym skrępowani.

Rzeczywistość wirtualna jest obecna w dwóch podstawowych segmentach runku:

- Rozrywka – w szczególności gry komputerowe. O tej części jest coraz głośniej ze względu na niedawny wykup technologii *Oculus Rift* przez *Facebook`a.* Technologia ta była znana od dawna, ale dopiero to przejęcie nagłośniło całą sytuację.
- Szkolenia – jednym z najlepszych przykładów są symulatory tworzone z myślą o pilotach samolotów. Budowane są specjalistyczne maszyny wraz z oprogramowaniem, które mają jak najwierniej odwzorować świat rzeczywisty i pomóc w szkoleniu. Należy jednak pamiętać, że jest to jedynie symulacja i cały świat jest jedynie wyrenderowany.

3.2.1.2 Różnice

W Tabela 4 przedstawiamy przykładowe różnice między rzeczywistością rozszerzoną, a rzeczywistością wirtualną. Ważne jest, aby przed przystąpieniem do zapoznania się z dalszą częścią pracy uświadomić sobie jak z pozoru podobne technologie mocno się różnią.

Tabela 4 Rzeczywistość rozszerzona vs Rzeczywistość wirtualna. Źródło: Opracowanie własne.

Rzeczywistość wirtualna	Rzeczywistość rozszerzona
Użytkownik ma kontakt jedynie ze sztucznym światem wykreowanym przez programistę. Może on odpowiadać rzeczywistemu (symulacja), ale nie musi.	Użytkownik ma stały kontakt z otoczeniem (rzeczywistym). Oprogramowanie/urządzenie ma na celu jedynie rozszerzenie informacji/bodźców, jakie docierają do użytkownika.
Wymagane jest specjalistyczne urządzenie tzw. *Headset* który pozwoli na odseparowanie użytkownika od świata rzeczywistego.	Wystarczy komputer lub *smartphone* czyli urządzenie codziennego użytkownika. Znacząco zwiększa to dostępność technologii.
Całkowita separacja użytkownika od świata rzeczywistego. Oprogramowanie pełni rolę typowo zadaniową, czyli jest kreowane w jednym celu (szkolenie, rozrywka).	Pozwala na normalne funkcjonowanie użytkownika, zwiększa ilość informacji, jakie do niego docierają, poprawia jakość pracy (*Industry 4.0),* zwiększa efektywność. Oprogramowanie wraz z odpowiednim urządzeniem może być wielozadaniowe (*Google Glass)*

3.2.2 Zastosowanie

Rzeczywistość rozszerzona posiada bardzo dużo zastosowań. Prawda jest taka, że nadal jest to swoista nisza rynkowa i jest to nadal niewyczerpany temat. Świadczą o tym nowe oraz coraz bardziej śmiałe projekty takiej jak:

- Projekt *Tango* rozpoczęty przez firmę Google (zobacz [8]).
- Google Glass (zobacz [9])
- Tworzony przez *U.S. Air Force* hełm dla pilotów *F-35* oparty o rzeczywistość rozszerzoną do wsparcie ich w walce (zobacz [10])
- Okulary z wyświetlaczem HUD dla astronautów tworzone przez *Osterhout Design Group* dla rządu amerykańskiego i NASA [11]

Przedstawiliśmy tylko kilka stosunkowo młodych i nowych pomysłów, które pokazują jak wielki potencjał drzemie w tej technologii. Szacuje się, że wartość rynku związanego z rzeczywistością rozszerzoną, do roku 2018 będzie wynosić ponad miliard dolarów. Jest to wzrost o blisko 15% w stosunku do roku 2013 [12].

3.2.2.1 Wojsko

Jednym z odbiorców technologii rzeczywistości rozszerzonej od dawna było wojsko. Popularnym przykładem są wyświetlacze HUD (*Head-Up Display*) montowane w myśliwcach bojowych, które maja na celu dostarczanie pilotowi dodatkowych informacji takich jak wysokość, kierunek lotu itp.

Rysunek 1 Head-Up Display. Źródło: [13]

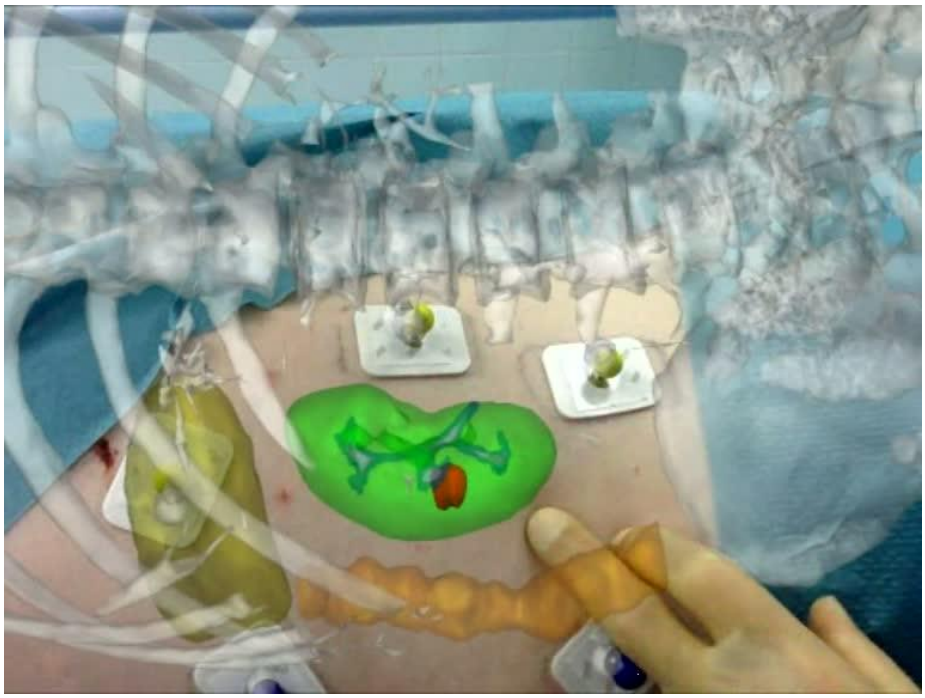
Wraz z rozwojem technologii trwają pracę nad dostarczeniem specjalnych okularów - na wzór *Google Glass* – dla żołnierzy piechoty, które miałyby dostarczać najważniejszych informacji o aktualnym położeniu oraz sytuacji oddziału. Mają one na celu zwiększyć ilość informacje jakie docierają do ludzi na polu bitwy, a tym samym zwiększyć ich szansę na podjęcie prawidłowych decyzji.

Rzeczywistość rozszerzona w przypadku zastosowań militarnych ma na celu zdobycie przewagi na przeciwnikiem. Lepsza jakość informacji, lepszy wywiad, szybsze dostrzeżenie potencjalnego zagrożenia, te wszystkie czynniki zwiększają bezpieczeństwo, a równocześnie zwiększają szanse na wygranie potyczki. Możliwe, że dzięki tej gałęzi technologia ta ma największe szanse na rozwój, tak jak miało to miejsce w przypadku internetu.

3.2.2.2 Medycyna

Medycyna posiada bardzo szerokie pole do zastosowania rzeczywistości rozszerzonej. Jednym z bardzo ciekawych przykładów jest – stosunkowo świeży – projekt utworzony przez firmę *Evena* który pracuje nad goglami *Eyes-On*. Mają one na celu obrazowanie naczyń krwionośnych pacjenta w czasie rzeczywistym [14].

Innym przykładem zastosowania są próby obrazowania narządów w trakcie operacji. Polega to na tym, że chirurg widzi również te narządy, które są ukryte głębiej. Ma to na celu pozwolić lekarzowi na pełną kontrolę oraz pełen wgląd w aktualny stan pacjenta.



Rysunek 2 Wizualizacja narządów w czasie rzeczywistym. Źródło: [15]

Technologia rzeczywistości rozszerzonej w medycynie dąży do minimalizowania błędów jakie mogą popełniać ludzie, wspomóc ich w podejmowaniu decyzji, a tym samym zwiększenie bezpieczeństwa pacjentów.

3.2.2.3 Architektura

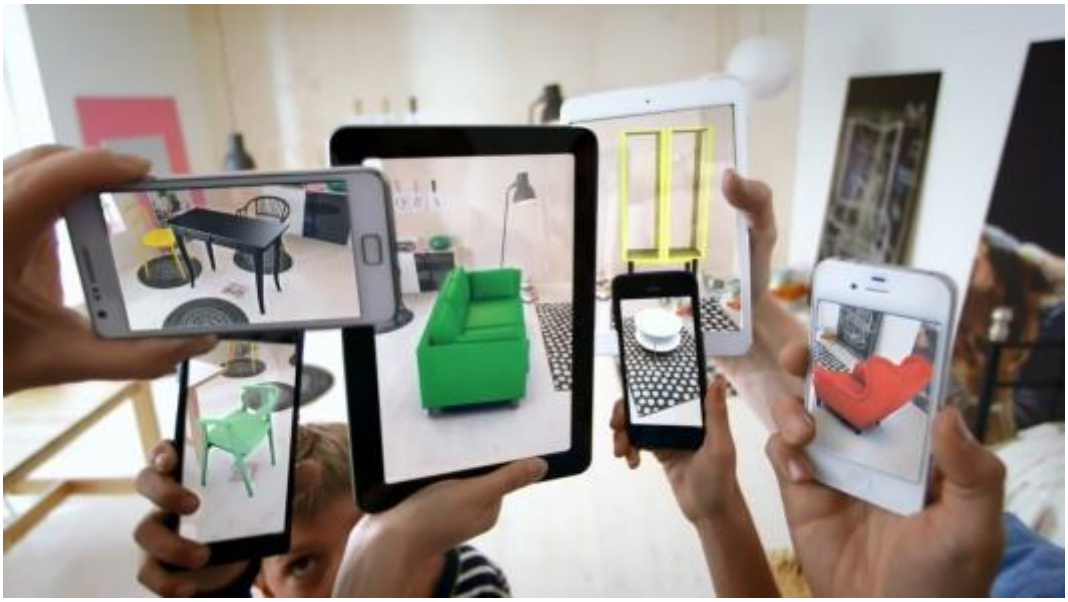
W przypadku branży budowlanej najczęstsze zastosowanie rzeczywistości rozszerzonej dotyczy poprawienia jakości prezentowanej oferty klientowi. Ma to na celu:

- Poprawę warstwy wizualnej prezentacji
- Poprawę jakości informacji jakie dostarczane są potencjalnemu klientowi
- Zwiększenie zainteresowania potencjalnego klienta

Rysunek 3 Prezentacja modelu 3D z katalogu. Źródło: [16]

Na rysunku (Rysunek 3) przedstawiliśmy w jaki sposób zwykły katalog z projektami domów może przemienić się w wirtualną prezentację, która dostarcza dodatkowych informacji klientowi. Jest to bardzo wygodne rozwiązanie pozwalające w prosty i przystępny sposób dostarczyć wszystkich najważniejszych informacji.

Innym przykładem jest oprogramowanie tworzone na potrzeby *IKEA* która pozwala na rozplanowanie produktów z katalogu wewnątrz własnego domu.



Rysunek 4 Katalog IKEA. Źródło: [17]

Po załadowaniu danego produktu z katalog za pośrednictwem telefonu lub tabletu, użytkownik może nanieść dany produkt na przestrzeń swojego domu i zobaczyć w jaki sposób będzie się on prezentował.

3.2.2.4 Podsumowanie

W tym rozdziale chcieliśmy wykazać jak szerokie zastosowanie może mieć rzeczywistość rozszerzona w dzisiejszym świecie. Co więcej widać wyraźnie, że cel zastosowania tej technologii jest również bardzo zróżnicowany i jest ściśle powiązany segmentem rynku:

- Zdobycie przewagi
- Zwiększenie bezpieczeństwa
- Poprawa jakości informacja
- Poprawa prezentacji
- Zabawa
- Rozrywka

Rzeczywistość rozszerzona to dziedzina, która stale się rozwija i będzie się rozwijać wraz z nowinkami technologicznymi. Każda poprawa urządzeń takich jak laptop, *smartphone* otwiera nowe możliwości rozwoju oprogramowania opartego o rzeczywistość rozszerzoną. To technologia jest aktualnie problemem numer jeden wielu ciekawych i innowacyjnych projektów. Jednak nie jest to problem, którego nie da się pokonać.

Należy jednak pamiętać o tym, że jak wszystko, również rzeczywistość rozszerzona, posiada pewne wady.

Po pierwsze, aktualnie rzeczywistość rozszerzona wykorzystywana jest głównie na telefonach i tabletach. Jest to mało wygodne, mało intuicyjne, a przede wszystkim zaburza w pewien sposób prostote

dostępu do danych cyfrowych nałożonych na świat rzeczywisty jaką powinna ze sobą nieść. Oczywiście powstają kolejne projekty związane z okularami pozwalającymi na korzystanie z tej technologii na dużo wyższym poziomie, jednak nadal są to prototypy lub bardzo specjalistyczne urządzenia.

Po drugie, cena. Rozmawiając o wysokiej klasy sprzęcie czy oprogramowaniu związanym z rzeczywistością rozszerzoną, musimy liczyć się z wysokimi kosztami. Po pierwsze same urządzenia są stosunkowo drogie. Po drugie, wiele dziedzin związanych z tą technologią są w ciągłej fazie rozwoju, dlatego ilość specjalistów jest stosunkowo niska, a poziom skomplikowania wymaga osób o naprawdę dużej wiedzy.

Pomimo, że powyższe wady wydają się bardzo istotne, uważamy, że technologia rzeczywistości rozszerzonej będzie w przyszłości elementem równie codziennym jak radio czy telewizor. To tylko kwestia czasu, gdy stanie się ona tania i dostępna dla wszystkich.

3.3 Założenia biblioteki

W ramach Naszej pracy magisterskie chcieliśmy osiągnąć dwa podstawowe cele:

- Przybliżyć technologię rzeczywistości rozszerzonej
- Utworzyć interesujący prototyp, który zaprezentuje najważniejsze elementy związane z rzeczywistością rozszerzoną

Nasz prototyp jest podzielone na dwa elementy. Zbiór narzędzi pozwalający na szybsze wytwarzania aplikacji mobilnych opartych o rzeczywistość rozszerzoną. Gra – jej założenia opisana w dalszej części rozdziału – której zadaniem jest przedstawienie:

- Koncepcji rzeczywistości rozszerzonej
- Wykorzystania wstępnej wersji biblioteki pozwalającej na szybszą produkcję aplikacji podobnego typu

Wstępna wersja biblioteki ma za zadanie pomóc podczas wykorzystywania najpopularniejszych elementów związanych z rzeczywistością rozszerzoną i platformami mobilnymi. Dla przykładu są to: lokalizacja (np. gps) oraz kamera.

3.3.1 Zakres działania biblioteki

W ramach tworzonej biblioteki skupiliśmy się na trzech najważniejszych obszarach:

1. Kamera
2. Sensory
3. Grafika 3D

Obszary te pozwalają na kompleksowe przedstawienie koncepcji rzeczywistości rozszerzonej oraz możliwości jakie ze sobą niesie.

3.3.2 Podstawowe założenia

W trakcie budowania Naszej biblioteki kierowaliśmy się kilkoma podstawowymi założeniami. Wynikają one ze specyfiki wytwarzanego oprogramowania.

- Ponowne wykorzystanie
- Dostarczenie metod wykonujących konkretne zadania
- Odseparowanie od warstwy kontroli aplikacji
- Rozszerzalność

3.3.2.1 Ponowne wykorzystanie

Staraliśmy się, aby w ramach Naszej pracy, zestaw narzędzi nad którym pracowaliśmy był w jak największym stopniu gotowy do ponownego użycia. W tym celu dbaliśmy o to, aby żaden fragment logiki aplikacji nie znalazł się w klasach odpowiedzialnych za dostarczenie stosownych metod.

Dzięki odpowiednim zabiegom, Nasza biblioteka może zostać podpięta do zewnętrznych projektów (niezwiązanych z Naszym prototypem) dzięki czemu program będzie miał dostęp do metod i narzędzi, które wytworzyliśmy w trakcie pisania tej pracy.

3.3.2.2 Dostarczenie metod wykonujących konkretne zadania

Wszystkie funkcjonalności w ramach Naszej biblioteki zostały wytworzone w 100%. Oznacza to, że dostarczone narzędzia są w pełni sprawne i gotowe do użycia. Realizują konkretne zadania i dostarczają konkretne funkcjonalności w ramach większej całości.

3.3.2.3 Odseparowanie od warstwy kontroli aplikacji

Podczas tworzenia Naszego prototypu równolegle pracowaliśmy nad biblioteką oraz aplikacją, która zaprezentuje Naszą koncepcję. Praca w takim trybie może spowodować omylne umieszczenie fragmentu logiki aplikacji w metodach, które potem zostaną udostępnione w formie biblioteki narzędziowej. W celu uniknięcia takiej sytuacji należy przed rozpoczęciem pracy prawidłowo przemyśleć i zaprojektować poszczególne fragmenty systemu. Tylko w ten sposób nie doprowadzi się do sytuacji przenikania dwóch projektów.

3.3.2.4 Rozszerzalność

Biblioteka jaka zostanie wytworzona w ramach niniejszej pracy magisterskiej będzie jedynie początkiem. Jej głównym zadaniem będzie przedstawienie pewnej koncepcji, możliwych rozwiązań, możliwych dróg rozwoju.

Jak w innych narzędziach tego typu, w przypadku rozwoju przez osoby trzecie, rozwój powinien odbywać się poprzez rozszerzanie kodu, a nie jego modyfikowanie.

3.3.3 Potencjalne zagrożenia

Dostrzegamy pewne potencjalne zagrożenia związane z Naszą biblioteką. Obawiamy się kwestii związanych z renderowanie grafiki 3D i jej spięciem z sensorami. Natura sensorów jest stosunkowo zmienna i zależna od wielu czynników np. otoczenia czy urządzenia na których dokonywany jest pomiar. Zakładamy, że podczas tworzenia poszczególnych narzędzi będziemy musieli zadbać o poprawę jakości odczytów.

Jak w przypadku wielu aplikacji mobilnych, również w Naszym przypadku istotnym aspektem jest zużycie baterii. Podczas pracy będziemy musieli dbać o to, aby budowane metody były jak najmniej skomplikowane pod względem złożoności. Musimy zadbać to ponieważ każda dodatkowo operacja obliczeniowa zużyje dodatkową porcję energii. Aby Nasza biblioteka mogła być kiedyś rozpatrywana jako warta rozwoju i użycia musi być pod tym względem jak najbardziej oszczędna.

3.4 Koncepcja przykładu zastosowania

W niniejszym rozdziale postaramy się przybliżyć podstawowe koncepcje i założenia prototypu gry mobilnej, która zostanie wytworzony w ramach pisania pracy. Gra stanowi jedynie przykład zastosowania zarówno rzeczywistości rozszerzonej jak i tworzonej przez Nas biblioteki. W dalszej części pracy wyjaśnimy czym kierowaliśmy się podczas doboru tematyki aplikacji, która ma być przykładem zastosowania.

Naszym celem jest utworzenie aplikacji, która będzie wykorzystywała popularne założenie geocaching`u co pozwoli Nam na połączenie wielu elementów takich jak:

- Kompas
- Pomiar pola magnetycznego
- Odczyt GPS
- Grafika 3D

Wszystko to razem połączone w celu utworzenia rzeczywistości rozszerzonej w oparciu o popularne i proste zasady.

3.4.1 Geocaching – słowo o pierwowzorze

Założenia tworzonej gry powstały w oparciu o znaną na całym święcie zabawę zwaną „Geocaching”, która to zdobywa swoją popularność od połowy roku 2000. Jej idea jest niezwykle prosta i bazuje na grupie pasjonatów, którzy naprzemiennie ukrywają i szukają przedmiotów zlokalizowanych w dowolnym miejscu na święcie. Oryginalna zabawa jest nierozerwalnie powiązana jest z faktem odkodowania sygnału GPS (zobacz: [18]), który to dzięki swoim odczytom umożliwia lokalizacje skrytek (od angielskiego słowa „*caching*”) z ukrytymi wcześniej rzeczami. Fakt uwolnienia sygnału wykorzystał Amerykanin Dave Ulmer, który to jako pierwszy postanowił ukryć wiadro w lesie i

udostępnić jego współrzędne innym entuzjastą technologii GPS a jego pomysł wkrótce wyewoluował w zabawę zwaną *Geocahchingiem*.

Przeważnie skrytki umieszcza się w miejscach turystycznie atrakcyjnych, tak aby przyciągał jak największą liczbę poszukiwaczy. Równie ważnym elementem jest pomysłowe ukrycie przedmiotu, poprzez jego maskowanie, oraz dodanie zadań logicznych, których rozwiązanie jest niezbędne w celu pójścia dalej. Niemniej jednak finalnie ostatni pojemnik i tak znajdujemy dzięki uzyskanym koordynatą. Po odnalezieniu skrytki na pewno znajdziemy w niej papierowy dziennik zwany „loogbook", w którym to musimy się własnoręcznie wpisać – stanowi to fundamentalną zasadę zabawy. Jeśli zaś chodzi o ukryty w skrytce skarb, nie ma żadnych odgórnych wymagań czym ów przedmiot powinien być. Oczywiście trzeba zadbać by nie był to przedmiot łatwo psujący się albo niebezpieczny czy nielegalny. Wszystko po za tym pozostaje w gestii osoby ukrywającej pojemnik (zobacz [19]).

3.4.2 Podstawowe założenia

Dla celów tej pracy postanowiono rozbudować idee *Geocaching* o elementy rzeczywistości rozszerzonej. Główne założenia tworzonej gry pozostają zgodne z jej pierwowzorem. Tak więc całość będzie oparta o sygnał GPS a głównym zadaniem gracza będzie znalezienie ukrytego wirtualnego skarbu. Osoba korzystająca z tej aplikacji na początku będzie mogła określić promień w którym zamierza się poruszać oraz ilość ukrytych elementów. Zakładamy, że każdy z elementów będzie swojego rodzaju puzzlem, składającym się na finalny skarb.

Po wybraniu wstępnych opcji graczowi zostanie wyświetlona mapa z naniesionymi punktami w których to znajdują się poszczególne elementy w wyznaczonym promieniu. Gra nie będzie ingerować w kolejność lokalizowania i zbierania przedmiotów, także jedynie od osoby uczestniczącej w zabawie będzie zależał wybór trasy. Gracz ma możliwość wybrania punktu do którego będzie kierowany przez kompas wskazujący kierunek. Dla urozmaicenia rozrywki aplikacja nie będzie wyznaczać dokładnej trasy. O tym w jaki sposób oraz którędy dotrze się do celu decyduje gracz.

W momencie gdy grający zbliży się do jednego z elementów układanki system wygeneruje obiekt w formie warstwy na obrazie z kamery. Renderowany będzie obiekt 3D, który poprzez wykorzystanie sensorów wbudowanych w urządzeniu, będzie można oglądać z różnych stron. Celem gry jest zebranie/odnalezienie wszystkich przedmiotów.

3.4.3 Możliwe opcje rozbudowy

Opisywana powyżej idea pozwala w późniejszym czasie na rozważeniu dodania dodatkowych elementów rozgrywki takich jak:

- Tryb dla wielu graczy
- Zadania logiczne umożliwiające odnalezienie kolejnych koordynatów
- Zadania logiczne umożliwiające podniesienie odnalezionego przedmiotu

- Zbieranie danych, umożliwiających tworzenie historii danego gracza bądź budowanie rankingów
- Zbudowanie platformy pozwalającej na stworzenie społeczności wokół gry

3.5 Motywacja wyboru tematu

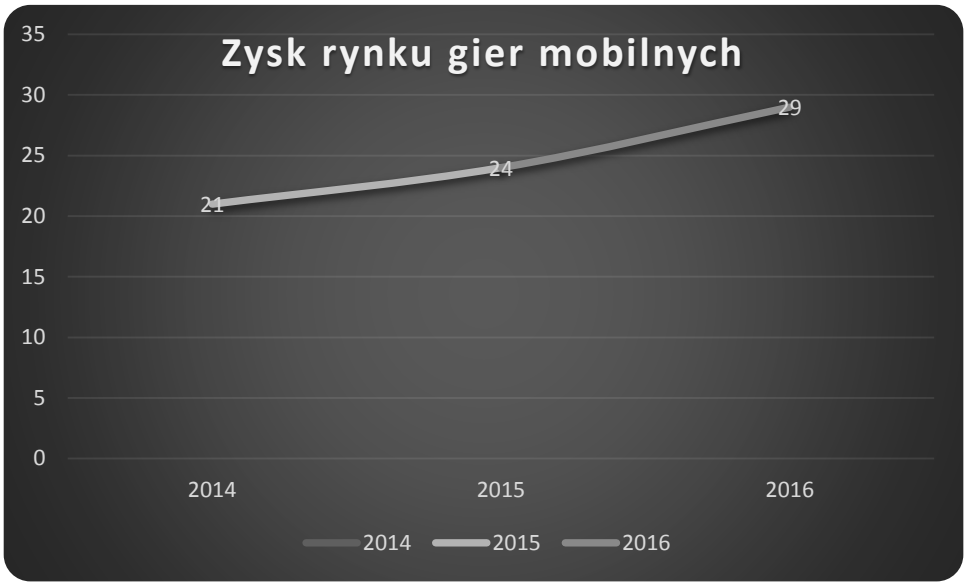
Decydując się na wybór tematu naszej pracy przede wszystkim chcieliśmy skupić się na nowoczesnych technologiach, działających na dynamicznie rozwijającym się rynku. Dlatego też zdecydowaliśmy się stworzyć oprogramowanie na urządzenia mobilne. Dalszy proces analizy tematu zawęził wybór do urządzeń działających pod kontrolą systemu Android.

W powyższym rozdziale przedstawiliśmy wyniki świadczące o tym jak wielkim rynkiem jest rynek urządzeń mobilnych, jak dynamicznie rozwija się sprzedaż smartphone`ów. Opisaliśmy również zalety rozszerzonej rzeczywistości.

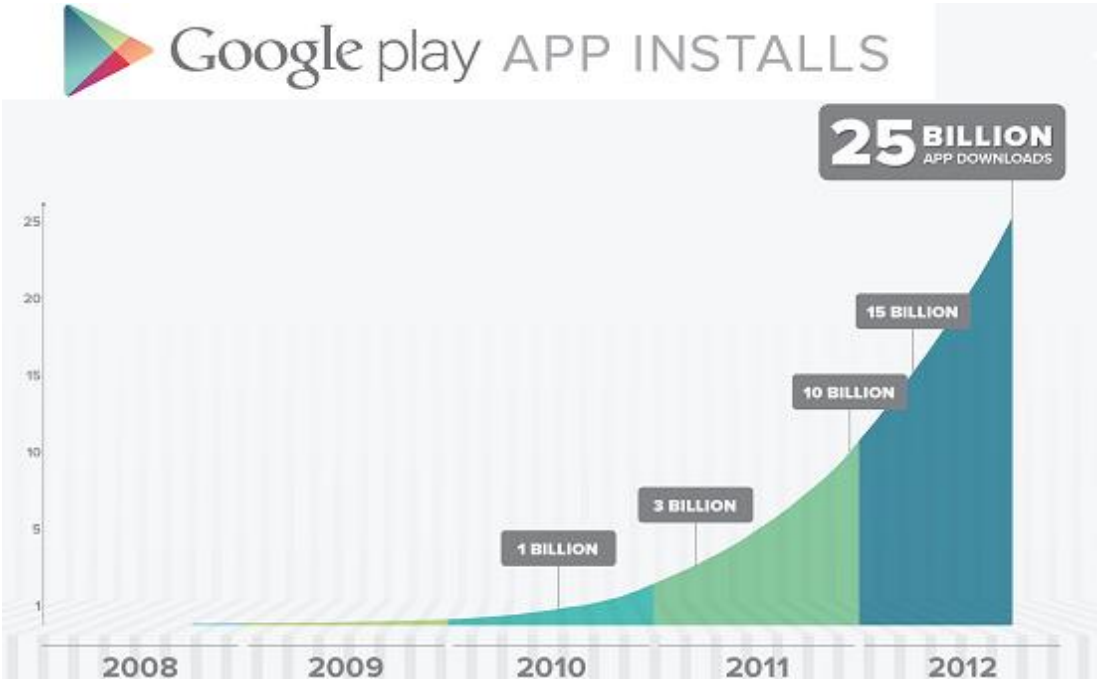
W niniejszej pracy Nasz dwuelementowy prototyp składa się z biblioteka oraz aplikacja mobilna. Uznaliśmy, że utworzenie biblioteki pozwoli Nam na ogólne i szerokie przedstawienie tematu. Pozwoli na utworzenie użytecznego narzędzia, które może okazać bardzo przydatne w trakcie dalszej nauki i rozwijania wiedzy w tematyce rzeczywistości rozszerzonej. Co więcej Nasza biblioteka będzie miała wiele zastosowań. Jesteśmy przekonani, że będzie można ją wykorzystać nie tylko do tworzenia gier mobilnych, ale również aplikacji *stricte* biznesowych.

Do tej pory pomijaliśmy uzasadnienie wyboru rynku gier, zamiast na przykład aplikacji biznesowych. Chcieliśmy, aby oprogramowanie, które wykorzystamy do zaprezentowania rzeczywistości rozszerzonej oraz Naszej biblioteki było w jak najlepszym stopniu dostosowane do aktualnych realiów rynkowych.

Po pierwsze, gry mobilne wydały nam się zwyczajnie tematem dużo ciekawszym i bardziej interesującym niż wspomniane wcześniej aplikacje biznesowe. Jednak decydujący wpływ na wybór tematu miał czynnik ekonomiczny. Jak widać na wykresach (Wykres 1 i Wykres 2) prognozowany jest dalszy wzrost zysku z tytułu gier mobilnych. Ciekawie wygląda krzywa pobrań wszystkich aplikacji poprzez oficjalny sklep Google. Niecałe dwa lata od osiągnięcia pierwszego miliarda pobrań aplikacji, zajęło pobranie kolejnych 14 miliardów aplikacji. Natomiast kolejne 10 miliardów zostało już pobrane w nieco ponad pół roku. Pokazuje to jak wielką dynamiką wzrostu charakteryzuje się ten rynek.



Wykres 1 Szacunkowy zysk (w miliardach dolarów) z rynku gier mobilnych lata 2014-2016. Źródło: [20]



Wykres 2 Liczba pobranych aplikacji z Google Play 2008-2012. Źródło: [21]

Reasumując możliwość dotarcia do tak wielkiej ilości odbiorców, poprzez tworzenie ciekawego projektu przy wykorzystaniu rozszerzonej rzeczywistości zadecydowały podczas kształtowania się tego tematu.

4 Wybrana technologia

W rozdziale tym zostaną przybliżone technologie jakie zostały wybrane do wytworzenia biblioteki jak i stworzenia prototypu. Wyjaśnimy czym kierowaliśmy się podczas dokonania wyboru, a także przybliżymy szczegóły techniczne elementów składowych.

4.1 Platforma systemowa – Android

Zarówno tworzona przez nas biblioteka jak i prototyp gry zostały stworzone pod konkretny system mobilny – Android. Wybór tego systemu nie był przypadkowy, gdyż jest to obecnie najpopularniejszy system, który charakteryzuje się dynamicznym rozwojem. W dalszych podrozdziałach postaramy się pokazać odpowiednie fakty, które potwierdzają ten opis systemu Android.

4.1.1 Historia systemu

W tej pracy zdecydowaliśmy się na stworzenie oprogramowaniu w oparciu o system Android. Jego początki sięgają roku 2003, kiedy to mała firma *Android INC.* rozpoczęła swoją działalność na terenie Kalifornii w USA, gdzie rozpoczęła tworzenie swojego autorskiego rozwiązanie w oparciu o jądro Linux (zobacz [22]). W roku 2005 roku następuje moment zwrotny – bez którego najprawdopodobniej nie był by możliwy tak dynamiczny rozwój systemu jaki znamy, czyli wykupienie firmy *Android INC.* przez potentata rynku internetowego, firmę *Google* (zobacz [23]).

Od lipca 2005 roku do wypuszczenia wersji 1.0 systemu sygnowanego przez logo zielonego robota minęły prawie 3 lata. Wydanie nowego systemu było poprzedzone opublikowanie pierwszej wersji Android SDK[1] w 2007 roku. Debiutancka wersja w znacznym stopniu różniła się od rozwiązania znanego obecnie, nie obsługując chociażby wielodotyku, czy klawiatury ekranowej. Pomimo to stanowiła solidną bazę do rozwoju najpopularniejszego obecnie mobilnego systemu operacyjnego na święcie (zobacz [24]).

Rok później zostaje wydana wersja 1.5, która poza implementacją nowych funkcji (m. in. Klawiatura dotykowa) zapoczątkowała również zmienioną konwencje nazewnictwa i tak od tej wersji każdy kolejny system od *Google*, obok oznaczeń numerycznych nosił nazwę związaną z słodyczami.

W tym samym roku (2009) zostaję wydana wersja 1.6 Donut (ang. pączek), która wprowadziła obsługę rozdzielczości innych niż 480x320px, co umożliwiło instalacje Android na znacznie większej ilości smart fonów. Wartym odnotowania jest fakt, że w momencie premiery tej wersji system ten posiadał niewielki udział w rynku, szacowany na około 2%.

Pomiędzy wydawaniem kolejnych wersji systemu została wprowadzona nowa linia mobilnych urządzeń sygnowanych logiem *Google* zwana *Nexus*. Jej główną zaletą jest działanie w oparciu o czystą, niemodyfikowaną żadnymi nakładkami wersje Androida. Ten fakt plus dodatkowo konkurencyjna cena miały doprowadzić do dynamicznego wzrostu znaczenia oprogramowania na rynku globalnym. Kolejne

[1] SDK - Software development kit

wersje Androida wprowadzały dodatkowe funkcje, poprawiały wydajność i stabilność rozwiązania. Wybrane wersje systemu (wraz z wybranymi zmianami) zostały wypisane w Tabela 5:

Tabela 5 Wybrane wersje systemu Android wraz z przeglądem funkcji. Źródło: [25] [26]

Wersja systemu (wersja API)	Nazwa systemu	Premiera	Zmiany
1.0 (API level 1)	Apple Pie	23.09.2008 r.	Pierwsza wersja systemu, pozbawiona obsługi multidotyku, czy też klawiatury ekranowej
1.5 (API level 3)	Cupcake	30.04.2009 r.	Klawiatura ekranowa Automatyczny obrót ekranu Obsługa widżetów
1.6 (API level 4)	Donut	15.09.2009 r.	Obsługa rozdzielczości WVGA oraz QVGA Integracja aparatu, kamery oraz galerii
2.2 (API level 8)	Froyo	20.05.2010 r.	Automatyczne aktualizacje aplikacji pochodzących z oficjalnego sklepu Android Tethering przez WIFI Instalacja aplikacji na karcie pamięci
2.3.3 (API level 10)	Gingerbread	22.02.2011 r.	Uproszczony interfejs Poprawiona energooszczędność
3.0 (API level 11)	Honeycomb	24.01.2011 r.	Jest to specjalna wersja androida przeznaczona jedynie na tablety, w związku z tym główne zmiany to: Zoptymalizowanie aplikacji pod względem wyższych rozdzielczości Obsługa urządzeń USB Umieszczenie paska nawigacji bezpośrednio na ekranie
4.0 (API level 14)	Ice Cream Sandwich	19.10.2011 r.	Wsparcie dla rozdzielczości 720p Wsparcie dla procesorów 2 rdzeniowych Przesyłanie plików za pomocą technologi NFC Poprawiony interfejs

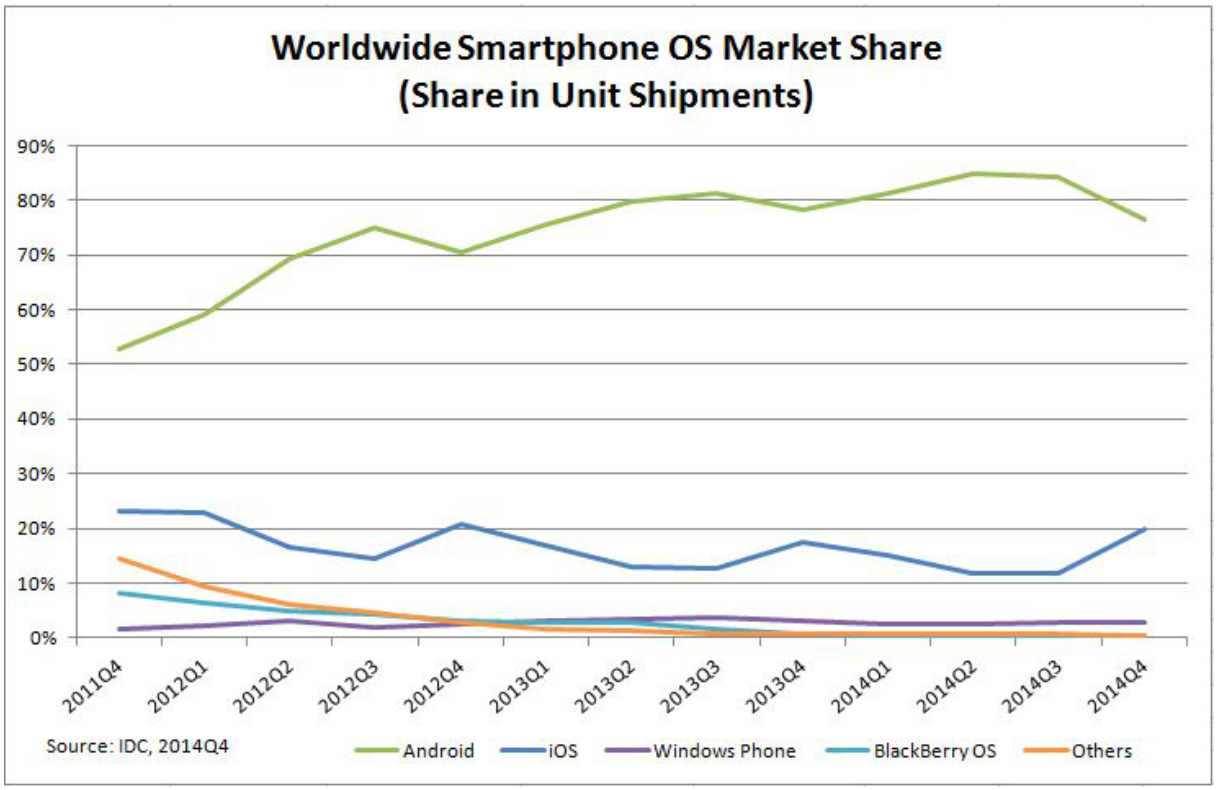
4.4 (API level 19)	KitKat	31.10.2013 r.	Dostęp do asystenta Google z ekranu startowego Zoptymalizowanie systemu względem mniej wydajnych urządzeń Poprawiono szereg błędów związanych z niespodziewanym restartowaniem aplikacji jak i całego systemu
5.0 (API level 21)	Lollipop	03.11.2014 r.	Przebudowany interfejs w całości oparty o nowy design (Material Design) Wsparcie dla procesorów 64-bitowych Project Volta mający za zadanie wydłużenia pracy na baterii Wsparcie Bluetooth 4.1

4.1.2 Wybór wersji

Wybierając system operacyjny na który to napisana zostanie nasza aplikacja kierowaliśmy się przede wszystkim chęcią tworzenia oprogramowania dla jak największej rzeszy odbiorców. W związku z tym postanowiliśmy przyjrzeć się danym statystycznym z kilku ostatnich lat i na ich podstawie dokonać wyboru.

Tabela 6 Systemy mobilne – udział w rynku lata 2010-2014. Źródło: [27]

Okres	Android	iOS	Windows Phone	BlackBerry OS	Pozostałe
4 kwartał 2014	76.6%	19.7%	2.8%	0.4%	0.5%
4 kwartał 2013	78.2%	17.5%	3.0%	0.6%	0.8%
4 kwartał 2012	70.4%	20.9%	2.6%	3.2%	2.9%
4 kwartał 2011	52.8%	23.0%	1.5%	8.1%	14.6%
1 kwartał 2010	23.3%	16.0%	4.9%	16.0%	40.2%



Wykres 3 Systemy mobilne – udział w rynku lata 2011-2014. Źródło: [27]

Powyższe zestawienia pokazuje jak zmieniał się udział poszczególny systemów mobilnych na przestrzeni ostatnich kilku lat. Wartym zauważenia jest dynamiczny wzrost znaczenia systemu Android , który w przeciągu 4 lat zyskał na popularności o ponad 50% osiągając udział w rynku na poziomie przekraczającym 75%.

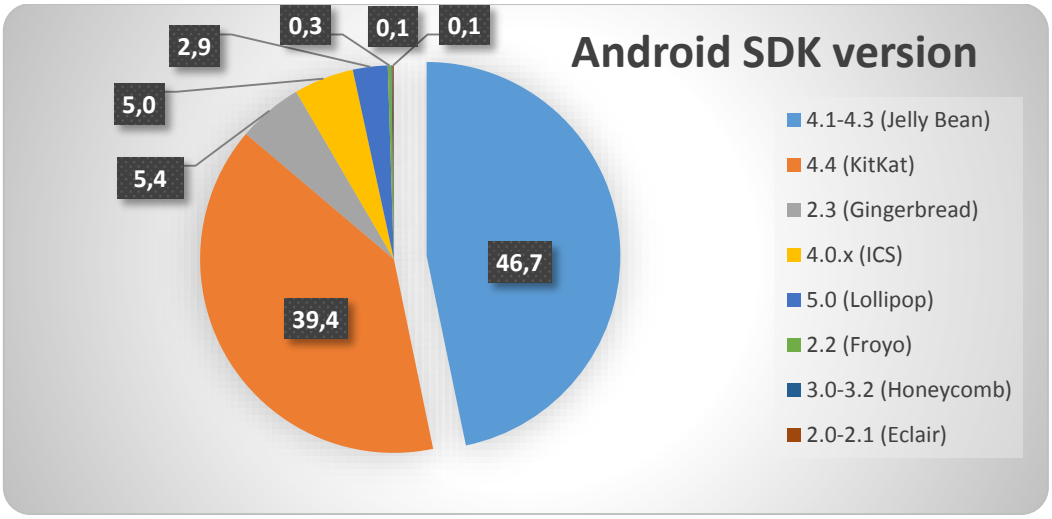
Kolejnym istotnym faktem wynikającym z powyższych danych statystycznych jest ogromny spadek udziału pozostałych systemów takich jak np. Symbian (wykorzystywany w min. urządzeniach Nokii) z poziomu 40% do poziomu marginalnego wynoszącego poniżej 1%. Pokazuje to, że nowoczesne smartphone`y wymagają bardziej rozbudowanego oprogramowania, które będzie potrafiło odpowiednio spożytkować ich potencjał. Dla nas jednak najważniejszą informacją jest fakt, że obecnie dominującym system jest Android i dlatego to właśnie on został wybrany jako baza dla naszego rozwiązania.

Po dokonaniu wyboru platformy docelowej niemal od razu ujawnił się jeden z największych problemów tego systemu – duża fragmentacja ze względu na jego wersje oraz odmienne implementacje systemu wprowadzane przez poszczególnych producentów urządzeń. Android jako system stosunkowo młody, działający na dynamicznie zmieniającym się rynku (ogromny postęp technologiczny, wpływającym na szybki wzrost mocy obliczeniowej obecnych urządzeń mobilnych) zmuszony był do wprowadzenia szybkiego cyklu wydawniczego. W wyniku tego na rynku działają skrajnie różne wersje systemu. Aktualnie najnowszą wersją Android jest wersja 5.x działające na API w wersji >21, jednocześnie wraz z tą wersją niemal 6% udział posiada wersja 2.3 współpracująca z API w wersji 10.

Jak widać jest to znacząca dysproporcja. My jednak ponownie postanowiliśmy oprzeć się na danych statystycznych, które to umieściliśmy w Tabela 7.

Tabela 7 Udział poszczególnych wersji systemu Android – 2.02.2015 r. Źródło: [28]

Wersja Android	Udział w rynku	Zmiana w ciągu ostatnich 30 dni
4.1-4.3 (Jelly Bean)	46.7%	-3%
4.4 (KitKat)	39.4%	4%
2.3 (Gingerbread)	5.4%	-8%
4.0.x (ICS)	5.0%	-8%
5.0 (Lollipop)	2.9%	47%
2.2 (Froyo)	0.3%	-11%
3.0-3.2 (Honeycomb)	0.1%	-9%
2.0-2.1 (Eclair)	0.1%	-12%



Wykres 4 Udział poszczególnych wersji systemu Android – 2.02.2015 r. Źródło: [28]

Podczas analizy powyższych danych, niemal od razu nasuwa się wniosek, że największą popularnością cieszą się wersje systemu z serii 4.x osiągając ponad 90% udział. Niemniej jednak trzeba pamiętać, że poszczególne wersje Android z rodziny 4.x różnią się dość znacznie wersją API począwszy od API poziomu 14 aż do API poziomu 19. Dlatego też po wyborze serii 4.x kolejnym krokiem było wybranie jej konkretnej instancji. Do wyboru zostały nam dwie najpopularniejsze wersje – Jelly Bean (4.1-4.3) z ponad 46% udziałem, oraz wersja KitKat z niemal 40% udziałem. Pierwszym naturalnym wyborem byłbym Android Jelly Bean z około 7% przewagą w udziale nad KitKatem, jednak ponownie znaczący okazał się wybór względem poziomu API. Google pod szyldem Jelly Bean wydał system od wersji 4.1 do 4.3 korzystające z trzech różnych wersji API 16-18, natomiast system w wersji 4.4 korzysta z jednego poziomu API 19. Mając na uwadze to wszystko, zdecydowaliśmy się ostatecznie na korzystanie z systemu Android w wersji 4.4 KitKat wraz z API poziomu 19.

4.1.3 Android SDK

Tworząc oprogramowanie działające pod systemem Android jednym z pierwszych kroków jakie musi wykonać programista jest zapoznanie się z zestawem narzędzi Android SDK. Zestaw ten złożony jest z dwóch części

- **SDK Tools –** jest elementem wymaganym niezależnym od wersji Androida i wykorzystywanym w każdej aplikacji na ten system.
- **Platform Tools -** jest zestawem narzędzi modyfikowanym pod konkretną wersję systemu Android

Do podstawowych zalet tego zestawu narzędzi należy jego modularność, dzięki której w prosty sposób możemy zarządzać jakie elementy są aktualnie zainstalowane na naszej maszynie. Prosty proces instalacji i deinstalacji zapewniony przez narzędzia Android SDK Manager, pozwala na bieżącą organizować liczbę posiadanych modułów. Do samych modułów zaliczane są takie elementy jak:

- Obrazy z wersjami systemu Android
- Wszelkie sterowniki
- Przykładowe projekty
- Dokumentacja
- Biblioteki
- Emulatory

Tak jak wspomnieliśmy wcześnie niezbędnym elementem każdej aplikacji jest zestaw narzędzi **SDK Tools.** W jego skład wchodzi wiele narzędzi umożliwiających takie akcje jak:

- AVD Manager – pozwala na zarządzanie wirtualnymi maszynami
- Debugowanie aplikacji, oraz emulacja samych urządzeń (emulator i DDMS)
- Narzędzia do symulacji pamięci wewnętrznej, kart SD urządzeń (mksdcard)

- Tworzenie i zarządzanie bazami danych oparte o SQLite

Drugim elementem składowym Android SDK jest Platform tools, a więc zestaw narzędzi aktualizowany przy okazji wydania nowej wersji systemu. Umożliwia on kompilację tworzonej aplikacji pod konkretną wersje systemu mobilnego. Jednocześnie każda kolejna wersja jest tak zaimplementowana, żeby być zgodną z poprzednimi. W jego skład wchodzi Android Debug Bridge, umożliwiające kontrole urządzeń bądź emulacji z system Android, poprzez między innymi uruchamianie/instalacje aplikacji, czy też przesyłanie plików. Kolejnym często wykorzystywanym narzędziem wchodzącym w skład Platform tools jest fastboot, umożliwiający wgrywanie na urządzenie dowolnego obrazu systemu, zarządzanie partycjami systemowymi, a także dostęp do bootloadera (zobacz [29]).

Do wygodniej pracy z Android SDK zalecane jest korzystanie z IDE, chociaż oczywiście nie jest to obowiązkowe. Firma Google od maja 2013 roku udostępnia IDE nazywane Android Studio, który jest alternatywą do popularnego Eclipsa wraz z rozszerzeniem ADT [30]. Więcej na temat wyboru IDE została przedstawiona w rozdziale czwartym.

4.2 Biblioteka jMonkeyEngine – Warstwy 3D

jMonkeyEngine (JME) jest to silnik grafiki 3D napisany w pełni w języku JAVA. Jest to projekt typu *open source* opublkowany na zasadach licencji *BSD license*.

Rysunek 5 Przykład wykorzystania jMonkey. Źródło: [31]

jMonkeyEngine posiada specjalne SDK pozwalające na wykorzystanie go na platformie mobilnej *Android.*

Do Naszej pracy została wybrana najnowsza wersja silnika oznaczona numerem 3.0. Głównymi argumentami za doborem tej technologii są:

- Pełne wsparcie dla platformy *Android*
- Szybkie działanie silnika – jest to bardzo ważny czynnik przy aplikacjach mobilnych ze względu na ograniczenie możliwości obliczeniowych
- Bogata dokumentacja oraz rozbudowane WIKI
- Duże wsparcie społeczności skupionej wokół *jME*

Historia rozwoju silnika *jMonkeyEngine* również miała wpływ na wybór. Zadecydowały dwa podstawowe fakty. Od początku silnik był rozwijany przez stały zespół, zapewnia to ciągłość myśli przewodniej oraz celu do jakiego oprogramowanie dąży. Po drugie wysoki poziom zaangażowania społeczności co zapewnia dobre testowanie w prawdziwych projektach oraz dobre wsparcie w przypadku natrafienia na problem.

Rysunek 6 Przykład użycia jMonkey. Źródło: [32]

Tabela 8 Historia jMonekEngine. Źródło: Opracowanie własne.

Data	Zdarzenie
jMonkeyEngine 0.1 – 2.0	
Rok 2003	Rozpoczęcie prac nad silnikiem przez Mark`a Powella

Styczeń 2004	Do Mark`a dołącza Joshua Slack. Wspólnie przez kolejne 2 lata tworzą główny zespół rozwojowy. Są wspierani przez społeczność skupioną wokół *jME*
15.09.2008	Joshua Slack ogłasza, że wycofuje się z projektu. Jego odejście spowodowało bardzo dużą stagnację i niemal całkowicie zahamowało dalszy rozwój silnika.
jMonkeyEngine 3.0	
01.04.2009	Kirill Vainer rozpoczyna projekt *jMonkeyEngine 3.0*
24.06.2009	Oficjalne ogłoszenie i utworzenie gałęzi wersji 3.0. W tym czasie wytwarza się wyraźnie zarysowany zespół (*core team)* który zajmuje się jego rozwojem. Sprawami organizacyjnymi zajmuje się Erlend Sogge Heggen.
17.05.2010	Wypuszczenie pierwszej wersji *alfa*
07.08.2010	Całkowita przebudowa strony internetowej projektu, zmiana nazwy oraz domeny, która jest wykorzystywana po dziś dzień.
22.10.2011	Wypuszczenie pierwszej wersji *beta*
15.01.2014	Wypuszczenie pierwszej stabilnej wersji, która jest nadal rozwijana.

4.3 Środowisko programistyczne

Przy okazji opisywania Android SDK wspomnieliśmy, że do wygodnego tworzenia oprogramowania zalecane jest wykorzystywania IDE (z ang. Integrated Development Environment), czyli zintegrowanego środowiska programistycznego. Firma Google w roku 2013 zaproponowała Android Studio jako narzędzie dedykowane dla programistów, tworzących aplikacje na system Android. Z miejsca stało się ono głównym rywalem dla popularnego Eclipse działającego z rozszerzeniem ADT. Na potrzeby tej pracy nie będziemy skupiać się na porównaniu tych dwóch rozwiązań, gdyż jest to temat dość rozległy, a każde z tych dwóch rozwiązań posiada szereg plusów jak i minusów. My zdecydowaliśmy się korzystać z środowiska programistycznego Eclipse, gdyż towarzyszy nam ono od początku studiów jak i przy okazji naszej codziennej pracy.

4.3.1 Eclipse

Platforma ta powstała 7 listopada 2004 roku na bazie uwolnionego przez Rational IBM kodu. Od tamtego czasu projekt jest rozwijany przez fundacje non-profit, która została założona przez firmę IMB w styczniu tego samego roku. Do jej głównych zadań należy [33] [34]:

- Zarządzanie kierunkiem rozwoju projektu Eclipse
- Koordynacja projektów powiązanych z Eclipsem
- Działania marketingowe
- Opracowywanie zasad licencjonowania oprogramowania
- Zapewnienie infrastruktury informatycznej

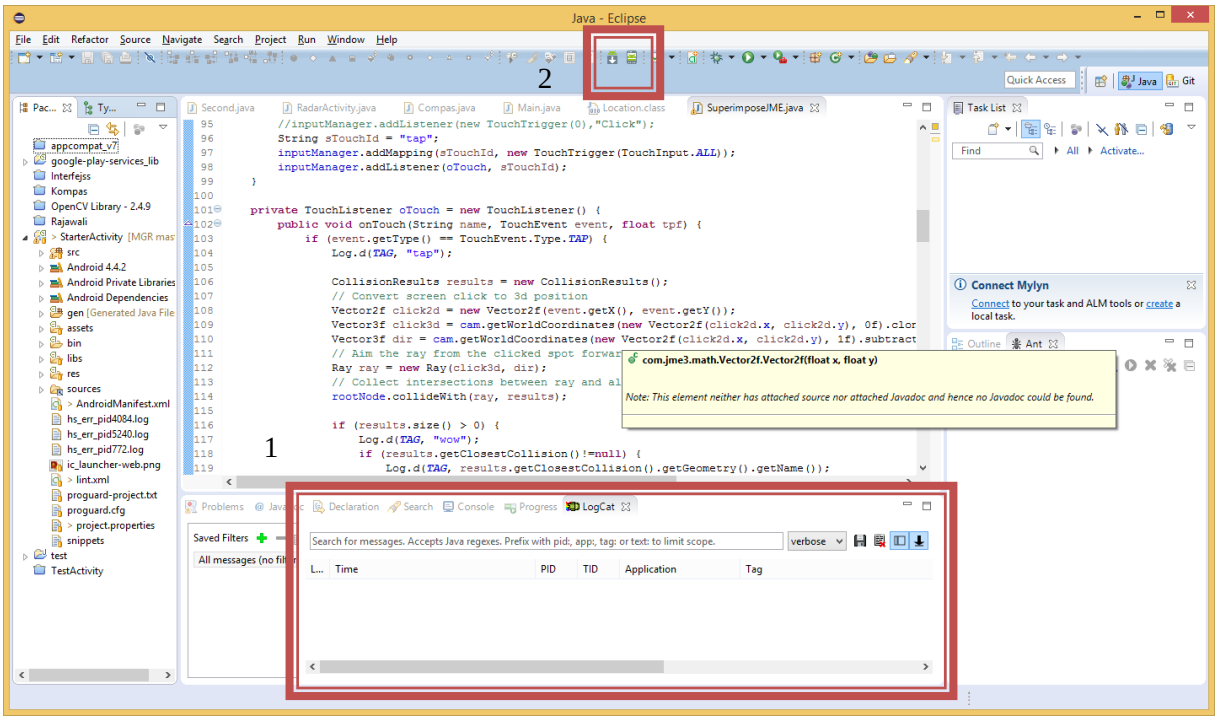
Taki sposób zarządzania pozwolił na stworzenie narzędzia, które szybko zdobyło uznanie wśród programistów. Jego najważniejszą cechą jest obsługa wtyczek (z ang. Plugins), która umożliwia dostosowywania środowiska do własnych potrzeb poprzez dowolne zarządzanie funkcjonalnościami. Zastosowanie odpowiednich wtyczek pozwala między innymi na:

- Tworzenie aplikacji min. w Javie, C/C++, PHP
- Modelowanie aplikacji w oparciu o język UML
- Zapewnie współprace serwerami aplikacji i baz danych
- Rozwiania Webserwisów
- Tworzenie aplikacji mobilnych na platformę Android poprzez rozszerzenie ADT

Pierwsza wydana wersji Eclipse była oznaczona numerem 3.0, natomiast aktualna wersja to 4.4.1 Luna wydana 26 września 2014 roku i to właśnie ta wersja zostanie wykorzystana na potrzeby tego projektu [34].

4.3.2 Rozszerzenie ADT

Zgodnie z tym co wspomnieliśmy w poprzednim podrozdziale główną zaletą środowiska programistycznego Eclipse jest zdolność do obsługi rozszerzeń. Zastosowanie jednego z nich – ADT (skrót od angielskich słów Android Developer Tools, co można tłumaczyć jako narzędzia dla programisty na platformę mobilną Android) – w znacznej mierze upraszcza i porządkuje proces tworzenia aplikacji. Po jego instalacji uzyskujemy dostęp do wielu przydatnych narzędzi, których najważniejsze elementy działania opiszem [35].



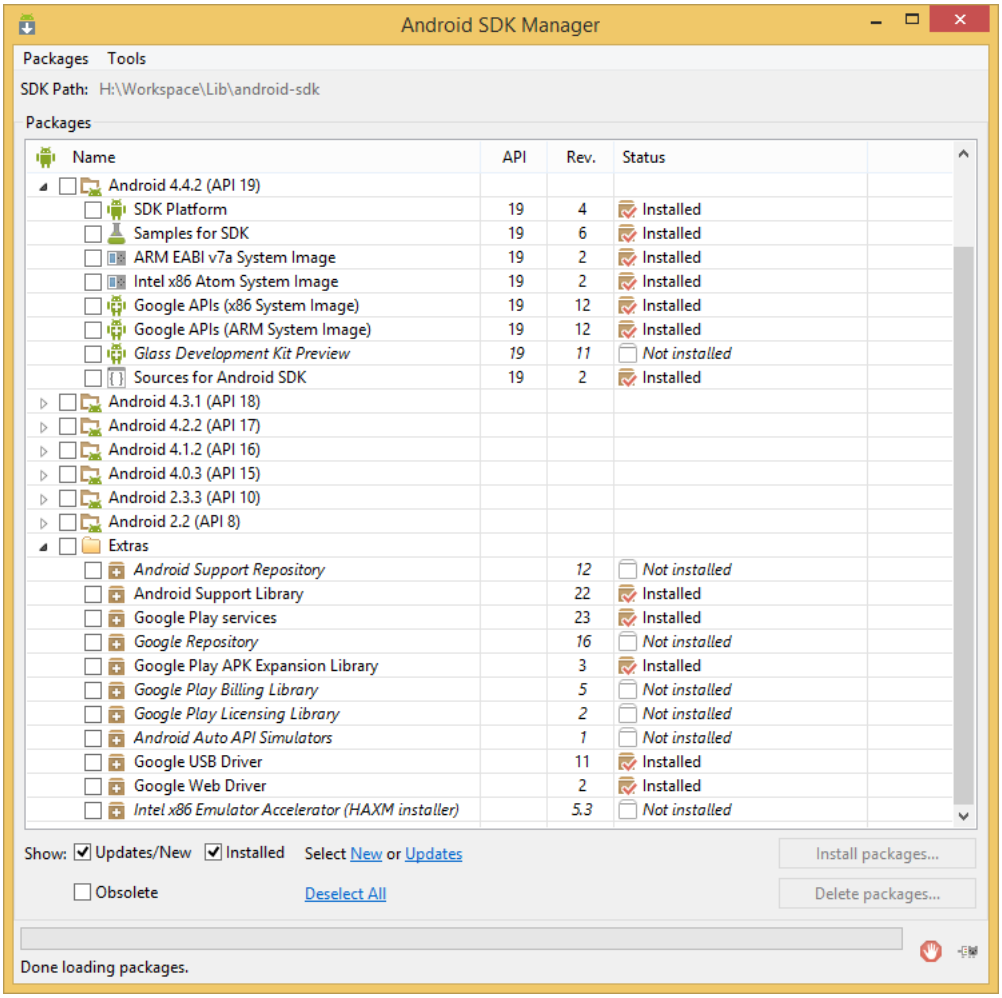
Rysunek 7 Eclipse Luna z rozszerzeniem ADT – główne okno programu

Na zrzucie ekranowym (Rysunek 7) zostało przedstawione główne okienko w trybie widoku projektowym, na którym to zaznaczyliśmy elementy wchodzące w skład ADT.

Numerem 1 została oznaczona konsola z komunikatami nazwana „LogCat". Jest to niezbędne narzędzie podczas analizy działania naszej aplikacji. Wyświetlane na niej są takie informacje jak:

- Sekwencje komunikatów z procesu umieszczania, instalowania i uruchamiania aplikacji na urządzeniu docelowym (zarówno fizycznym jak i wirtualnym)
- Wyświetlanie wiadomości o błędach
- Wyświetlanie informacji podanych przez programistę poprzez wykorzystaniu polecania *LOG*. Jest to najłatwiejszy sposób na śledzenie przebiegu wykonywania się aplikacji, podczas gdy urządzenie jest powiązane z komputerem
    - Przykładowa składnia Log.d(TAG,"Komunikat");

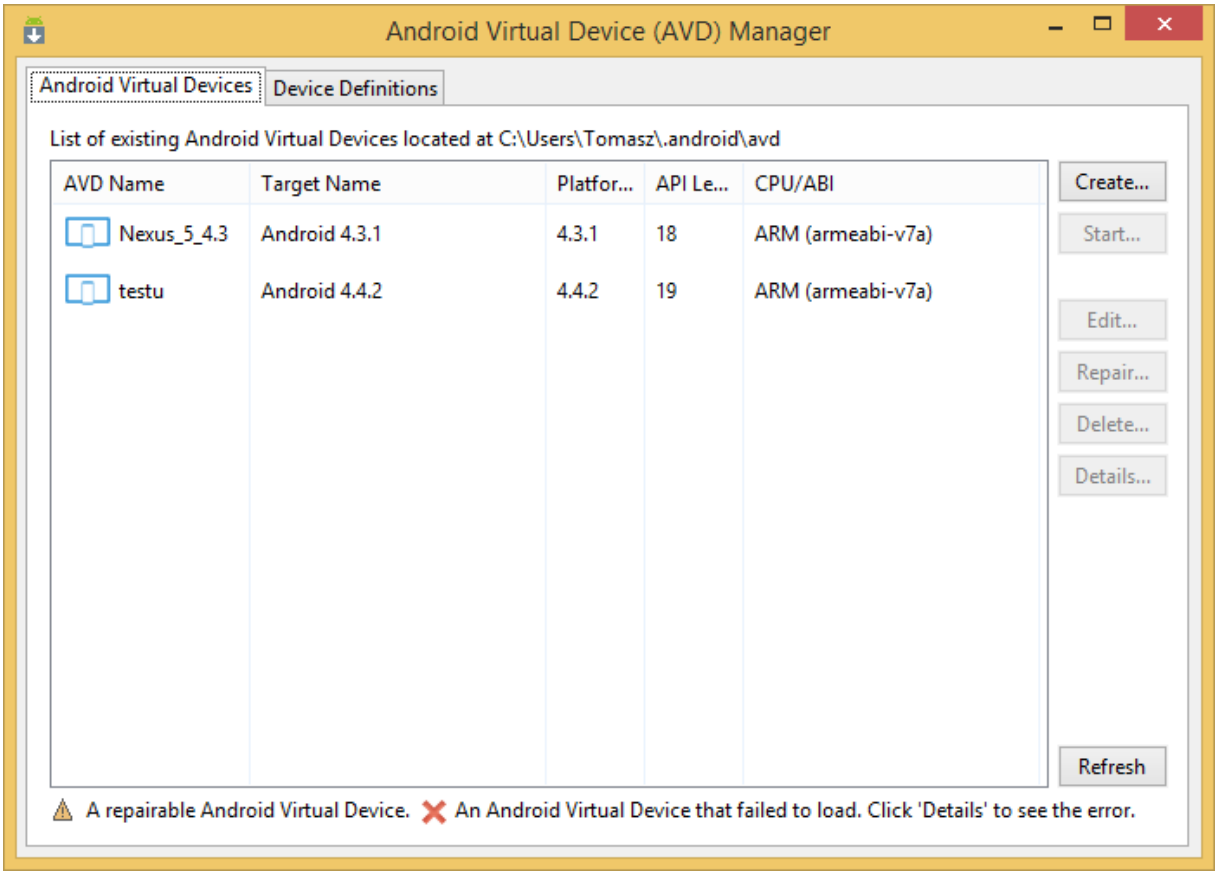
Numerem 2 zostały oznaczone dwa skróty. Pierwszy z nich jest Android SDK Manager, czyli narzędzie które umożliwia w łatwy sposób zarządzać składnikami opisywanego wcześniej Android SDK.



Rysunek 8 Eclipse Luna ADT – Android SDK Manager. Źródło: Opracowanie własne.

Jak widać na powyższym zrzucie ekranowym na potrzeby tej pracy zostało zainstalowane min. takie składniki jak:

- Android SDK z elementami dedykowanymi pod wersje 4.4.x KitKat, wraz z API poziomu 19
    - Wraz z obrazami systemu w wersji 4.4.x umożliwiającymi tworzenie urządzeń wirtualnych
- Google Play services – umożliwiające korzystanie między innymi z map Google Maps
- Niezbędne sterowniki do sterowania urządzeniami z systemem Android



Rysunek 9 Eclipse Luna ADT – Android Virtual Device. Źródło: Opracowanie własne.

Drugim elementem oznaczonym liczbą z numerem 2 jest AVD (Android Virtual Device), czyli narzędzie umożliwiające tworzenie wirtualnych urządzenie. Dzięki wykorzystaniu tego narzędzie jesteśmy w stanie testować zachowanie gotowej aplikacji na różnych urządzeniach o skrajnych parametrach technicznych, ale również różniących się wersją systemu. Takie rozwiązanie pozwala na zmniejszenie kosztów tworzenia oprogramowania, przy jednoczesnym zachowaniu procesu niezbędnych testów.

Jak wspomnieliśmy wcześniej najlepiej jest testować niemal gotową aplikacje przy wykorzystaniu tego narzędzia. Związane jest to z największą wadą wirtualizcji, czyli dość słabą

optymalizacją. Porotwanie oprogramowania i wirtualizacja urządzenia sprawia, że korzystanie z niego staję się bardzo czasochłonne [36].

4.3.3 Google Maps

Jednym z podstawowych założeń tworzonej w tej pracy gry jest nanoszenie punktów na mapę, oraz późniejsze nawigowanie do nich przy wykorzystaniu mechanizmu przypominającego w działaniu kompas. W uzyskaniu takiego efektu pomocnym okazało się skorzystanie z biblioteki „Google play services library", w skład której wchodzi interfejs umożliwiający wykorzystywanie mechanizmów usługi Google Maps działającej od lutego 2005 roku. Firma Google poprzez swój serwis umożliwia między innymi:

- Wyszukiwanie obiektów (budynków, atrakcji turystycznych itp.)
- Wyświetlanie map
- Wyświetlanie zdjęć lotniczych
- Dla wybranych lokalizacji dostęp usługi zwanej Street View, która to umożliwia oglądanie panorama ulic (ale również budynków i innych obiektów) w 360°
- Wyznaczanie tras przejazdu, wraz z opisem trasy krok po kroku

Jak wspomnieliśmy wcześniej firma Google udostępniła programistą Android specjalną bibliotekę zawierającą API dla Google Maps. Wykorzystanie jego możliwości możliwe jest po uzyskaniu darmowego klucza, a następnie umieszczenie go w manifeście aplikacji. Jest to nieskomplikowany proces, po zakończeniu którego uzyskujemy dostęp do większości funkcjonalności oryginalnego serwisu.

4.4 Język JAVA

Pisząc aplikacje na platformę Android trudno nie wspomnieć o języku programowania JAVA, czyli o języku w którym napisana została większość aplikacji. W tym podrozdziale postaramy się pokrótce opisać historię powstania tego jednego z najpopularniejszych języków programowania. Również opiszemy najważniejsze naszym zdaniem cechy charakteryzujące JAVE, oraz przedstawimy nieco danych statystycznych, udowadniających nasz powyższe zdanie o popularności tego języka.

4.4.1 JAVA – historia

Początki języka JAVA datowane są na rok '91 XX wieku, kiedy to w firmie Sun powstał pomysł stworzenie prostego multiplatformowego języka programowania. Za rozwój projekt odpowiedzialny był zespół pod kierownictwem Jamesa Goslinga. W wyniku ich pracy powstał nowatorski język działający w oparciu o wirtualną maszynę, co pozwoliło na uruchamianie kodu niezależnie od konfiguracji fizycznej maszyny na której interpreter JAVY był odpalony.

Nie wszyscy wiedzą, że początkowo język miał się nazywać Oak (z ang. Dąb), a jego bezpośrednią inspiracją było ogromne drzewo zdobiące siedzibę firmy Sun. Podczas pracy nad semantyką języka,

jeden z członków zespołu odkrył, że istniał już język noszący taką nazwę. W wyniku tego zespół był zmuszony do zmiany nazwy, a obecna nazwa została wybrana niejako przez przypadek podczas spaceru po kawę [37].

Pierwszym projektem w którym wykorzystano JAVE był projekt nowatorskiego interfejsu do zarządzania sprzętem domowym (takim jak telewizory, oświetlenie, telefony). Opierał się on o pełnoekranowym interfejsie, którego obsługa odbywała się za pomocą dotykowych ekranów. Jednak za oficjalną datę powstania języka uznawany jest rok 1995, kiedy to na konferencji w San Francisco ogłoszono wydanie nowej wersji przeglądarki (Netscape Navigator zobacz: [38]) wspierającej obsługę apletów napisanych w języku JAVA. Ruch ten pomógł umocnić się nowemu językowi [39] [40].

W roku 2009 firma Oracle wykupiła akcje Sun za kwotę opiewająco na 7,4 miliarda dolarów, dzięki czemu stała się właścicielem języka JAVA. Pomimo początkowych obaw wielu środowisk powiązanych z tą technologią, firma Oracle w dalszym ciągu rozwija język wydając w roku 2014 wersję 1.8 [41] [42].

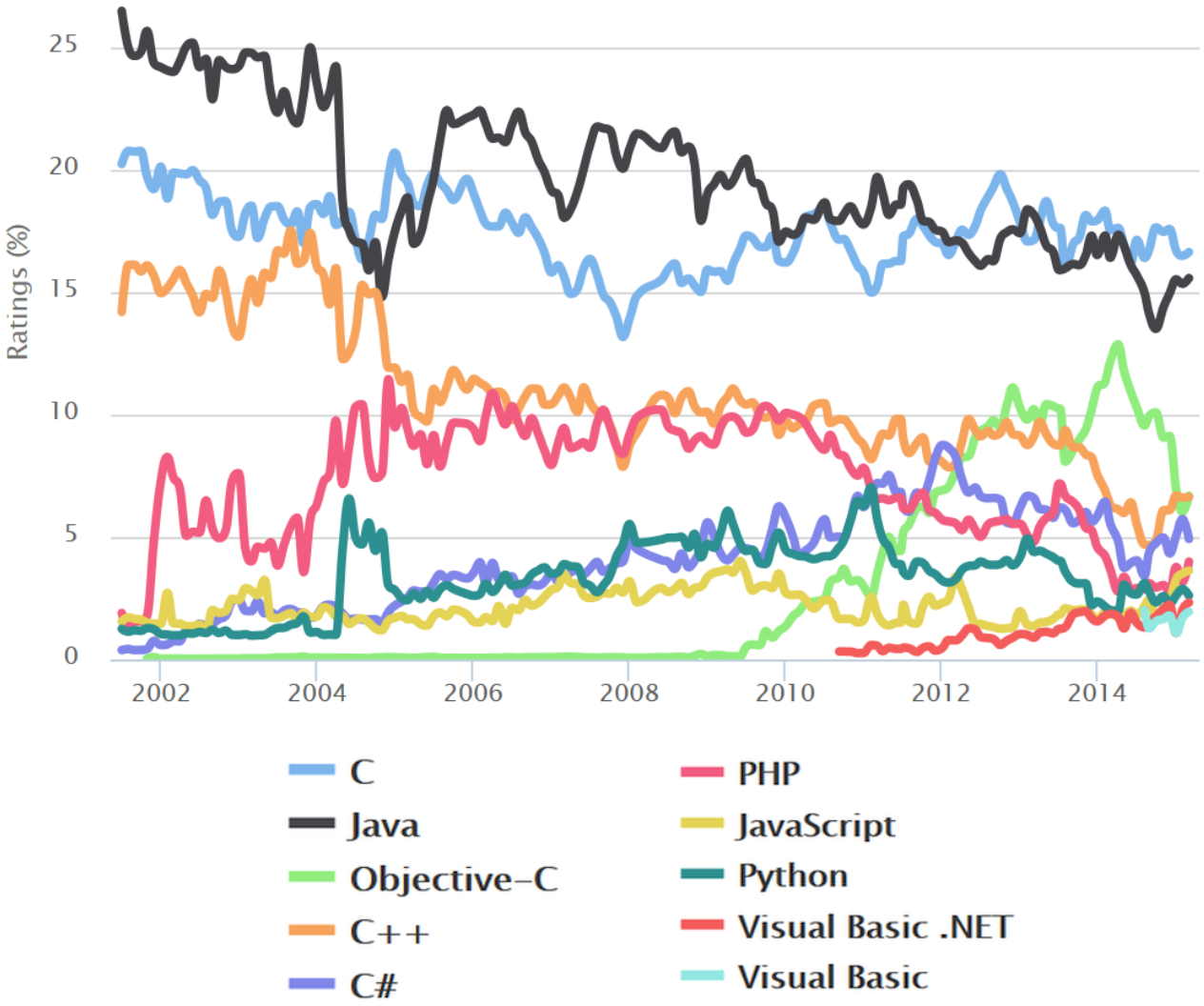
4.4.2 Główne założenia języka JAVA

Java opiera się na kilku kluczowych założenia. Najważniejsze z nich to:

1. **Obiektowość –** od samego początku język ten był silnie ukierunkowany na obiektowość, dzięki czemu dane jak i akcje zostały pogrupowane w klasy obiektów. Jedynymi odstępstwami od tej koncepcji są typy proste, do których zaliczamy między innym int, czy też float. Jednakże nawet typy proste posiadają swoje odpowiedniki obiektowe rozszerzające ich możliwości.
2. **Dziedziczenie –** język JAVA posiada nadrzędną klasę **Object,** po której to dziedziczą wszystkie pozostałe klasy. Zastosowanie tego mechanizmu, wraz z utworzeniem klasy bazowej umożliwiło zapewnienie zestawu wspólnych podstawowych możliwości dla wszystkich obiektowych bytów języka. Zaliczamy do nich:
    a. Identyfikacje
    b. Kopiowanie
    c. Porównywanie
    d. Kasowanie
    e. Wsparcie dla współbieżnego wykonania
3. **Multiplatformowość –** jak wspomnieliśmy w poprzednim podrozdziale, jednym z głównych założeń przyświecających tworzeniu nowego języka, była chęć tworzenia oprogramowania niezależnego od maszyny fizycznej. W języku JAVA Multiplatformowość została zapewniona poprzez kompilacje kodów źródłowych do kodu pośredniego, który następnie jest wykonywany poprzez maszynę wirtualną. Główną zaletą tego rozwiązania jest możliwość przenoszenia oprogramowania pomiędzy różnymi platformami sprzętowymi. Wadą zaś jest fakt, że powstały

kod pośredni musi być interpretowany, w wyniku czego sam program jest wolniejszy od programów skompilowanych do kodu maszynowego [43].

4.4.3 Java na tle innych języków programowania



Wykres 5 Ranking popularności języków programowania. Źródło: [44]

Jak w wynika z powyższego wykresu JAVA od ponad dekady jest jednym z najczęściej wykorzystywanych języków programowania na świecie. Obecnie jego wykorzystywanie szacowane jest na niewiele ponad 16% i zgodnie z raportem firmy TIOBE, oznacza to że jest to obecnie najpopularniejszy język programowania. Jest to co prawda wynik znacząco gorszy niż wyniki z czasów jej największej popularności (powyżej 25% w roku 2001), lecz w dobie mnogości różnych języków programowanie jest to ciągle wynik bardzo dobry.

5 Analiza i wnioski

W tym rozdziale skupiamy się wokół wymagań jakie zostały wypracowane w ramach analizowania potrzeb związanych z tworzonym przez Nas prototypem. Opiszemy tu wymagania funkcjonalne, niefunkcjonalne, a także podstawowe schematy oraz diagramy czyli wszystko co niezbędne do prawidłowego wytworzenia oprogramowania. W wymaganiach postanowiliśmy wyraźnie rozdzielić kwestie związane z biblioteką oraz grą.

5.1 Analiza wymagań

Zgodnie z powszechnie przyjętymi procesami inżynierii oprogramowania, jednym z pierwszych etapów wytwarzania nowego projektu jest zebranie wymagań. W tych podrozdziałach znajdują się zarówno wymagania funkcjonalne jak i niefunkcjonalne z uwzględnieniem podziału na bibliotekę, oraz prototyp gry.

5.1.1 Wymagania funkcjonalna – prototyp

W Tabela 9 zawarte są wymagania funkcjonalne jakie zostały postawione przed budowaną w ramach pracy biblioteką. Wymagania te są minimum, które pozwolą wraz z dalszym rozwojem, a zbudowanie kompleksowego i wydajnego podczas pracy zbioru narzędzi do budowania mobilnych aplikacji opartych o rzeczywistość rozszerzoną.

Tabela 9 Wymagania funkcjonalne - biblioteka. Źródło: Opracowanie własne.

Lp.	Nazwa wymagania	Opis wymagania	Ograniczenia
1.0	Obsługa sensorów	Biblioteka musi pozwalać na stosunkowo prostą i szybką obsługę sensorów. Klasyczne podejście wynikające z SDK jest skomplikowane i wymaga wiele uwagi od programisty co może przyczynić się do nieumyślnych błędów. Biblioteka musi zapewnić wygodny interfejs dostępowy.	Brak
1.1	Poprawa jakości odczytu	Ze względu na znane problemy z odczytami, metody dostępowe biblioteki powinny być wyposażone w mechanizmy, które zapewnią podniesienie jakości odczytów	Brak

2.0	Podstawowa obsługa 3D	Biblioteka musi umożliwiać wygenerowanie obiektu trójwymiarowego na ekranie	Brak
2.1	Manipulacje obiektem 3D	Nieodłącznym elementem aplikacji rzeczywistości rozszerzonej ze wsparciem 3D jest bazowa manipulacja obiektami.	Manipulacja w ramach prototypu musi zapewnić przynajmniej ruch na osi X
2.2	Animacja 3D	Biblioteka powinna wspierać wbudowane animacje 3D importowanych obiektów	Musi być taka możliwość nie tylko na etapie inicjalizacji, ale w dowolnym momencie działania aplikacji
3.0	Kamera	Ze względu na niemożliwy do rozdzielenia sprzęg na linii oprogramowanie – kamera, biblioteka musi być wyposażona w elementy pozwalające na bazową inicjalizację kamery pod kątem obsługi obiektów 3D	Brak

5.1.2 Wymagania funkcjonalne – gra

Tabela 10 zawierająca wymagania funkcjonalne jakie zostały postawione przed budowaną w ramach pracy grą. Wymagania te zostały dobra według dwóch – najważniejszych Naszym zdaniem – kryteriów.

1. Możliwość wykorzystania elementów opracowanej biblioteki
2. Prezentacja jak największej ilości elementów, które mogą zostać wykorzystane do tworzenia rzeczywistości rozszerzonej

Tabela 10 Wymagania funkcjonalna - gra. Źródło: Opracowanie własne.

Lp.	Nazwa wymagania	Opis wymagania	Ograniczenia
1.0	Wybór obszaru poszukiwania	Użytkownik wybiera opcję nowej gry. Za pomocą suwaka wybiera obszar w jakim chce, aby system wylosował obiekty do odnalezienia.	Obszar podawany jest w kilometrach i nie może być mniejszy niż 1 kilometr.

2.0	Wybór ilości punktów	Użytkownik wybiera opcję nowej gry. Za pomocą suwaka wybiera ilość elementów jaka ma zostać wylosowana na mapie.	Ilość elementów nie może być mniejsza niż 1 (jeden)
3.0	Wyświetlanie mapy	Użytkownik kładąc telefon na płasko przechodzi do mapy.	Mapa wykorzystuje silnik Google Maps.
3.1	Zaznaczenie obszaru na mapie	System automatycznie na mapie zaznacza obszar w jakim generowane są punkty. W obrębie obszaru wyświetlane są markery. Każdy marker odpowiada pojedynczemu punktowi do odszukania.	Brak
3.2	Oznaczenie odszukanych punktów	System oznacza na mapie punkty, które zostały już odnalezione.	Brak
3.3	Wybranie punktu	Użytkownik naciska na marker w celu wyświetlenia jego nazwy. Następnie naciska na nazwę w celu uruchomienia trybu wyszukiwania.	Użytkownik może wybrać tylko jeden punkt na raz.
4.0	Wybór kierunku	Po wybraniu punktu na obrazie z kamery nałożona jest igła kompasu wskazująca kierunek w jakim znajduje się wybrany punkt. Kompas ten zawsze wskazuje wybrany punkt.	Aplikacja nie nawiguje bezpośrednio do punktu, a jedynie wskazuje kierunek do punktu
5.0	Generowanie modelu 3D	W momencie, gdy użytkownik znajduje się w pobliżu szukanego obiektu system generuje model 3D, który jest wizualizacją odnalezionego obiektu.	Brak
5.1	Wykrycie ruchu użytkownika względem obiektu	W trakcie, gdy użytkownik zmienia swoje położenie system w czasie rzeczywistym modyfikuje kąt generowania obiektu. Użytkownik ma wrażenie, że porusza się wokół rzeczywistego obiektu	Brak

5.2	Interakcja z obiektem	Po wykryciu obiektu, użytkownik ma możliwość naciśnięcia na obiekt. Akcja ta spowoduje wyświetlenie zebranie obiektu.	Brak
6.0	Wyświetlenie zebranych elementów	Użytkownik z menu kontekstowego może wyświetlić listę elementów, które zostały zebrane, oraz jakie zostały jeszcze do zebrania.	Brak
6.1	Przejście do elementu na mapie	Naciśnięcie na jeden z elementów z listy przenosi użytkownika do mapy, która jest wycentrowana na danym elemencie.	Brak
7.0	Zakończenie rozgrywki	System cały czas kontroluje czy użytkownik zebrał wszystkie elementy. W momencie, gdy zbierze ostatni wyświetla komunikat z podsumowaniem rozgrywki. Podsumowanie zawiera informację o: • Dacie rozpoczęcia • Dacie zakończenia • Łącznym czasie	

5.1.3 Wymagania niefunkcjonalne – biblioteka

Dzięki doświadczeniu zawodowemu Naszego zespołu byliśmy wstanie wypracować rozsądne i przemyślane wymagania niefunkcjonalne. Podczas ich opracowywania kierowaliśmy się tym, czego w rzeczywistości oczekiwalibyśmy od biblioteki w momencie wykorzystywania jej podczas produkcji oprogramowania

Lp.	Cecha	Priorytet	Miara
1	Konfiguracja	Musi	Biblioteka musi pozwalać na konfigurację podstawowych zagadnień (np. pliki związane z obiektami 3D)
2	Konfiguracja - prostota	Powinna	Konfiguracja musi być jak najprostsza
3	Wykorzystanie	Musi	Biblioteka musi być możliwa do wykorzystania na platformie Android

4	Możliwość wykorzystania w innych projektach	Musi	Biblioteka musi być możliwa do wykorzystania w ramach innych projektów

5.1.4 Wymagania niefunkcjonalne - gra

Wymagania niefunkcjonalne stawiane przed wszystkimi projektami związanymi z aplikacjami mobilnymi są bardzo restrykcyjne szczególnie w kwestii zagadnień takich jak:

- Wygoda obsługi
- Responsywność
- Intuicyjność
- Design

Zestawienie tych elementów/wymagań zostały zaprezentowane w Tabela 11.

Tabela 11 Wymagania niefunkcjonalne - responsywność. Źródło: Opracowanie własne.

Lp.	Cecha	Priorytet	Miara
1	Responsywność	Musi	Zmiana pomiędzy ekranami aplikacji nie może trwać dłużej niż 1 sekunda. Nie dotyczy rozruchu aplikacji podczas bazowego ładowania modelu 3D

Tabela 12 Wymagania niefunkcjonalne - niezawodność. Źródło: Opracowanie własne.

Lp.	Cecha	Priorytet	Miara
1	Niezawodność działania	Musi	Aplikacja nie może posiadać nieobsłużonych wyjątków. Każde zachowanie potencjalnie generujące błąd/zwracające wyjątek musi być prawidłowo obsłużone

Tabela 13 Wymagania niefunkcjonalne – intuicyjność/design. Źródło: Opracowanie własne.

Lp.	Cecha	Priorytet	Miara
1	Prosta obsługa	Musi	Interfejs aplikacji musi być maksymalnie uproszczony. Poszczególne pozycje powinny być umieszczone w miejscach zgodnych z zasadami projektowania aplikacji na systemie *Android*
2	Czytelny design	Musi	Design aplikacji musi być czytelny tzn.

			• Kolorystyka dobrana tak, aby czytanie nie sprawiała problemów • Ikony, guziki powinny być na tyle duże, aby ich używanie nie sprawiało problemów • Muszą być stosowane ogólnie przyjęte ikonografiki, aby użytkownik wiedział do czego służą
3	Prosty dostęp	Powinien	Opcje aplikacji powinny być dostępne na pierwszym poziomie zagłębienia. Nawigacje i ustawienia nie powinny posiadać wielu zagnieżdżeń
4	Łatwość użytkowania	Powinien	Przeciętny użytkownik po rozegraniu jednej gry powinien wiedzieć gdzie znajdują się najważniejsze funkcje gry

Tabela 14 Wymagania niefunkcjonalne – obsługiwane systemy. Źródło: Opracowanie własne.

Lp.	Cecha	Priorytet	Miara
1	Obsługiwane systemy	Musi	Aplikacja musi działać pod Androidem w wersjach: • 4.3 • 4.4
2	Obsługiwane systemy	Powinien	Aplikacja powinna działać na jak największej ilości wersji systemu Android

Tabela 15 Wymagania niefunkcjonalne – projekt i rozwój. Źródło: Opracowanie własne.

Lp.	Cecha	Priorytet	Miara
1	Możliwość rozbudowy	Musi	System powinien być zaprojektowany w taki sposób, aby przyszły rozwój mój możliwie jak najprostszy
2	Projekt	Musi	Aplikacja musi być zaprojektowana zgodnie ze standardami przyjętymi przy projektowaniu aplikacji na systemie Android

Tabela 16 Wymagania niefunkcjonalne - standardy programowania. Źródło: Opracowanie własne.

Lp.	Cecha	Priorytet	Miara
1	Platforma	Musi	Systemu musi być zbudowany w oparciu o Android SDK
2	Język	Musi	System musi być zbudowany w oparciu o język JAVA. Niedopuszczalne jest wykorzystanie jakiegokolwiek multiplatformowego framework`a do budowania aplikacji w oparciu o inny język programowania.

5.2 Największe trudności

Wybierając temat od początku zdawaliśmy sobie sprawię z niektórych problemów na jakie natrafimy. Mamy tu na myśli przeszkody technologiczne związane zarówno z samą aplikacją, jak i biblioteką, która była budowana w ramach niniejszej pracy. Przykładami są:

- Niedokładność sensorów
- Duże zużycie baterii
- Trudniejsze – niż w przypadku standardowych aplikacji mobilnych – testowanie
- Prostota wykorzystania biblioteki

W tym rozdziale skupiamy się na opisaniu tych trudności, co ze sobą niosły, wnioski jakie wyciągnęliśmy podczas pracy.

Ważne jest, że opisywane problemy dotyczą dwóch elementów: technologii, które zostały wykorzystane oraz tych, które dotyczą projektu opartego o konkretne założenia. Dodatkowo – tam gdzie to możliwe – postaramy się przybliżyć trudności z perspektywy aplikacji i biblioteki.

5.2.1 Pomiary GPS

Prototyp jest silnie uzależniony od odbiornika GPS i jego pomiarów. Wykorzystywany jest zarówno w celu osadzenia szukanego obiektu w świecie rzeczywistym, jak również w ustaleniu położenia użytkownika względem obiektu. Z tych względów prawidłowy odczyt koordynatów był rzeczą kluczową.

Niestety odbiorniki GPS w urządzeniach mają bardzo zmienną charakterystykę co w sposób znaczący utrudnia odpowiednie dostosowanie i konfiguracja.

W trakcie naszych testów zaobserwowaliśmy bardzo odmienne „zachowanie” odbiornika podczas ciągłego ruchu oraz w momencie, gdy użytkownik stoi nieruchomo. W pierwszym przypadku niedokładność pomiaru w wielu przypadkach była znośna. W zależności od urządzenia wahała się od 2

do 6 metrów, zaś skoki koordynatów[2] były stosunkowo rzadkie. Pozwoliło to na stosunkowo skuteczne ustawienie wewnętrznego kompasu gry na pozycję obiektu.

Zupełnie odmienne zachowanie miało miejsce w momencie, gdy użytkownik zatrzymywał się na dłuższą chwilę w miejscu. Błąd odczytu GPS potrafił sięgać nawet 50 metrów. Automatycznie powodowało to, że jakikolwiek losowy skok koordynatów potrafił diametralnie zniekształcić zachowanie gry. W dalszej części pracy przedstawione będą dane pokazujące problemy związane z odczytem.

W przypadku GPS i jego wykorzystania w Naszym prototypie istotne jest, że nie chodzi o samą niedokładność odczytu GPS, a jego zmienną i skokową naturę. W przypadku gdybyśmy mieli do czynienia wyłącznie z niedokładnością odczytu, ale taką, która byłaby stosunkowo stała można byłoby zastosować odpowiednie metody, które pozwoliłyby na jej uwzględnienie. Największym problemem w tym przypadku stanowi właśnie losowość i zmienne zachowanie uzależnione od tego co aktualnie użytkownik robi. Jest to czynnik niemalże losowy, którego stabilizacja okazała się bardzo trudna.

Dodatkowo sposób w jaki zmienia się sposób działania odbiornika w zależności od zachowania w przypadku Naszego prototypu jest wysoce niezadowalający. Jak opisane na początku rozdziału, odbiornik GPS w urządzeniach mobilnych traci na dokładności w momencie, gdy użytkownik znajduje się w jednym miejscu lub jego przemieszczenie jest małe. W przypadku Naszego prototypu właśnie w tym momencie – gdy użytkownik stoi – potrzebna jest największa dokładność, bo z reguły oznacza to, że obiekt jest blisko i chcemy na niego spojrzeć przez obiektyw Naszej kamery. W tym momencie każdy skok czy szum będzie miał negatywny wpływ na działanie aplikacji i jej odbiór przez użytkownika.

Powyższe obserwacje i testy doprowadziły Nas do wniosków, których nie mogliśmy w żaden sposób zignorować podczas budowania Naszej biblioteki i klas związanych z lokalizacją. Uznaliśmy, że Nasze metody dostępowe muszą zadbać o to, aby przyszli programiści nie musieli martwić się kwestiami związanymi z dokładnością i zmiennością wyników, a przynajmniej, aby zwracane wyniki były na tyle dokładne by mogły zostać wykorzystane do większości realizacji. Automatycznie zrodziło to kolejną komplikację. W przeciwieństwie do sensorów[3] odczyt pozycji nie zawsze musi być częsty i bardzo dokładny, czasami ważniejsza jest przybliżona pozycja i mniejsza ilość operacji, dlatego trzeba pamiętać o tym, aby nie zmuszać programisty do stosowania metod wygładzających i udostępnić maksymalnie generyczne metody odczytu pozycji z opcjami wyboru. Był to element, który musieliśmy silnie wziąć pod uwagę na etapie projektowania.

---

[2] Przez skoki koordynatów rozumiemy losowe zmiany współrzędnych urządzenia. Skoki te z reguły odbywają się w zakresie błędu GPS. Losowy jest moment skoku, kierunek, odległość oraz częstotliwość.

[3] W przypadku sensorów szum i zakłócenia zawsze stanowią problem, w przypadku analizowania pozycji użytkownika nie zawsze niezbędny jest idealny wynik, są przypadki w których rzadki i nie do końca dokładny odczyt są jak najbardziej do przyjęcia

5.2.2 Szum sensorów

Sensory były ważnym elementem obu prototypów. Od strony aplikacji, jednym z kluczowych narzędzi był kompas. Z kolei jeśli chodzi o bibliotekę, ważne była ich obsługa.

Aplikacja posiadała dwa tryby kompasu:

- Standardowy – określający kierunki geograficzne
- Nawigacyjny – określający kierunek do obiektu. Tak jak standardowy kompas zawsze wskazuje północ, tak nawigacyjny zawsze wskazywał poszukiwany obiekt.

Głównym problem w kontekście tworzenia kompasu wykorzystującego sensory urządzenia mobilnego, są mechanizmy składowe jego działania. Opierają się one na pomiarze otaczającego pola magnetycznego w trzech osiach (x,y,z). Pomiar podawany jest w jednostce μT [45] i uwzględnia położenie telefonu z akcelerometru. Niestety powoduje to, że odczyty z kompasu mogą być w bardzo prosty sposób zaburzone przez znajdujące się w pobliżu metalowe obiekty oraz inne urządzenia emitujące własne pole magnetyczne. W przypadku aplikacji, która jest wykorzystywana w terenie – głównie miejskim i zabudowanym - są to istotne problemy.

Kompas w obrębie prototypu wykorzystywany jest do trzech podstawowych zadań:

1) Wyznaczenie kierunku oraz określenie azymutu na który patrzy się w danym momencie użytkownik
2) Wyznaczenie kierunku na jakim znajduje się poszukiwany obiekt i wskazanie kierunku
3) Sterowanie obiektem 3D – w zależności od kierunku patrzenia w połączeniu z obiektem 3D jesteśmy w stanie dokonać rotacji kamery w celu oglądania obiektu w zakresie 360 stopni w osi X

Wszystkie problemy z odczytami oraz szumem nastręczyły wiele komplikacji podczas implementacji i mają wpływ na finalny odbiór aplikacji jako całości. Ponieważ każdy odczyt w sposób bezpośredni jest przekładany na element interfejsu (kompas wewnętrzny, obiekt 3D) to każdy skok w sposób negatywny wpływa na całą rozgrywkę. W dalszej części pracy przedstawione będą dane pokazujące problemy związane z odczytem.

W ramach Naszej biblioteki musieliśmy zadbać o to, aby informacje zwracane przez odpowiednie metody były jak najdokładniejsze. Dużym wyzwaniem było wypracowanie odpowiednich metod pozwalających na poprawę jakości wyników. Musieliśmy tak zaprojektować poszczególne klasy i metody, aby wymiana metod „wygładzania” nie wpływała na działanie samej biblioteki. Założyliśmy, że wraz z rozwojem platformy android, urządzeń itp. Mogą powstać lepsze metody dlatego ich wymiana powinna być jak najmniej problematyczna.

5.2.3 Zużycie baterii

Czynniki takie jak:

- Ciągłe korzystanie z GPS

- Ciągłe korzystanie z sensorów
- Renderowanie obiektów 3D
- Kalkulacja pozycji

Mają bardzo negatywny wpływ na zużycie baterii. W przypadku urządzeń mobilnych oraz aplikacji, która ma zapewniać rozrywkę w terenie bardzo ważne jest, aby bateria wytrzymywała jak najdłuższy czas i miała jak najmniejszy wpływ na normalne użytkowanie urządzenia.

W trakcie Naszych testów zauważyliśmy, że tempo zużycia baterii jest drastycznie duże. Urządzenie od pełnego naładowania potrafiło rozładować się w ciągu 2-3 godzin. Zużycie na takim poziomie znacząco może w dużym stopniu utrudnić rozgrywkę w przypadku, gdy użytkownik nie ma dostępu do miejsca w którym w dowolnym momencie będzie mógł doładować urządzenie.

Kolejnym problemem związanym z baterią jest fakt, że praktycznie każde działanie mające na celu zmniejszenie zużycia energii, negatywnie wpływa na pozostałe aspekty rozgrywki. Dla przykładu jeśli zmniejszymy częstotliwość odczytów GPS w celu jego mniejszego wpływu na baterię to skuteczność kalkulacji pozycji może znacząco zmaleć.

W przypadku budowania Naszej biblioteki aspekt zużycia baterii był bardzo istotny na etapie projektowania wielu metod. Złożoność obliczeniowa ma bezpośrednie przełożenie na obciążenie baterii dlatego podczas dobierania rozwiązań oraz ich implementacji musieliśmy kierować się jak najmniejszą złożonością obliczeniową. Nie zawsze było to możliwe, dla przykładu niektóre z Naszych metod dostępowych pracują na narzędziach związanych z biblioteką jMonkey Engine. W takich przypadkach cała złożoność jest po stronie zewnętrznej biblioteki. Nie mniej jednak tam gdzie było to możliwe staraliśmy się być jak najbardziej efektywni.

5.2.4 Odczyty sensorów, a obiekt 3D

Prawidłowe wyświetlenie obiektu szukanego w przestrzeni w momencie, gdy znajduje się on w zasięgu wzroku okazało się bardzo problematyczne ze względu sumaryczną niedokładność wszystkich sensorów. Prace nad wyświetleniem obiektu względem pozycji użytkownika rozpoczęliśmy od ustalenia warunków granicznych dla sytuacji idealnej. Warunki te:

- Jednoznaczne określenie pozycji szukanego obiektu
- Jednoznaczne określenie pozycji użytkownika
- Wyznaczenie położenia obiektu względem użytkownika
- Określenie czy obiekt jest w zasięgu w polu widzenia kamery
- Prawidłowa animacja obiektu na podstawie ruchu kamery

Spełnienie tych warunków zapewniłoby idealną prezentację w czasie rzeczywistych, która w 100% odpowiada zachowaniu użytkownika. Niestety ze względu na problemy związane z sensorami nie udało się zrealizować wszystkich założeń w pełni, a niektóre wymagały dostosowania do napotkanych realiów.

Bezproblemowo zrealizowane zostało kryterium jednoznacznego określenia pozycji szukanego obiektu. Ze względu na fakt, że jest to obiekt statyczny i wybiera go użytkownik z pośród losowy wybranych koordynatów mogliśmy zawsze ze 100% skutecznością określić położenie obiektu w przestrzeni.

W przypadku jednoznacznego określenia pozycji użytkownika pojawiły się już problemy, ponieważ mogliśmy określić jedynie przybliżoną pozycję użytkownika. Zgodnie z rozdziałem 5.2.1 mogliśmy określić gdzie znajduje się w danej chwili użytkownik z dokładnością do kilku metrów. Jednak największą przeszkodą w realizacji tego warunku były losowe skoki koordynatów z którymi musieliśmy sobie poradzić.

Efektem niedokładności GPS i problemów z jednoznacznym określeniem pozycji użytkownika były problemy z określeniem położenia obiektu względem użytkownika. Jednakże to kryterium udało się zrealizować. Dzięki rozwiązaniom opisanym w rozdziale 6 oraz wykorzystaniu dodatkowo kompasu byliśmy w stanie zrealizować zagadnienie związane z kierowaniem użytkownika do poszukiwanego obiektu. Niestety szum sensorów oraz losowe skoki z koordynatów mają negatywny wpływ na odczucia użytkownika podczas rozgrywki. Wahania wskazań oraz opóźnienie wynikające z metod stabilizujących (opisane w dalszej części pracy) pozbawiły płynności i efektowności niektóre elementy.

Pomimo wszystkich problemów kryterium dotyczące określenia czy obiekt znajduje się w polu widzenia zostało zrealizowane, a skuteczność metod jest wysoka. Metoda zastosowana do tego przypomina bramkę logiczną, która mówi reszcie programu czy można wyświetlić obiekt. Poziom niedokładności tej metody zależy od dwóch podstawowych czynników:

- Kąt widzenia kamery. Na ten czynnik nie mamy żadnego wpływu ponieważ wynika z tego na jakim urządzeniu aplikacja jest testowana. Mały kąt widzenia kamery sprawia, że aplikacja jest w większym stopniu narażona na wahania czujników. Wynika to z faktu, że podczas kontroli zakresu azymutów warunki skrajne (najniższy azymut, najwyższy) napotykane są częściej niż w przypadku dużego kąta widzenia
- Wahania czujników. W celu określenia czy obiekt znajduje się w polu widzenia dokonywana jest kontrola azymutów (opisana w dalszej części pracy). Jeśli obiekt dopiero wchodzi w pole widzenia i dochodzi do nagłego skoku odczytu obiekt może w sposób losowy pojawić się i zniknąć z ekranu. Efekt ten widoczny jest przy warunkach brzegowych i obniża efektowność aplikacji i może mieć negatywny odbiór ze strony użytkownika

Ostatnim warunkiem jaki musiał został spełniony była prawidłowa animacja obiektu 3D w polu widzenia użytkownika. Kryterium to ma dwa aspekty:

- Programistyczny – polega na wytworzeniu metod, które pozwolą na sterowanie obiektem 3D w ramach sceny
- Użytkowy – polega na dostarczeniu jak największego realizmu przedstawionej sceny. Użytkownik musi być przekonany, że to co widzi na obiektywie faktycznie reaguje na jego zachowanie.

- Lokacyjny – polega na prawidłowym umieszczeniu obiektu w obrębie sceny na kamerze. W tym przypadku dysponujemy dwoma układami współrzędnych. Zewnętrznym – tym który dotyczy człowieka i obiektu w obrębie świata rzeczywistego – i wewnętrznym – tym, który dotyczy obiektu w obrębie sceny przedstawianej na kamerze.

Aspekt programistyczny został w pełni zrealizowany. Wytworzone zostały metody, które pozwalają przyszłym programistom na proste sterowanie obiektem w zależności od ich potrzeb bez potrzeby zagłębiania się w meandry silnika odpowiadającego za generowanie obiektów trójwymiarowych.

W przypadku aspektu użytkowego, po raz kolejny pierwszoplanową rolę odegrały losowe skoki koordynatów. Ze względu na nagłe i częste zmiany odczytów zachowanie obiektu była skokowa lub opóźniona. Kwestia czy skokowa czy opóźniona zależy od wariantu zastosowanego rozwiązana – oba rozwiązania opisane są w dalszej części pracy.

W przypadku aspektu lokacyjnego rozwiązaliśmy problem prawidłowego przeniesienia i przeliczenia pozycji z układu zewnętrznego do układu wewnętrznego. Dodatkowo udostępniliśmy możliwość dla programistów, aby dostosowali wyniki do wielkości obiektów jakie generują. Podczas testów napotkaliśmy problemy wynikające z generowania różnych obiektów 3D dlatego uznaliśmy, że element ten może mieć dużą wartość dla osób korzystających z tego narzędzia. Należy jednak pamiętać, że skuteczność przełożenia jest częściowo uwarunkowana skutecznością pomiaru. Jeśli pomiar będzie niedokładny i/lub błędny to położenie obiektu w obrębie sceny może być zupełnie inne niż autor wstępnie założył.

5.2.5 Obsługa obiektu 3D

Ze względu na stosunkowo skomplikowaną obsługę obiektów 3D uznaliśmy, że aby Nasza biblioteka miała sens musi posiadać metody, które upraszczają ten proces. Największym wyzwaniem na etapie projektowania tych metod było zapewnienie jak największej prostoty dla programistów którzy będą z tych narzędzi korzystać. Naszym celem było wystawienie prosty funkcji, które odpowiadają za ustawienie podstawowych parametrów w taki sposób, aby programista nie musiał się przejmować zagadnieniami związanymi z kontrolą istnienia modelu czy aktualizacją generacji sceny. Liczymy, że wykorzystanie takich metod znacząco ułatwi i skróci czas operacji na obiekcie 3D.

5.2.6 Wnioski

Jak widać w powyższych rozdziałach ilość problemów na jakie natrafiliśmy podczas tworzenia prototypu – zarówno gry jak i biblioteki - była bardzo liczna. Niektóre były stosunkowo proste do rozwiązanie, inne problematyczne, ale skutecznie rozwiązanie, z kolei inne nie do obejścia i mające znaczący wpływ na działanie aplikacji.

Technologia rzeczywistości rozszerzonej jest fascynująca i możliwa do wykorzystania na urządzeniach mobilnych jednak nadal wykorzystanie jej wszystkich możliwości w pełnym zakresie nie jest w pełni możliwym.

Wszystkie komplikacje na jakie trafiliśmy w bardzo dużym stopniu zmniejszają grywalność oraz odbierają efektowność rozgrywce. Istnieją pewne potencjalne, alternatywne rozwiązania, jak na przykład:

- analiza obrazu
- rozpoznawanie przestrzeni[4]

Niestety rozwiązania te nie są w pełni zgodne z założeniami Naszego prototypu przez co nie mogliśmy z nich skorzystać. Naszym celem było oparcie się wyłącznie tym co przeciętna osoba posiada na co dzień przy sobie, a cały wirtualny świat w pełni przenieść i osadzić w świecie rzeczywistym. Nie chcieliśmy stosować półśrodków i sztucznych obejść dlatego podeszliśmy do tematu kompleksowo i dążyliśmy do uzyskania maksymalnej skuteczności i jakości wykorzystując standardowe systemy i urządzenia. Niestety, jak pokazała praca i wszystkie testy, na takie rozwiązanie jest jeszcze zbyt wcześnie. Według Nas, koncepcja za jakiś czas – wraz z rozwojem technologii smartphone`ów oraz czujników i urządzeń w nich montowanych – koncepcja ta będzie mogła zostać w dużo większym stopniu wykorzystana, jej skuteczność wzrośnie, a jakość rozgrywki będzie nieporównywalnie większa.

5.3 Proponowane rozwiązania

Proponowanym rozwiązaniem jest lekka, szybka aplikacja zapewniająca wielogodzinną rozrywkę. Pokazuje również jakie możliwości niesie ze sobą rzeczywistość rozszerzona przy wykorzystaniu zwykłego telefonu.

Rozgrywka rozpoczyna się od wybrania parametrów gry, a kończy po zebraniu wszystkich wylosowanych elementów.

5.3.1 Aktorzy

Aplikacja ze względu na swoją prostotę – od strony użytkowej – posiada jedynie dwa typy aktorów.

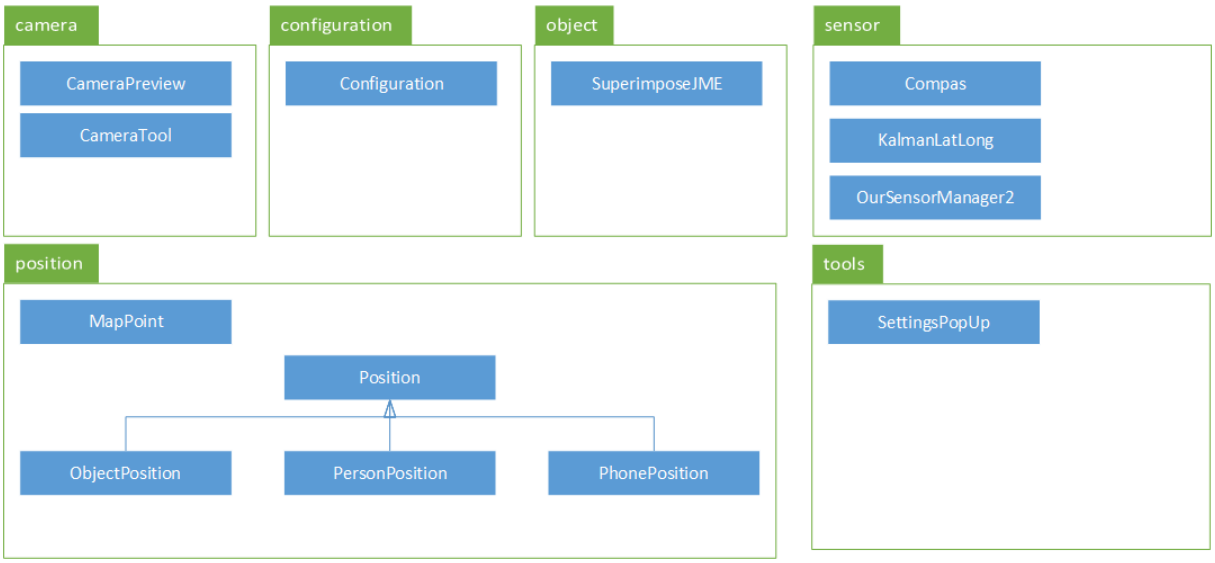
Rysunek 10 Aktorzy obecni w systemie. Źródło: Opracowanie własne.

[4] W rozdziale Rozwiązania alternatywne postaraliśmy się przybliżyć inne rozwiązania

Aplikacja obsługiwana jest przez aktora „Użytkownik”. To on wchodzi z nią we współpracę oraz posiada dostęp do wszystkich jej funkcjonalność. Z kolei za kontrolowanie rozgrywki oraz sprawdzanie jej parametrów odpowiada aktor „System”. Aktorzy wchodzą ze sobą w bezpośrednią interakcję.

5.3.2 Struktura biblioteki

Podczas projektowania Naszej biblioteki, zależało Nam na logicznym podziale poszczególnych pakietów. Oczywiście zdajemy sobie sprawę z tego, że nie jest to rzecz krytyczna, nie mniej jednak ma ona znaczenie podczas rozwoju oprogramowania. Planowana struktura biblioteki została umieszczona na Rysunek 11.



Rysunek 11 Stuktura biblioteki. Źródło: Opracowanie własne

5.3.2.1 Pakiet Camera

W ramach tego pakietu, przewidujemy umieszczać klasy związane z obsługą kamery. Powinny znaleźć się tu narzędzia, które pozwalają na inicjalizowanie oraz sterowanie kamerą.

5.3.2.2 Pakiet Configuration

W ramach tego pakietu, przewidujemy umieszczać klasy oraz inne elementy związane z konfiguracją biblioteki. Jest to również – Naszym zdaniem – najlepsze miejsce na umieszczanie różnego typu klas enumeracyjnych itp. Elementów.

5.3.2.3 Pakiet Object

Pakiet odpowiedzialny za obsługę obiektów 3D. W ramach Naszego prototypu będzie przechowywać klasę odpowiedzialną za inicjalizację elementów takich jak sceny czy obiekty 3D czy metody sterujące.

5.3.2.4 Pakiet Sensor

Pakiet odpowiedzialny za obsługę szerokiego pojęcia jakim są sensory. Pakiet ten powinien zawierać elementy nie tylko związanego z bezpośrednią obsługą sensorów, ale również z elementami pomocniczymi, np. klasami posiadającymi logikę poprawiającą odczyty.

5.3.2.5 Pakiet Position

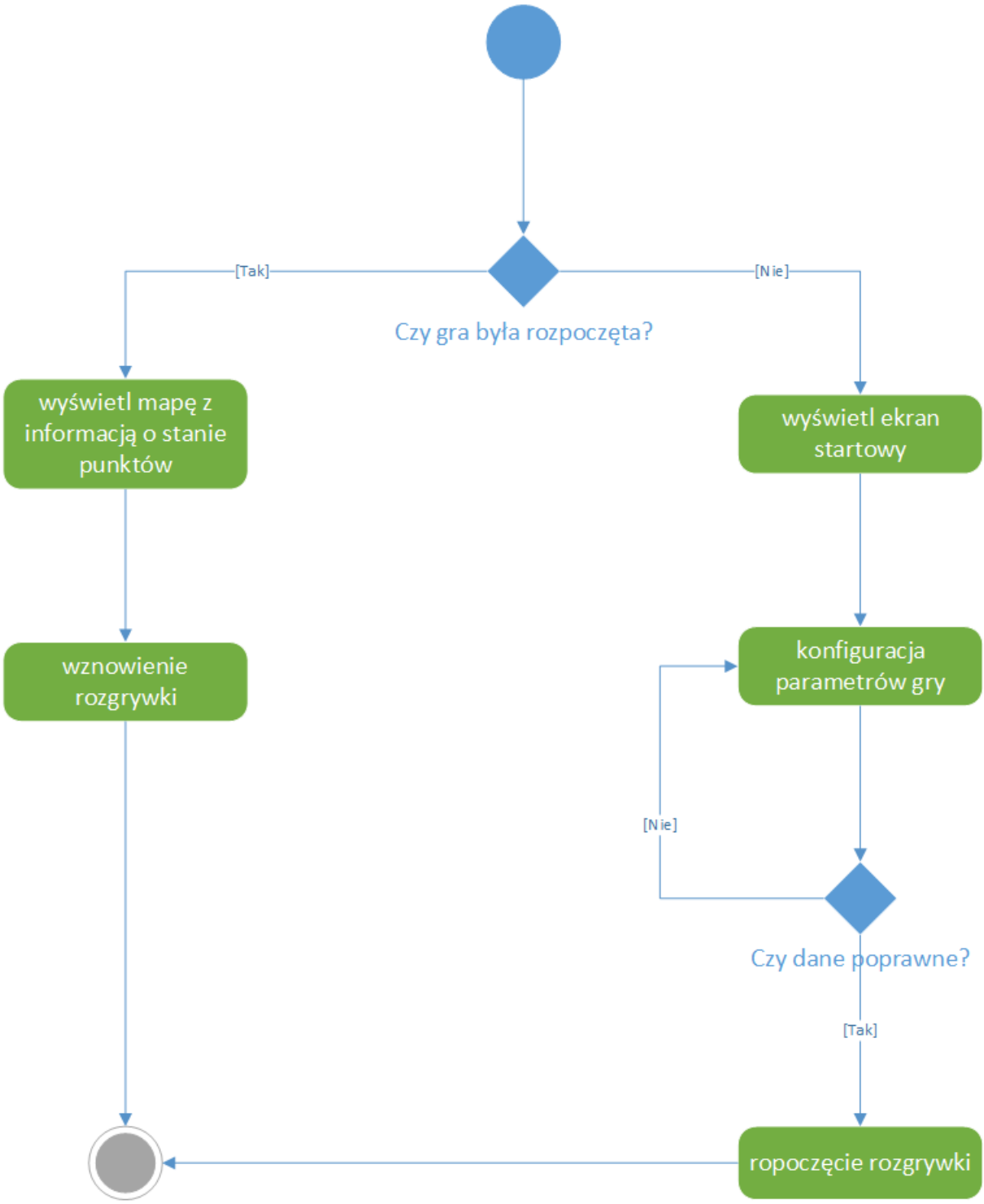
W ramach pakietu *Position* planujemy obsłużyć wszystko co związane z pozycją. Powinny znajdować się to modele opisujące pozycje różnych elementów, jak i również klasy narzędziowe związane z logiką określania, wyliczania, kierowania do pozycji.

5.3.2.6 Pakiet Tools

Pakiet charakteryzujący się szerokim pojęciem „narzędzia”. W Naszym zamiarze służyć do grupowania klas typowo narzędziowych, niekoniecznie związanych z logiką operacji rzeczywistości rozszerzonej.

5.3.3 Rozpoczęcie gry

Przypadek użycia zaprezentowany na Rysunek 12., który to przedstawia aktywność aktora „Użytkownik”.



Rysunek 12 Przypadek - rozpoczęcie gry. Źródło: Opracowanie własne.

Przykładowy scenariusz dla aktora „Użytkownik”

Użytkownik uruchamia aplikację. Po pomyślnym jej uruchomieniu System decyduje czy:

- Aplikacja ma wyświetlić ekran startowy

- Wznowić rozgrywkę

Ekran startowy

W przypadku ekranu startowego System pozwala na rozpoczęcie nowej rozgrywki. Na ekranie konfiguracji Użytkownik ma możliwość podania:

- Ilości elementów jakie ma odnaleźć
- Promienia w jakim punkty mają być rozmieszczone

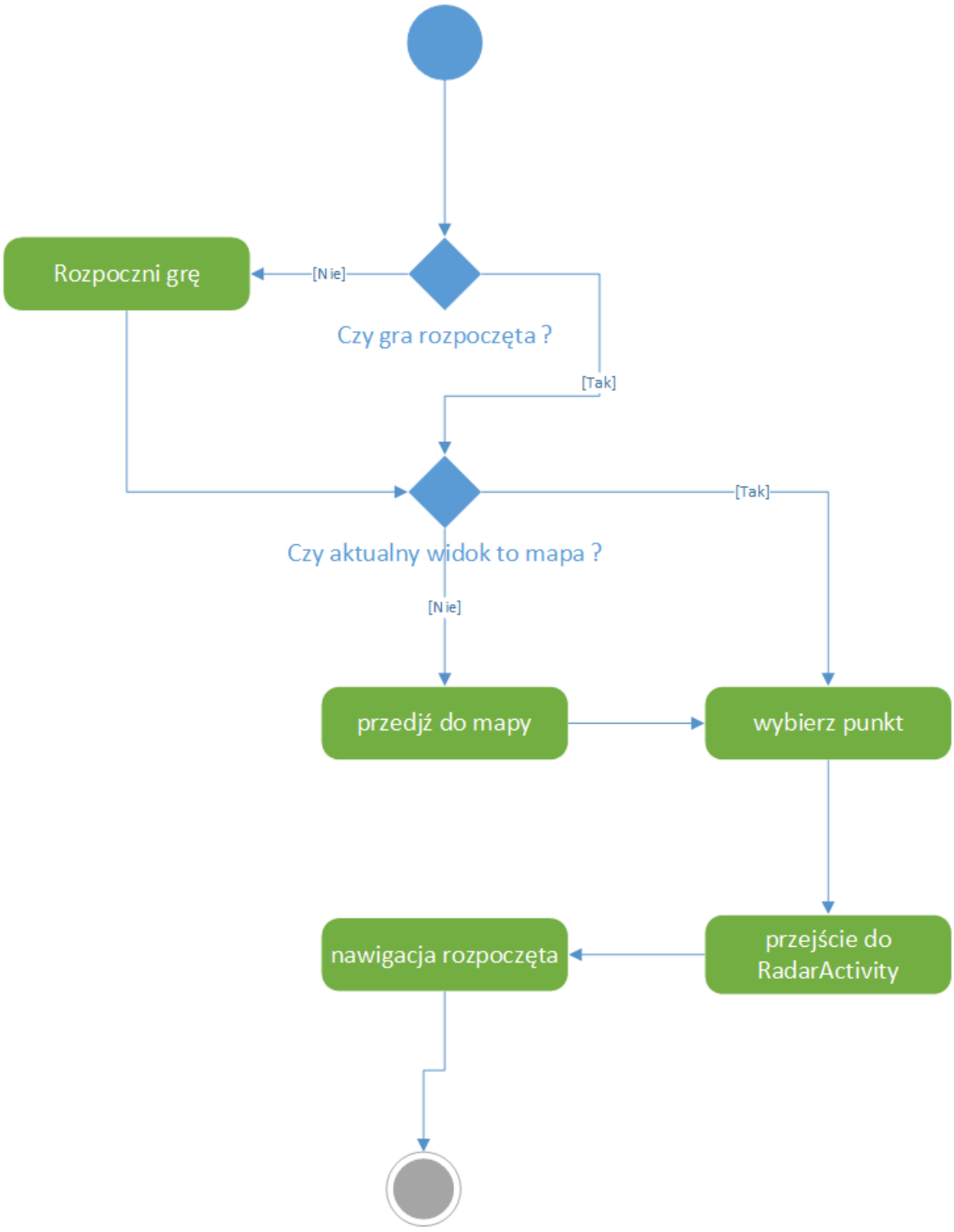
Po ich podaniu, System dokonuje walidacji poprawności danych. Jeśli dane zawierają błąd Użytkownik zostanie o tym poinformowany. Jeśli dane są podane prawidłowo System rozpocznie rozgrywkę.

Wznowienie rozgrywki

System rozpoczyna rozgrywkę w momencie w jakim została ostatnio zakończona. Aplikacja uruchomiona jest z pominięciem ekranu startowego i/lub konfiguracyjnego. Wszystkie punkty, które były wylosowane znajdują się w tych samych miejscach wraz z zachowanymi stanami odnalezienia.

5.3.4 Nawigowanie do punktu

Przypadek użycia zaprezentowany na Rysunek 13, który przedstawia aktywność aktora „Użytkownik”.



Rysunek 13 Przypadek – nawigowanie do punktu. Źródło: Opracowanie własne.

Przykładowy scenariusz dla aktora „Użytkownik”

Użytkownik uruchamia aplikację. Po pomyślnym jej uruchomieniu użytkownik musi przejść do LocatorActivity czyli aktywności odpowiedzialnej za mapę.

Ekran mapy

System na mapie prezentuje punkty w wybranym prez użytkownika obszarze. Na mapie zaprezentowane w sposób graficzny są:

- Obszar
- Punkty do wyboru

W przypadku punktów do wyboru, system rozróżnia trzy stany:

1. Punkt do odszukania
2. Punkt do którego zmierza gracz
3. Punkt odnaleziony

W celu wybrania punktu do odszukania gracz wybiera punkt, a następnie potwierdza wybór przez naciśnięcie na jego nazwę.

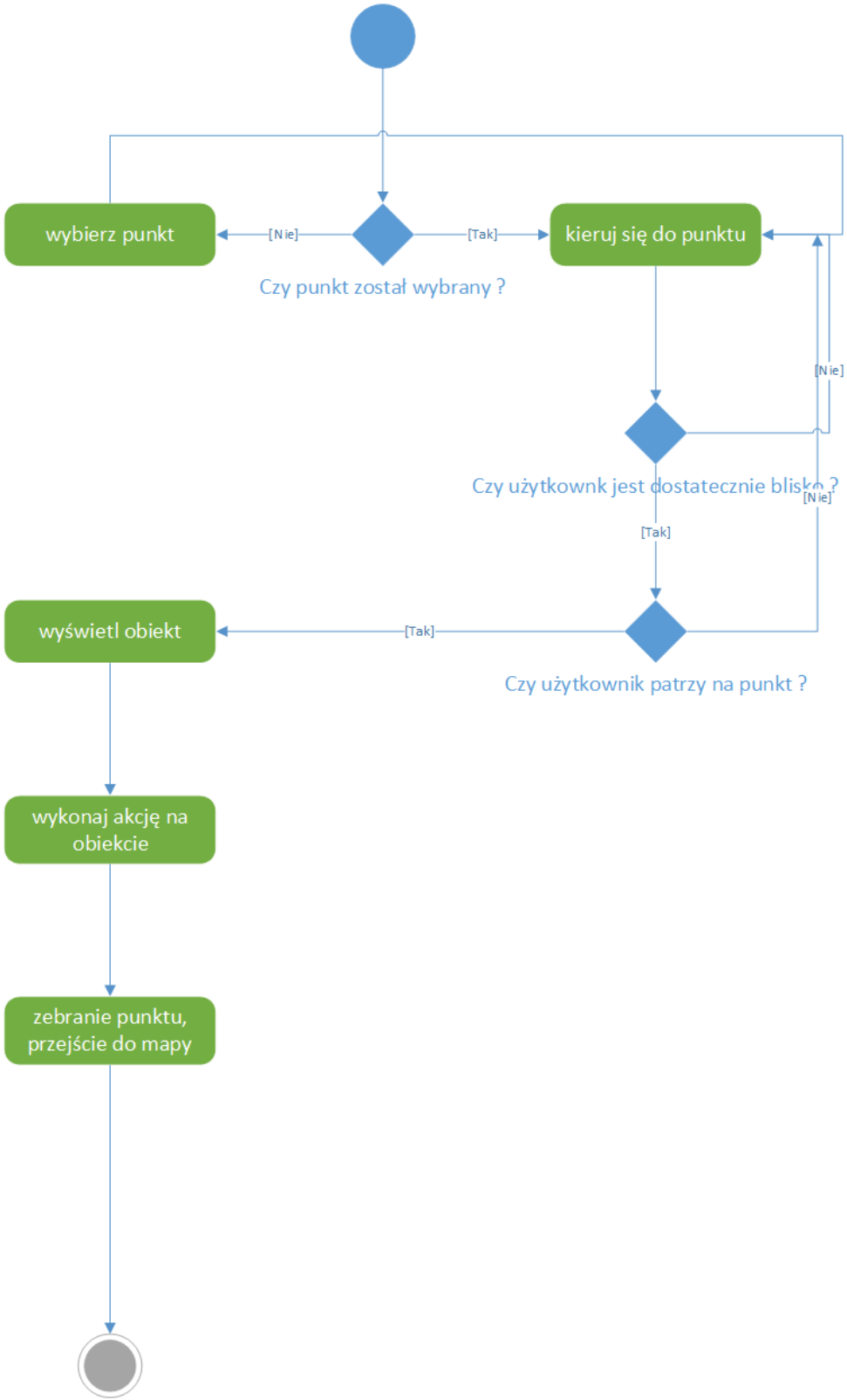
Nawigowanie do punktu

Od tego momentu rozpoczyna się nawigacja do punktu. System wskazuje kierunek w jakim ma się poruszać gracz, lecz nie wyznacza jednoznacznie trasy.

Przypadek użycia kończy się w momencie zebrania obiektu lub wybrania innego obiektu do odnalezienia.

5.3.5 Zebranie obiektu

Przypadek użycia zaprezentowany z Rysunek 14 przedstawia aktywność dla aktora „Użytkownik”.



Rysunek 14 Przypadek – zebranie obiektu. Źródło: Opracowanie własne.

Przykładowy scenariusz dla aktora „Użytkownik”

Użytkownik uruchamia aplikację. Po pomyślnym jej uruchomieniu użytkownik musi przejść do LocatorActivity czyli aktywności odpowiedzialnej za mapę. Na mapie dokonuje wyboru punktu do którego chce dotrzeć.

Nawigowanie do punktu

Po dokonaniu wyboru punktu do jakiego użytkownik chce dotrzeć, System za pomocą kompasu wewnętrznego prezentuje kierunek w jakim należy się kierować.

Sprawdzenie warunków widoczności

System w czasie rzeczywistym kontroluje czy obiekt spełnia warunki wyświetlania:

1. Obiekt jest dostatecznie blisko
2. Obiekt znajduje się w polu widzenia kamery

Jeśli oba warunki są spełnione użytkownik może zebrać obiekt poprzez wykonanie na nim akcji (naciśnięcie na obiekt palcem na ekranie)
Zebranie punktu kończy przypadek użycia.

5.4 Rozwiązania alternatywne

W tym rozdziale skupimy się na opisaniu rozwiązań alternatywnych. Będą to metody i/lub półśrodki, które można zastosować w celu poprawy pewnych aspektów prototypu. Przedstawione rozwiązania mogą zmieniać bazowe założenia jakie przyjęliśmy. Każda taka zmiana jest traktowana przez nas jako wada rozwiązania oraz dodatkowy koszt jaki ze sobą niesie.

5.4.1 Wykorzystanie zewnętrznych urządzeń

Jedną z opcji może być wykorzystanie dodatkowego odbiornika GPS w formie urządzenia zewnętrznego, które łączy się z telefonem np. za pośrednictwem złącza bluetooth. Rozwiązanie to ma kilka zasadniczych wad:

1. Wymaga zakupu dodatkowego urządzenia, które użytkownik będzie posiadał przy sobie
2. Cena. Przy założeniu tworzenia gry dla przeciętnej osoby, korzystanie z zewnętrznego urządzenia powinno być jedynie opcją. W przypadku gdyby był to element wymagany wtedy taka aplikacja miałaby małe szanse zaistnienia
3. Wymaga wykorzystania kolejnego elementu jakim jest bluetooth co dodatkowo zwiększy zużycie baterii

Rozwiązanie – mimo wad – może mieć pozytywny wpływ na działanie mechaniki gry, ponieważ odczyty z takiego urządzenia mogą być dokładniejsze i stabilniejsze.

5.4.2 Wykorzystanie animacji sceny

Wykorzystanie animacji obiektu 3D może pozwolić na częściowe oszukanie gracza, oraz pozwoliłoby na poprawę odbioru – grywalności – obiektu przez gracza. Wymagałoby jednak to rezygnację z jednego z kluczowych założeń jakim jest zakotwiczenie obiektu do konkretnej pozycji GPS na świecie.

Rozwiązanie to polega na utworzeniu animacji w ramach renderowanego obiektu 3D i uruchomienie jej niezależnie od zachowania gracza. Aplikacja w momencie, gdy wykryje, że użytkownik widzi obiekt musi jedynie dokonywać zmiany punktu widzenia kamery wewnętrznej po to żeby pokazać inną część animacji. Pozwala to na niwelację problematycznego sterowania obiektem i oparcie się jedynie określenie punktu widzenia. W ten sposób usuwany jest jeden z dwóch problematycznych czynników związany z wyświetlaniem obiektu 3D w oparciu o lokalizację.

5.4.3 Analiza obrazu w celu zakotwiczenia

Aplikacja przed wyświetleniem obiektu może dokonać analizy obrazy i wybrać jego fragment, a obiekt zakotwiczyć do obrazu. Wiąże się to z rezygnacją z jednego z podstawowych założeń jakim jest kotwiczenie obiektu do pozycji GPS.

Założenie to może być realizowane na dwa sposoby:

1. Wyszukanie charakterystycznego fragmentu, np. płaska przestrzeń, fragment na bazie figury geometrycznej przez analizę konturów
2. Nauczenie aplikacji rozpoznawania konkretnych obiektów takich jak np. auto, budynek

Następnie po pozytywnym rozpoznaniu, aplikacja musiałaby zakotwiczyć obiekt do rozpoznanego fragmentu obrazu. Zadaniem aplikacji byłoby określenie czy obiekt jest w polu widzenia, zanalizować obraz i zakotwiczyć obiekt do niego. Przemieszczenie obiektu na ekranie byłoby zależne od przesunięcia rozpoznanego fragmentu obrazu.

Zastosowanie tej technologii dodatkowo ogranicza grywalność, ponieważ wymaga dobrej widoczności i odpowiednich warunków atmosferycznych co powoduje, że rozgrywka może odbywać się jedynie w takich, które spełniają odpowiednie kryteria.

5.4.4 Podsumowanie

Jak widać, istnieją różne rozwiązania poszczególnych problemów na które natrafiliśmy podczas tworzenia prototypu. Jednak nie są to rozwiązania idealne, które małym kosztem/nakładem rozwiązują problemy definitywnie. Rozwiązania alternatywne, owszem mogą być skuteczne w pewnych dziedzinach jednak przeważnie drastycznie zmieniają założenia prototypu, tym samym ich wykorzystanie zmusza do stworzenia trochę innego produktu niż ten który miał pierwotnie powstać.

Rozwiązania te dodatkowo ukazują pewną ułomność technologii na których się skupiliśmy. Fakt, że istnieje potrzeba szukania i stosowania półśrodków w celu uzyskania maksymalnej efektowności –

często kosztem efektywności – świadczy o tym, że na pewne rozwiązania rzeczywistości rozszerzonej w oparciu o urządzenia mobilne jest po prostu jeszcze zbyt wcześnie.

6 Realizacja projektu

W ramach rozdziału „Realizacja projektu” chcemy przedstawić najciekawsze i najważniejsze zagadnienia z jakimi się spotkaliśmy podczas tworzenia Naszego prototypu. Opiszemy elementy, które zaprojektowaliśmy i zaprogramowaliśmy w celu wykorzystania w innych podobnych projektach, a także opiszemy rozwiązania najtrudniejszych zagadnień. Rozdział ten posiada dużą wartość dla osób, które są zainteresowane zagadnieniami związanymi z rzeczywistością rozszerzoną i chcą zapoznać się z alternatywną koncepcją rozwiązań popularnych problemów.

6.1 Składowe tworzonej biblioteki

W tym rozdziale zawarte są informacje o najważniejszych elementach związanych z metodami/narzędziami, które mogą być wykorzystane przez innych programistów w innych projektach. Cel jaki Nam przyświecał podczas tworzenia tych narzędzi to zapewnić maksymalne uproszczenie procesów, które często są skomplikowane lub powtarzalne. Narzędzia te skupione są wokół:

- Sensorów
- Kamery
- Obsługi 3D

6.1.1 Obsługa sensorów

System Android w swoim SDK posiada dość dobrą implementacje do zarządzania sensorami. Niemniej jednak rozwiązanie to posiada kilka niedoskonałości, szczególnie widoczne w projekt podobnych do prototypu przedstawionego w tej pracy. Zauważyliśmy, że przy użytkowaniu kilku sensorów cześć linii kodu jest powtarzana. Również domyślnie nie są stosowane żadne mechanizmy wygładzania odczytów. W związku z tym podjęliśmy próbę usprawnienia/ułatwienia obsługi wielu sensorów. Przed przystąpieniem do implementacji ustaliliśmy, że nasza implementacja musi spełniać następujące założenia:

- Być maksymalnie prostą dla użytkownika końcowego
- Ograniczyć ilość powtarzającego się kodu
- Posiadać mechanizmy wygładzające odczyty, które są transparentne dla użytkownika końcowego

W celu uzyskanie odpowiedniego efektu postanowiliśmy obudować klasę `SensorManager`, która to standardowo odpowiada za obsługę sensorów. Aby rozpocząć korzystanie z tak przygotowanej klasy, użytkownik musi wykonać kod z Listing 1:

Listing 1 Przykład wykorzystania zarządcy sensorów. Źródło: Opracowanie własne

```
Public class Test extends Activity{
```

```
      Private OurSensorManager2 sensorManager;

      protected void onCreate(Bundle savedInstanceState) {

            sensorManager = new OurSensorManager2(this);
            sensorManager.addSensor(Sensor.TYPE_ACCELEROMETER,10,0);
            sensorManager.addSensor(Sensor.TYPE_MAGNETIC_FIELD,65,0);
      }
}
```

Tych kilka linijek kodu umożliwiło aktywowanie obsługi dwóch sensorów, przy jednoczesnym ustawieniu mechanizmów służących do wygładzania odczytów. Dodanie kolejnego sensora sprowadza się do wywołania metody `addSensor` analogicznie jak w powyższym przypadku sensora akcelerometru, czy pola magnetycznego. Zanim jednak wyjaśnimy dokładniej jak działa metoda `addSensor` skupmy się na samej inicjalizacji instancji obiektu naszej klasy. Można zrobić to wykorzystując jeden z dwóch udostępnionych konstruktorów:

- Konstruktor jednoargumentowy (użyty w powyższym listingu), który przyjmuje obiekt klasy `Context`. Obiekt ten jest niezbędny w celu inicjalizacji standardowego managera systemu Android.
- Konstruktor dwuargumentowy pozwala dodatkowo na ustawienie trybu czułości z jaką będą pracować nasze sensory. W przypadku pierwszego konstruktora wartość ta zostanie ustawiona na wcześniej zdefiniowany tryb: `SensorManager.SENSOR_DELAY_GAME`

Osoby bardziej zaznajomione z system Android zapewne zauważą pewne drobne ograniczenie idące za tym rozwiązaniem. Mianowicie tryb czułości pracy sensorów w naszym rozwiązaniu będzie taki sam w przypadku wszystkich sensorów. Oczywiście inna implementacja umożliwiająca zarządzanie czułością per sensor byłaby nie tylko możliwa, ale i stosunkowo prosta, to jednak świadomie postanowiliśmy ograniczyć taką możliwość. Z naszych obserwacji wynika, że w większości przypadków czułość wszystkich sensorów w obrębie danej aplikacji jest taka sama. Dlatego też w celu uproszczenia obsługi sensorów zdecydowaliśmy się zastosować mechanizm jednej czułości dla wszystkich obsługiwanych sensorów.

Wracając do metody `addSensor`, która to umożliwia nam dodanie wybranego sensora do naszego managera. Przyjmuje ona zawsze trzy argumenty:

- `Integer iSensorId` – jest to numeryczny identyfikator sensora, który zapisany jest w Androidowej klasie `Sensor` jak wartość statyczna finalna zmienna.
- `Integer iNumberOfResultsForMedian` – zmienna ta mówi nam ile ostatnich pomiarów powinno być zapamiętywanych dla danego sensora. Parametr ten bezpośrednio wpływa na wygładzenie odczytów z danego urządzenia i nie powinien być przesadnie wysoki (zbyt duża liczba pamiętanych wyników wpływa na zwiększone opóźnienie).

- `int iPrecision` – ostatni parametr przyjmowany przez metodę `addSensor`. Jego zadaniem jest określenie do ilu miejsc po przecinku ma być zaokrąglany wynik pomiarów danego sensora. Dzięki temu również możemy wygładzić odczyty kosztem dokładności pomiaru.

Jak widać inicjalizacja obiektu zarządcy sensorów nie jest skomplikowana. Również dodawanie poszczególnych sensorów odbywa się jedynie poprzez wywołanie jednej metody. Niewątpliwym plusem skorzystania z tego mechanizmu jest dodawany z automatu system wygładzania odczytu. Ostatnią funkcją, które w sposób nieco niejawny pełni ta metoda jest sprawdzanie, czy dany sensor jest obsługiwany przez urządzenie użytkownika. Odbywa się to poprzez sprawdzenie listy dostępnych sensorów danego sprzętu mobilnego, a w przypadku jego braku zostanie wyświetlony odpowiedni monit w formie okienka POP-UP.

Projektując własny manager do sensorów nie można zapomnieć o kwestii zarządzania energią. W tym celu zaimplementowaliśmy odpowiednie metody `onPause(),onDestroy()` i `onResume()`, które kolejno zadbają najpierw o wyrejestrowanie wszystkich sensorów, a następnie po przywróceniu aplikacji, ponownie je zarejestrują. Taki mechanizm pozwala na oszczędzanie energii poprzez zatrzymanie pobierania wartości z sensorów, gdy nasza aplikacja jest zatrzymana. Użytkownik końcowy musi jedynie zadbać o umieszczenie wywołania tych metod w swoim kodzie na przykład tak jak na Listing 2.

Listing 2 Obsługa akcji wstrzymania i wznowienia. Źródło: Opracowanie własne.

```
Public class Test extends Activity{

     Private OurSensorManager2 sensorManager;
     /*[…]*/

     @Override
     public void onResume() {
            super.onResume();
            sensorManager.onResume()
     }
     @Override
     public void onPause() {
            super.onPause();
            sensorManager.onPause()
      }

     @Override

     public void onDestroy() {

           super.onDestroy();

          sensorManager.onDestroy()
     }
}
```

6.1.2 Obsługa 3D

Obsługa silnika 3D może przysporzyć wielu programistom sporej dawki problemów. Są to zagadnienia o stosunkowo wysokim progu wejścia, dlatego Naszym celem było utworzenie metod pozwalających na podstawową obsługę obiektów.

Dzięki zastosowaniu Naszych metod dostępowych, programistę nie interesuje obsługa aktualizacji poszczególnych klatek scen. Jego jedynym zadaniem jest podanie odpowiednich parametrów, a klasa narzędziowa zajmuje się niezbędnymi operacjami.

6.1.2.1 Rotacja

Jedną z podstawowych operacji jest rotacja kamery. Rotacja nie zmienia położenia samego obiektu w obrębie wewnętrznego układu współrzędnych, a jedynie zmienia położenie kamery względem pozycji startowej. W celu wykonania rotacji należy wywołać metodę:

Listing 3 Definicja metody rotate. Źródło: Opracowanie własne.

```
rotate(float x, float y, float z)
```

Parametry jakie są przyjmowane przez metodę to kąt o jaki ma być wykonana rotacja w poszczególnych osiach (X,Y,Z). Należy pamiętać, że rotacja jest wykonywana o kąt względem ostatniej podanej wartości. Tak więc jeśli chcemy wykonać jedynie rotację o kąt 10 stopni w osi Y należy wykonać następującą operację.

Listing 4 Sposób wywołania rotacji. Źródło: Opracowanie własne.

```
 ((com.fixus.towerdefense.model.SuperimposeJME) app).rotate(0f, 10,
0f);
```

Dzięki takiemu wywołaniu rotacja zostanie wykonana tylko w jednej osi.

6.1.2.2 Skalowanie

Inną metodą na manipulację obiektem 3D, który często jest wykorzystywany to skalowanie obiektu. Skalowanie polega na przekształceniu modelu o zadany współczynnik na każdej z osi. Biblioteka domyślnie inicjalizuje obiekt przeskalowany do wartości `0.025f` na każdej z osi. W przypadku gdyby zaszła potrzeba zmiany udostępniliśmy metodę:

Listing 5 Definicja metody `scale`. Źródło: Opracowanie własne.

```
scale(float x, float y, float z);
```

Metoda ta dokona skalowania na każdej z osi o wartości podane przez programistę.

6.1.2.3 Przemieszczenie obiektu

Przemieszczenie obiektu[5] to jedno z kluczowych zadań. Odbywać się może na dwa sposoby:

- Z wykorzystaniem animacji
- Natychmiastowa zmiana położenia

Wykorzystanie animacji polega na przemieszczeniu obiektu między dwoma punktami kontrolnymi. Punktem startowym jest obecna pozycja obiektu, a punktem końcowym jest pozycja do której chcemy przemieścić obiekt. Wykorzystanie wewnętrznych mechanizmów biblioteki jMonkey pozwala na płynny ruch co znacząco uatrakcyjnia korzystanie z aplikacji.

W obrębie Naszego prototypu ruch odbywał się wyłącznie po osi X. Definicja udostępnionej metody jest bardzo prosta:

Listing 6 Definicja metody `startCinematic`. Źródło: Opracowanie własne.

```
startCinematic(float x)
```

Najważniejsze w efekcie końcowym jest fakt, że osoba korzystająca z biblioteki nie musi przejmować się obsługą tworzenia punktów kontrolnych oraz uruchamiania samej animacji, ponieważ wszystkie niezbędne operacje wraz z aktualizacją sceny obsłużone są przez Naszą bibliotekę.

Ponieważ nie każdy obiekt w świecie rzeczywisty porusza się w ten sposób, a także sama szybkość ruchu może zależeć od wykonanej operacji, udostępniliśmy również drugą wersję metody, która pozwala na ustawienie przez programistę czasu inicjalnego oraz prędkości.

Listing 7 Definicja metody `startCinematic`. Źródło: Opracowanie własne.

```
startCinematic(float x, float initialDuration,float speed)
```

- initialDurtaion – jest to parameter, który decyduje o ty mile trwać będzie animacja
- speed – jest to parametr, który decyduje o prędkości animacji. W przypadku, gdy chcemy, aby animacja trwała dokładnie tylko tyle co inicjalny czas należy podać wartość 1. Każda wartość większa od 1 spowoduje, że animacja wykona się w czasie wyliczonym według wzoru:

  ```
  czas animacji = initialDuration / speed
  ```

Przy częstych aktualizacjach – jak np. aktualizacja pozycji obiektu na podstawie ruchu telefonu – metoda ta doskonale niweluje efekt skokowego ruchu, uprzyjemnia korzystanie z aplikacji, poprawia grywalność. Należy jednak pamiętać, że jeśli aktualizacja pozycji będzie częsta oraz zmiana będzie

[5] Przemieszczenie obiektu to translacja w obrębie wewnętrznego układu współrzędnych

pozycji będzie duża, powstanie efekt opóźnienia. Opóźnienie to spowodowane jest tym, że w przypadku ruchu w odcinkach:

A – B

B – C

To ruch na odcinku B-C możliwy jest dopiero po zakończeniu animacji na odcinku A-B. W przypadku naszego prototypu, aby maksymalnie zniwelować ten efekt wykorzystaliśmy dwie rzeczy: poprawę jakości pomiarów, zwiększenie częstotliwości aktualizacji (w celu częstszej aktualizacji o mniejsze wartości).

Istnieje też alternatywna metoda zmiany pozycji obiektu. Polega na bezpośredniej zmianie pozycji obiektu w obrębie wewnętrznego układu współrzędnych. Nie posiada ona żadnej animacji. Można ją z powodzeniem stosować, gdy ruch obiektu nie jest istotny, np. gdy obiekt nie jest widoczny na scenie. W Naszym prototypie również do tego przygotowaliśmy metodę dostępowa do tego typu przesunięcia:

Listing 8 Definicja metody `move`. Źródło: Opracowanie własne.

```
move(float x, float y, float z)
```

W odróżnieniu od rotacji, zmiana współrzędnych w wewnętrznym układzie współrzędnych polega na bezpośrednim ustawieniu ich wartości w układzie XYZ. Wynika to z faktu, że biblioteka jMonkey nie wspiera bezpośrednio przesunięcia innego niż o wektor w przestrzeni. To znaczy, że przy każdorazowej zmianie należy wyliczać wszystkie trzy współczynniki lub zadbać o ustawienie dla poszczególnych osi aktualne wartości. Aby usprawnić proces ustawiania obiektu – zarówno na potrzeby prototypu jak i przyszłego programowania z wykorzystaniem biblioteki - przygotowaliśmy dodatkowe metody z których programista może korzystać w momencie, gdy przesunięcia chce dokonać tylko w jednej osi na raz:

Listing 9 Definicja metody `moveOneAxis`. Źródło: Opracowanie własne.

```
moveOneAxis(float value, String axis)
```

Metoda ta na podstawie wartości *axis* ustawi wartość na wskazanej osi na wartość równą parametrowi *value*. Jest to wygodny sposób przesunięcia tylko w jednej płaszczyźnie. Gdyby zaszła potrzeba tworzenia kombinacji np. przesunięcia w dwóch płaszczyznach do dyspozycji są jeszcze metody:

Listing 10 Definicje metod przemieszczenia. Źródło: Opracowanie własne.

```
moveOnlyX(float x)
moveOnlyY(float y)
```

```
moveOnlyZ(float z)
```

Metody przesunięcia tylko w jednej płaszczyźnie w sposób automatyczny ustawiają pozostałe osie na poprzednie wartości. Dzięki temu programista nie musi zajmować się wyliczaniem i pamiętaniem o prawidłowym ustawieniu wszystkich trzech parametrów.

6.1.2.4 Automatyczna zmiana odległości

W ramach Naszego prototypu istotne było, aby system potrafił przełożyć odległość między użytkownikiem, a obiektem na położenie obiektu na ekranie. W związku z tym uznaliśmy za stosowne, że należy utworzyć metodę, którą inni programiści będą mogli wykorzystać.

Listing 11 Definicja metody `setUserLocation`. Źródło: Opracowanie własne.

```
setUserLocation(Location personLocation, Location objectLocation)
```

6.1.2.5 Wykorzystanie wewnętrznej animacji obiektu

Niektóre obiekty 3D[6] posiadają wbudowaną animację, która może zostać wykorzystana w projektach zarówno rozrywkowych jak i typowo biznesowych. Z tego względu postanowiliśmy udostępnić programistom możliwość wykorzystywania animacji.

Listing 12 Definicja metody `animationStart`. Źródło: Opracowanie własne.

```
animationStart(String animName, LoopMode loopMode, float animSpeed);
```

Metoda ta pozwala na uruchomienie animacji obiektu o:

- Wskazanej nazwie: `animName`
- We wskazanym trypie: `loopMode`
    - `LoopMode.Cycle` – *tryb ten będzie powodował, że animacja będzie odtwarzana od początku do końca i od końca do początku*
    - `LoopMode.DontLoop` – *tryb ten dezaktywuje pętle. Animacja zostanie odtworzona do końca, a następnie zatrzymana*
    - `LoopMode.Loop` – *tryb ten zapętla animację permanentnie*
- Ze wskazaną prędkością. Domyślna prędkość każdej animacji to *1f*

W przypadku gdyby programista podał nazwę nieistniejącej animacji, biblioteka zwróci stosowną wiadomość do logu systemowego oraz zwróci wyjątek.

---

[6] W tym podrozdziale obiekt 3D to definicja obiektu, który będzie renderowany

Dla zachowania symetrii dodaliśmy oczywiście metodę pozwalającą na wyłączenie animacji:

Listing 13 Definicja metody `animationStop`. Źródło: Opracowanie własne.

```
animationStop();
```

Po analizie dwóch powyższych metod doszliśmy do wniosku, że ciągłe budowanie animacji za pomocą metody `animationStart` często bywa niepotrzebne. W większości wypadków wystarczy jedynie początkowa inicjalizacja animacji. Ze względu na to dodaliśmy trzecią metodę sterująca animacjami:

Listing 14 Definicja metody `toogleAnimation`. Źródło: Opracowanie własne.

```
toogleAnimation(boolean switcher);
```

Rola tej animacji to zmiana trybu (włączona / wyłączona) animacji na podstawie parametru switcher. Zestaw tych trzech metod pozwala na szybkie i proste sterowanie animacją podczas działania aplikacji.

6.1.3 Obsługa kamery

Podstawowym narzędziem w tego typu aplikacjach jest kamera. W przypadku aplikacji, która dodatkowo renderuje obiekty trójwymiarowe jako *overlay*[7] dodatkowo dochodzi pojęcie sceny, czyli tego co prezentowane jest za pomocą silnika grafiki 3D.

Prace rozpoczęliśmy od przygotowania klasy `CameraPreview` której zadaniem jest utworzenie widoku kamery w trybie pełnoekranowym. W celu inicjalizacji widoku kamery należy:

1. Utworzyć obiekt klasy `CameraPreview` w momencie uruchomiena aktywności

Listing 15 Utworzenie podglądu kamery. Źródło: Opracowanie własne.

```
mPreview = new CameraPreview(this, mCamera, mCameraCallback)
```

2. Poinformować System o potrzebie wyświetlenia podglądu

Listing 16 Dodanie podglądu do warstwy prezentacji. Źródło: Opracowanie własne.

```
ViewGroup.LayoutParams lp = new ViewGroup.LayoutParams(1, 1);
```

[7] Overlay czyli powłoka nałożona obraz kamery.

```
addContentView(mPreview, lp);
```

Parametrami są:

- Kontekst wywołania
- Obiekt kamery
- Callback kamery

Aby zdjąć z programisty potrzebę tworzenia i inicjalizowania wszystkich elementów związanych z kamerą powstałą klasa narzędziowa `CameraTool`. Jej zadaniem jest bazowa inicjalizacja. Tak więc, aby utworzyć obiekt `mCamera` należy:

I. Utworzyć obiekt klasy `CameraTool`

Listing 17 Utworzenie narzędzi kamery. Źródło: Opracowanie własne.

```
this.cTools = new CameraTool();
```

II. Pobrać instancję kamery

Listing 18 Utworzenie instancji kamery. Źródło: Opracowanie własne

```
mCamera = this.cTools.getCameraInstance();
```

III. Zainicjalizować bazowe parametry

Listing 19 Inicjalizacja parametrów kamery. Źródło: Opracowanie własne.

```
mCamera = this.cTools.initializeCameraParameters(mCamera);
```

Callback pozwala na anlizę informacji zwracanych przez kamerę, a także dokonać synchronizacji animowanej sceny 3D z obrazem rzeczywistym prezentowanym na ekranie. Zgodnie z SDK należy utworzyć obiekt klasy `Camera.PreviewCallback()` i zaimplementować obsługę metody `onPreviewFrame(byte[] data, Camera c);` Metoda ta wywoływana jest przy każdej klatce obrazu. W przypadku Naszego prototypu pozwoliło Nam to na reagowanie na każdą zmianę w czasie rzeczywistym. Jest to ważny element w tego typu aplikacjach.

6.1.4 Konfiguracja obiektu

Jedną z kluczowych dla Nas rzeczy podczas budowania biblioteki, była opcja umożliwienia konfigurowania obiektów według potrzeb programisty. Nasze pierwotny pomysł dotyczył przyjęcia generycznej metody nazewniczej jednak rodziło to dwa zasadnicze problemy:

- Komplikowało sytuację rozwoju biblioteki, a mimo, że to jedynie prototyp nie chcieliśmy stosować rozwiązań, które ograniczałyby rozwój biblioteki
- Wymagałoby od autora materiałów 3D nazywania obiektów zawsze w ten sposób, co na dłuższą metodą może być niewygodne

Z tego względu wprowadzona została klasa, która umożliwia konfigurowanie elementów. W wersji prototypowej pozwala ona na ustawienie:

- Ścieżki do pliku obiektu
- Ścieżki do pliku z teksturą
- Ścieżki do materiału

Klasa ta zawiera jedynie pola statyczne, co pozwala na inicjalizację na początku działania programu, a następnie nieprzejmowanie się ustawieniami 3D. Poszczególne pola to:

Listing 20 Pola konfiguracyjne. Źródło: Opracowanie własne.

```
public static String modelPath;
public static String texturePath;
public static materialPath;
```

Dodatkowo dzięki zastosowaniu odpowiedniej konfiguracji, Nasza biblioteka pozwala na przeciążanie domyślnych ustawień dostarczanych przez *jMonkeyEngine*. Można dokonać przez przeciążenie plików, jakie udostępnia. Jeśli programista w ramach swojego pliku w obszarze katalogu *assets* odtworzy strukturę katalogów, która jest dostarczona przez *jME* i zapisze plik zgodnie ze standardem *jME* dokona przeciążenia domyślnych ustawień. Jest to bardzo wygodna metoda, która nie wymaga żadnych zabiegów programistycznych.

6.2 Wygładzanie odczytów z sensorów

W tworzonej przez nas aplikacji bardzo dużą rolę odgrywają sensory wbudowane w urządzenie mobilne. Właściwości owych sensorów wykorzystywaliśmy do takich rzeczy jak przełączanie pomiędzy mapą a widokiem z kamery, nawigowaniem do obiektu, czy wreszcie wyświetlaniem wirtualnego obiektu na ekranie smartphone. Niestety już podczas pierwszych testów okazało się, że dokładność jak i czułość tych sensorów dalece odbiegała od naszych oczekiwań, w wyniku czego musieliśmy zmierzyć się z problemem wygładzania odczytów płynących z sensorów. Zadanie to okazało się wyjątkowo

ciekawym zagadnieniem i okazją do sprawdzenia wielu rozwiązań. Poniżej postaramy się przybliżyć wybrane przez nas techniki.

6.2.1 Sensory – uśrednianie, mediana i zaokrąglanie wyników

Pierwszym z problemów na jakie się natknęliśmy były duże wahania w odczytach kompasu. Pomimo względnie nieruchomego trzymania telefonu, wskazówka naszego pseudo kompasu potrafiła odchylać się o kilkanaście (czasem nawet kilkadziesiąt) stopni w różnych kierunkach. Nasza analiza odczytów poszczególnych sensorów ukazała, że odczyty z sensorów przeważnie są podawane z dużą dokładnością kilku miejsc po przecinku. Znając wymagania naszej gry wiedzieliśmy, że aż tak dokładne dane nie przynoszą nam żadnych korzyści a prowadzą jedynie do zjawisk przez nas niepożądanych tak jak właśnie niekontrolowane wahania kompasu. Do poradzenia sobie z tym problemem postanowiliśmy wykorzystać najprostszą z możliwych metod, czyli zaokrąglanie. Żeby korzystanie z tego mechanizmu odbywało się w sposób jak najmniej inwazyjny i problematyczny postanowiliśmy dodać odpowiednie implementacje do naszej klasy obsługującej podstawowe sensory. Dzięki temu podczas inicjalizacji wybranego sensora możemy podać liczbę miejsc po przecinku do której to będą zaokrąglane odczyty.

Listing 21 Fragment metody dodającej sensor. Źródło: Opracowanie własne

```
public void addSensor(Integer iSensorId, Integer
iNumberOfResultsForMedian,int iPrecision){

     /*[…]*/
     new BigDecimal(oMedian).setScale(iPrecision,
         BigDecimal.ROUND_HALF_UP).floatValue();
     /*[…]*/
}
```

Niestety zastosowanie samego zaokrąglania odczytów nie przyniosło oczekiwanych rezultatów, gdyż wartości pobierane z sensorów różniły się często o wiele dziesiątych punktowych. Po kolejnej analizie tym razem już zaokrąglonych wyników zauważyliśmy pewną zależność, która charakteryzował x kolejnych wyników, przy względnie nieruchomej pozycji telefonu. Mianowicie raz na kilka poprawnych wyników pojawiał się błędny/przekłamany odczyt różniący się o kilka dziesiętnych punktowych. Pierwszym naszym pomysłem rozwiązania tego problemu było zastosowanie wartości maksymalnej zmiany wartości w jednostce czasu, jednak pomysł ten posiadał dość znaczącą wadę. Przykładowo jeśli wprowadzilibyśmy taki prosty algorytm w przypadku akcelerometru uniemożliwiłoby to szybkie ruchy telefonem, poprzez uznanie nowych odczytów jako błędne. W związku z tym musieliśmy zastosować jakieś udoskonalenie tego rozwiązania.

Postanowiliśmy zacząć kolejkować ostatnie odczyty z sensorów zachowując ostatnie n – rezultatów a następnie obliczać z nich wartość uśrednioną. Pomysł ten niestety również posiadał pewną

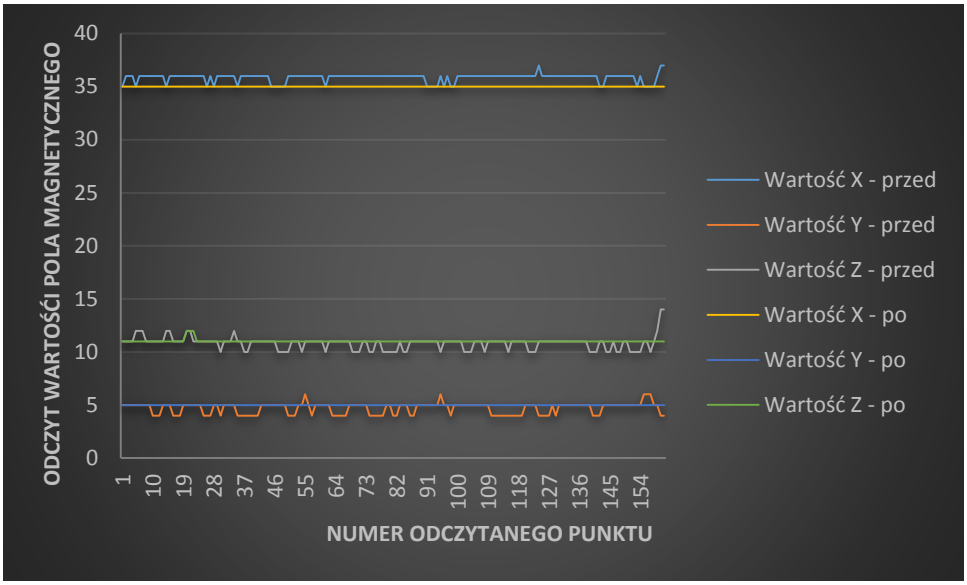
niedoskonałość, która ponownie objawiała się poprzez wahania odczytu. Oczywiście wahania te były już dużo mniejsze niż te wynikające z niemodyfikowanych odczytów, niemniej jednak ciągle były niesatysfakcjonujące. Mając jednak na uwadze wszystkie nasze poprzednie spostrzeżenia jak i dotychczasową implementacje uznaliśmy, że znacznie lepsze rezultaty powinniśmy uzyskać poprzez zastosowanie innego narzędzia statystycznego zwanego medianą. Jest to wartość środkowa określana też jako wartość przeciętna. Dzięki właściwością polegającym na większej odporności na wartości skrajne jest często wykorzystywana jako lepsza alternatywa średniej [46]. Zastosowanie tego mechanizmu sprawiło uzyskanie odczytów charakteryzujących się dużą mniejszą amplitudą odczytów, co z kolei doprowadziło do znacznego wygładzenia wartości zwracanych przez sensory. Ponownie implementacje postanowiliśmy umieścić bezpośrednio w naszej klasie obsługującej sensory, tak aby korzystanie z niej było maksymalnie uproszczone.

Listing 22 Implementacja mediany. Źródło: Opracowanie własne

```
private Float getMedian(Float[] oTmp,int iPrecision){

     if(oTmp.length == 0){

          return 0.0f;

     }else{
          Arrays.sort(oTmp);
          double oMedian;
          if (oTmp.length % 2 == 0){
               oMedian = ((double)oTmp[oTmp.length/2] +
               double)oTmp[oTmp.length/2 - 1])/2;
          }else{
               oMedian = (double) oTmp[oTmp.length/2];
          }
          return new BigDecimal(oMedian).setScale(iPrecision,
          BigDecimal.ROUND_HALF_UP).floatValue();
     }
}
```

Jak widać powyżej implementacja mediany jest dość prosta, a jej inicjalizacja ponownie odbywa się już podczas dodawania sensora. Stworzenie takie struktury umożliwia transparentne korzystanie z wygładzania odczytów przez użytkownika. Wykres 6 zawiera odczyty z sensora pola magnetycznego przed i po zastosowaniu omawianego algorytmu. Dane zostały wykorzystane przy pomocy smartphone przymocowanego do statywu. Po umieszczeniu urządzenia na statywie odczekano 10 sekund w celu ustabilizowania sensorów. Następnie kolejno zostało zebranych 160 odczytów pola magnetycznego, co przy obecnych ustawieniach przekłada się na około 10 sekundowy pomiar. Na potrzeby pomiarów wielkość próbkowania mediany została ustawiona na czterdzieści kolejnych odczytów. Jak można

odczytać z wykresu, zastosowany mechanizm pozwolił na uzyskanie niemalże idealnie gładkiego odczytu (odchył nastąpił jedynie raz w przypadku wartości Z).



Wykres 6 Wygładzanie odczytów – sensor pola magnetycznego. Źródło: Opracowanie własne.

6.2.2 Filtr dolnoprzepustowy

Gdy mieliśmy już odpowiednie mechanizmy umożliwiające odpowiednie wygładzanie poszczególnych sensorów kolejnym wyzwaniem było ostateczne wygładzenie rezultatów powstałych przy mieszaniu odczytów różnych sensorów. Przykładem takiego działania w naszej pracy jest kompas, który działa w oparciu o dwa sensory (akcelerometr i czytnik pola magnetycznego). Mając medianę poszczególnych sensorów uzyskiwaliśmy wyniki charakteryzujące się dość małymi wahaniami odczytu, jednak ciągle istniało miejsce na wprowadzenie poprawek. Do tego celu zastosowaliśmy implementacje znanego z układów elektronicznych filtru dolnoprzepustowego wraz z sposobem kolejkowania zastosowanego przy obsłudze sensorów.

Listing 23 Wygładzanie odczytów z kompasu. Źródło: Opracowanie własne

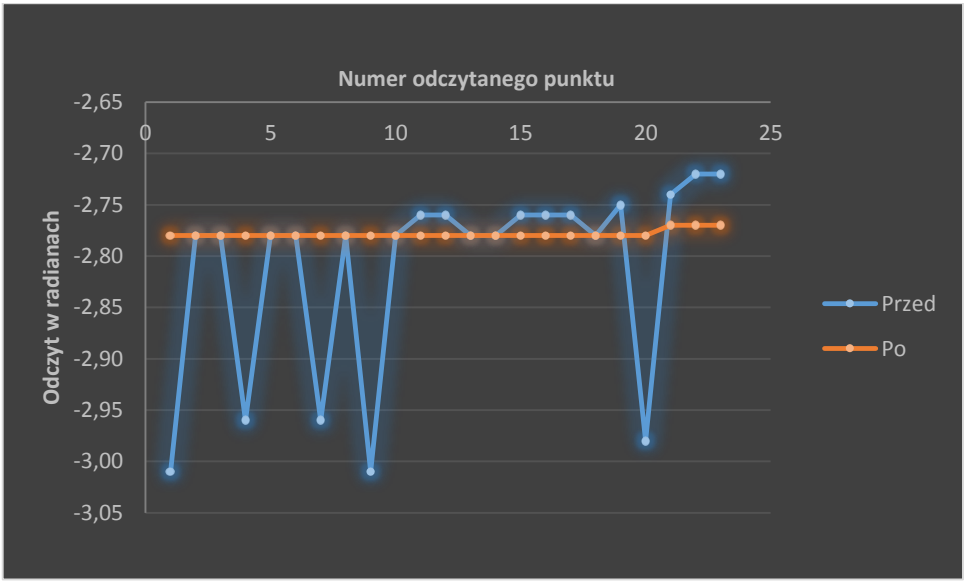
```
private static void addAzimutToQue(float radians){

      sumSin += (float) Math.sin(radians);
      sumCos += (float) Math.cos(radians);
      azimutQueue.add(radians);

      if(azimutQueue.size() > QUEUE_LENGTH){
        float old = azimutQueue.poll();
        sumSin -= Math.sin(old);
        sumCos -= Math.cos(old);
      }
}
private static float getAziumutAverage(){
```

```
        int size = azimutQueue.size();
        return (float) Math.atan2(sumSin / size, sumCos / size);
}
```

Na powyższym listingu znajduje się kawałek kodu implementujący założenia filtra dolnoprzepustowego. W wyniku jego działania otrzymujemy wynik pozbawiony szumów podany w radianach. W kodzie również pojawiają się funkcje trygonometryczne sinus i cosinus, które to umożliwiają uzyskanie faktycznego środkowego punktu pomiędzy dwoma wartościami kompasu. Żeby lepie to zobrazować posłużymy się przykładem. Jak wiadomo kompas można zobrazować jako dowolny okrąg, a więc zawiera się on w przedziale <0,360> stopni. Mając dwa przykładowe odczyty z kompasu 350 stopni, oraz 10 stopni, przy wykorzystaniu normalnej średnie uzyskalibyśmy wynik 180, ponieważ:

$$\frac{350+10}{2} = 180$$

Natomiast w przypadku kompasu środkową wartością w dla tych dwóch odczytów jest 0. Połączenie tego algorytmu wraz z wyliczaniem mediany dla poszczególnych sensorów umożliwiło wygładzenie odczytów kompasu. W chwili obecnej ruchy igły kompasu są mniej chaotyczne, oraz niemalże całkowicie zostały zniwelowane jej skoki w przypadku trzymania urządzenia względnie nieruchomego.



Wykres 7 Filtr dolnoprzepustowy – wygładzanie odczytów kompasu. Źródło: Opracowanie własne.

Na powyższym wykresie zostały zebrane 23 odczyty z kompasu. Kolorem niebieskim oznaczone zostały wyniki przed zastosowaniem algorytmów wygładzających, pomarańczowym zaś efekt działania mediany, oraz filtru dolnoprzepustowego. Wybór odczytu z kompasu jest o tyle interesujący, że wykorzystuję on wszystkie nasze pomysły, oraz co istotne jest wynikiem działania dwóch sensorów: akcelerometru i czytnika pola magnetycznego. Ponownie punkty zostały zebrane poprzez

przymocowanie telefonu do statywu i jego unieruchomienie. Następnie po odczekaniu 10 sekund po uruchomieniu aplikacji, zostały zebrane kolejno 23 odczyty.

6.2.3 Filtr Kalmana – estymacja odczytu GPS

Algorytmy zastosowane w przypadku sensorów i kompasu okazały się nie być wystarczające w przypadku próby prób wygładzania odczytu koordynatów z nadajników GPS. Po analizie dostępnych nam materiałów i zawartych w nich propozycji rozwiązania problemu zdecydowaliśmy się zastosować wariant wykorzystujący filtr Kalmana [47]. Jest to rodzaj algorytmu rekurencyjnego, który umożliwia estymacje wektora stanu. Nie będziemy jednak opisywać dokładnie wszystkich założeń tego algorytmu, gdyż nie jest to temat naszej pracy. Na potrzeby tej pracy i tego problemu najważniejsze jest, że dzięki jego zastosowaniu jesteśmy w stanie:

- Estymować nowe współrzędne.
- Uwzględnić minimalne przesunięcie w czasie, dzięki czemu niwelujemy nadmierne skoki wynikające z błędu odbiornika sygnału GPS. Polega to na tym, że wykorzystany jest współczynnik mówiący jaką maksymalną odległość można pokonać w ciągu danego czasu.

Listing 24 Implementacja filtru Kalmana. Źródło: Opracowanie własne.

```
public void Process(double dNewLatitude, double dNewLongitude,
     float fAccuracy, long lTimeStampInMilliseconds) {
          /*
           * Sprawdzenie, czy wartosc odchylenia pomiaru jest
           * wieksza niz zalozona minimalna wartosc (1 m)
           */
          if (fAccuracy < MIN_ACCURACY){
               fAccuracy = MIN_ACCURACY;
          }
          if (fVariance < 0) {
               // brak inicjalizacj czyli mamy pierwsze odpalenie
               this.lLastTimestamp = lTimeStampInMilliseconds;
               dLastLatitude = dNewLatitude;
               dLastLongitude = dNewLongitude;
               fVariance = fAccuracy * fAccuracy;
          } else {
               //Metoda Klamana - czesc wlasciwa
               long lTimestampDiff = lTimeStampInMilliseconds
```

```
                        - this.lLastTimestamp;
                if (lTimestampDiff > 0) {
                     /*
                      * warunek sprawdzajacy czy mielismy zmiane w
czasie
                      * jesli taka zmiana miala miejsce, to musimy
przeliczyc
                      * wspolczynik kowariancji
                      */
                     fVariance += lTimestampDiff * fMetersPerSecond
                               * fMetersPerSecond / 1000;
                     this.lLastTimestamp = lTimeStampInMilliseconds;
           }

                //wzmocnienie maciezy Kalmana
                float fKalmanGain = fVariance / (fVariance +
fAccuracy * fAccuracy);
                //zastosowanie wzmocnienia
                dNewLatitude += fKalmanGain * (dNewLatitude -
dNewLatitude);
                dLastLongitude += fKalmanGain * (dNewLongitude -
dLastLongitude);
                //nowa kowariancja (IdentityMatrix - K) *
Covarariance
                fVariance = (1 - fKalmanGain) * fVariance;
           }
}
```

Powyżej została umieszczona uproszczona implementacja filtru Kalmana. Uwzględnia ona sprawdzenie przesunięcia w czasie względem przebytej odległości. Zastosowanie tego algorytmu umożliwiła nam wygładzenie odczytu z urządzenia GPS, dodatkowo nie generując nadmiernego narzutu potrzebnego na obliczenia.

6.2.4 Wygładzanie sensorów – podsumowanie

Zastosowane w tej pracy metody niwelowania wahań/szumów występujący w odczytach sensorów w znacznym stopniu poprawiły jakość uzyskiwanych wyników. Problem błędnych skoków została zmarginalizowany, dzięki czemu między innym udało się zniwelować wahania igły kompasu. Rozwiązanie to oczywiście nie jest pozbawione wad. Główną z nich stanowi opóźnienie wynikające z konieczności uzbierania odpowiedniej ilości danych potrzebnych do wyliczenia mediany. Niemniej jednak efekt wygładzenie sensorów jest na tyle poprawny, że zdecydowaliśmy się na wprowadzenie

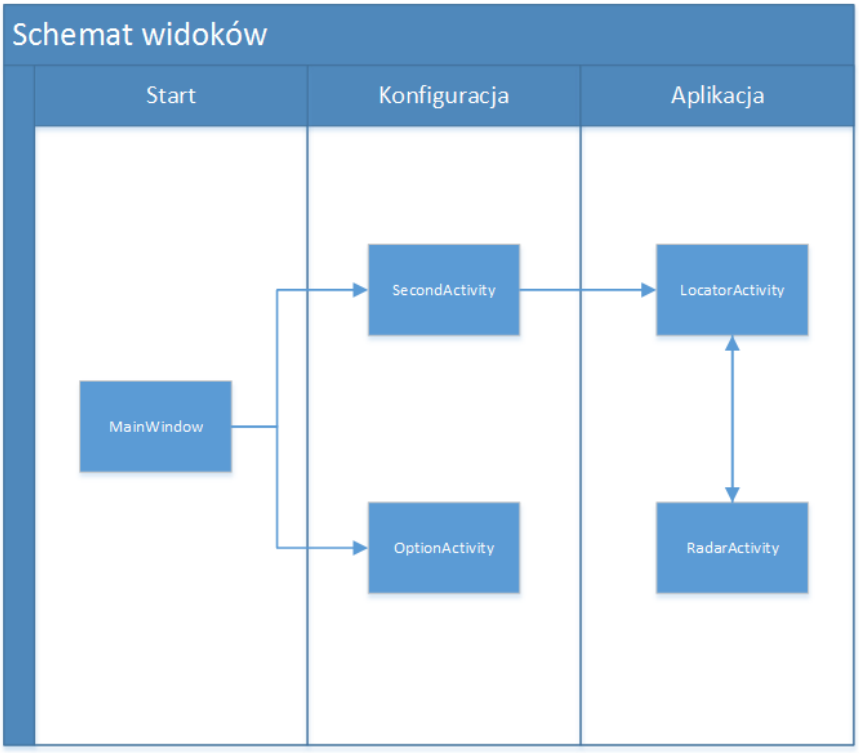
wszystkich powyżej opisanych metod do prototypu. Opisane przez nas mechanizmy oczywiście posiadają miejsce na dalszy rozwój szczególnie w kwestii zmniejszenia generowanego opóźnienia.

6.3 Projekt aplikacji

Podczas projektowania aplikacji przyświecał nam jeden podstawowy cel, chcieliśmy, aby sama aplikacja była prosta, a wszystkie jej składowe elementy były przydatne i łatwo używalne w innych projektach. Dla tego też wszystkie składowe były projektowane pod kątem przyszłych wdrożeń oraz przyszłego rozwoju. Stosowaliśmy się do dobrych praktyk przedstawionych przez Google w dokumentacji dla developerów [47].

6.3.1 Schemat widoków

Diagram z Rysunek 15 przedstawia schemat widoków, jakie zostały wykorzystane w ramach prototypu. Na diagramie zawarty jest również schemat przejść pomiędzy poszczególnymi widokami.



Rysunek 15 Schemat widoków. Źródło: opracowanie własne

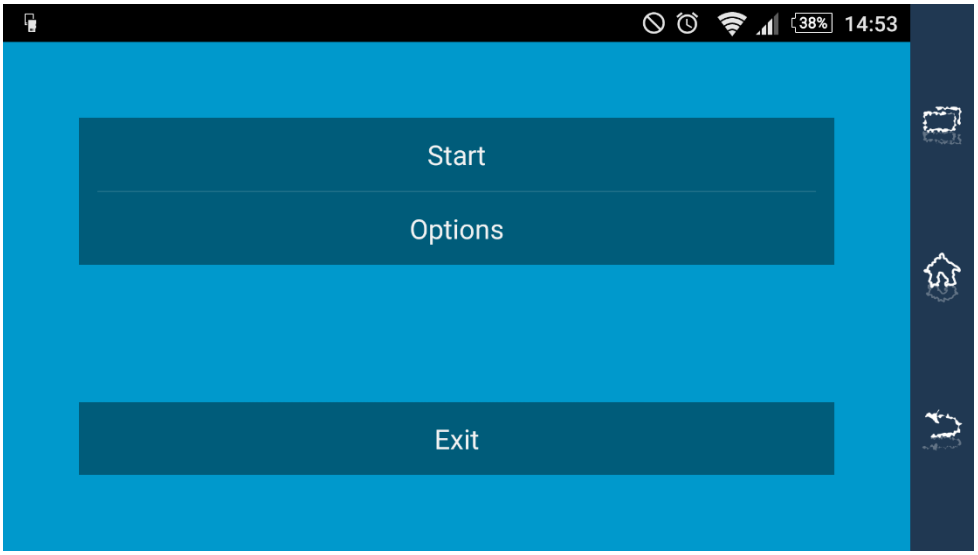
MainWindow

Widok ustawiony za pomocą manifestu jako widok domyślny.

Listing 25 Przykład konfiguracji trybu uruchomienia. Źródło: Opracowanie własne.

```
<category android:name="android.intent.category.LAUNCHER" />
```

Pozwala użytkownikowi na rozpoczęcie nowej rozgrywki lub przejścia do okna zawierającego preferowane ustawienia lub zamknięcie aplikacji.

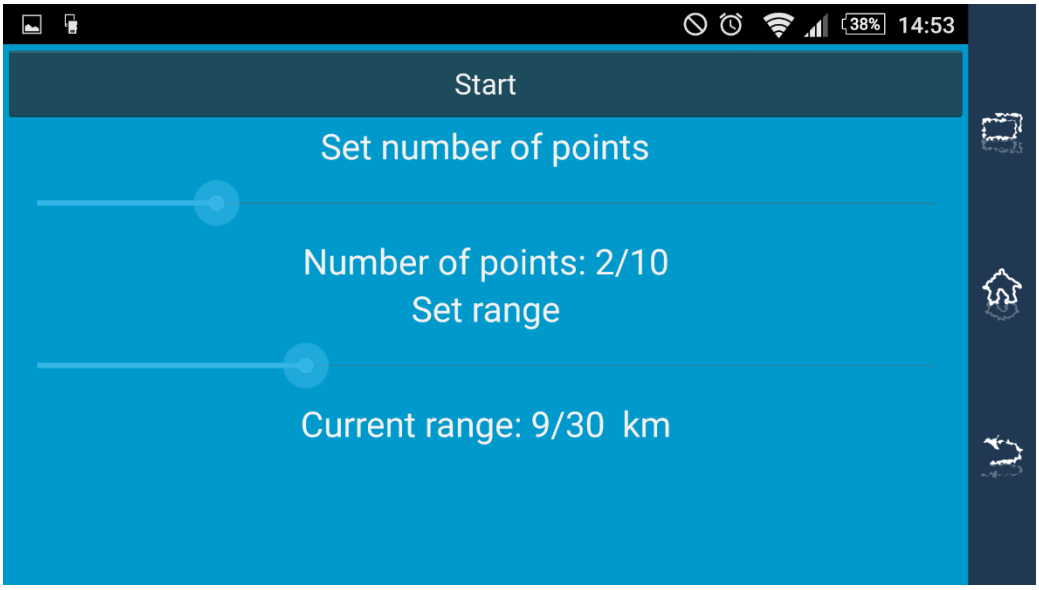


Rysunek 16 MainWindow - screeen. Źródło: Opracowanie własne

OptionActivity

Widok ten, nie jest wykorzystywany w ramach Naszego prototypu. Został jednak utworzony celowo pod przyszły rozwój aplikacji.

SecondActivity

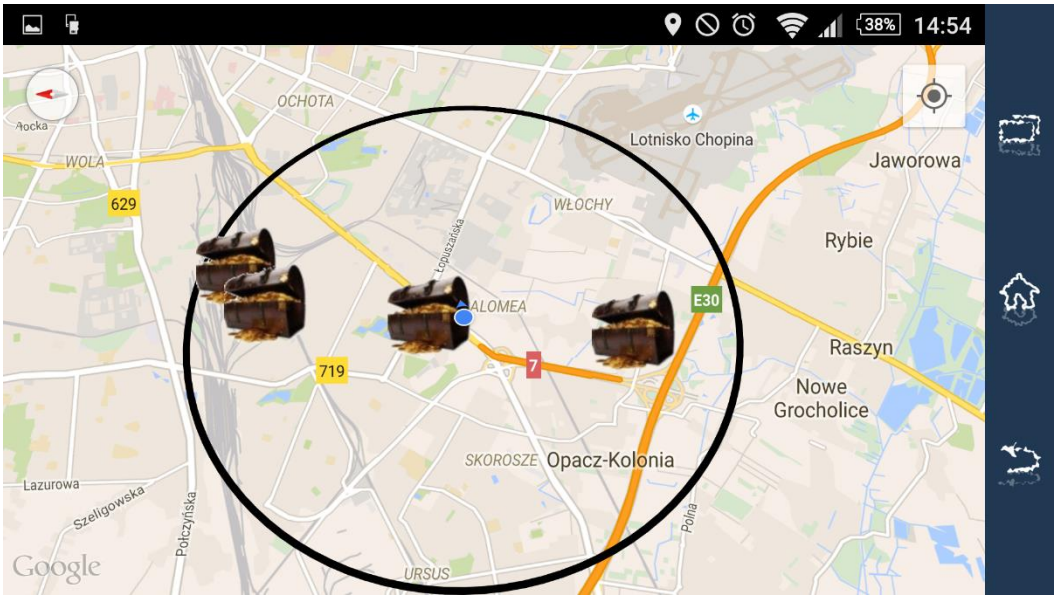


Rysunek 17 SecondActivity – screen. Źródło: Opracowanie własne

Widok służący do ustawienia parametrów dotyczących:

- Ilości obiektów do odszukania
- Promień w jakim obiekty mają być rozlosowane

LocatorActivity



Rysunek 18 LocatorActivity - screen. Źródło: Opracownie własne

Widok w całości oparty o Google Maps API v2 [48]. Pozwala na:

- Ustalenie swojej obecnej pozycji
- Określenie gdzie są granice wybranego obszaru
- Ustalenie, które obiekty zostały do odszukania, a które już zostały odszukane
- Wybranie obiektu do odszukania

RadarActivity

Rysunek 19 RadarActivity - screen. Źródło: Opracowanie własne

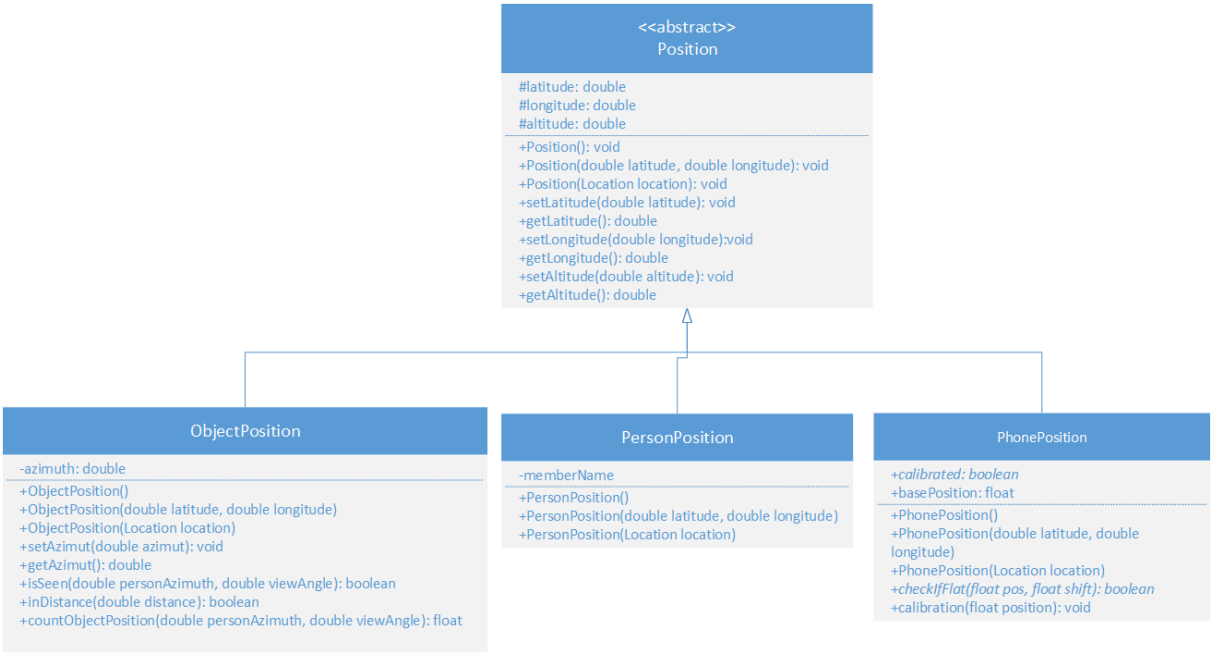
Jest to widok, który można uznać za najważniejszy. To w jego ramach prezentowane są:

- Kierunek w jakim należy podążać, aby trafić do obiektu
- Obiekt – gdy jest w zasięgu widzenia
- Animacja 3D obiektu

6.3.2 Klasy pozycji

Na potrzeby prototypu, ale i również przyszłego rozwoju, utworzyliśmy klasy pozycji poszczególnych typów obiektów. Rozróżniamy:

- Pozycję osoby
- Pozycję obiektu
- Pozycję telefonu



Rysunek 20 Diagram klas pozycji. Źródło: Opracowanie własne

6.3.2.1 ObjectPosition

Klasa `ObjectPosition` reprezentuje pozycję każdego obiektu. Poza podstawowymi informacjami, które pozwalają na przechowywanie informacji o położeniu w przestrzeni posiada szereg metod kontrolnych. Jedną z najważniejszych metod w ramach tej klasy jest metoda `isSeen` która pozwala określić czy obiekt w danym momencie znajduje się w polu widzenia użytkownika.

W celu zminimalizowania opóźnienia w kontroli, metoda ta jest wywoływana w każdej klatce przetwarzanego obrazu. Do ustalenia czy obiekt znajduje się w polu widzenia potrzebne są dwa parametry:

- Azymut na który patrzy użytkownik
- Kąt widzenia kamery

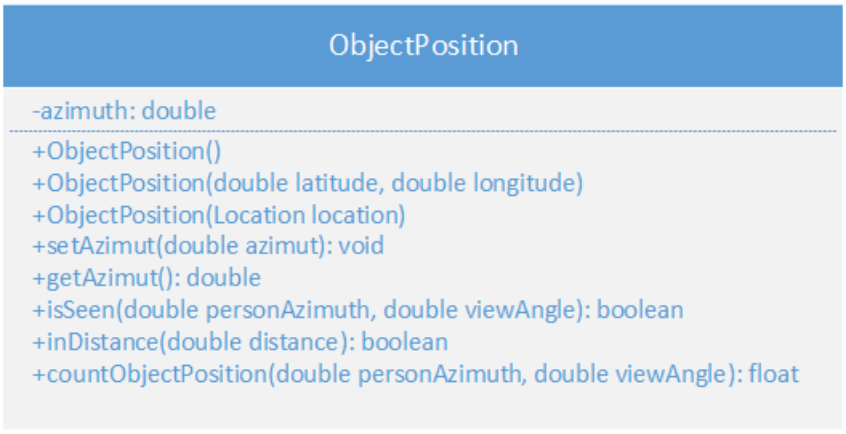
Kąt widzenia kamery można przyjąć lub pobrać z SDK (udostępnione są stosowne metody). Azymut na który patrzy użytkownik pobierany jest z odpowiednich metod Naszego kompasu. Mając te wartości możemy określić, czy azymut na którym obecnie znajduje się obiekt jest w polu widzenia kamery użytkownika. Pierwsza operacja do wykonania to określenie dolnego i górnego skrajnego azymutu jak widzi kamera w danym momencie poprzez uwzględnienie kąta widzenia kamery:

Listing 26 Ustalenie limitów do sprawdzenia pola widzenia. Źródło: Opracowanie własne.

```
double halfAngle = viewAngle / 2;

double topLimit = personAzimuth + halfAngle;
```

```
double bottomLimit = personAzimuth - halfAngle;
```

Następnie wystarczy sprawdzić czy azymut obiektu znajduje się w zadanym przedziale. Trzeba jednak pamiętać o wartościach granicznych. Azymut musi być z zakresu <0; 360> to znaczy, że w przypadku zbyt niskiego (poniżej 0) limitu dolnego lub zbyt dużego (powyżej 360) limitu górnego należy dokonać stosownego przekształcenia, aby wartość była w odpowiedniej ćwiartce okręgu.

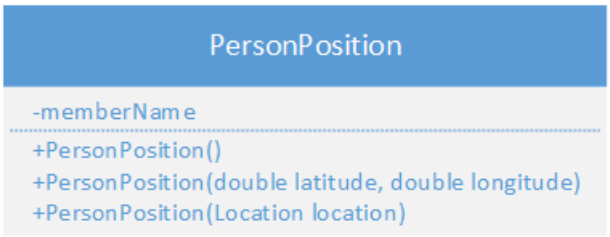


Rysunek 21 Klasa ObjectPosition. Źródło: Opracowanie własne

Dodatkowo klasa `ObjectPosition` posiada metody pozwalające na skontrolowanie czy obiekt znajduje się w polu widzenia w kontekście odległości (metoda `inDistane`) oraz kalkulująca pozycję obiektu na osi X podczas renderowania go na ekranie.

6.3.2.2 PersonPosition

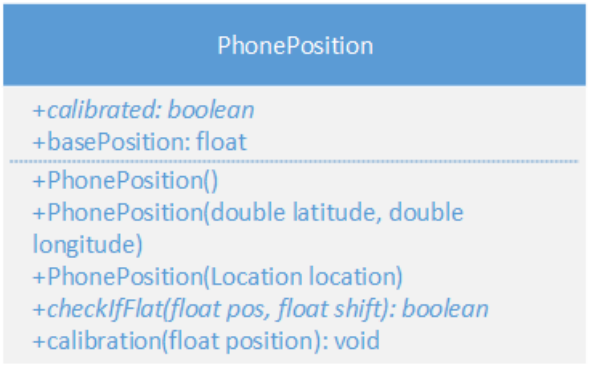
Klasa narzędziowa pozwalająca na określenie i przetwarzanie pozycji osoby. Stanowi ona też furtkę do przyszłego rozwoju aplikacji, np. do trzymania i określania pozycji członków drużyny.



Rysunek 22 Klasa PersonPosition. Źródło: Opracowanie własne

6.3.2.3 PhonePosition

Klasa definiująca pozycję telefonu w przestrzeni. W odróżnieniu od `ObjectPosition` oraz `PersonPosition` uwzględnia również fizyczną pozycję telefonu względem podłoża (np. zwraca informację czy telefon leży płasko względem podłoża). Pozwala to na przykład na sterowanie aplikacją co jest częstym elementem aplikacji opartych o rzeczywistość rozszerzoną.



Rysunek 23 Klasa PhonePosition. Źródło: Opracowanie własne

6.4 Potencjalne dalsze kierunki rozwijania prototypu

W tym rozdziale skupiamy się na przedstawieniu potencjalnych, przyszłych form rozwoju aplikacji. Mają one na celu wzbogacenie rozgrywki oraz podniesieniu wartości komercyjnej.

Dodatkowo opiszemy również dalsze proponowane przez Nas kierunki rozwoju samej biblioteki w celu zwiększenia jej atrakcyjności dla programistów.

6.4.1 Kierunki rozwoju – biblioteka

Przedstawiamy listę potencjalnych kierunków w jakich można rozwijać utworzony przez Nas prototyp biblioteki do budowania aplikacji mobilnych opartych o rzeczywistość rozszerzoną. Punkty te są niejako wnioskami jakie wyciągnęliśmy z testów i obserwacji.

6.4.1.1 Rozbudowa wsparcia kamery

Ciekawym kierunkiem rozwoju biblioteki byłaby rozbudowa obsługi kamery. W ramach Naszego prototypu, biblioteka zapewnia podstawowe wsparcie dla kamery czyli jej inicjalizację oraz ustawienie podstawowych parametrów. Już sam to zdejmuje dużą ilość pracy z programisty, jednak ciekawym pomysłem mogłoby być wprowadzenie domyślnej obsługi analizy klatek. Przy założeniu, że programista chcę zrealizować bliźniaczy lub podobny projekt, wprowadzenie domyślnej obsługi analizy klatek wydaje się ciekawym pomysłem.

Idąc dalej tym tropem, w ramach dalszego rozwoju można przemyśleć wprowadzenie kilku trybów analizowania klatek. Łatwo sobie wyobrazić projekt w ramach którego analizowanie każdej klatki jest zbędne. Można wprowadzić konfigurację pozwalającą na zmniejszenie ilości operacji w zależności od potrzeb. Każde takie zmniejszenie będzie miało pozytywny wpływ na zużycie baterii, ponieważ zmniejszy to obciążenie na jednostce obliczeniowej. Należy jednak pamiętać, że ilość klatek zależy od wielu czynników, dlatego mechanizm ten należy przemyśleć i przygotować go tak, aby wybrany tryb adaptował się do aktualnego wykorzystania.

6.4.1.2 Obsługa wielu obiektów 3D

W ramach Naszego prototypu mogliśmy ograniczyć do równoczesnego renderowania jednego obiektu trójwymiarowego. Była to ilość wystarczająca do przedstawienia Naszego konceptu oraz doskonałe wyjście dla przyszłego rozwoju biblioteki.

W trakcie analizy wymagań i projektowania doszliśmy do wniosku, że rozwój biblioteki pod kątem tworzenia skomplikowanych scen trójwymiarowych nie jest ścieżką jaką chcemy obierać. Z reguły jest to sprawa bardzo unikatowa i wynikająca ze specyfiku danego projektu, dlatego próba stworzenia generycznego narzędzia do tego mija się z celem i nie będzie miała pola do szerokiego zastosowania.

Po przeanalizowaniu całego zagadnienia, uważamy, że najlepszym sposobem na zapewnienie obsługi skomplikowanych scen byłby rozwój pod kątem obsługi wielu obiektów 3D. Chodzi tu o ich inicjalizację, renderowanie oraz bazową obsługę. W przypadkach, gdy to nie będzie wystarczające, programista zawsze będzie mógł rozszerzyć dostarczoną klasę. Dzięki temu zabiegowi zyska niezależność związaną z renderowaniem scen 3D, będzie mógł zaimplementować własną logikę, a zyska swobodę i wygodę związaną z obsługą wielu obiektów 3D i część pracy zostanie przerzucona z niego na bibliotekę.

6.4.1.3 Narzędzia GUI

Dobrym pomysłem byłby rozwój biblioteki i utworzenie narzędzi pozwalających na wprowadzenie do aplikacji interfejsu graficznego związanego z warstwą kamery. Chodzi tu o połącznie GUI z obiektami 3D prezentowanymi na ekranie. W przypadku standardowych widoków SDK Androida doskonale sobie z tym radzi i nie powinno się próbować tego zamienić. Rozwój powinien być skupiony wyłącznie na dostarczeniu narzędzi związanych z obiektem 3D.

6.4.1.4 Budowanie tekstowych overlay`ów

W ramach rozbudowy narzędzi do interakcji z użytkownikiem, biblioteka mogłaby zostać wyposażona w metody pozwalające na tworzenie tekstowych overlayów. Jest to prosta, ale często wystarczająca forma prezentacji informacji na ekranie. Rzeczywistość rozszerzona nie musi zawsze być

prezentowana z wykorzystaniem najbardziej efektownych środków przekazu. Pamiętajmy, że ta technologia jest związana z łączeniem świata cyfrowego z rzeczywistym, dlatego taka forma również może być bardzo przydatna.

6.4.2 Kierunki rozwoju – gra

Proces tworzenie prototypu gry, napotkane problemy jak i wiele godzin spędzonych nad testowaniem prototypu pozwoliły na określenie różnych dróg rozwoju aplikacji. Zebraliśmy te naszym zdaniem najciekawsze i opisaliśmy je w podrozdziałach począwszy od 6.4.2.1.

6.4.2.1 Zdobywanie części całego obiektu

W celu uatrakcyjnić rozgrywki rozbudowany może zostać element związany z obiektami trójwymiarowymi. Aplikacja musiałaby zostać rozwinięta o obsługę wielu różnych obiektów 3D. Każda nowa gra losowałaby obiekt 3D z kolekcji, a następnie dzielony na tyle części ile punktów wybrał gracz.

Zbieranie obiektów polegałoby na zbieraniu elementów układanki. Zebranie wszystkich powodowałoby złożenie całego obiektu którym gracz mógłby sterować (w każdej płaszczyźnie) w celu zapoznania się z nim.

Sensowność tego rozszerzenia polegałoby na dostarczeniu dużej ilości i wysokiej jakości treści trójwymiarowej, tak, aby oglądanie było prawdziwą nagrodą.

Efektem ubocznym tego rozwiązania byłby znaczący wzrost kosztów związanych z dostarczaną treścią.

6.4.2.2 Rozgrywka rankingowa

Rozgrywka rankingowa ma na celu wykorzystanie potencjału gry sieciowej. Każdy gracz gra zupełnie niezależnie od innego, dodatkowo aplikacja zostanie wzbogacona o system punktacji.

Punktowanie powinno uwzględniać takie zmienne jak:

- Ilość punktów
- Promień poszukiwań

Powinien również uwzględnić czas w jakim wszystkie punkty zostaną odnalezione. W celu wyrównania szans graczy, aplikacja może dodatkowo wyliczać średnią prędkość poruszania się gracza, aby uwzględnić czy porusza się on pieszo czy środkiem lokomocji co ma znaczący wpływ na czas przemieszczania między poszczególnymi obiektami.

Po każdym odnalezieniu wszystkich obiektów, aplikacja wysyła ostateczny wynik na serwer i generuje dostępny dla wszystkich ranking. Takie rozwiązanie bazujące na rywalizacji między graczami może zachęcić większą rzeszę ludzi do rozgrywki.

6.4.2.3 Rozgrywka drużynowa

Rozgrywka drużynowa ma pozwolić grać wraz ze swoimi znajomymi. Z racji, że jest to gra terenowa, wspólna rozgrywka może zainteresować wiele osób do skorzystania z aplikacji.

Rozbudowa systemu wymagałaby dwóch znaczących zmian w mechanizmach rozgrywki:

- Wprowadzenie możliwości wybrania wielkości drużyny wraz z utworzeniem mechanizmu łączenia graczy w zespół za pośrednictwem połączenia internetowego
- Synchronizacja poszukiwanych i odszukanych obiektów

W przypadku wprowadzenia mechanizmu drużyny aplikacja musi zostać rozszerzona o mechanizm łączenia graczy w zespół oraz synchronizację obiektów. Oznacza to, że po stronie serwera powinien istnieć mechanizm, który stworzy obiekt zespołu złożony z poszczególnych graczy – może być wymagana autoryzacja w celu jednoznacznej identyfikacji. Obiekt gracza – po stronie serwera – musi przechowywać informację o tym jakiego obiektu aktualnie poszukuje, a informacja ta musi być przekazywana pozostałym graczom.

Dodatkowo interesującym dodatkiem może być umożliwienie śledzenia członków zespołu i przedstawianie ich pozycji na mapie.

Najefektywniejszym sposobem rozbudowy prototypu o funkcjonalność rozgrywki drużynowej byłoby rozszerzenie klas `ObjectPosition` i `PersonPosition`. Są to klasy służące do identyfikacji elementów w przestrzeni, dzięki czemu są naturalnym wyborem podczas tworzenia możliwości trzymania stanu obiektów i graczy.

6.4.3 Podsumowanie

Jak widać aplikacja ma nadal wiele możliwych dróg rozwoju, które miałyby na celu zwiększenie atrakcyjności rozgrywki. Warto zauważyć, że nic nie stoi na przeszkodzie, aby zastosować te i inne możliwe rozszerzenia równolegle. Na przykład można zaimplementować moduł rozgrywki drużynowej, a także obsługę systemu punktowania. Wykorzystanie tych dwóch elementów, automatycznie tworzy nam trzecią opcję rozwoju jakim jest podział punktacji na zespoły oraz gry indywidualne. W ten sposób docieramy do coraz szerszego grona potencjalnych odbiorców.

Wiadomo, że każdy rozwój aplikacji związany jest z dodatkowym nakładem pracy, a co za tym idzie, ma to przełożenie finansowe. W związku z tym przed rozpoczęciem jakiejkolwiek pracy nad dodatkowymi funkcjonalnościami i/lub treściami dostępnymi w obrębie aplikacji należy dokonać dokładnej analizy rynku i przemyśleć, która ścieżka ma największe szanse na maksymalizację zysku w stosunku do zainwestowanego czasu i/lub pieniędzy.

Z kolei biblioteka ma niemal nieograniczone możliwości rozwoju. Wszystko zależy od nakładu pracy jaki chce się w to włożyć. To czym zajęliśmy się w trakcie pisania niniejszej pracy to elementy pozwalające na przedstawienie Nam konceptu i założeń jakie przyjęliśmy. W porównaniu do możliwości jakie niesie ze sobą technologia rzeczywistości rozszerzonej to jedynie czubek góry lodowej.

Co więcej uważamy, że Nasza biblioteka to idealny projekt dla podejścia polegającego na rozwoju w oparciu o partycypacje społeczności zainteresowanej tą tematyką. Rzeczywistość rozszerzona jest na tyle ciekawym i szerokim zagadnieniem, że zebranie zespołu pasjonatów w celu zbudowania w pełni funkcjonalnego, skończonego, generycznego narzędzia nie powinno stanowić problemu. Zwracamy na to uwagę, ponieważ jest to również pewien kierunek rozwoju oprogramowania. Często zdarza się, że o ostatecznej formie produktu końcowego decyduje zespół który go buduje, a także forma i metoda jego rozwoju.

7 Testy

Stworzony na potrzeby tej pracy prototyp w jednej z końcowych faz projektu został poddany procesowi testowemu. Wyróżniliśmy dwa rodzaje testów, przy pomocy których postanowiliśmy sprawdzić realizacje założeń koncepcyjny gry, oraz spróbować zmierzyć satysfakcje użytkowników. W związku z tym powstały scenariusze testowe jak i została wyselekcjonowana grupa użytkowników, którzy zostali poddani uproszczonym testom użyteczności (z angielskiego Usability).

7.1 Testy funkcjonalne

Wszystkie przedstawione tu testy funkcjonalne zostały opisane przy wykorzystaniu scenariuszy testowych (mających formę tabelaryczną). Integralnymi częściami każdego takiego scenariusza są następujące elementy:

- **Nazwa scenariusza** – zwarty opis przypadku testowego
- **Warunki początkowe** – określające stan początkowy jaki musi być osiągnięty przed przystąpieniem do realizacji scenariusza
- **Kroki** – rozpisanie kolejnych działań, które musi podjąć osoba testująca, w celu realizacji postawionego zadania
- **Oczekiwany rezultat** – pole to zawiera kryteria akceptacji poprawności zrealizowanego scenariusza testowego
- **Uzyskany rezultat** – zawiera krótką informacje z uwagami osoby testującej
- **Wynik testu/status** – jest to ostateczna ocena (plus krótki opis napotkanego problemu, jeśli wystąpił) osoby testującej zgodna z następującym schematem:
    - Błąd krytyczny – ocena taka zostanie przyznana w przypadku gdy realizacja scenariusza testowego jest niemożliwa/
    - Wada inna – status ten zostanie przyznany scenariuszowi, którego realizacja zakończyła się sukcesem. Niemniej jednak podczas jego wykonywania wystąpił jakiś inny błąd, niewpływający bezpośrednio na testowany przypadek użycia.
    - Wynik pozytywny – przypadek testowy zrealizowany bez problemów i wszystkie kryteria akceptacji zostały spełnione

Wszystkie przedstawione scenariusze testowe posiadają własne dodatkowe warunki początkowe, określające między innymi miejsce w aplikacji z którego nastąpi rozpoczęcia realizacji zadania. Dodatkowo przed rozpoczęciem każdego z opisywanych przypadków testujących, muszą zostać spełnione warunki ogólne:

- Urządzenie z systemem Android w wersji co najmniej 4.4
- Wsparcie dla sensorów:
    - Akcelerometru
    - Pola Magnetycznego

- GPS
- Minimum 1 GB pamięci ram
- Zainstalowana aplikacja z prototypem naszej gry

Tabela 17 Scenariusz testowy 01: Ustawianie właściwości gry. Źródło: Opracowanie własne.

Nazwa scenariusza	01. Ustawianie kryteriów gry zgodnie z wymaganiami użytkowników	
Warunki początkowe	Moduł GPS powinien być włączony, po czym należy uruchomić prototypu gry. Przypadek testowy rozpoczyna się od wyświetlenia głównego okna zawierającego przycisk „Start”.	
Realizacja scenariusza testowego		
Krok	Opis	Uwagi
1	Z menu głównego wybrać przycisk „Start”	
2	Na następnym ekranie wybrać dowolną liczbę punktów z zakresu <1,10>, oraz ustawić dowolny zasięg („Set range”) w zakresie <1,30>	
3	Przejście do ekranu wyświetlającego mapę	
4	Weryfikacja poprawności wyświetlanych informacji zgodna z kryteriami akceptacji.	
5	Scenariusz testowy należy powtórzyć trzykrotnie, wybierając inne wartości w kroku numer 2	
Oczekiwany rezultat (Kryteria akceptacji)	• Promień w którym zostały rozmieszczone punkty, będzie wyglądał na poprawny. Nie ma konieczności dokładnego sprawdzania. • Liczba wygenerowanych i zaznaczonych na mapie punktów jest zgodna z tą wybraną w pkt. 2. • Lokalizacja użytkownika jest punktem centralnym względem wygenerowanego pola gry.	
Uzyskany rezultat	Zgodny z oczekiwaniem.	
Wynik testu/status	Wynik pozytywny	

Tabela 18 Scenariusz testowy 02: Możliwość wyboru punktu. Źródło: Opracowanie własne.

Nazwa scenariusza	02. Możliwość wyboru wygenerowanych punktów na mapie

Warunki początkowe		Scenariusz testowy rozpoczyna się w momencie ustawienia punktów na mapie w scenariuszu numer 01 (minimalna ilość wybranych punktów powinna wynosić 2).
Realizacja scenariusza testowego		
Krok	Opis	Uwagi
1	Na ekranie mapy użytkownik wybiera dowolny wygenerowany punkt oznaczony ikonką skrzyni z skarbem: 	Nad wybraną ikonką powinna pojawić się „chmurka” zawierająca ciąg znakowy (napis).
2	Klikniecie wyświetlonego napisu	
3	Użytkownik powinien zostać przeniesiony do widoku aparatu	
4	Powrót do widoku mapy	
5	Ikonka uprzednio wybranego punktu powinna być zmieniona na muszkę strzelniczą: 	
6	Ponowne wybranie tego punktu powinno być niemożliwe, co powinno zostać sprawdzone poprzez próbę jego wybrania (Kroki od 1 do 2)	
7	Wybór innego dostępnego punktu i wykonanie punktów od 1 do 6, a następnie sprawdzenie kryteriów akceptacji.	
Oczekiwany rezultat (Kryteria akceptacji)	• Aktywny punkt nie może być ponownie wybrany • Ikona aktualnie wybranego punktu zmienia się • Po zmianie aktualnie wybranego punktu następuje odpowiednie ustawienie ikon	
Uzyskany rezultat	Zgodny z oczekiwanym	
Wynik testu/status	Wada inna – podczas jednego z testów wystąpił błąd opisany jako błąd silnika renderującego.	

Tabela 19 Scenariusz testowy 03: Możliwość zebrania wygenerowanego punktu. Źródło: Opracowanie własne.

Nazwa scenariusza	03. Możliwość zebranie wygenerowanego punktu

Warunki początkowe		Scenariusz testowy rozpoczyna się po zrealizowaniu punktu numer 3 z scenariusza testowego 02. Użytkownik powinien być w widoku aparatu.
Realizacja scenariusza testowego		
Krok	Opis	Uwagi
1	Podążać za wskazaniami strzałki symulującej zachowanie kompasu w celu znalezienia obiektu. Telefon powinien być ustawiony w pozycji horyzontalnej, bezpośrednio przed użytkownikiem, a strzałka być skierowana ku górze	Najlepiej wybrać punkt najbliższy naszej lokalizacji, oraz początkowo wybrać dużą liczbę punktów w promieniu 1 kilometra.
2	W odpowiedniej odległości od punktu na ekranie aparatu powinien wyświetlić się obiekt. Gdy to nastąpi należy w niego kliknąć.	
3	Użytkownik powinien zostać przeniesiony na obraz mapy	
4	Sprawdzić czy ikonka zebranego punktu została zmieniona na taką: 	
5	Próba ponownego wybrania tego punktu	
6	Wybranie innego punktu i przejście kroków 1-5, a następnie ocena kryteriów akceptacji	Ponownie najlepiej wybrać najbliżej położoną lokalizacje
Oczekiwany rezultat (Kryteria akceptacji)	• Zmiana ikony na mapie po zebraniu punktu • Blokada możliwości wybrania raz zebranego punktu • Wskazania strzałki prowadzą do wybranego punktu • Wyświetlenie animacji obiektu 3D w wybranej lokalizacji • Możliwość zebrania obiektu 3D	
Uzyskany rezultat	Zgodny z oczekiwanym	
Wynik testu/status	Wynik pozytywny	

Tabela 20 Scenariusz testowy 04. Sprawdzenie aktywacji modułu GPS. Źródło: Opracowanie własne.

Nazwa scenariusza	04. Sprawdzenie aktywacji modułu GPS

Warunki początkowe		Moduł GPS powinien być wyłączony. Po sprawdzeniu, że GPS jest wyłączony powinien zostać uruchomiony prototyp z grą.
Realizacja scenariusza testowego		
Krok	Opis	Uwagi
1	Wybór przycisku „Start”	
2	Ustawienie wartości: Liczba punktów: 4 Zasięg: 1 kilometr	
3	Na ekranie powinien zostać wyświetlony monit o braku włączonego GPS	
4a	Wybranie przycisku „Cancel”, po którym powinno nastąpić wyjście z aplikacji. Ponowne wykonanie punktów 1-3 a następnie wykonanie punktu 4b.	
4b	Wybór przycisku „Open settings”	
5	Wyświetlenie odpowiednie zakładki ustawień urządzenia mobilnego umożliwiającego włączenie GPS. Należy włączyć GPS i powrócić do aplikacji	
6	Wygenerowanie lokalizacji zgodnie z kryteriami określonymi w kroku numer 2	
Oczekiwany rezultat (Kryteria akceptacji)	• Pojawienie się monitu informującego o braku aktywnego GPS • Anulowanie akcji włączenia GPS powoduje zamknięcie prototypu • Możliwość przejścia do odpowiednich ustawień uruchamiających moduł GPS • Po uruchomieniu GPS punkty są generowane	
Uzyskany rezultat	Niezgodny z oczekiwanym	
Wynik testu/status	Błąd krytyczny: Monit o braku dostępności modułu GPS uruchamia się wielokrotnie, po wybraniu opcji cancel nic się nie dzieje. Po wybraniu przycisku „Open settings” zostaje wyświetlona właściwa	

	karta ustawień, jednak nawet po aktywacji GPS monit cały czas się wyświetla

Tabela 21 Scenariusz testowy 04. Możliwość zakończenia gry. Źródło: Opracowanie własne.

Nazwa scenariusza		05. Próba zakończenia gry
Warunki początkowe		Scenariusz testowy rozpoczyna się w momencie ustawienia punktów na mapie w scenariuszu numer 01 (minimalna ilość wybranych punktów powinna wynosić 2, a promień 1km).
Realizacja scenariusza testowego		
Krok	Opis	Uwagi
1	Wybieramy pierwszy punkt z mapy	
2	Odnajdujemy wybrany punkt i zbieramy przypisany mu obiekt	
3	Powtarzamy czynności 1-2 dla wszystkich pozostałych punktów	
4	Po zebraniu ostatniego punktu wybieramy przycisk „OK” z wyświetlonego monitu	
Oczekiwany rezultat (Kryteria akceptacji)	• Pojawienie się monitu informującego o zakończeniu gry • Zmiana wszystkich ikon na mapie	
Uzyskany rezultat	Zgodny z oczekiwanym	
Wynik testu/status	Wynik pozytywny	

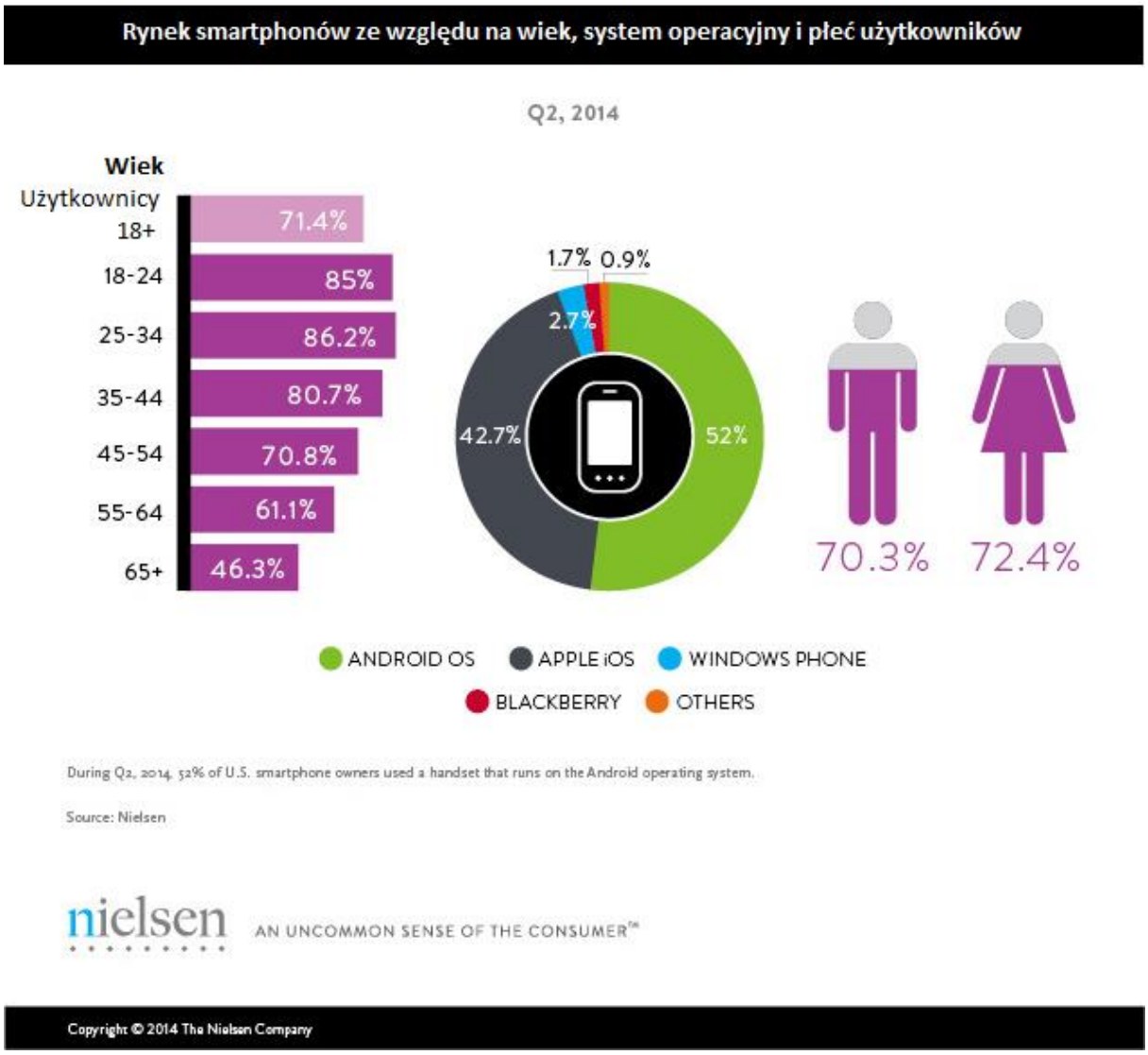
7.2 Testy użyteczności

W poprzednim podrozdziale skupiliśmy się na testach funkcjonalnych, które pozwoliły nam sprawdzić realizacje podstawowej funkcjonalności prototypu. Niemniej jednak testy tę w żaden sposób nie sprawdzały zadowolenia i łatwości realizacji celów użytkownika, a więc pomijana była kwestia Usability. W języku polskim termin ten najczęściej jest tłumaczony jako użyteczność i powiązany jest między innymi z analizą/miarą satysfakcji i łatwości obsługi produktu przez użytkownika końcowego. Wymienia się dwa najpopularniejsze sposoby badania użyteczności:

- Testy eksperckie – jest to rodzaj analizy przeprowadzonej przez tzw. Eksperta, który posiada szeroką wiedzę z zakresu projektowania interfejsów. Do jego zadań należy analiza rozmieszczenia elementów, intuicyjność rozwiązania, ale również sprawdza kontrast pomiędzy poszczególnymi elementami. Również dobry ekspert zwraca uwagę na przystosowanie testowanego bytu pod względem niepełnosprawnych (związane z terminem dostępności. Testy dotyczy głównie stron internetowych).

- Testy z użytkownikami – są rodzajem testów dających najlepsze efekty. Przeprowadzanie regularnych testów na grupie 5 osób, pozwala na redukcje błędów o 80%. Istnieje wiele metod i narzędzi umożliwiających przeprowadzenie badania. Najczęściej jednak wykorzystuję się postać moderatora, który jednocześnie jest obserwatorem sporządzającym notatki z realizacji zadań [49] [50].

W naszej pracy zdecydowaliśmy się przeprowadzić testy z użytkownikami. Jest to metoda nieco trudniejsza, ale w przypadku naszego prototypu bardziej zasadna. Zanim jednak przejdziemy do przeprowadzania testów należy określić grupę docelową naszego prototypu. W pierwszej kolejności określmy profil statystycznego użytkownika smartphone.



Rysunek 24 Rynek smartphone - profil użytkownika. Źródło: [51]

Jak widać na powyższej infografice (Rysunek 24 Rynek smartphone - profil użytkownika. Źródło:) penetracja rynku najwyższe wartości osiąga dla grupy wiekowej 18-34 lata. Natomiast rozkład z rozróżnieniem na płeć użytkownika jest niemalże identyczny. Jednakże powyższe informacje ciągle nie

są wystarczające, gdyż należy pamiętać że grupą docelową są specyficzni użytkownicy smartphone – gracze. Sprawdzając dane statystyczne przeprowadzone na potrzeby rynku USA, okazuje się że statystyczny użytkownik urządzenia mobilnego posiada zbliżone cechy, jak osoba grająca w gry mobilne. Średnia wieku wynosi w przybliżeniu 28 lat. Również w przypadku graczy płeć nie odgrywa bardziej znaczącej roli. 87% mężczyzn oraz 80% kobiet, posiadających urządzenia mobilne wykorzystuje je do grania w gry [52]. Podsumowując: profil odbiorcy naszego prototypu powinien charakteryzować się następującymi cechami:

- Wiek: 18-34 lata
- Płeć: dowolna
- Użytkownik smartphone
- Korzystający z gier mobilnych

Po określeniu grupy docelowej wytypowaliśmy 5 osób wpisujących się w wyżej wymieniony profil. Wśród testujących znalazły się 2 kobiety i 3 mężczyzn w wieku 18-29 lat. Również postawiliśmy na różnorodność w kwestii posiadanego wykształcenia, jego kierunku, oraz co najważniejsze z różnym stopniem obycia z smartphone. Jako przypadki testowe posłużyły nam lekko zmodyfikowane scenariusze z testów funkcjonalnych, a więc osoby testujące musiały wykonać następujące zadania:

- Rozpocząć grę z określonymi parametrami
- Znaleźć i zebrać obiekt
- Obsłużyć ekran z mapą
- Zakończyć grę

Wytypowaliśmy moderatora, którego zadaniem było kontrolowanie przebiegu testów, przy jednoczesnym zachowaniu następujących zasad:

- Braku podpowiedzi w przypadku problemów osoby testującej
- Nie odpowiadaniu na pytania, które mogły by doprowadzić do rozwiązania problemu (ewentualne pytania będą zapisywana, a odpowiedź zostanie udzielona po zakończeniu testu)
- Obserwacja zachowań osoby testującej, jej uwag, oraz trudności z jakimi się mierzyła
- Przeprowadzenie wywiadu po przeprowadzonym teście w celu zebrania opinii

Równocześnie wszystkie osoby przed podejściem do zadań były proszone o wyrażanie szczerych opinii i nieobawianie się krytykowania mankamentów występujących w prototypie. Każdy z testujących został poinformowany, że jedynie takie opinie są wartościowe w przypadku tego badania, oraz że oceniane jest konkretne rozwiązanie a nie osoby za nie odpowiedzialny. Informacja ta była o tyle istotna, że z powodu ograniczonych środków (a więc brak możliwości zapłacenia testerom za ich pracę) osoby te były werbowane z kręgu znajomych autorów tej pracy dyplomowej.

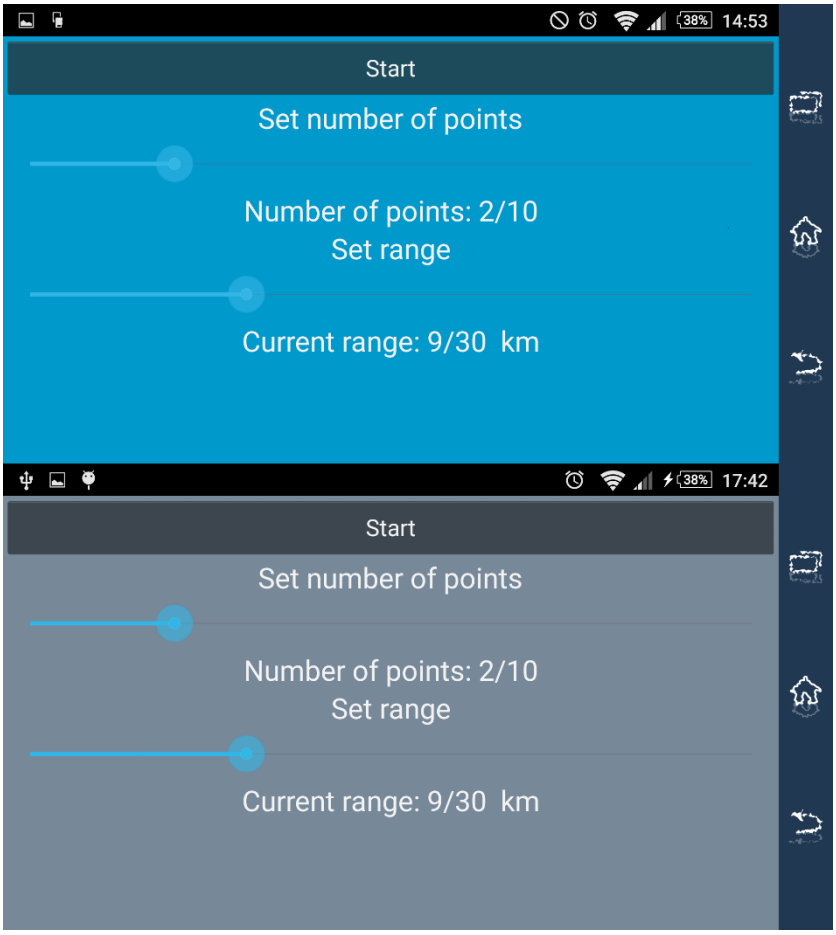
Wszyscy użytkownicy poradzili sobie z postawionymi zadaniami, realizując je z mniejszą lub większą ilością problemów. Niemniej jednak wszyscy mieli zastrzeżenia do interfejsu, które przekładają się na poniższe punkty:

- Gra powinna rozpocząć się od widoku mapy a nie z widoku kamery
- Brak samouczka
- Interfejs użytkownika jest nieczytelny (głównym zastrzeżeniem był dobór tła i kontrolek, które się ze sobą zlewały)
- Widok mapy powinien być zawsze w pozycji horyzontalnej (zablokowanie możliwości zmiany orientacji tej aktywności). Wynika to z faktu, że gra miała tendencje do powracania do widoku mapy, w przypadku gdy wybieranie celu odbywało się kiedy mapa była wyświetlana w pozycji pionowej
- Brak możliwości zakończenia działania gry przy próbie zamknięcia aplikacji z poziomu widoku aparatu. Zamiast zakończenia działania, gra przenosi nas gracza do menu opcji

7.3 Wnioski

Przeprowadzone testy pozwoliły na wykrycie kilku błędów, oraz potencjalnych kierunków rozwoju aplikacji. W bezpośrednim następstwie zgłoszonych błędów zmianie uległy następujące elementy:

- Został poprawiony kontrast pomiędzy tekstem a tłem w ustawieniach gry. Dzięki temu zabiegowi kontrast z poziomu 1.05 wzrósł do wartości 3,79. Zmiany można zauważyć na zrzucie ekranowym umieszczonym na Rysunek 25:



Rysunek 25 Zmiana kontrastu pomiędzy tłem a tekstem (Pierwszy zrzut pokazuje interfejs przed, drugi po zmianach). Źródło: Opracowanie własne.

- Wyeliminowano problem z oknem informującym o braku sygnału GPS
- Po ustawieniu parametrów gry ładowany jest widok mapy z możliwością wyboru celu
- W aktywności wyświetlającej mapę został ustawiony domyślny widok horyzontalny

8 Podsumowanie i wnioski

Projekt jakiego podjęliśmy się w ramach pisania Naszej pracy okazał się bardzo dużym wyzwaniem. Zademonstrował z jakimi problemami może spotkać się programista podczas wykorzystywania i łączenia wielu różnych składowych. Mimo swojej popularności i ciągłego rozwoju technologii, proste rzeczy nadal przysparzają wielu problemów. Mamy tu na myśli tak z pozoru trywialne rzeczy jak:

- Niedokładność odczytu
- Szum odczytu
- Zużycie baterii
- Zakłócenia
- Złożoność obliczeniowa
- Prawidłowa separacja klas

Wszystkie problemy na jakie natrafiliśmy w trakcie całego projektu były dogłębnie analizowane w celu znalezienia najbardziej optymalnego i skutecznego rozwiązania. Niejednokrotnie prowadziło to do wielogodzinnych testów, które prowadziły Nas w ślepą uliczkę. Okres pisania pracy to ciągłe wzloty i upadki. To czyni ten temat tak fascynującym.

Nauczyliśmy się bardzo wiele o technologii rzeczywistości rozszerzonej. Zrozumieliśmy jakie są jej wady, ale jeszcze lepiej poznaliśmy jej zalety. Na koniec pisania pracy dostrzegamy jak wielkie możliwości daje i jak ogromne perspektywy stoją przez nią. Jesteśmy przekonani, że rzeczywistość rozszerzona wraz z czasem oraz rozwojem technologii będzie coraz mocniej wkraczać w życie codzienne zwykłych użytkowników.

GeocachingAR to prototyp, który miał na celu przybliżyć koncepcję wykorzystania rzeczywistości rozszerzonej na urządzeniach mobilnych. Miał pokazać jakie są możliwości łączenia wielu różnych elementów, które przeciętna osoba niejednokrotnie ma przy sobie. Elementów, które często były tworzone z zupełnie innym przeznaczeniem.

Jak każdy prototyp, również ten, aby osiągnąć maksimum swoich możliwości wymaga dalszego rozwoju. Według Nas, aby móc myśleć o spieniężeniu tej aplikacji należy skupić się na następujących aspektach:

- Rozwinięcie warstwy grywalności
- Rozwinięcie warstwy prezentacji

Warstwa grywalności – aktualnie Nasz prototyp posiada jeden, prosty tryb gry. Jest on w pełni wystarczający, aby zaprezentować koncepcję, którą chcieliśmy przedstawić, jednak zbyt ubogi, aby wciągnąć rzeszę osób w wielogodzinną zabawę. Prawidłowo przemyślana i zbudowana warstwa logiczna gry pozwoli na zainteresowanie wielu osób i będzie miała pozytywny wpływ na ewentualny zarobek.

Warstwa prezentacji – czyli to co widzi gracz. To najtrudniejsze z wyzwań. Prezentacja – czyli to co widzi gracz i z czym wchodzi w interakcję – jest uzależniona od czynników takich jak:

- GPS
- Pomiary
- Rzutowanie obiektów

Czyli te elementy, które przysparzają najwięcej problemów. Możliwe, że, aby osiągnąć komercyjny sukces musi dojść do zmiany pewnych fundamentalnych założeń jakie przyjęliśmy przy budowie Naszego prototypu.

Budowana przez Nas biblioteka okazała się bardzo dużym i fascynującym wyzwaniem. Wymagała od Nas nie tylko dogłębnego zgłębienia tematu rzeczywistości rozszerzonej, ale również prawidłowego podejścia od strony inżynieryjnej. Jedyne zastosowanie się do najważniejszych zasad związanych z zagadnieniami takimi jak: analiza, projektowanie, planowanie mogło zapewnić, że uda Nam się doprowadzić projekt do celu z sukcesem. W kontekście niniejszej pracy magisterskiej, zarówno aplikacja jak i biblioteka odgrywały niemal równorzędne role. Gdyby powstała sama biblioteka to podejście do tematu byłoby zbyt płytkie, a wnioski niewystarczające. Z kolei sama gra charakteryzowałaby się zbyt wąskim spojrzeniem na temat. Generyczność biblioteki niejako zmusiła Nas do szerokiego spojrzenia na temat rzeczywistości rozszerzonej.

Uważamy, że temat ten może być dalej rozwijany, a jego zastosowania ogranicza jedynie ludzka wyobraźnia oraz technologia. Jest to bardzo dobra wiadomość dla rzeczywistości rozszerzonej, ponieważ ludzka wyobraźnia nie zna granic, a technologia stale się rozwija. Z czasem będziemy coraz częściej korzystać z aplikacji tego typu w rzeczach, które teraz wydają się bardzo egzotyczne jak:

- Okulary
- Deski rozdzielcze na szybach aut
- Nauka

9 Bibliografia

[1] European Commission. *Wikipedia.* [Online] [Zacytowano: 01 04 2015.] http://europa.eu/rapid/press-release_IP-14-145_en.html.

[2] Urządzenia Mobilne. *Wikipedia.* [Online] [Zacytowano: 01 04 2015.] http://pl.wikipedia.org/wiki/Urz%C4%85dzenie_mobilne.

[3] Smartfon. *Wikipedia.* [Online] [Zacytowano: 01 04 2015.] http://pl.wikipedia.org/wiki/Smartfon.

[4] Nurski Miron. Pierwszy miliard smartfonów. [Online] [Zacytowano: 01 04 2015.] http://komorkomania.pl/636,w-ubieglym-roku-po-raz-pierwszy-sprzedano-az-miliard-smartfonow.

[5] *International Data Corporation (IDC).* [Online] [Zacytowano: 02 04 2015.] www.idc.com.

[6] Matjaz Mihelj, Domen Novak i Novak Begus. *Virtual Reality Technology and Applications (Intelligent Systems, Control and Automation: Science and Engineering).* 2013. ISBN: 9400769091.

[7] Grigore Burdea i Philippe Coiffet. *Virtual Reality Technology.* 2003. ISBN: 7121007770.

[8] Pojekt Tango. *Google.* [Online] [Zacytowano: 20 04 2015.] https://www.google.com/atap/project-tango/about-project-tango/index.html.

[9] Google Glass. *Google.* [Online] [Zacytowano: 20 04 2015.] https://www.google.com/glass/start/U.

[10] Feature lockheed. *Airforce Technology.* [Online] [Zacytowano: 20 04 2015.] http://www.airforce-technology.com/features/featurelockheed-f-35-successes-programme-setbacks/featurelockheed-f-35-successes-programme-setbacks-3.html.

[11] Freethy Evan. NASA and ODG Bring Augmented Reality Into Space. *AUGMENTED Blog.* [Online] [Zacytowano: 20 04 2015.] http://blog.metaio.com/2015/04/06/nasa-and-odg-bring-augmented-reality-into-space/.

[12] Rohan Mr. Augmented Reality and Virtual Reality Market worth $1.06 Billion by 2018. *Markets and Markets.* [Online] [Zacytowano: 20 04 2015.] http://www.marketsandmarkets.com/PressReleases/augmented-reality-virtual-reality.asp.

[13] Wikipedia ES. [Online] 25 Maj 2015. http://upload.wikimedia.org/wikipedia/commons/4/4f/Hud_on_the_cat.jpg.

[14] THE COMPANY EVENA RELEASES “EYES-ON. [Online] [Zacytowano: 20 04 2015.] http://medicalaugmentedreality.com/2014/02/the-company-evena-releases-eyes-on/.

[15] Team Zesty Blog. [Online] Zesty, 2014. https://blog.zesty.co.uk/wp-content/uploads/2015/03/Augmented-Reality-in-Healthcare-7.jpg.

[16] Violeti Kiran. Virtual View App - Augmented Reality Property App. [Online] 20 Marzec 2014. http://www.realareal.com/wp-content/uploads/2014/03/Picture000118.png.

[17] Ridden Paul. Gizmag | New and Emerging Technology News. [Online] 14 Wrzesień 2013. http://images.gizmag.com/gallery_lrg/ikea-augmented.JPG.

[18] Global Positioning System(GPS) . *Wikipedia.* [Online] [Zacytowano: 25 02 2015.] http://pl.wikipedia.org/wiki/Global_Positioning_System .

[19] Geocaching - History. *Geocaching.* [Online] [Zacytowano: 25 02 2015.] http://www.geocaching.pl/geocaching.php.

[20] Zbigniew Kowal. *komorkomania.pl.* [Online] [Zacytowano: 08 04 2015.] http://zbigniew.kowal.komorkomania.pl/1787,ile-warty-bedzie-rynek-gier-mobilnych-w-2016.

[21] Paterek Marcin. 25 miliardów aplikacji pobranych z Google Play. *dobreprogramy.pl.* [Online] http://www.dobreprogramy.pl/25-miliardow-aplikacji-pobranych-z-Google-Play,News,36416.html.

[22] Stron Domowa. *Linux.* [Online] [Zacytowano: 25 03 2015.] https://www.linux.com/ .

[23] Google about company. *Google.* [Online] [Zacytowano: 25 03 2015.] http://www.google.com/about/company/.

[24] Podział rynku systemów mobilnych. *International Data Corporation (IDC).* [Online] [Zacytowano: 25 03 2015.] http://www.idc.com/prodserv/smartphone-os-market-share.jsp .

[25] Pisarski Michał. Android - historia najpopularniejszego mobilnego systemu operacyjnego. *komputerswiat.pl.* [Online] [Zacytowano: 26 03 2015.] http://www.komputerswiat.pl/artykuly/redakcyjne/2014/06/android---historia-najpopularniejszego-mobilnego-systemu-operacyjnego.aspx.

[26] Android - wersje sytemu. *Wikipedia.* [Online] [Zacytowano: 26 03 2015.] http://pl.wikipedia.org/wiki/Wersje_systemu_Android.

[27] Smartphone OS Market Share. *International Data Corporation.* [Online] http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[28] Top Android versions. [Online] http://www.appbrain.com/stats/top-android-sdk-versions.

[29] Program rozruchowy. *Wikipedia.* [Online] [Zacytowano: 25 03 2015.] http://pl.wikipedia.org/wiki/Program_rozruchowy.

[30] Android. *Wikipedia.* [Online] [Zacytowano: 03 04 2015.] http://pl.wikipedia.org/wiki/Android_SDK.

[31] Paper Machete Games. [Online] Lipiec 2013. http://www.papermachetegames.com/wp-content/uploads/2013/07/jme01.png.

[32] Paper Machete Games. [Online] Lipiec 2013. http://www.papermachetegames.com/wp-content/uploads/2013/07/jme022.png.

[33] Eclipse Fundation. *Wikipedia.* [Online] [Zacytowano: 10 04 2015.] http://pl.wikipedia.org/wiki/Eclipse_Foundation.

[34] Eclipse - strona domowa. [Online] [Zacytowano: 10 04 2015.] http://www.eclipse.org/.

[35] ADT Plugin Release Notes. *Developer Android.* [Online] [Zacytowano: 13 04 2015.] http://developer.android.com/tools/sdk/eclipse-adt.html.

[36] Roman Wantoch-Rekowski. *Android w praktyce – Projektowanie aplikacji.* Warszawa : PWN, 2014. strony 9-14. 9788301178130.

[37] Neuman. [Online] [Zacytowano: 30 04 2015.] http://eff10.internetdsl.tpnet.pl/www/neuman/java/historia.htm.

[38] Netscape Navigator. [Online] [Zacytowano: 30 04 2015.] http://pl.wikipedia.org/wiki/Netscape_Navigator.

[39] Java start. [Online] [Zacytowano: 30 04 2015.] http://javastart.pl/static/wprowadzenie/historia-javy/ .

[40] Oracle - history of Java. [Online] [Zacytowano: 30 04 2015.] http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html.

[41] Kowasz Wojciech. Dobreprogramy - Oracle przejmuje Sun. [Online] [Zacytowano: 30 04 2015.] http://www.dobreprogramy.pl/Oracle-kupuje-Sun-za-74-mld-dolarow,News,10443.html.

[42] Golański Adam. Webhosting - Oracle kupiło Sun Microsystems. Co to oznacza dla świata IT? [Online] [Zacytowano: 30 04 2015.] http://webhosting.pl/Oracle.kupilo.Sun.Microsystems.Co.to.oznacza.dla.swiata.IT.

[43] Java. *Wikipedia.* [Online] [Zacytowano: 30 04 2015.] http://pl.wikipedia.org/wiki/Java.

[44] Tiobe. *TIOBE Index for April 2015.* [Online] [Zacytowano: 30 04 2015.] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

[45] Google - Android Developers. [Online] http://developer.android.com/guide/topics/sensors/sensors_overview.html.

[46] Marek Cieciura Janusz Zacharski. *PODSTAWY PROBABILISTYKI Z PRZYKŁADAMI ZASTOSOWAŃ W INFORMATYCE CZĘŚĆ II STATYSTYKA OPISOWA.* Warszawa : http://cieciura.net/mp/pdf/czesc2.pdf, 2011.

[47] Kim Phil. *Kalman Filter for Beginners: with MATLAB Examples.* brak miejsca : A-JIN Publishing, 2011. 978-1463648350.

[48] Google best pracice. [Online] http://developer.android.com/guide/practices/index.html.

[49] Google Maps Android API. [Online] https://developers.google.com/maps/documentation/android/.

[50] Krug Steve. *Nie każ mi myśleć.* Gliwice : Helion, 2010. 978-83-246-2772-1.

[51] Kasperski Marek i Boguska-Torbicz Anna. *Projektowanie stron WWW.* Gliwice : Helion, 2008. 978-83-246-1291-8.

[52] http://www.nielsen.com/. *MOBILE MILLENNIALS: OVER 85% OF GENERATION Y OWNS SMARTPHONES.* [Online] [Zacytowano: 08 06 2015.] http://www.nielsen.com/us/en/insights/news/2014/mobile-millennials-over-85-percent-of-generation-y-owns-smartphones.html.

[53] http://skillz.com/. *65 Mobile Gaming Stats to Impress Your Friends.* [Online] [Zacytowano: 08http://skillz.com/ 06 2015.] http://skillz.com/blog/2013/03/05/65-mobile-gaming-stats-to-impress-your-friends/.

[54] Java. *Wikipedia.* [Online] [Zacytowano: 30 04 2015.] http://en.wikipedia.org/wiki/Java_%28programming_language%29#/media/File:Java_logo_and_wordmark.svg.

[55] Wikipedia. *Google Maps.* [Online] [Zacytowano: 11 05 2015.] http://pl.wikipedia.org/wiki/Mapy_Google#/media/File:Google_Maps.svg.

10 Spisy

10.1 Ilustracje

10.2 Tabele

10.3 Wykresy

10.4 Listingi