

**AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE**

Wydział Inżynierii Metali i Informatyki Przemysłowej



PRACA DYPLOMOWA INŻYNIERSKA

pt.

„Wirtualny spacer po terenie AGH z wykorzystaniem technologii VR”

Imiona i nazwiska dyplomantów: Paweł Brzoza, Marcin Szumlański

Kierunek studiów: Informatyka Stosowana

Nr albumów: 278884, 279034

Promotor: dr inż. Tomasz Dębiński

Recenzent: dr hab. inż. Łukasz Rauch

Podpisy dyplomantów:

Podpis promotora:

Kraków 2018

„Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie znieksztalcia taki utwór, artystyczne wykonanie, fonogram, videogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej " sądem koleżeńskim "”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.”

Kraków, dnia 13.01.2018r.

Podpisy dyplomantów:

Spis treści:

Wprowadzenie.....	5
Cele i zakres pracy	5
1. Rzeczywistość wirtualna, rozszerzona i mieszana na przykładzie dostępnych rozwiązań	6
1.1. Rzeczywistość wirtualna.....	6
1.2. Rzeczywistość rozszerzona	7
1.3. Rzeczywistość mieszana	10
2. Jak działają okulary VR?	13
2.1. Stereoskopia	13
2.2. Anaglify	13
2.3. Optyka okularów VR	15
2.4. Mankamenty i ich rozwiązania.....	15
3. Wybór wykorzystywanego sprzętu.....	18
3.1. Uzasadnienie	18
3.2. Zestawienie specyfikacji technicznej urządzeń	18
3.3. Opis wybranych okularów VR.....	19
4. Zdjęcia sferyczne – pozyskiwanie i właściwości.....	22
4.1. Zdjęcie sferyczne, media 360	22
4.2. Wykorzystany sprzęt	23
5. Praca z silnikiem graficznym Unity.....	25
5.1. Niezbędne elementy instalacyjne	25
5.2. Poszczególne elementy środowiska	26
5.3. Składniki projektu	27
5.4. Struktura sceny.....	29
5.5. Elementy biblioteki Google VR SDK.....	33
5.6. Implementacja mechaniki przejść i funkcji GazeClick	33
5.8. Export aplikacji na urządzenia z systemem Android	38
6. Aplikacja wykonana przy użyciu VR View w JavaScript.....	40
6.1. Wstęp	40
6.2. Implementacja.....	40
6.2.1. VR View.....	40
6.2.2. Ładowanie nowej zawartości	42
6.2.3. Hotspotty	43
6.3. Hosting.....	44

7. Testowanie aplikacji	46
7.1. Wydajność programów	46
7.2. Porównanie jakościowe.....	50
8. Badania „user experience”.....	52
8.1. Część 1 – pytania otwarte	53
8.2. Część 2 – pytania otwarte	59
8.3. Wnioski wynikające bezpośrednio z badania	62
9. Perspektywy rozwoju i przyszłość wirtualnej/rozszerzonej rzeczywistości	64
9.1. Rozwój wyświetlaczy	64
9.2. Systemy śledzenia wzroku.....	65
9.3. Inne projekty	67
9.4. Metasoczewki.....	68
10. Podsumowanie.....	70
11. Spis rysunków i tabel	71
11.1. Rysunki	71
11.2. Tabele	72
12. Bibliografia	73

Wprowadzenie

Na przestrzeni ostatnich lat trudno nie zauważyc ogromnego postępu w zastosowaniu wirtualnej rzeczywistości (ang. VR - Virtual Reality). Dzięki ułatwieniom i coraz większej dostępności sprzętu do obsługi VR, ludzie zaczynają eksperymentować z prostymi aplikacjami czy filmami pozwalającymi przetestować wirtualny świat. Przedmiotem pracy jest rozpoznanie możliwości dostępnych na rynku, a następnie implementacja aplikacji do wirtualnego „spaceru” po terenie AGH dla platformy Android oraz wersję dostępną w przeglądarkach internetowych. Przeanalizowano różnice pomiędzy różnymi typami wirtualnej rzeczywistości i przedstawiono jak działa omawiana technologia. Wspomniano również o procesie pozyskiwania zdjęć sferycznych, następnie zbudowano dwie aplikacje. Przedstawiono proces tworzenia „gry” w środowisku Unity, jak i poza nim wykorzystując czysty JavaScript, wszystko to wspomagając się SDK (ang. software development kit) od firmy Google. Zaprezentowano wyniki testów aplikacji oraz badania „user experience” na grupie 21 osób. Zostały przedstawione perspektywy rozwoju jakie daje technologia oraz kilka ciekawych projektów, które mają szansę zdobyć popularność w niedalekiej przyszłości. Całość zakończono podsumowaniem, w którym autorzy wyciągnęli wnioski z realizacji pracy.

Cele i zakres pracy

Główym celem było zapoznanie się z technologią VR, następnie rozpoznanie rozwiązań aktualnie dostępnych na rynku i wybranie możliwie niskobudżetowego zestawu, który w pełni umożliwia korzystanie z przedstawianej technologii. Pozwoliło to wybrać odpowiedni sprzęt oraz oprogramowanie do tworzenia wirtualnej rzeczywistości po czym, stworzono aplikacje.

Program występuje w 2 wersjach. Pierwsza, zaimplementowana za pomocą silnika graficznego Unity, a druga przy pomocy języka skryptowego JavaScript, aby zbadać możliwości, uzyskać jak najlepszy efekt końcowy i zdecydować w oparciu o przyjęte kryteria, która technologia jest odpowiednia.

1. Rzeczywistość wirtualna, rozszerzona i mieszana na przykładzie dostępnych rozwiązań

1.1. Rzeczywistość wirtualna

Wszystkie technologie „modyfikujące” rzeczywistość zmieniają w pewnym sensie sposób w jakim ją postrzegamy, jednakże rzeczywistość wirtualna (ang. VR – Virtual Reality) całkowicie zmienia środowisko, które nas otacza. Merriam-Webster definiuje rzeczywistość wirtualną[1] jako sztuczne środowisko, które jest odbierane za pomocą takich bodźców jak wzrok czy słuch. Jest dostarczone przez komputer w którym akcje użytkownika częściowo determinują, co stanie się w tym otoczeniu.

To, w jaki sposób użytkownik wchodzi w interakcję z otoczeniem zależy od platformy której używa. Niektóre zestawy VR zostały zaprojektowane do używania siedząc w jednym miejscu i ruszając się za pomocą kontrolera, tak jak np. w grach komputerowych. Różnicą jest tutaj jednak to, że ekran jest powiązany z głową użytkownika i zasłania jego dużą część pola widzenia „zanurzając” w wirtualnym 360-cio stopniowym świecie.

Oculus Rift jest dobrze znany jako jeden z najbardziej popularnych zestawów VR aktualnie dostępnych masowo na rynku, ale wymaga zarówno komputera (do którego podłączone są gogle) jak i osobnego kontrolera do działania. Efektywnie, jest to dający złudzenie zanurzenia się w rzeczywistości wirtualnej ekran, zakładany na głowę. Podobnie działa PlayStation VR oraz HTC Vive – są to zbliżone technologicznie rozwiązania.

Z drugiej strony są np. Google Cardboard i Samsung Gear VR (jakościowo solidniejszy), które oferują podobne doświadczenia – ogólnie niższej jakości, ale również w niższej cenie. Plusem tych urządzeń jest to, że nie muszą być przewodowo podłączone do komputera. Zamiast tego do stworzenia doświadczenia VR używają smartfona podłączonego do gogli.

Ze względu na prostotę i niski koszt zdecydowano się wybrać tą technologię do stworzenia projektu.

Istnieją również bardziej rozbudowane technologie VR, tworzące hiperrealistyczne środowisko, które pozwala na fizyczne poruszanie się i manipulację - korzystają z tego np. wirtualne parki rozrywki (Rys. 1).



Rysunek 1 - park rozrywki z VR[2]

W takim wirtualnym świecie, użytkownik może fizycznie poruszać się po przestrzeni wokół nas, z atrapami obiektów. Użytkownik sam jest niejako kontrolerem i może zanurzyć się całkowicie w wyrenderowanym świecie. Często w takim miejscu można natknąć się również na efekty takie jak np. sztuczny deszcz, ciepło/zimno, aby gra była jeszcze bardziej interaktywna i immersyjna. Jak nietrudno się domyśleć rozwój technologii VR przyczynił się do wzrostu popularności tego typu gier.

Najprościej myśleć o VR jako o całkowicie odrębnym i sztucznym świecie zaprojektowanym by zmieniać rzeczywistość i „pochłaniać” w niej użytkownika. Nic nie jest prawdziwe, wszystko jest wirtualne.

1.2. Rzeczywistość rozszerzona

Rzeczywistość rozszerzona[3] (ang. AR – Augmented Reality) czerpie z faktycznej, istniejącej rzeczywistości i zmienia pewne jej aspekty przez obiektyw smartfona, parę okularów czy wspomniane google. Z rzeczywistością rozszerzoną użytkownik zawsze widzi co jest przed nim, ale z dodaną „wirtualną otoczką”. Merriam-Webster definiuje rzeczywistość rozszerzoną jako uatrakcyjnioną wersję rzeczywistości stworzoną przy użyciu technologii by nakładać cyfrowe informacje na obraz obserwowany przez urządzenie.

Rzeczywistość rozszerzona może przyjąć wiele form, ale jej najbardziej powszechne użycie jest w smartfonach. Najbardziej znaną obecnie aplikacją jest gra Pokemon GO! (Rys. 2).



Rysunek 2 - aplikacja na smartphone'y o nazwie „Pokemon Go!”[4]

Istnieją również aplikacje, które na bieżąco, po skierowaniu kamery w odpowiednie miejsce pokazują informacje o np. pobliskiej restauracji itp. Wszystkie te aplikacje używają aparatu smartfona, aby pokazywać wygląd na żywo tego co jest wokół, ale nakładają informacje na wyświetlacz. AR znajdziemy w wielu innych aplikacjach na smartfony, np. Tłumacz Google umożliwia bardzo praktyczną funkcjonalność (Rys. 3). Pozwala po skierowaniu kamery na tekst w jednym języku przetłumaczyć i zamienić go na inny. Dzieje się to automatycznie, bądź do wyboru jest druga opcja zrobienia zdjęcia i zaznaczenia obszaru, który ma zostać przetłumaczony.



Rysunek 3 - zrzut ekranu z aplikacji Tłumacz Google, przedstawiający możliwości rozszerzonej rzeczywistości

AR jest dostępna również na Google Glass. Urządzenie było jednym z pierwszych, które korzystało z rozszerzonej rzeczywistości poza smartfonami, ale stało się źródłem kontrowersji z powodu obaw dotyczących prywatności, wysokiej ceny, małej ilości istotnych funkcjonalności. Jednakże porażka Google Glass nie przeszkadzała innym firmom w próbach popularyzacji tej pasywnej metody interakcji (względem chociażby VR) i tchnięcia nowego życia w AR – są to np. LAFORGE Optical albo Optinvent. Istnieje wiele zwolenników, ale nie brakuje też przeciwników, którzy uważają, że urządzenia takie jak Google Glass nigdy nie osiągną sukcesu. Od roku 2015 na rynku pojawiła się nowa odsłona w wersji Enterprise Edition (Rys. 4), którą zainteresowało się już ponad 50 firm. Firma Google widzi w tym potencjał, a firmy zaczynają realizować zyski ze wzrostu efektywności pracowników czy ze wzrostu produkcji[5].



Rysunek 4 - okulary Google Glass[6]

Ogólnie, o rozszerzonej rzeczywistości można myśleć jak o nowej warstwie nałożonej na istniejącą rzeczywistość, ale nie przemieszanej z nią. Okulary być może zaliczają się do tej kategorii, ale to smartfony znajdują w AR największe zastosowanie – jako nakładka na pole widzenia smartfona, bez interakcji jako część większego środowiska.

1.3. Rzeczywistość mieszana

Rzeczywistość mieszana[7] (ang. MR – Mixed Reality) zabiera AR na nowy poziom i jest w zasadzie tym, czym wszyscy początkowo chcieli lub oczekiwali od AR. Zamiast być tylko pewną warstwą nałożoną na świat, rzeczywistość mieszana odnosi się do połączenia cyfrowo renderowanych obiektów w świat rzeczywisty.

Techopedia[8] definiuje MR jako typ hybrydowego systemu, na który składają się zarówno fizyczne oraz wirtualne elementy. Wielu ekspertów ocenia MR jako podziałkę pomiędzy całkowicie fizycznym środowiskiem bez wirtualnych elementów, a całkowicie wirtualnym. W takim połączonym środowisku fizyczne i cyfrowo obiekty koegzystują i wchodzą w interakcję w czasie rzeczywistym.

O ile omawiane pojęcie ma prawo bytu na smartfonach, to okulary do VR bardziej wpasowują się w tę definicję, ponieważ są zwyczajnie immersywne. Jednymi z bardziej znanych gogli MR są, Microsoft HoloLens (Rys. 5), Meta 2

(Rys. 6) oraz Magic Leap One (choć nie są jeszcze szeroko dostępne na rynku, lub są jeszcze na etapie wdrażania; Rys. 7).



Rysunek 5 - Microsoft HoloLens[9]



Rysunek 6 - Leap One[10]



Rysunek 7 - Meta 2[11]

Potrafią skanować pomieszczenie i przede wszystkim „rozumieć” przestrzeń w której się znajdują, dzięki czemu mogą precyzyjniełączyć cyfrowe obiekty do istniejącego, rzeczywistego środowiska. Z wyświetlonymi obrazami przez okulary można wchodzić w interakcję, zupełnie tak, jakby były prawdziwe. Cyfrowy świat zostaje połączony z rzeczywistym, a poprzez gogle można wchodzić w interakcje i oddziaływać.

Technologie zmieniające rzeczywistość mogą być czasami dezorientujące, ponieważ wszystkie zazębają się ze sobą. Z pewnością rzeczywistość mieszana i rozszerzona dzielą ze sobą bardzo wiele i mają podobne przypadki użycia. W pewnym sensie o MR można myśleć jako o podzbiorze bardzo zaawansowanej rzeczywistości rozszerzonej, gdyż MR udoskonala rzeczywistość, ale w bardziej gruntowny i integracyjny sposób. AR dodaje cyfrową warstwę do rzeczywistości, którą widzimy. MR łączy wszystko bardziej gładko i zapewnia więcej interakcji dla użytkownika.

2. Jak działają okulary VR?

2.1. Stereoskopia

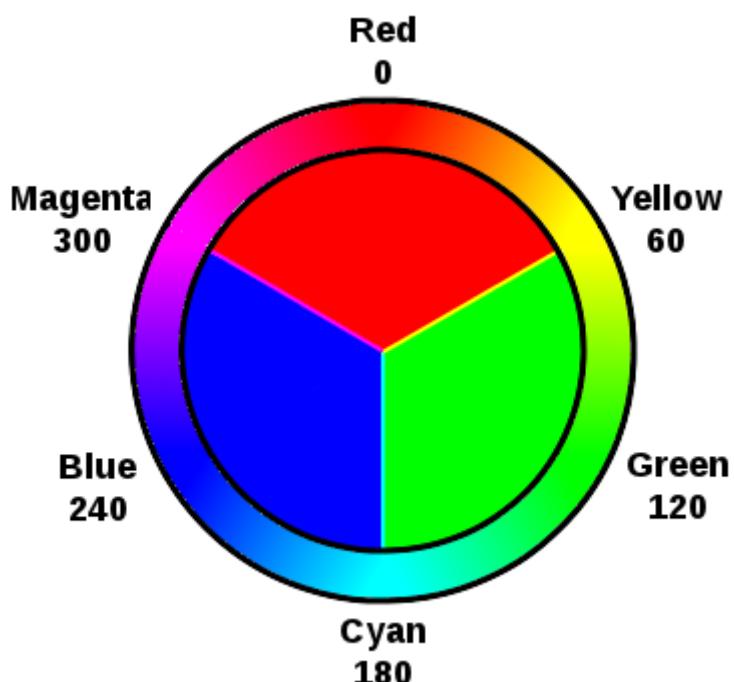
Aby rozpocząć omawianie mechanizmu działania nowoczesnych okularów, dobrze zacząć od stereoskopii. Stereoskopia[12] to mechanizm tworzenia złudzenia trójwymiarowego za pomocą dwóch ujęć tego samego obiektu, różniących się nieznacznie przesunięciem (dylatacją) dając wrażenie widzenia stereoskopowego. Co to widzenie stereoskopowe?

Cytując wydawnictwo naukowe PWN[13]: „widzenie stereoskopowe, widzenie głębi przestrzeni, polegające na postrzeganiu trójwymiarowości przedmiotów i ich przestrzennego rozmieszczenia”. W rzeczywistości nasze oczy nie widzą tego samego obrazu tylko dwa nieznacznie przesunięte i mózg jest w stanie stworzyć z nich jeden z efektem głębi – tzw. obraz cyklopowy.

Klasycznie obraz stereoskopowy, czyli stereogram[14] to para dwóch zdjęć 2D (po 1 na każde oko) przy czym stereoskopowe zdjęcie nosi nazwę stereogramu i do obserwacji efektu trójwymiarowego wykorzystuje się okulary zwane stereoskopem. Zasadniczo obraz stereoskopowy to para dwóch zdjęć dwuwymiarowych tej samej sceny (po jednym na każde oko) przy czym perspektywa jednego jest nieznacznie przesunięta względem drugiego celem uzyskania złudzenia ww. widzenia stereoskopowego.

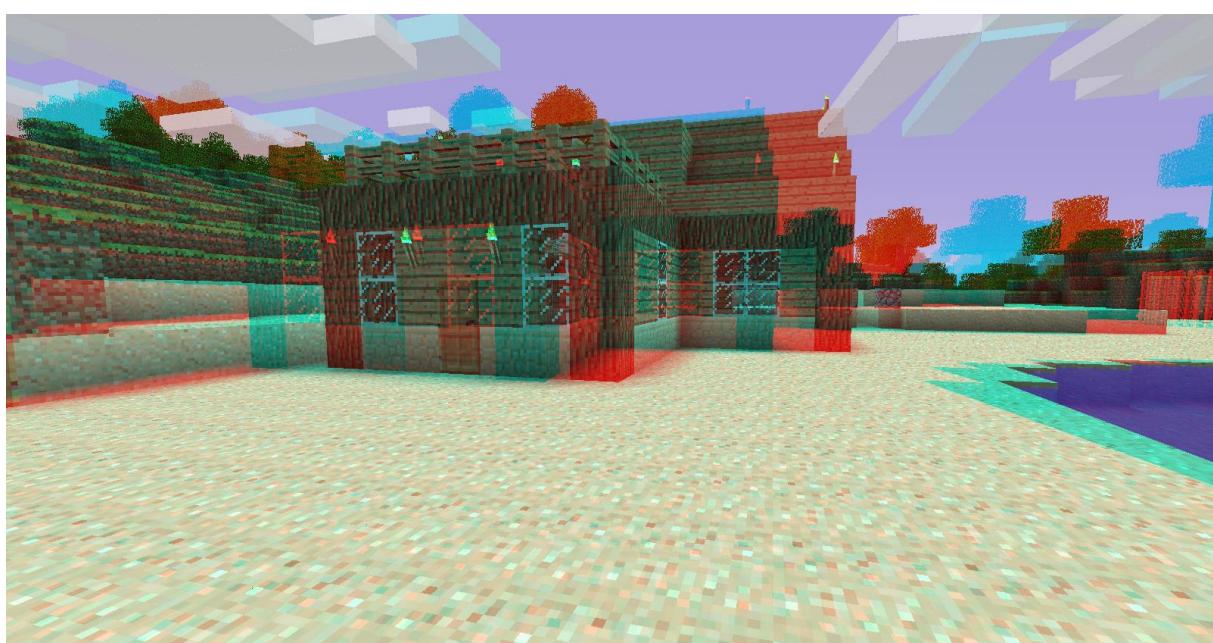
2.2. Anaglify

Istnieją specjalne stereogramy tzw. anaglify[15] szczególnie popularne w przeszłości, których stworzenie polega na nałożeniu na siebie obrazu w różnych barwach (Rys. 8-9) z poziomym przesunięciem. Zazwyczaj jako 2 kolory wybiera się te, które leżą po przeciwnych stronach względem siebie na kole barw. Najbardziej typową parą są szkła cyjan-czerwień lub cyjanowe szkła, wygaszające w dużym stopniu światło czerwone.



Rysunek 8 - koło kolorów RGB[16]

Każdy z dwóch obrazów dociera do odpowiedniego oka, ukazując stereograficzny obraz. Kora wzrokowa łączy dwa obrazy w trójwymiarową scenę. Zaletą tego rozwiązania jest cena, ale posiada również sporo wad, przede wszystkim obraz jest w zaburzonych kolorach względem oryginalnych.



Rysunek 9 - tryb anaglifowy w grze Minecraft[17]

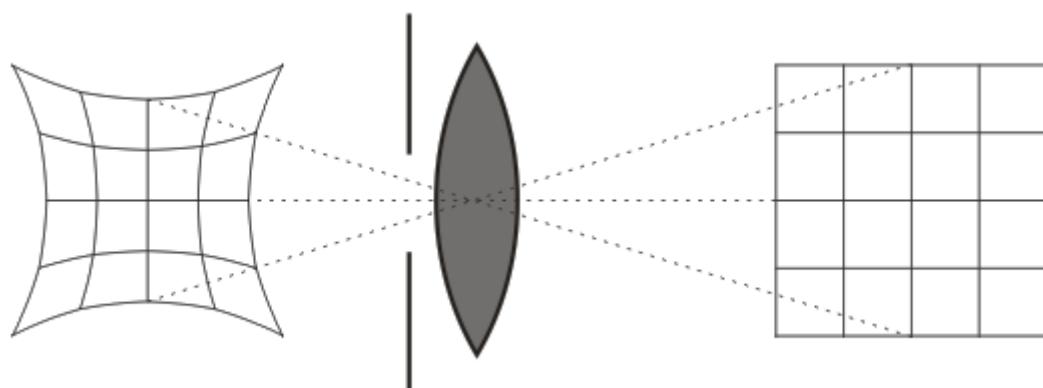
2.3. Optyka okularów VR

Okulary VR korzystają z podobnego mechanizmu. Składają się z 2 soczewek, które odpowiednio zakrzywiają dwa lekko przesunięte względem siebie obrazy tej samej sceny. W taki sposób można, stworzyć efekt głębi. Jednak jeśli obraz jest zbyt blisko oczu, wtedy soczewka oka nie może wystarczająco dopasować się i obraz na siatkówce powstaje niewyraźny – punkt ogniskowania przesuwa się za siatkówkę. Wyświetlacz w goglach jest w odległości kilku cm od oczu, dlatego soczewki dodatkowo zakrzywiają światło, które wpada do oka tak, aby obraz na siatkówce pozostał wyraźny i możliwe było złudzenie widzenia go dalej niż jest w rzeczywistości - mapując zbliżony obraz na szersze pole widzenia.

2.4. Mankamenty i ich rozwiązania

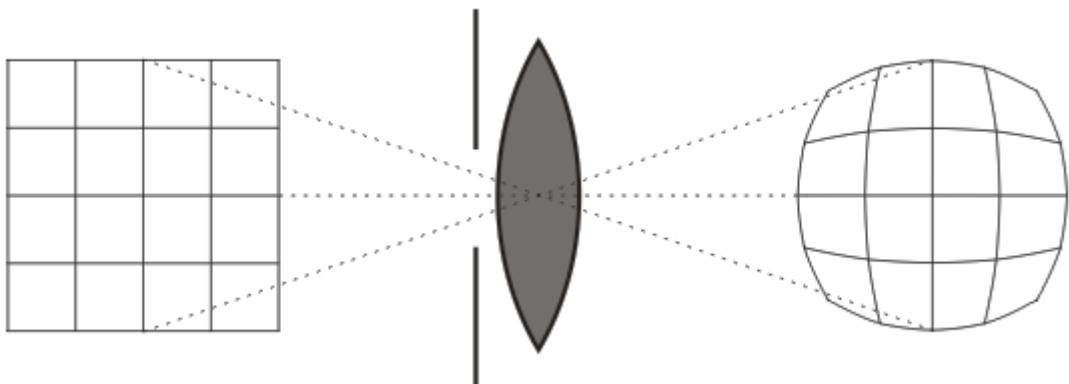
Nie jest to jednak rozwiązanie bez wad. Im większe pole widzenia tym powstają większe zniekształcenia obrazu tzw. pincushion effect[18].

Po prawej obraz z wyświetlacza, po lewej obraz widziany przez obserwatora po przepuszczeniu go przez soczewkę.

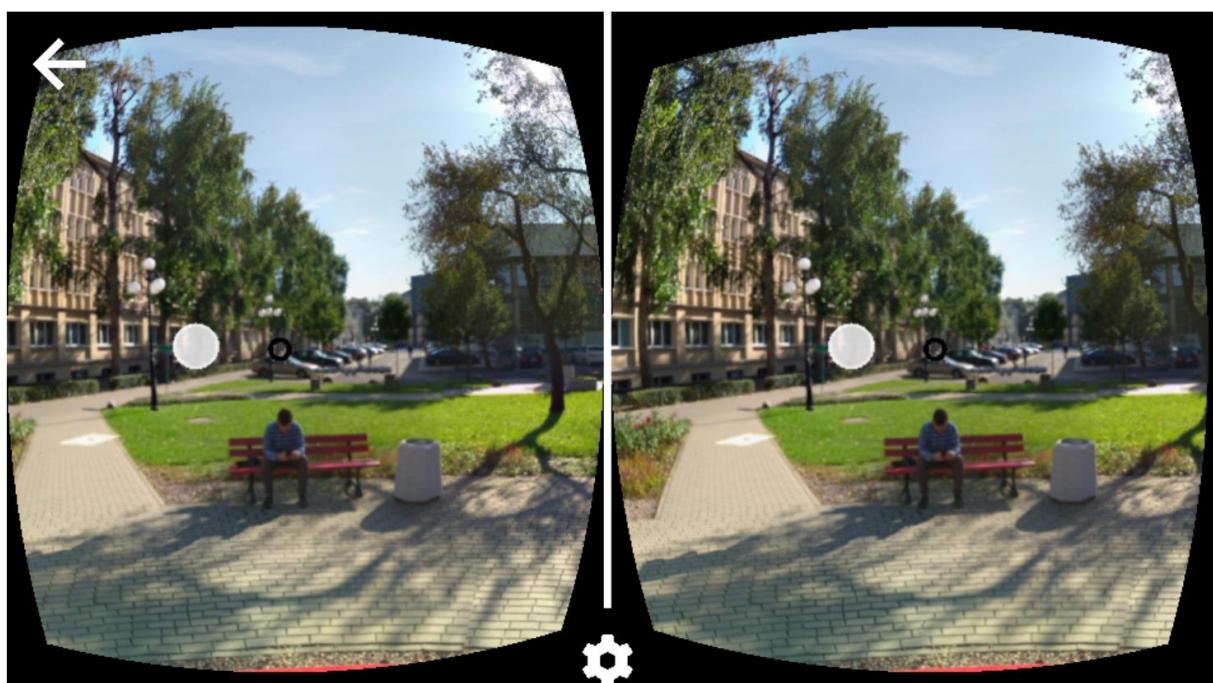


Rysunek 10 - skupienie soczewek 1 [18]

Rozwiązuje się to programowo i na obraz na wyświetlaczu nakłada się tzw. barrel distortion, dzięki czemu finalny obraz wygląda neutralnie.



Rysunek 11 - skupienie soczewek 2 [18]



Rysunek 12 - barrel distortion na przykładzie zaimplementowanej aplikacji webowej

Wskutek używania znieksztalcających soczewek występuje również tzw. aberracja chromatyczna. Cytując wydawnictwo naukowe PWN[19]: „aberracja chromatyczna, chromatyzm, wada soczewkowych układów opt. spowodowana zależnością współczynnika załamania światła materiału soczewek od długości fali świetlnej (dyspersja światła); przejawia się w powstawaniu na obrazie barwnych obwódek (zmniejszających jego ostrość) wskutek rozszczepienia światła przy załamaniu w materiale soczewki.”

Problem ten poza zmniejszaniem immersyjności doświadczenia, jest również główną przyczyną mdłości i bólu głowy podczas użytkowania aplikacji VR przez niektóre osoby. Chromatyczną aberrację usuwa się również sprzętowo (a raczej ogranicza).

Zdjęcie przed (góra) i po (dół) korekcji chromatycznej aberracji:



Rysunek 13 - naprawa chromatycznej aberracji [20]

3. Wybór wykorzystywanego sprzętu

3.1. Uzasadnienie

Podczas rozważań nad wyborem urządzeń do obsługi okularów oraz samych gogli, brano pod uwagę kilka czynników decydujących. W przypadku smartphone'ów, warto zaznaczyć, iż mówimy tu o wyborze do samej obsługi technologii VR, a nie do robienia zdjęć (ten temat zarezerwowany został na kolejny rozdział). Decydujące czynniki to: rozdzielcość i ilość kolorów wyświetlacza, przekątna ekranu, waga, system operacyjny, a dla okularów największe znaczenie miały: cena, jakość wykonania, optyka, wygoda, możliwości regulacji.

3.2. Zestawienie specyfikacji technicznej urządzeń

Analizując urządzenia odpowiedzialne za renderowanie obrazu, na wstępnie zostały odrzucone wysokobudżetowe rozwiązania takie jak Oculus Rift czy HTC Vive, czyli technologie zintegrowane. Jakość doświadczeń z pewnością byłaby większa, natomiast takie rozwiązania są kosztowne. Dlatego zdecydowano się na wybór zalewających masowo rynek rozwiązań, okularów w których możliwe jest umieszczenie smartphone'a i uruchamianie aplikacji. Mając do dyspozycji trzy modele telefonów sprawdzono ich charakterystykę sprzętową (Tab. 1).

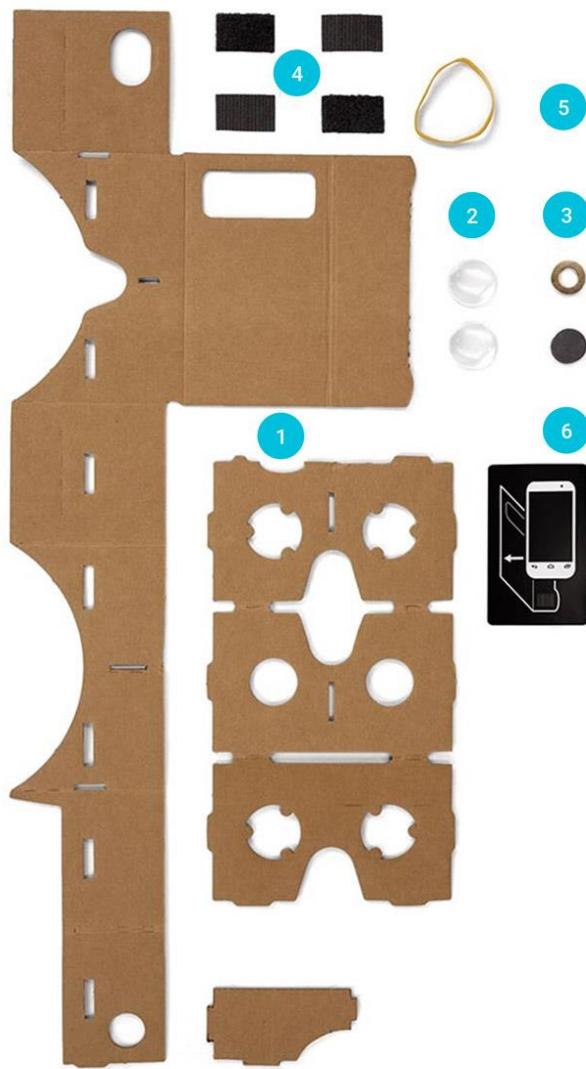
Tabela 1 - przedstawia porównanie istotnych parametrów sprzętowych

SMARTPHONE	XIAOMI REDMI NOTE 4X	SONY XPERIA Z1	HTC ONE M7
ROZDZIELCZOŚĆ WYŚWIETLACZA	1080 x 1920 [px]	1080 x 1920 [px]	1080 x 1920 [px]
RODZAJ WYŚWIETLACZA	Kolorowy / IPS TFT 16M kolorów	Kolorowy / IPS TFT 16M kolorów	Kolorowy / Super LCD 16M kolorów
PRZEKĄTNIA EKRANU	5,50"	5,00"	4,70"
SYSTEM OPERACYJNY	Android 7.0	Android 5.1.1	Android 4.1.2
WAGA	165 [g]	170 [g]	143 [g]
PROCESOR GRAFICZNY	Adreno 506	Adreno 330	Adreno 320

Na podstawie danych przedstawionych w tabeli uznano, że najlepszym z dostępnych urządzeń okazał się Xiaomi Redmi Note 4X. Wygrał ze swoimi rywalami przede wszystkim przekątną ekranu oraz procesorem graficznym. Od Sony Xperia Z1 lepszy miał również system operacyjny i był lżejszy (co przy zestawie VR jest wyjątkowo ważne z uwagi na wygodę późniejszego użytkowania). Warto zauważyć, że HTC ONE M7 nie miałby nawet możliwości obsługi aplikacji Cardboard przez wersję androida niższą od 4.4, dlatego został wyeliminowany.

3.3. Opis wybranych okularów VR

Na potrzeby późniejszych badań wybrano dwa modele gogli. Pierwszy rodzaj to najprostsza możliwa forma prezentowanych okularów – Google Cardboard (Rys. 14). Charakteryzuje się prostą budową, dlatego uzyskanie tego modelu jest niezwykle łatwe, ponieważ można je zbudować samemu. Na stronie producenta można znaleźć instrukcję złożenia, a wszystkie części, potrzebne do konstrukcji można kupić w Internecie lub w sklepie z narzędziami i są to kolejno: (1) tektura, (2) soczewki, (3) magnesy, (4) rzepy, (5) gumka recepturka, (6) chip NFC (ang. Near Field Communication).



Rysunek 14 - Google Cardboard [21]

Drugie, to gogle chińskiej firmy FiiT VR 2S. Przy wyborze tych okularów, kierowano się pozytywnymi opiniami oraz kilkoma recenzjami znalezionymi w Internecie. Przedstawiony zestaw składa się z gogli VR, wykręcanych(regulowanych) soczewek, pasków na głowę oraz szmatki do czyszczenia soczewek. Dodatkowo w soczewkach regulowany jest ich rozstaw, co pozwala na dokładne dopasowanie do wyświetlanego obrazu oraz wyregulowanie ostrości poprzez wkręcanie, bądź wykręcanie soczewek. Plastikowa część jest bardzo solidna i lekka. Co z pewnością odróżnia FiiT VR od wersji kartonowej to wygoda i możliwość przymocowania headset'u do głowy za pomocą elastycznych pasków. Wyściółka, która znajduje się pomiędzy głową, a plastikiem jest bardzo miękka i przyjemna w dotyku, co ważne przepuszcza powietrze dlatego nie ma problemu z parowaniem soczewek. Montaż telefonu jest bardzo szybki za sprawą prostoty skonstruowanego zamocowania.



Rysunek 15 - okulary FiiT VR 2S [22]

4. Zdjęcia sferyczne – pozyskiwanie i właściwości

4.1. Zdjęcie sferyczne, media 360

Zdjęcie sferyczne (inaczej fotosfera, zdjęcie 360) – zdjęcie uchwytyjące kompletną scenę jako jeden obraz, która to scena oglądana jest obracając się wokół jednego centralnego punktu. Zazwyczaj tworzona na zasadzie łączenia ze sobą wielu zdjęć wykonanych w rotacji 360 stopni lub używając specjalnych kamer 360. Zdjęcie może być również generowane całkowicie komputerowo, a także można łączyć ze sobą komputerowe efekty i fotografie. Media 360 (zdjęcia i filmy) można tworzyć w formacie mono lub stereo. Muszą być one przechowywane w odpowiednim formacie np. cubic lub equirectangular panorama. Format equirectangular jest obecnie najszerzej wspierany - w surowej formie zdjęcie jest płaskie, a efekt wirtualnej rzeczywistości uzyskuje się odpowiednio je przekształcając. Przykładowe wykonane zdjęcie (format equirectangular) przedstawiono na rysunku (Rys. 16).



Rysunek 16 - zdjęcie sferyczne w formacie equirectangular

Istnieją również inne formaty tzw. niepełne, które nie pokrywają całej sfery, jednak w pracy nie były wykorzystywane.

4.2. Wykorzystany sprzęt

Do zrobienia zdjęć wykorzystano aplikację Google Street View[23], statyw FIRST C-3570 (Rys. 17) oraz telefon Xiaomi Redmi Note X4 (Rys. 18). Aplikacja Google Street View to aplikacja od firmy Google, która pozwala m.in. na łatwe robienie zdjęć sferycznych oraz udostępnianie ich. Cały proces jest prosty, aplikacja przy użyciu żyroskopu w telefonie i po określaniu lokalizacji naznacza punkt na kamerze w rzeczywistości rozszerzonej. Po skupieniu się na nim w czasie 2-3 sekund w bezruchu automatycznie robi zdjęcie, które następnie nakładane jest na wygenerowaną grafikę 3D, nałożoną na obraz z kamery. Następnie zdjęcie to staje się punktem odniesienia i wokół niego pojawiają się kolejne punkty, które po skupieniu się na nich robią kolejne zdjęcia. W efekcie, jeżeli użytkownik porusza się w osiach mając kamerę w jednym miejscu może zapełnić wirtualną grafikę 3D, 360 stopni, mozaiką zdjęć. Kiedy zostaną wykonane wszystkie zdjęcia (w przypadku użytego aparatu kilkudziesiąt) poddawane są one scaleniu do zdjęcia sferycznego.

W omawianym przypadku z racji niskiego budżetu, wykorzystano przedstawiany we wcześniejszym rozdziale telefon Xiaomi Redmi Note X4 z aparatem o specyfikacji technicznej:

- matryca 13 Mpix
- ogniskowa 4mm oraz z przysłoną f/2.0.



Rysunek 17 - statyw stabilizujący aparat[24]



Rysunek 18 - smartphone wykorzystywany do robienia zdjęć 360[25]

Statyw bardzo ułatwił osadzenie punktu orientacji w jednym miejscu, dzięki czemu zdjęcia były stabilne i aplikacja lepiej je scalała. Gdyby wybrać potencjalne ulepszenie całego procesu bez wątpienia telefon można wyposażyc np. w obiektyw szerokokątny (charakteryzujący się dużym kątem widzenia przy jak najmniejszym zniekształceniu geometrycznym w całym kadrze), który wpłynąłby na mniejszą liczbę zdjęć potrzebnych do wykonania całej sfery, a co za tym idzie przyspieszyłby proces jak również poprawiłby jakość – mniej zdjęć do scalania to mniej potencjalnych zniekształceń.

5. Praca z silnikiem graficznym Unity

Stworzenie aplikacji wymagało przejścia przez określone zadania, rozpoczynając od najważniejszych to:

- 1) Stworzenie projektu
- 2) Dodanie wymaganych skryptów od GoogleVR SDK
- 3) Utworzenie wirtualnego statywów z kamerą
- 4) Utworzenie 7 sfer oraz teksturowanie ich
- 5) Wprowadzenie odpowiednich obiektów w postaci strzałek umożliwiające przejścia do innych sfer
- 6) Umieszczenie ikon z informacją o obiektach
- 7) Utworzenie skryptu odpowiadającego za mechanikę przejść

5.1. Niezbędne elementy instalacyjne

Unity

Zintegrowane środowisko programistyczne (ang. IDE, Integrated, Development Environment), jakim jest Unity[26], pozwala tworzyć nie tylko gry komputerowe (z czym może być kojarzone), ale również pozwala na kreowanie wizualizacji, animacji czy materiałów interaktywnych. Wszystko to może zostać stworzone na płaszczyźnie dwu lub trójwymiarowej. W przedstawianym środowisku, jak w każdym innym profesjonalnym programie tego typu, można tworzyć, edytować, testować, bądź „konserwować” kod programu. W tym wypadku są to głównie skrypty pisane w języku programowania C#. Jedną z wielu zalet silnika Unity jest jego elastyczność. Odpowiednio skonfigurowany projekt pozwala na jego eksport na ponad 10 różnych platform:

- urządzenia mobilne	-	Android, IOS, Windows Phone
- technologie webowe	-	WebGL, Facebook
- komputery z systemami	-	Windows, Mac, Linux
- konsole do gier	-	PlayStation 4, PS Vita, Xbox One

Do implementacji zainstalowano najnowszą dostępną bezpłatną wersję personalną tj. 2017.3.0f3 z zaznaczonym komponentem „Android Build Support”.

Android Studio SDK

W celu prawidłowego działania środowiska Unity, należało zainstalować również środowisko Android Studio. Umożliwia ono między innymi przetestowanie aplikacji w wirtualnym urządzeniu, natomiast w projekcie skupiono się przede wszystkim na instalacji Android SDK w najniższej możliwej wersji Android 4.4 (KitKat) z API Level 19. Ta wersja i wyższe współpracują z GoogleVR SDK, stąd ten wybór. W łatwy sposób umożliwia to wbudowany w środowisko SDK Manager.

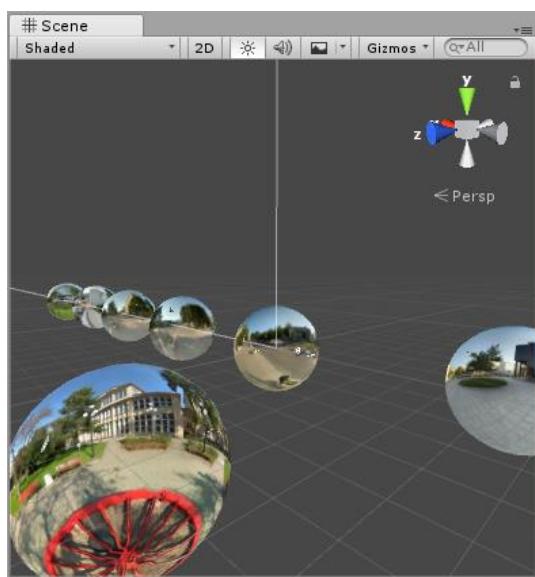
Java Development Kit

Zainstalowano również najnowsze biblioteki Java SE Development Kit w wersji 8u151. Co umożliwiło skompilowanie programu oraz jego późniejszą budowę/eksport do pliku .apk w celu instalacji na smartphonach z systemem Android.

Zarówno Android SDK jak i JDK, należało skonfigurować w ustawieniach Unity.

5.2. Poszczególne elementy środowiska

Scena (ang. scene; Rys.19) – pojedyncza przestrzeń dla obiektów, w której mogą na siebie oddziaływać. Jedna aplikacja może zawierać wiele scen, a w momencie przełączania się między nimi wymaga ponownego wyrenderowania.



Rysunek 19 - scena w środowisku Unity 3D

Obiekt gry (ang. game object) - każdy element, który występuje w scenie.

Komponenty (ang. components) – stanowią funkcjonalną część aplikacji tworzonych w Unity, zawiera je każdy obiekt gry, a te definiują np.

- pozycję i przekształcenie (ang. Transform)
- szczegóły wyświetlania takie jak tekstura (ang. Renderer)
- interakcję z innymi obiektami (ang. Collider)
- skrypty opisujące jego zachowania (ang. Script).

Ciekawą i wygodną rzeczą jest to, że każdy obiekt jest równocześnie kontenerem i nic nie stoi na przeszkodzie aby zagnieżdżała kolejne.

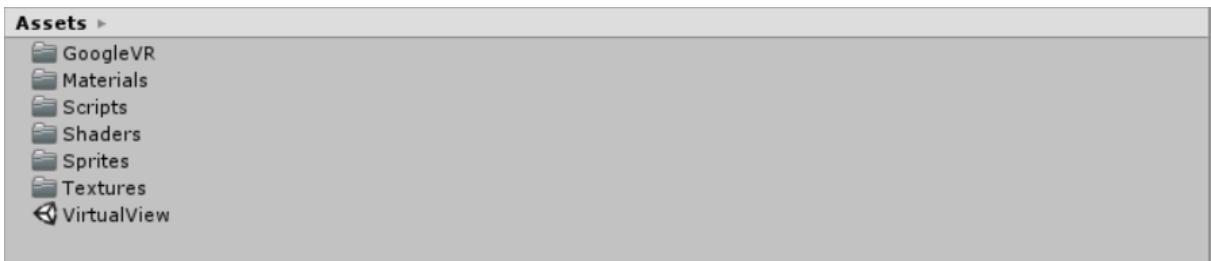
Bardzo ważnym typem obiektu jest kamera (ang. camera). Określa jaka część sceny 3D ma być w obecnej chwili wyświetlana. Pozycja kamery może się zmieniać w trakcie działania aplikacji, co może odpowiadać np. poruszaniu się gracza w przestrzeni sceny lub obracaniu przez niego głową.

Częścią aplikacji odpowiadającą za dynamikę oraz zachowania obiektów gry są skrypty. Unity pozwala uruchamiać skrypty napisane w językach C#, UnityScript (które jest składniowo podobne do JavaScript) oraz Boo. Zastosowanie języka C# jest bardzo praktyczne z uwagi na dwie istotne kwestie: powszechność i kompatybilność.

Za łatwiejsze wyszukiwanie obiektów w programie odpowiedzialne są krótkie ciągi znaków zwane tagami.

5.3. Składniki projektu

Tworzenie projektu rozpoczęło się od stworzenia odpowiedniej struktury (Rys. 20), czyli podziału na foldery zawierające odpowiednie moduły, z których powstał projekt. Przedstawiono kawałki zrzutów ekranu ze środowiska Unity.



Rysunek 20 - struktura folderów

Wszystkie składniki projektu znajdują się w folderze Assets (co w dosłownym tłumaczeniu w języku polskim może oznaczać wartości użyteczne, dla wygody stosowania w dalszej części pracy używana będzie angielska wersja). W jego skład wchodzą poszczególne pod foldery zawierające kolejno:

GoogleVR – zimportowane elementy z biblioteki: GoogleVRForUnity_1.110.0.unitypackage[27], będące dodatkowym wsparciem dla projektu, ważne skrypty to: GvrEditorEmulator, GvrControllerMain, GvrEventSystem i GvrReticlePointer, które dokładniej zostaną opisane niżej.

Materials – stworzone „materiały”, czyli elementy graficzne kontrolujące modele 3D, zawierające w sobie specjalne Shadery.

Scripts – skrypty, odpowiadające głównie za mechanikę programu

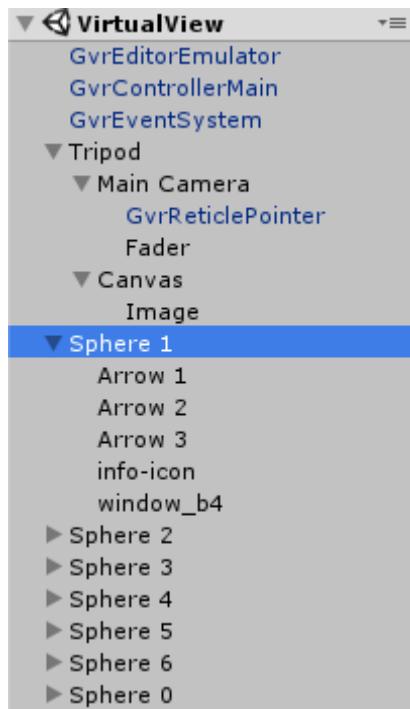
Shaders – folder zawierający tzw. Shadery[28], czyli kod napisany w języku HLSL (High-Level Shading Language)[29] lub CG (C for Graphics), które są do siebie bardzo zbliżone. Służą między innymi do opisu sposobu teksturowania modeli.

Sprites – elementy, rozszerzające rzeczywistość takie jak np. strzałki wskazujące kierunek (w przedstawianym projekcie pierwotnej wersji wykorzystywane modele ptaków co okazało się nie intuicyjne), ikony informacyjne, okienka informacji o budynkach oraz wskaźniki w postaci okręgów ułatwiające określenie czasu w momencie wykonywania tzw. GazeClick (dokładnie opisany w osobnym podrozdziale)

Tekstury – tekstury, czyli w omawianym przykładzie zdjęcia sferyczne

VirtualView – scena zawierająca obiekty

5.4. Struktura sceny

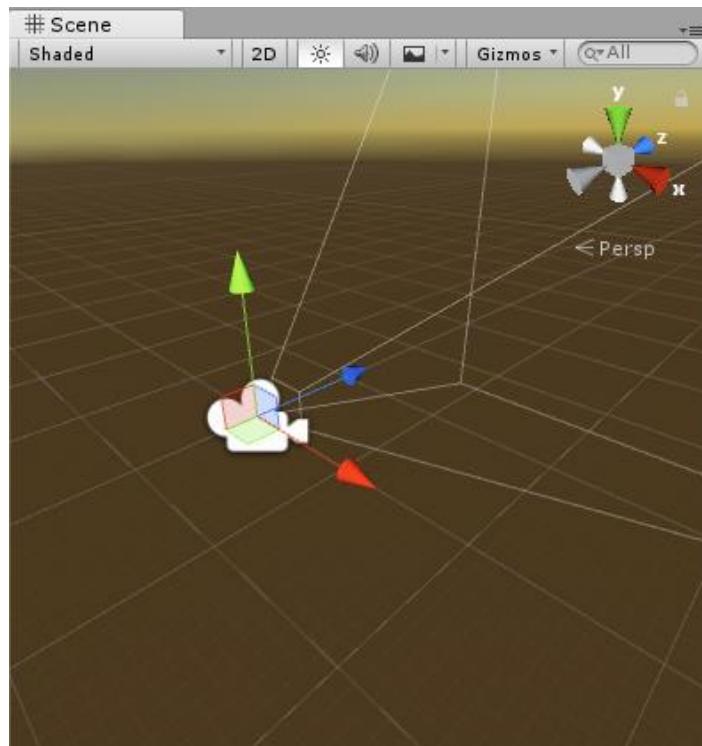


Rysunek 21 - struktura sceny

Na strukturę sceny(Rys. 21) składają się kolejno:

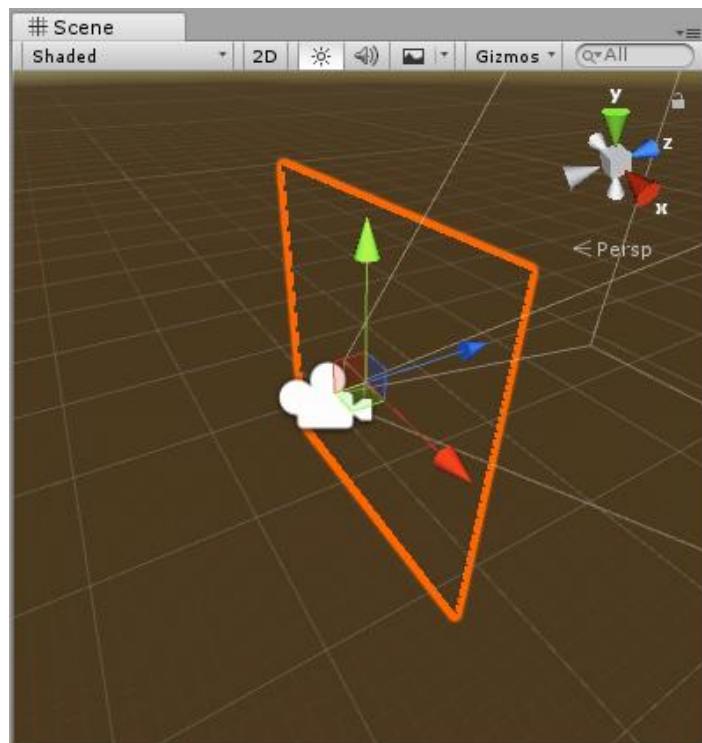
Cztery skrypty od GoogleVR SDK (3 bezpośrednio w elementach sceny + 1 w kamerze głównej, dokładniej zostaną opisane w podrozdziale „Elementy biblioteki GoogleVR SDK”)

Statyw (ang. Tripod, Rys. 22) – pusty obiekt, zawierający kamerę główną (ang. Main Camera).



Rysunek 22 - wirtualny statyw

Ta z kolei zawiera skrypt służący do wskazywania na inne obiekty oraz obiekt 3D typu czworokątnego (ang. Quad) o nazwie Fader (Rys. 23), który posłużył do ciekawszych przejść widoków kamery.



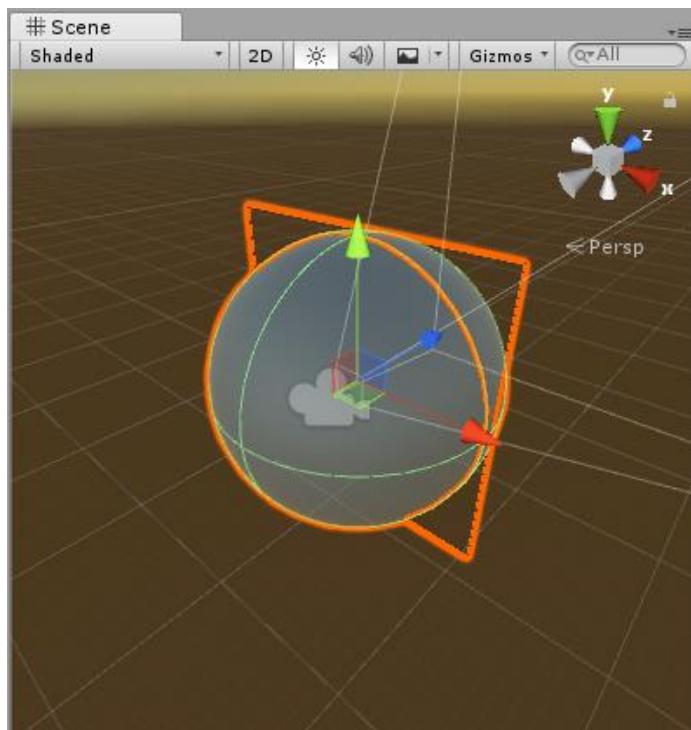
Rysunek 23 - fader

Statyw zawiera, również strefę Canvas[30], gdzie powinny znajdować się wszystkie elementy związane z interfejsem użytkownika (ang. UI). W omawianym przypadku jest to komponent typu Image odpowiadający za tzw. GazeClick (dokładniej opisany niżej).

Siedem sfer 3D z czego każda, zawiera strzałki (Rys. 36), oznaczające kierunki przemieszczania oraz ikony, dzięki którym, po najechaniu na nie, wyświetdana zostaje informacja o danym obiekcie (Rys. 37).

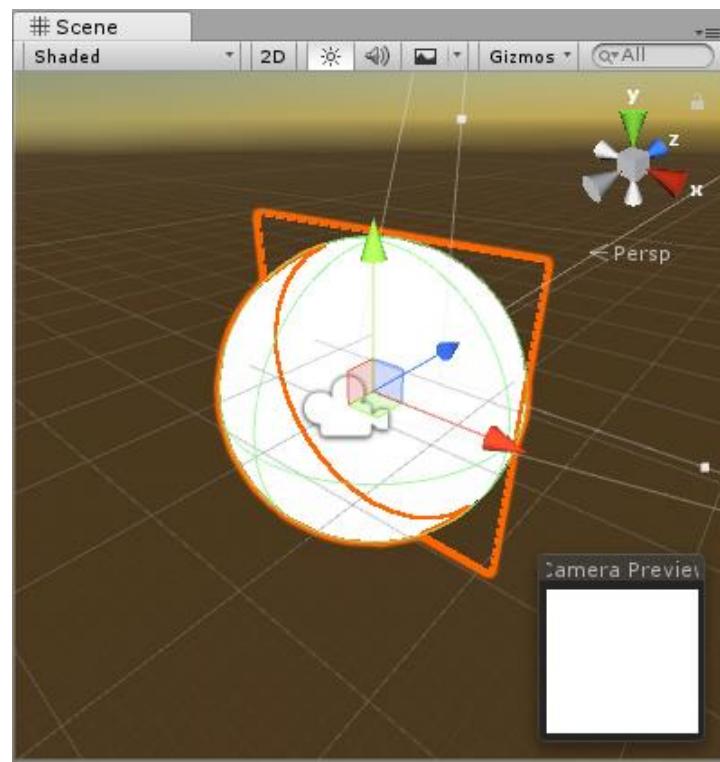
Wszystkie sfery tworzone są analogicznie do opisanej w następujący sposób:

Do statywów z kamerą dodatkowo utworzono obiekt 3D typu sfera (ang. Sphere, Rys. 24).



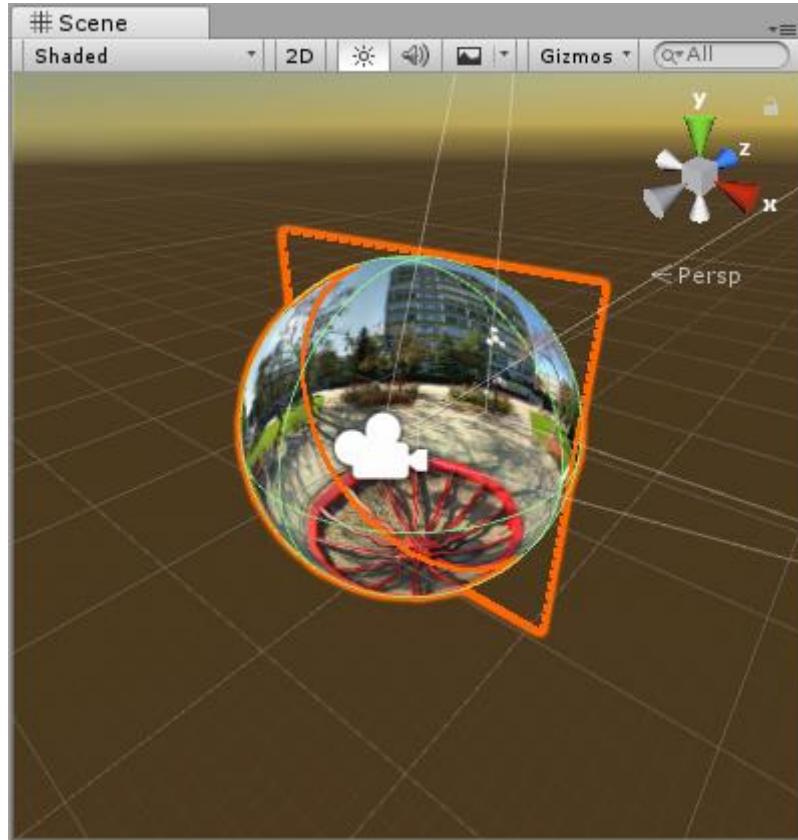
Rysunek 24 - pusta sfera 3D z kamerą

Po utworzeniu sfery dodano Shader o nazwie Insideout, który umożliwił odpowiednie teksturowanie sfery, obiekt nie będzie wymagał dodatkowego oświetlenia, domyślne renderujący siatkę na biało(Rys. 25).



Rysunek 25 - sfera wypełniona shaderem

Na koniec stworzono odpowiedni materiał za pomocą, wcześniej uzyskanych tekstur czyli zdjęć w formie 360 i dodano go do sfery co daje następujący efekt (Rys. 26).



Rysunek 26 - gotowa sfera, po wszystkich etapach budowania

5.5. Elementy biblioteki Google VR SDK

Przedstawiane skrypty to tzw. prefabrykaty (ang. prefabs) wprowadzające do projektu wsparcie w postaci kontrolerów[31]. Istnieją po to, aby łatwiej duplikować i zarządzać obiektami.

- GvrEditorEmulator – wszechobecny na scenie, pozwala na stymulowanie ruchu kamery poprzez ruch głowy lub myszki
- GvrControllerMain – wszechobecny na scenie, główny kontroler, odpowiedzialny za zarządzanie stanami i innymi kontrolerami
- GvrEventSystem – wszechobecny na scenie, zastępuje domyślny Event System Unity oraz osobny skrypt GvrPointerInputModule - zastępuje domyślny StandaloneInputModule, umożliwia zarządzanie eventami
- GvrReticlePointer – zawierający się w kamerze głównej, umożliwia wskazywanie na obiekty

5.6. Implementacja mechaniki przejść i funkcji GazeClick

Umożliwienie przechodzenia między sferami oraz uzyskanie dodatkowych efektów specjalnych umożliwia niżej opisany skrypt, który pozwala w projekcie na ciekawsze przejścia, czy też na wygodny „przycisk” ich zmiany. Na potrzeby implementacji napisano kod do obsługi tych możliwości. W dalszej części pracy zaprezentowane zostały fragmenty kodu skryptu o nazwie „SphereChangerWithGazeClick.cs” wraz z ich opisem.

Zastosowana mechanika przejść, polega na płynnym zakryciu pola widzenia kamery przy wyjściu ze sfery (ang. fade out) oraz odkryciu przy wejściu w nową (ang. fade in).

Wszystko zaczyna się od zdefiniowania odpowiednich pól (Rys. 27).

```
//This object should be called 'Fader' and placed over the camera
GameObject m_Fader;
float MyTime = 0f;
public Transform RadialProgress;
public Transform nextSphere{ get; set; }
```

Rysunek 27 - zdefiniowane pola w skrypcie

- m_Fader - obiekt klasy GameObject, obsługuje wcześniej już stworzony obiekt gry o nazwie Fader, który zostanie przypisany w konstruktorze
- MyTime - pole typu zmiennoprzecinkowego float, odpowiada za odmierzanie czasu, po którym następuje przejście
- RadialProgress - obiekt klasy Transform, obsługuje komponent Image będący zmienny w czasie
- nextSphere - obiekt typu Transform, zawiera charakterystyczny dla języka C# getter i seter, odpowiada bezpośrednio za zmianę sfery

Następnie metoda Awake() typu void (Rys. 28), tworzona w momencie tworzenia obiektu na scenie, jest wykonywana, gdy obiekt zostaje aktywowany. Traktowana jako konstruktor.

```
void Awake()
{
    //Find the fader object
    m_Fader = GameObject.Find("Fader");

    //Check if we found something
    if (m_Fader == null)
        Debug.LogWarning("No Fader object found on camera.");
}
```

Rysunek 28 - metoda Awake()

Zawiera przypisanie znalezioneego obiektu Fader oraz instrukcję warunkową, która dla przypadku nie znalezienia obiektu wypisuje w konsoli Unity ostrzeżenie.

```
public void ChangeSphere(Transform nextSphere)
{
    StartCoroutine(FadeCamera(nextSphere));
}
```

Rysunek 29 - metoda ChangeSphere()

ChangeSphere() - metoda typu void (Rys. 29), pobierająca jeden argument o nazwie nextShpere klasy Transform. Jej wywołanie odbywa się w metodzie Update(), która zostanie dokładniej wyjaśniona przy opisie funkcji GazeClick. Zostaje wywołana po przekroczenia czasu 3 sekund od nakierowania kurSORA na odpowiedni obiekt w tym przypadku są to obiekty obrazujące kierunki. ChangeSphere() wywołuje metodę StartCoroutine()[32] z klasy MonoBehaviour, która przyjmuje wywołanie innej metody o nazwie FadeCamera() powodującą oczekiwany efekt zakrycia i odsłonięcia kamery.

Miedzy tymi zdarzeniami występuje zmiana pozycji kamery na nową nextSphere wskazaną w środowisku Unity.

```
IEnumerator FadeCamera(Transform nextSphere)
{
    //Ensure we have a fader object
    if (_Fader != null)
    {
        //Fade the Quad object in and wait 0.75 seconds
        StartCoroutine(FadeIn(0.75f, _Fader.GetComponent<Renderer>().material));
        yield return new WaitForSeconds(0.75f);

        //Change the camera position
        Camera.main.transform.parent.position = nextSphere.position;

        //Fade the Quad object out
        StartCoroutine(FadeOut(0.75f, _Fader.GetComponent<Renderer>().material));
        yield return new WaitForSeconds(0.75f);
    }
    else
    {
        //No fader, so just swap the camera position
        Camera.main.transform.parent.position = nextSphere.position;
    }
}
```

Rysunek 30 - metoda FadeCamera()

FadeCamera(), metoda typu IEnumerator (Rys. 30) sprawdza, czy pobranie obiektu typu Fader powiodło się, jeśli tak to wywołuje kolejne współprogramy, które odpowiadają za zmianę, w przypadku niepowodzenia zmienia sferą bez specjalnego przejścia.

```
IEnumerator FadeOut(float time, Material mat)
{
    //While we are still visible, remove some of the alpha colour
    while (mat.color.a > 0.0f)
    {
        mat.color = new Color(mat.color.r, mat.color.g, mat.color.b, mat.color.a
            - (Time.deltaTime / time));
        yield return null;
    }
}

IEnumerator FadeIn(float time, Material mat)
{
    //While we aren't fully visible, add some of the alpha colour
    while (mat.color.a < 1.0f)
    {
        mat.color = new Color(mat.color.r, mat.color.g, mat.color.b, mat.color.a
            + (Time.deltaTime / time));
        yield return null;
    }
}
```

Rysunek 31 - metody FadeIn() oraz FadeOut()

Przedstawiony kod skryptu zawiera dwie metody typu IEnumerator o nazwach FadeOut() oraz FadeIn() pobierające dwa argumenty, pierwszy typu float i obiekt klasy Material (Rys. 31).

Dane metody odpowiadają za usuwanie w przypadku FadeOut() i dodawanie w przypadku FadeIn(), koloru alfa w obiekcie Fader, co prowadzi do całkowitej przeźroczystości obiektu bądź całkowitego ukazania się. Kolor jest stopniowo zmieniany w czasie.

Drugim wprowadzonym udogodnieniem jest GazeClick (Rys. 33), czyli funkcjonalność, która umożliwia „kliknięcie” obiektu na scenie po nakierowaniu na niego. Najechanie kursorem na dany obiekt włącza skrypt.

```
void Update()
{
    MyTime += Time.deltaTime;
    RadialProgress.GetComponent<Image>().fillAmount = MyTime/3;

    if (MyTime >= 3f) {
        ChangeSphere (nextSphere);
    }
}
```

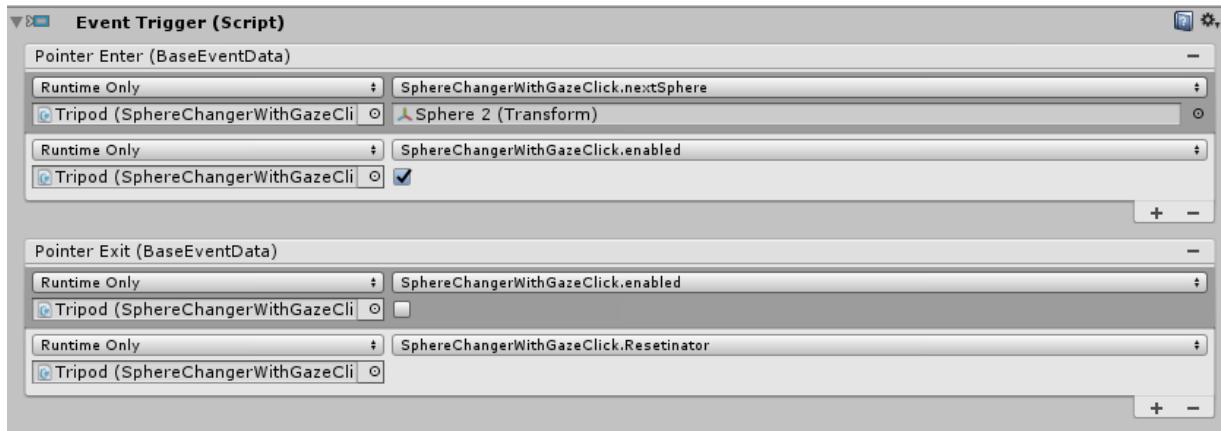
Rysunek 32 - metoda Update()

Update() – metoda wywoływana w każdej klatce (Rys. 32), jeśli MonoBehaviour jest włączone. Najczęściej używana metoda służąca do definiowania określonego zachowania w grze: przesunięcia, obracania, itd. Dla potrzeb uzyskania określonego zachowania, z każdym wykonaniem funkcji Update inkrementowana jest zmienna MyTime o dany czas uzyskany z metody deltaTime klasy Time. Następnie dopełniany jest promień okręgu i gdy przekroczy czas 3 sekund, bądź będzie mu równy, nastąpi wywołanie metody ChangeSphere.



Rysunek 33 - przedstawienie GazeClick

Na koniec ustawienie skryptu Event Trigger (Rys. 34) obsługującego zdarzenia.

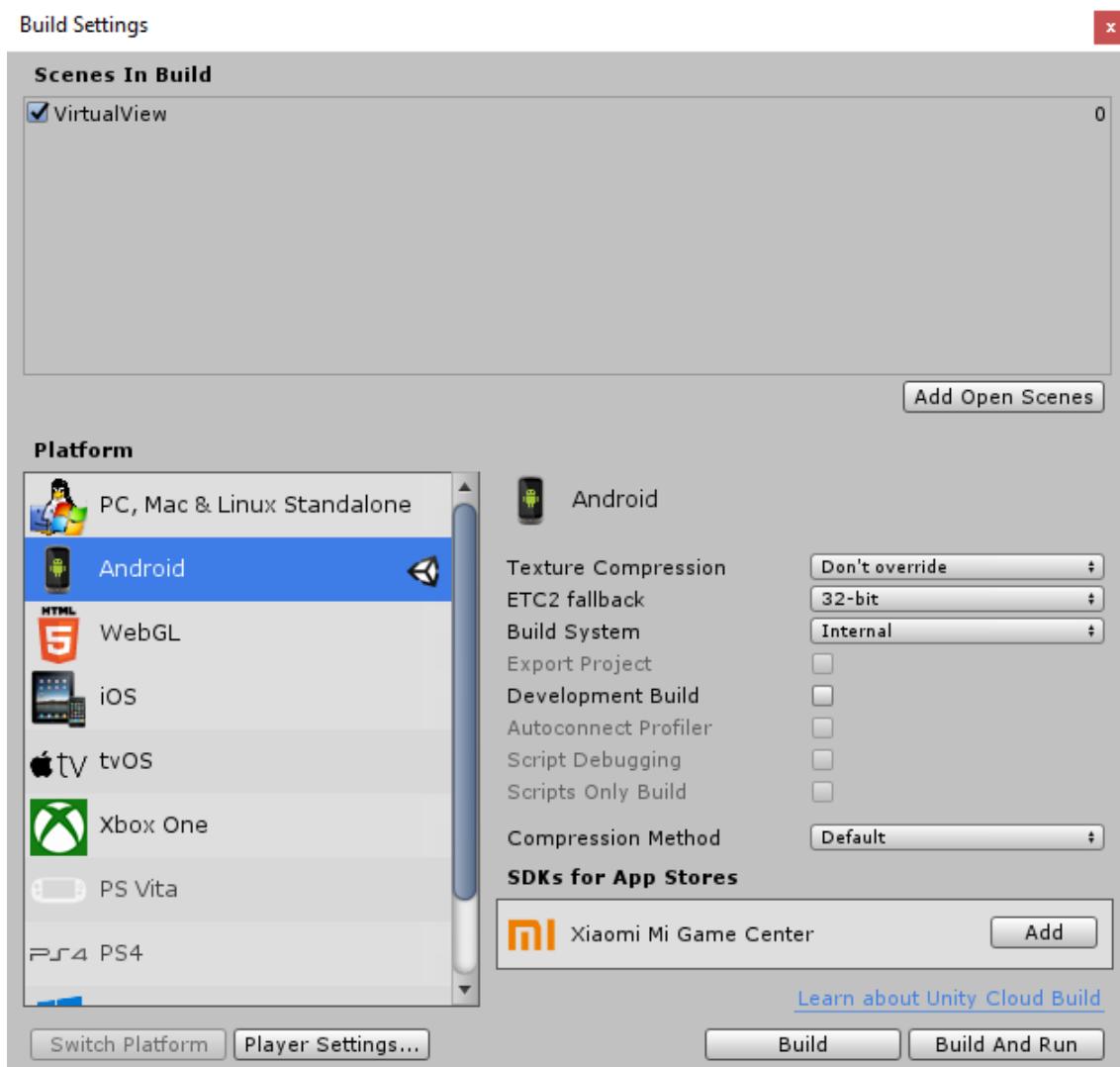


Rysunek 34 - okno przedstawiające ustawienia Event Trigerra

Przedstawiona ilustracja obrazuje w jaki sposób obsługiwane są przejścia oraz GazeClick. Zdarzenie Pointer Enter odpowiada za uruchomienie skryptu, natomiast Pointer Exit za jego wyłączenie. Dodatkowo w pierwszym, jako parametr przyjmowana jest sfera, na którą ma nastąpić zmiana, a w drugim przed wyłączeniem skryptu uruchamiana jest bardzo prosta metoda Resetinator(), która zeruje czas oraz postęp wypełnienia okręgu.

5.8. Export aplikacji na urządzenia z systemem Android

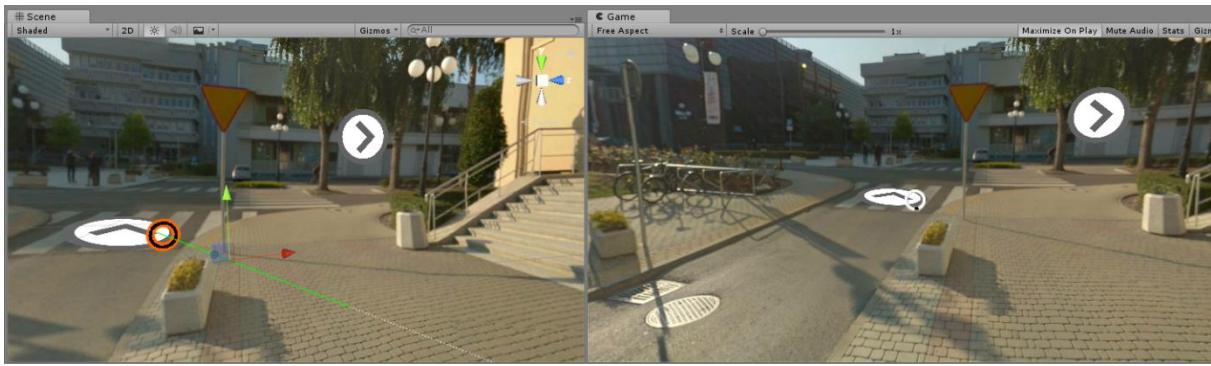
Za pomocą ustawień „Build Settings” (Rys. 35) w środowisku Unity można w łatwy sposób wyeksportować naszą aplikację na różne platformy systemowe.



Rysunek 35 - okno eksportu, budowania pliku .apk

Dodatkowo w ustawieniach Player Settings wymagane było wskazanie Virtual Reality SDK, w omawianym przypadku to Cardboard oraz Minimum API Level – zainstalowane wcześniej Android 4.4.

Po zbudowaniu i zainstalowaniu aplikacji na telefonie uzyskaliśmy widoczny niżej efekt (Rys. 38).



Rysunek 36 - przedstawienie występujących w projekcie strzałek do zmiany położenia; w lewym okienku widok sceny, po prawej widok gry



Rysunek 37 - przedstawienie funkcjonalności aplikacji



Rysunek 38 - widok z aplikacji zainstalowanej na smartphone'ie z systemem operacyjnym Android

Jak widać przedstawiony proces budowania aplikacji przebiegł pomyślnie, czego dowodem są powyższe zrzuty ekranów (Rys. 36 - 38).

6. Aplikacja wykonana przy użyciu VR View w JavaScript

6.1. Wstęp

Aplikację wykonano na wzór opracowanej w Unity w celach porównawczych.

Google udostępnia JavaScript API[33], które pozwala na wstawienie zdjęć oraz filmów 360 na stronę internetową tworząc i kontrolując zawartość elementu iframe lub jawnie deklarując element iframe. Zaletą tego rozwiązania prostota i czytelność.

6.2. Implementacja

Aby używać klasy VR View należy dołączyć źródło skryptu w kodzie HTML (Rys. 39) oraz stworzyć element div o id „vrview”, który po załadowaniu VR View zostanie zastąpiony elementem iframe o klasie „vrview”.

```
<!DOCTYPE html>
<html>
  <head>
    <title>VirtualView</title>
  </head>

  <body>
    <script src="http://storage.googleapis.com/vrview/2.0/build/vrview.min.js"></script>
    <script src="main.js"></script>

    <div id="vrview"></div>
  </body>
</html>
```

Rysunek 39 - kod HTML z dołączonymi skryptami i zadeklarowanym kontenerem na zawartość

6.2.1. VR View

Kiedy strona zostanie załadowana, zostanie wywołana również funkcja, która stworzy nową instancję VR View.

Instancję VR View stworzono wołając konstruktor, VRView.Player i przekazując selektor, który determinuje gdzie „wrzucić” VR View oraz przekazując kolekcję parametrów. Selektor powinien wskazywać na id elementu diva, który został zawarty w HTML-u. Parametrami mogą być m.in. źródło np. filmu lub zdjęcia (video/image), wymiary iframe (width/height), flagi np. is_debug do włączenia funkcji debugowania zawierającego m.in. licznik FPS-ów (Rys. 40).

```
window.addEventListener('load', onVrViewLoad)

function onVrViewLoad() {
    vrView = new VRView.Player('#vrview', {
        width: '100%',
        height: 480,
        image: 'img/B5.jpg',
        is_stereo: false
    });
}
```

Rysunek 40 - funkcja wywołująca konstruktor po załadowaniu strony

Ważne dla kolejnych części instrukcji - zdefiniowano obiekt scenes(Rys. 41), którego pola to poszczególne sceny o kluczu takim samym jak nazwa zdjęcia, które z kolei mają zdefiniowane swoje parametry (m.in. ścieżka do pliku zdjęcia) i pola hotspots przypisane dla każdej sąsiedniej sceny, które zawierają informację o położeniu i rozmiarze hotspota na danej scenie (na zdjęciu poniżej zdefiniowano scenę B5 i hotspot do sceny B4).

```
var scenes = {
    B5: {
        image: 'img/B5.jpg',
        is_stereo: false,
        hotspots: {
            B4: {
                pitch: 0,
                yaw: -120,
                radius: 0.05,
                distance: 1
            }
        }
    },
}
```

Rysunek 41 - przechowywanie informacji o obiektach

VR View rozgłasza różne rodzaje zdarzeń, które zostały być zarejestrowane używając funkcji VRView.on(String event, function handleEvent). Funkcja on() przyjmuje dwa argumenty: nazwa wydarzenia w postaci Stringa oraz funkcja do obsługiwanego (Rys. 42).

```
vrView.on('ready', onVRViewReady);
vrView.on('modechange', onModeChange);
vrView.on('click', onHotspotClick);
vrView.on('error', onVRViewError);
vrView.on('getposition', onGetPosition);
```

Rysunek 42 - obsługa eventów 1/2

Typy wydarzeń to (Rys. 43):

- ready : VR View załadował początkową zawartość i jest gotowy do przyjmowania komend.
- error : VR View zgłasza błąd, np. kiedy zawartość nie może być załadowana.
- click : VR View zarejestrował „klik” (tap na telefonie lub gaze w trybie VR). Może być użyte do przejścia do następnego slajdu w carousel lub rejestrowania, gdy użytkownik kliką na hotspot.
- modechange : VR View zmienia tryb, np. kiedy użytkownik wchodzi w tryb VR lub pełny ekran. Można tego używać, aby śledzić zachowanie użytkownika rejestrując wywołania zwrotne kiedy użytkownik zmienia tryby.
- getposition: VR View zarejestrował zmianę pozycji (atrybuty Yaw oraz Pitch wskazującego na kolejną pozycję w osi poziomej i pionowej)

```
function onVRViewReady(e) {
    console.log('onVRViewReady');
    loadScene('B5');
}

function onModeChange(e) {
    console.log('onModeChange', e.mode);
}

function onVRViewError(e) {
    console.log('Error! %s', e.message);
}

function onGetPosition(e) {
    console.log(e)
}
```

Rysunek 43 - obsługa eventów 2/2

6.2.2. Ładowanie nowej zawartości

Nową zawartość można załadować do iframe bez jego przeładowywania, używając funkcji setContentInfo()/setContent (). Funkcja ta przyjmuje URL do zdjęcia 360 lub filmu jako string i kolekcję innych parametrów. Zasadniczo może

używać tych samych parametrów co konstruktor VR View, poza width i height (Rys. 44).

```
function loadScene(id) {  
    console.log('loadScene', id);  
  
    vrView.setContent({  
        image: scenes[id].image,  
        preview: scenes[id].preview  
    });  
  
}
```

Rysunek 44 - ładowanie nowej sceny

6.2.3. Hotspedy

To miejsca na fotosferze, z którymi użytkownicy mogą wejść w interakcję. Hotspedy mogą być wyświetlane na wszystkich platformach, ale to w jaki sposób użytkownicy wchodzą w nimi z interakcją jest różny i zależy od platformy.

- Na desktopach, najechanie na hotspot zmienia je wizualnie, a kliknięcie aktywuje.
- W trybie mobile tap na hotspedy aktywuje go
- W trybie VR, wpatrywanie się w fotosferę zawierającą hotspedy powoduje pojawienie się celownika, który zmienia stan gdy zostanie ustawiony bezpośrednio w hotspot, wtedy tap gdziekolwiek aktywuje hotspot.

Hotspedy to okrągłe znaczniki na sferze z nadanym ID (String), koordynatami środka, promieniem oraz odlegością od kamery. Zostały dodane poprzez funkcję `vrView.addHotspot()` (Rys. 45). Koordynaty dzielą się na pitch i yaw i są one sferyczne. Domyślnie widok jest skierowany na (0,0). Zakres pitch to [-90,90] gdzie dodatnie wartości wskazują góre. Yaw to zakres [-180, 180] a dodatnie wartości wskazują na prawo.

```
// Dodaje wszystkie hotspotty do sceny
var newScene = scenes[id];
var sceneHotspots = Object.keys(newScene.hotspots);
for (var i = 0; i < sceneHotspots.length; i++) {
    var hotspotKey = sceneHotspots[i];
    var hotspot = newScene.hotspots[hotspotKey];

    vrView.addHotspot(hotspotKey, {
        pitch: hotspot.pitch,
        yaw: hotspot.yaw,
        radius: hotspot.radius,
        distance: hotspot.distance
    });
}
```

Rysunek 45 - dodawanie hotspotów do ładowanej sceny

Aby dodać wydarzenie, kiedy hotspot zostanie aktywowany użyto funkcji on(). Zasadniczo przy każdym kliku sprawdzamy czy trafiliśmy na hotspot, jeżeli tak to ładujemy odpowiednią scenę (Rys. 46-47).

```
vrView.on('click', onHotspotClick);
```

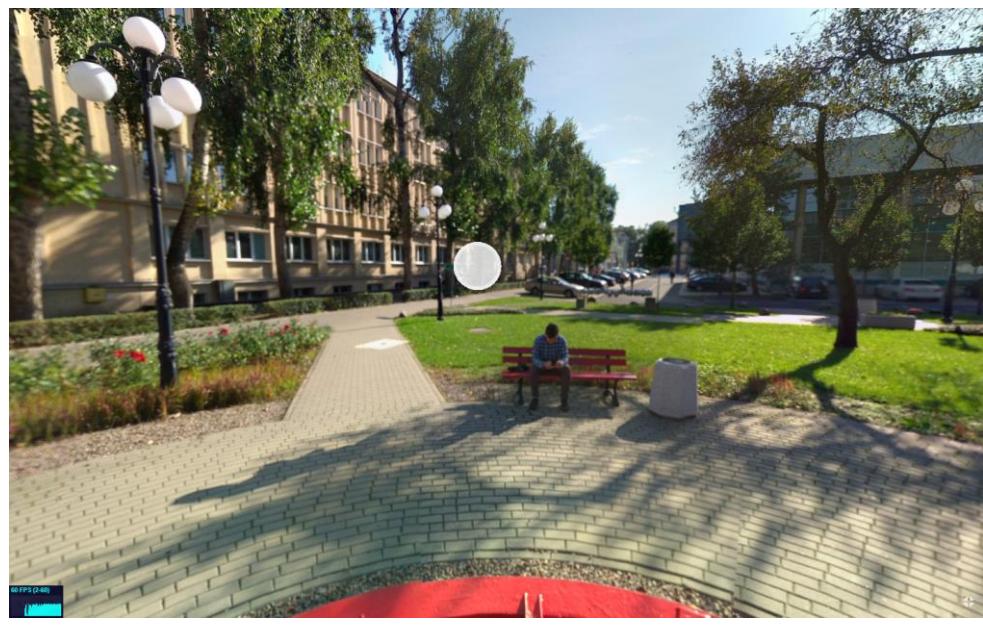
Rysunek 46 - obsługa kliknięcia

```
function onHotspotClick(e) {
    vrView.getPosition();
    console.log('onHotspotClick', e.id);
    if (e.id) {
        loadScene(e.id);
    }
}
```

Rysunek 47 - obsługa kliknięcia 2/2

6.3. Hosting

Aplikacja będąc aplikacją webową została udostępniona w sieci przy użyciu narzędzia „Chrome Web Server”, które pozwala na hosting stron internetowych/plików jako serwer HTTP urządzeniu posiadającemu przeglądarkę Google Chrome oraz dostęp do tych zawartości z dowolnego urządzenia w sieci lokalnej jak i publicznej. Dzięki temu można połączyć się ze stroną web przez smartfon, który podłączono do okularów, co pozwala na oglądanie zdjęć w trybie VR przez sieć.



Rysunek 48 - tryb desktopowy (poruszanie za pomocą kurSORA) – aplikacja webowa



Rysunek 49 - tryb VR – aplikacja webowa

7. Testowanie aplikacji

7.1. Wydajność programów

Badanie wydajności przeprowadzono na dwóch różnych telefonach oraz na dwóch różnych komputerach. Specyfikacja techniczna telefonów została zestawiona wcześniej w rozdziale 3 w tabeli 1, natomiast porównanie komputerów przedstawiono w tym rozdziale w tabeli 2. Przedstawienie wyników testów wydajnościowych zawarto zbiorczo w tabelach 3-5.

Tabela 2 - przedstawienie specyfikacji technicznej komputerów testujących

NAZWA KOMPUTERA	LENOVO 81BG	LENOVO Y580
PROCESOR	Intel i5-8250U 4x1.6-3.4	Intel i7-3630QM 4x2,4-3,4
KARTA GRAFICZNA	NVIDIA GeForce MX150	NVIDIA GTX 660m
PAMIĘĆ RAM	8GB DDR4	4GB DDR3
SYSTEM OPERACYJNY	MS Windows 10 x64	MS Windows 10 x64

Zarówno jeden jak i drugi komputer posiadają mocne podzespoły bazowe, jednakże notebook po lewej stronie jest około 4 lata młodszy od laptopa po prawej.

Tabela 3 - przedstawienie zużycia zasobów przez urządzenia w stanie spoczynku (wartości podawane w procentach, Rys. 51)

NAZWA URZĄDZENIA	XIAOMI REDMI NOTE 4X	SONY XPERIA Z1	LENOVO 81BG	LENOVO Y580
ZUŻYCIE	CPU	3	9	2
	GPU	0	0	0
	RAM	61	77	40

Tabela 4 - przedstawienie zużycia zasobów przez urządzenia podczas uruchomionej aplikacji Unity (wartości podawane w procentach; Rys. 52)

NAZWA URZĄDZENIA	XIAOMI REDMI NOTE 4X	SONY XPERIA Z1	LENOVO 81BG	LENOVO Y580
ZUŻYCIE	CPU	33	41	15
	GPU	35	36	19
	RAM	67	81	49

Tabela 5 - przedstawienie zużycia zasobów przez urządzenia podczas uruchomionej aplikacji JavaScript (wartości podawane w procentach; Rys. 53)

NAZWA URZĄDZENIA	XIAOMI REDMI NOTE 4X	SONY XPERIA Z1	LENOVO 81BG	LENOVO Y580
ZUŻYCIE	CPU	35	58	11
	GPU	30	44	20
	RAM	83	79	70

Badanie zostało przeprowadzone dla czterech urządzeń – dwóch smartphone’ów i dwóch laptopów. W przypadku pierwszej pary do odczytu zużycia używano aplikacji „Monitor System Lite”, natomiast w przypadku komputerów był to systemowy monitor zasobów. Na podstawie danych zebranych w tabelach zaobserwowano, że zarówno na poziomie telefonów jak i w przypadku notebooków dominuje sprzęt, który jest nowszy i zawiera podzespoły nowych generacji. Jest to niewielka różnica, ponieważ sprzęt testujący jest zbliżony do siebie. Duże różnice w wydajności aplikacji zaobserwowano między grupą smartphone’ów, a grupą notebooków, co wynika wprost z architektury i mocy obliczeniowej sprzętu.

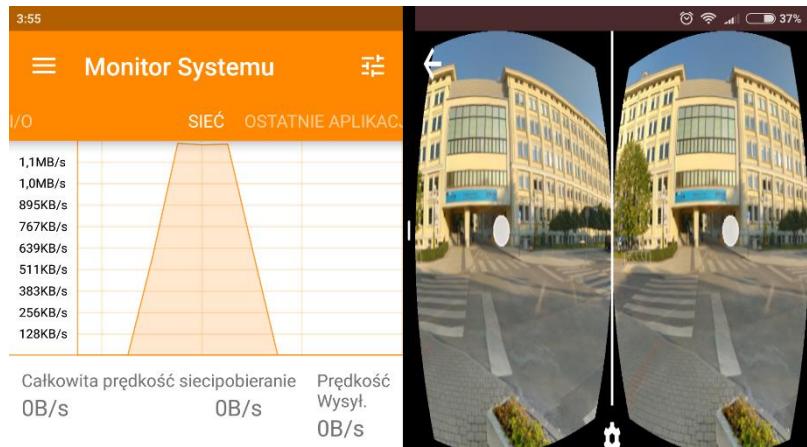
Porównując wyniki obciążenia sprzętu dwóch aplikacji można stwierdzić, że pod względem zużycia mocy obliczeniowej procesora i karty graficznej obie aplikacje wypadły bardzo wydajnie i zużywają mały procent zasobów. Tego samego nie można powiedzieć o zużyciu pamięci RAM w przypadku aplikacji JavaScript, gdzie wszystkie wartości były większe bądź równe 70%, co w porównaniu do programu napisanego w Unity wypada dużo gorzej.

Zużycie sieci (Rys. 50) wystąpiło jedynie w przypadku aplikacji webowej, w momentach przejścia/zmiany sfery 360 na inną. Objawiało się to również krótkim opóźnieniem mieszącym się w ułamku sekundy.

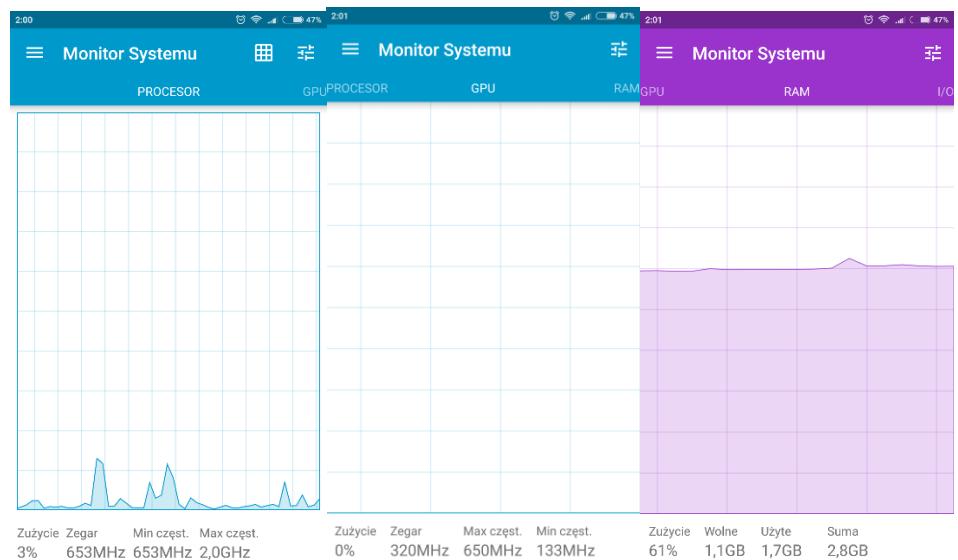
Zaobserwowano również, że obciążenie aplikacji nie przeszło obojętne przy zużyciu Baterii. Prognozowany czas rozładowania na urządzeniach mobilnych spadał średnio o godzinę.

Przyjrzano się również spadkom FPS'ów (ang. frames per second). Dla aplikacji stacjonarnej nie zauważono jakichkolwiek problemów, natomiast w przypadku sieciowej można było dopatrzyć się niewielkich skoków.

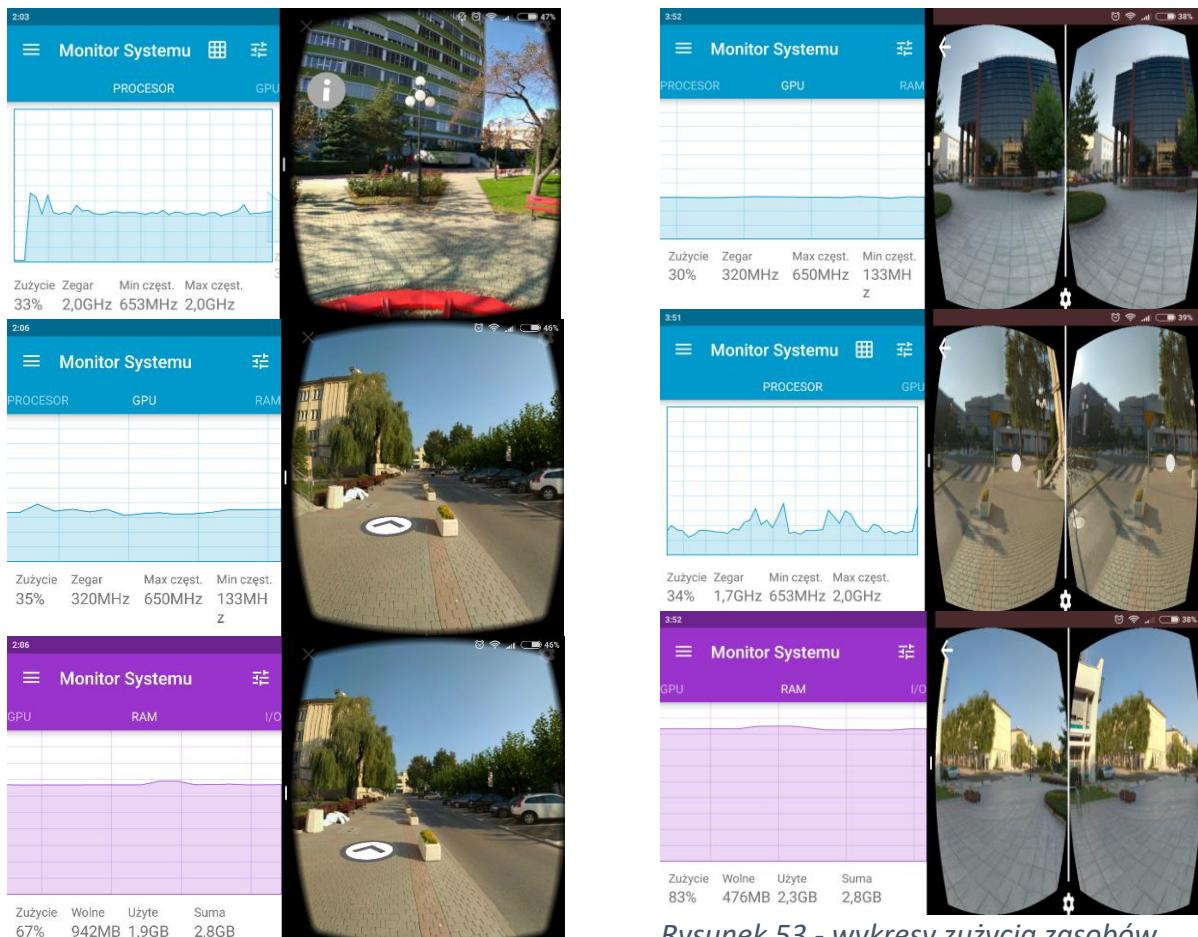
Przykładowe zrzuty ekranu z telefonu Xiaomi Redmi Note 4Xm, przedstawiające wyniki dla wszystkich trzech tabel:



Rysunek 50 - przedstawienie skoku prędkości sieci podczas testowania aplikacji webowej



Rysunek 51 - przedstawienie zasobów w czasie bezczynności smartphone'a



Rysunek 52 - wykresy zużycia zasobów przez aplikację napisaną w środowisku Unity

Rysunek 53 - wykresy zużycia zasobów przez aplikację napisaną w JavaScript

7.2. Porównanie jakościowe

Pomimo faktu, że użyto tych samych zdjęć w obu aplikacjach, ich jakość różniła się. Zestawiono ze sobą zrzuty ekranu z trybu full screen, czyli nie VR – owego (w którym obraz jest dodatkowo zniekształcaný w celu rekompensaty dla pewnych zjawisk opisanych szerzej w rozdziale 2.) zauważono wyraźnie lepszą jakość w przypadku aplikacji Web, która wykorzystuje rozwiązania Google'a do budowania sfery z gotowego zdjęcia. Dla porównania wybrano wycinki obszarów, gdzie różnica jest dobrze widoczna. (Rys. 54)



Rysunek 54 - porównanie jakości Unity - po lewej, Web - po prawej

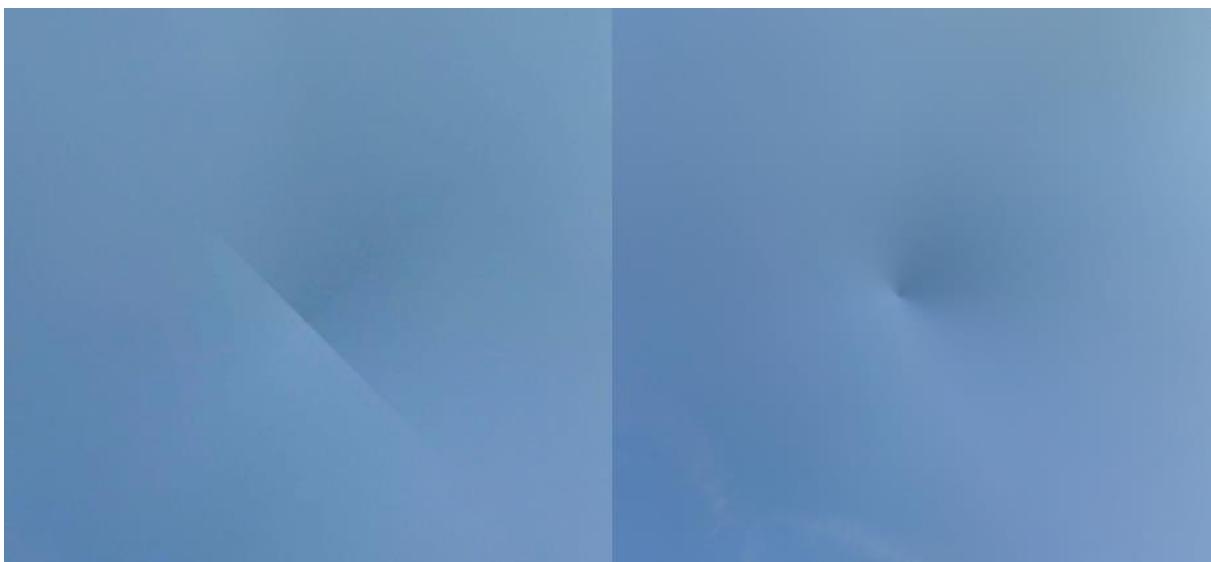
Pierwsza różnica na pierwszy rzut oka to zakłócenia i zdeformowania, które powstały w miejscach, gdzie zdjęcia zostały niedokładnie sklejone. W przypadku aplikacji Web zdjęcie wygląda mniej więcej tak, jak oryginał, jednak w Unity mamy do czynienia z dziwnymi wybrzuszeniami które pojawiły się po nałożeniu zdjęć na sferę. Jest to prawdopodobnie spowodowane inną implementacją shaderów przez oba rozwiązania. Poza tym, widoczna jest również niższa rozdzielcość fotosfery w przypadku Unity. Jest to spowodowane tym, że domyślnie Unity poddaje zdjęcia kompresji do maksymalnego rozmiaru 1024 KB. Jednak w dowolnym momencie jest możliwa zmiana ustawień. Porównanie kompresji do 1024 KB oraz jej braku umieszczone na Rys. 55.



Rysunek 55 - kompresja 1024 KB - po lewej, oraz jej brak - po prawej

Te niedociągnięcia, mimo swojej oczywistej natury, gdy je bezpośrednio porównamy nie są aż tak jednoznaczne w trakcie użytkowania aplikacji. Obraz i tak jest wystarczająco zniekształcanego software'owo (m.in. wspomniane w rozdziale 2. barrel distortion) jak i przez soczewki, dodając do tego niską rozdzielczość i PPI wyświetlacz, można dojść do wniosku, że bez odpowiedniego punktu odniesienia podczas używania aplikacji ciężko zauważać, że w jednym scenariuszu zdjęcie jest bardziej skompresowane od drugiego. Co innego miejscowe zniekształcenia pojawiające się spontanicznie na sferach – te są znacząco widoczne i w przypadku Unity niestety bardziej stanowią problem i odbierają immersję użytkowania, gdy już raz zwróciśmy na nie uwagę.

Inną różnicą jest sposób, w jaki sfera jest składana u sklepienia. W przypadku Web mamy do czynienia z dużo mniejszym zakłóceniem, przejawiającym się jednym charakterystycznym punktem (Rys. 56). W Unity jest to bardziej widoczne ponieważ biegun nie jest jednym punktem gdzie zbiegają się jednakowo części fotosfery, a linią, która jest mniej dyskretna.



Rysunek 56 - po lewej - Unity, po prawej - Web

8. Badania „user experience”

Pojęcie „user experience” odnosi się do całości wrażeń i nastawienia użytkownika danego produktu, systemu lub usługi, wynikające z jego stosowania. UX zawiera wszystkie preferencje, przekonania, wyobrażenia, emocje psychiczne oraz fizyczne reakcje, zachowania oraz dokonania, które następują przed, w trakcie i po używaniu produktu. Międzynarodowa organizacja normalizacyjna (ISO) wymienia 3 czynniki wpływające na doświadczenie użytkownika: system, użytkownika oraz kontekst użycia. Badanie nad UX polega na zaprojektowanie systemu, z którym interakcja będzie dostarczała jak najbardziej pozytywnych doświadczeń. Wymagania jakie stawia się danemu projektowi to: atrakcyjność, poręczność, przejrzystość, łatwość i przyjemność w użyciu.

Wychodzimy z założenia, że zbudowany projekt jest zorientowany na użytkownika i jego pozytywne doświadczenia. W związku z tym zorganizowano badania mające na celu ulepszenie obecnego projektu i ew. ukierunkowanie jego dalszego rozwoju. Badania odbyły się w prywatnym mieszkaniu, przy wykorzystaniu obu aplikacji oraz 2 rodzajów gogli - Google Cardboard i FiiT VR 2S.

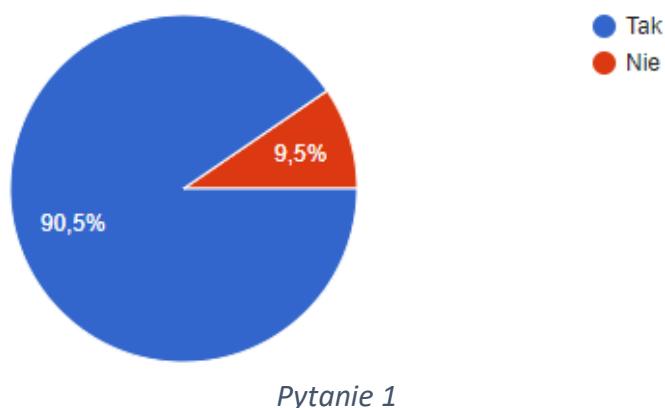
W badaniu wzięło łącznie udział 21 osób, w tym 76% mężczyzn (16) w przedziale wiekowym 19-24 lata, byli to głównie studenci lub absolwenci studiów technicznych. Warto zaznaczyć, że fakt ten mógł mieć pewien wpływ na subiektywność wydanych opinii. Mała próba ze specyficznego środowiska ankietowanych może też nieprecyzyjnie oddawać obraz odczuć przeciętnego użytkownika.

Badania podzielono na 2 części: w pierwszej została przedstawiona ankieta zawierająca kilkanaście pytań zamkniętych, zaś w drugiej poproszono o komentarz do 3 pytań, dla potencjalnych, przyszłych użytkowników, które pomogłyby zaprojektować jeszcze lepsze doświadczenie w przyszłości. Przeprowadzono również krótki wywiad związany z ww. pytaniami.

8.1. Część 1 – pytania otwarte

Czy uważasz, że badania UX w kontekście rozwoju tego projektu mają sens?

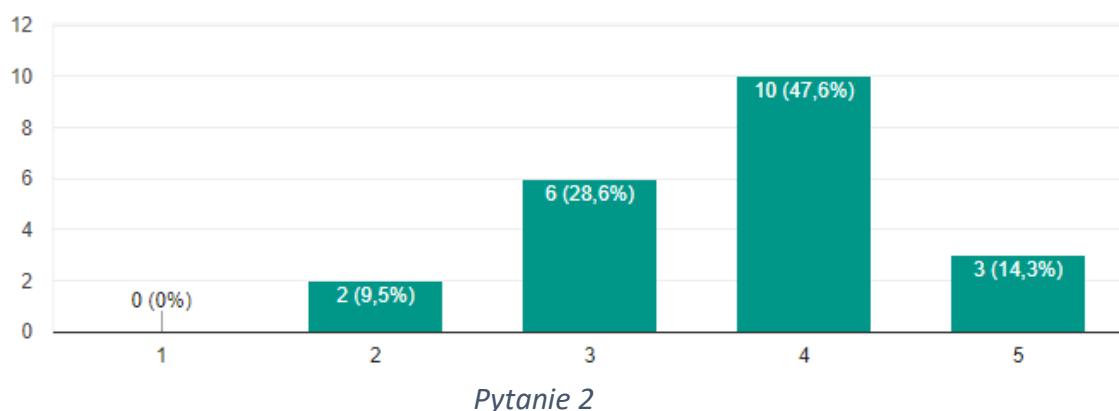
21 odpowiedzi



Pierwszym zbadano jaką wagę użytkownicy przywiązują do projektowania lepszego doświadczenia w kontekście takich aplikacji, odsetek odpowiedzi twierdzących mówi sam za siebie – ludzie są świadomi tego, że projektowanie UX to bardzo ważna część przy tworzeniu systemów interaktywnych.

Jak oceniasz jakość wykonanych zdjęć?

21 odpowiedzi

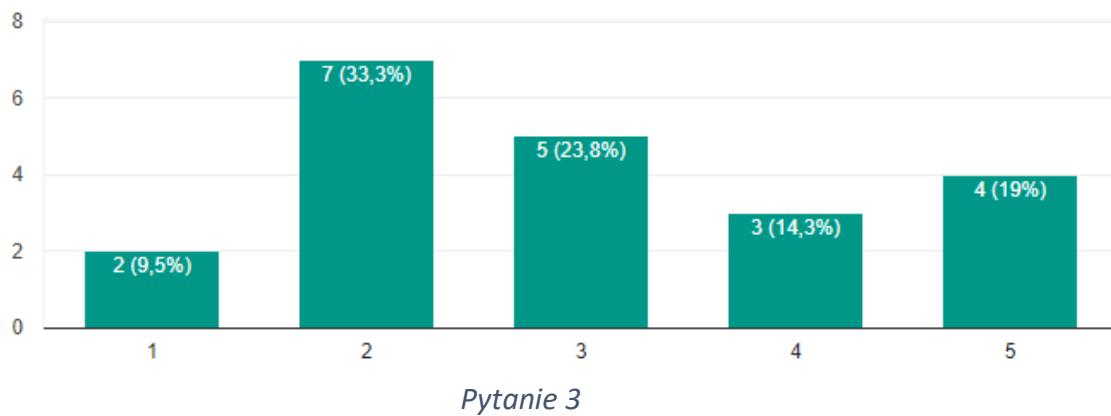


Zdjęcia zostały ocenione dobrze, jako niedociągnięcia zostały przedstawione niedokładne i pełne artefaktów bieguny sfery – wynika to z amatorskiego wykonywania zdjęć sferycznych, bez profesjonalnego sprzętu opierając się na

kilkudziesięciu zdjęciach. Ten element można poprawić używając wspomnianego wcześniej obiektywu szerokokątnego w smartfonie lub specjalnych dedykowanych aparatów do tworzenia zdjęć i filmów sferycznych np. Samsung Gear360 lub Rico Theta.

Jak oceniasz intuicyjność mechanizmu poruszania się po zdjęciach? (Aplikacja Unity)

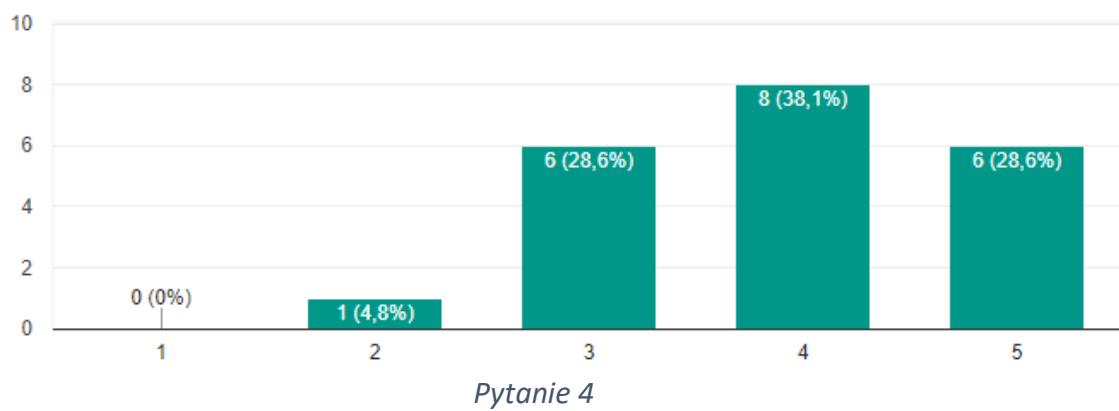
21 odpowiedzi



Wyniki były zdecydowanie niezadowalające, w aplikacji Unity jako znaczników do zmiany sceny użyto spriteów ptaków, które uznano za mocno nieintuicyjne i niewidoczne. W naszym odczuciu ptaki miały być oryginalne i artystyczne, jednak okazało się, że w większej części ankietowanych nie podobała się forma tej funkcjonalności.

Jak oceniasz intuicyjność mechanizmu poruszania się po zdjęciach? (Aplikacja Web)

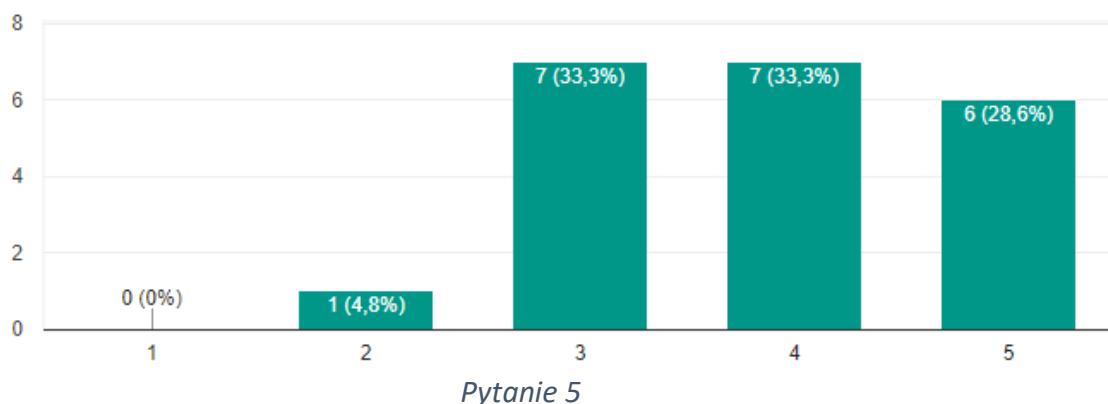
21 odpowiedzi



W aplikacji Web sam mechanizm poruszania się i zmiany scen był bardzo podobny do aplikacji Unity, poza samym znacznikiem, który był bardziej przyjazną i znaną użytkownikowi białą obręczą. Ta jedna różnica między mechanizmem tak bardzo spolaryzowała opinie.

Jak oceniasz wygodę mechanizmu uzyskiwania informacji o obiektach? (Aplikacja Unity)

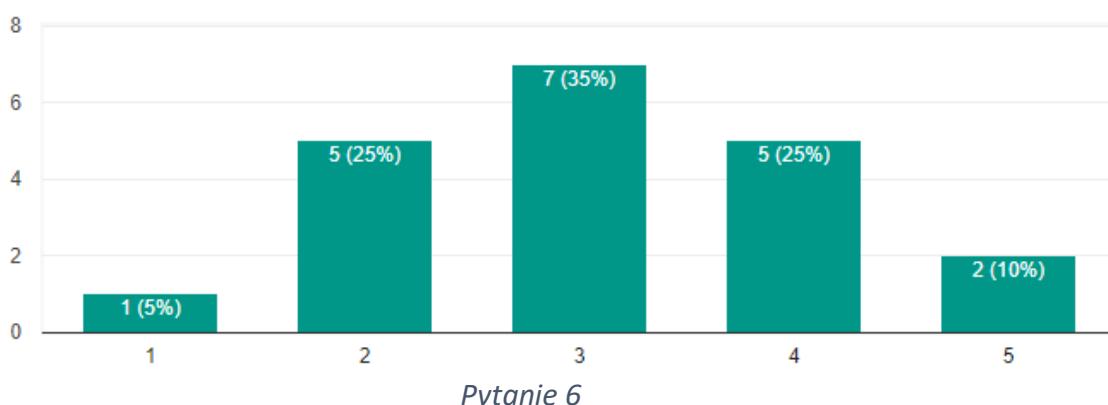
21 odpowiedzi



Uzyskiwanie informacji o obiekcie jako takie zostało dobrze przyjęte i uznane za ciekawe rozwiązanie, jednak niektórzy użytkownicy mieli problem z odczytaniem liter z opisów, co niestety bierze się ze słabej rozdzielczości obrazu wyświetlanego przez smartfon, który dodatkowo jest poddawany obróbkom oraz aberracji chromatycznej opisanej wyżej.

Jak bardzo było to immersywne doświadczenie?

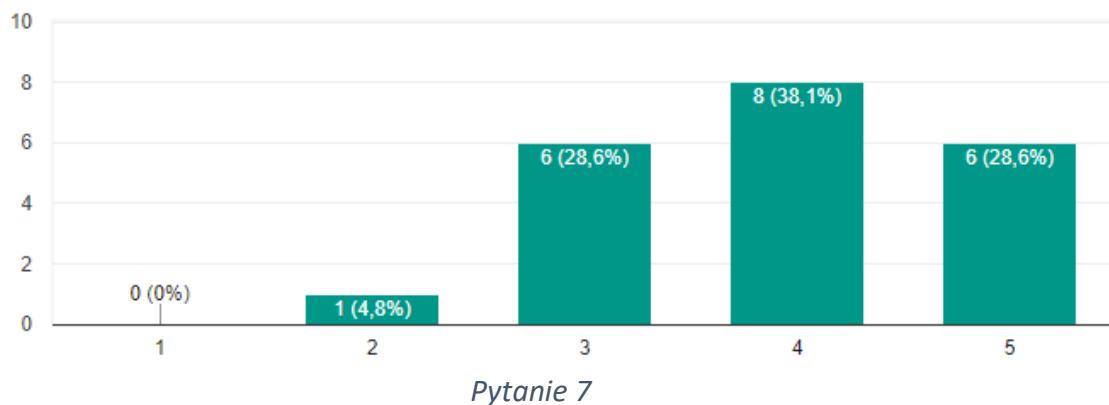
20 odpowiedzi



Immersyjność zaburzała przede wszystkim wspomniana słaba jakość obrazu spowodowana aberracją chromatyczną oraz zniekształceniami, które niestety obecnie są problemem układów VR.

Jak pozytywnie oceniasz całe doświadczenie?

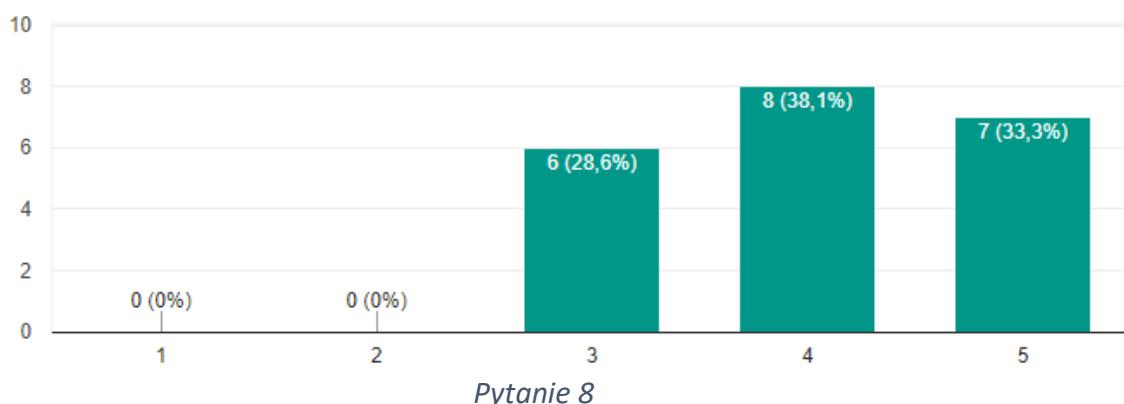
21 odpowiedzi



Doświadczenie zostało odebrane pozytywne, część badanych nie miała wcześniej bezpośredniej styczności z tą technologią, więc było to dla nich nowe i oryginalne doświadczenie.

Jak oceniasz wygodę okularów FiiT VR?

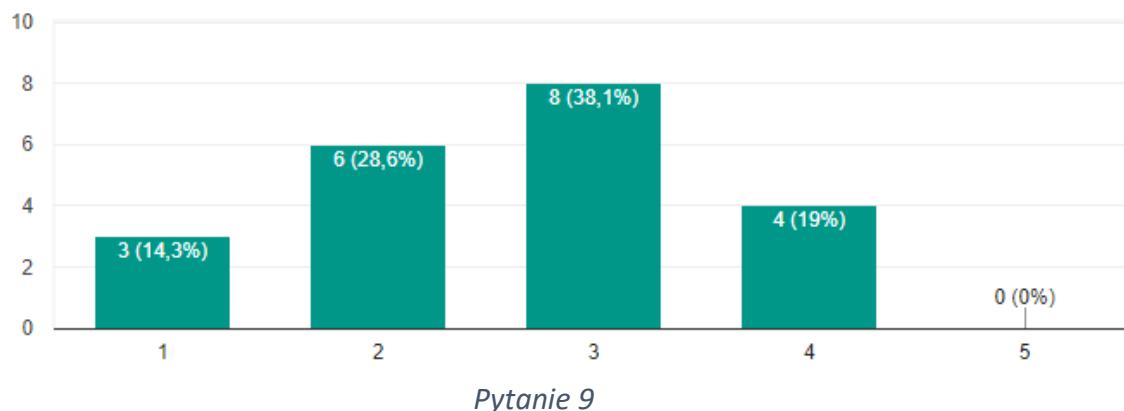
21 odpowiedzi



Okulary FiiT VR otrzymały wysoką ocenę. Są wygodne i mają różne udoskonalenia w postaci regulatorów.

Jak oceniasz wygodę okularów Google Cardboard?

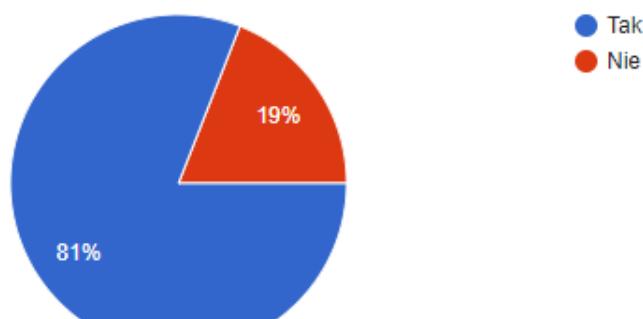
21 odpowiedzi



Nie było tutaj zaskoczenia, układ Google Cardboard jest z pewnością bardziej znaną opcją produkowaną i promowaną, jednakże pozostaje kartonem z soczewkami, co przekłada się na jego słabą ergonomię. Do plusów jednak na pewno zalicza się cenę.

Czy uważasz za sensowne istnienie możliwości wirtualnego zwiedzania różnych obiektów?

21 odpowiedzi

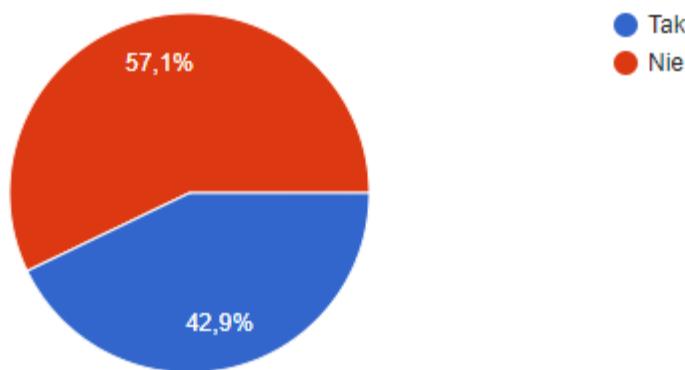


Pytanie 10

Większość głosów sugeruje, że takie projekty mogą mieć w przyszłości rację bytu i być wykorzystywane w celach komercyjnych lub edukacyjnych.

Czy uważasz, że wirtualne zwiedzanie tego typu mogłoby zastąpić prawdziwe?

21 odpowiedzi

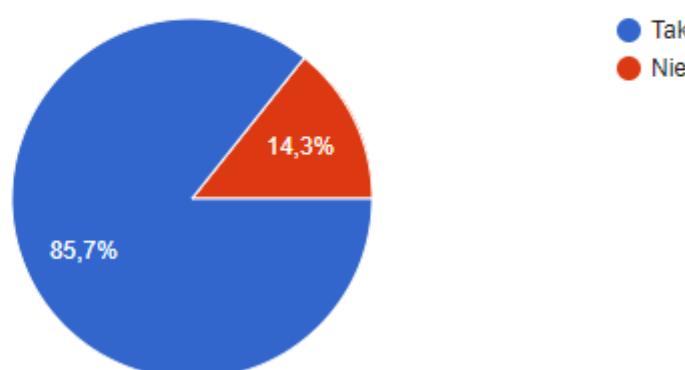


Pytanie 11

Na dzień dzisiejszy technologie wirtualnej rzeczywistości są na bardzo młodym etapie, można powiedzieć ta technologia dopiero się rozwija. Wynik nie dziwi – coś materialnego jest uznawane za bardziej autentyczne, a autentyczność to obecnie bardzo cenny towar. Istnieje w niej jednak wielki potencjał i jest możliwe, że w przyszłości technologia będzie w stanie bardzo wiernie odwzorowywać rzeczywistość – na tyle wiernie, że różnica pomiędzy nimi się zatrze.

Czy Twoim zdaniem jest to przyszłościowa aplikacja?

21 odpowiedzi

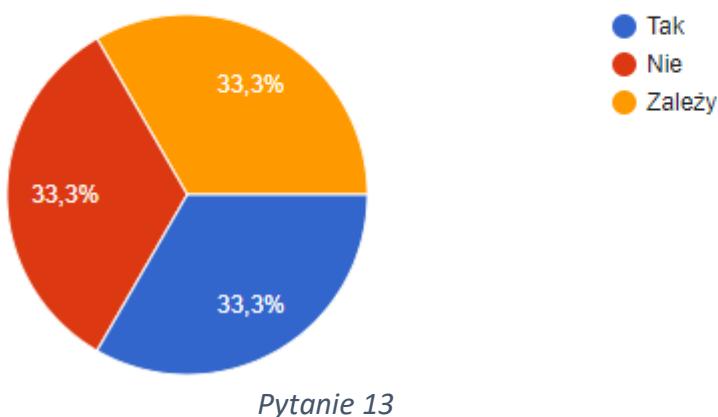


Pytanie 12

Tak jak zostało powiedziane wcześniej, technologia ta ma ogromny potencjał, a aplikacja pozwalająca wirtualnie zwiedzić pewien obszar świata wydaje się być dobrym rozwiązaniem.

Czy byłbyś w stanie zapłacić za możliwość wirtualnej wycieczki?

21 odpowiedzi



Czynniki od których zależy zostały przedstawione jako: jakość finalnego produktu, autentyczność wrażeń oraz cena.

8.2. Część 2 – pytania otwarte

Zostały zadane 3 pytania, a najbardziej powtarzające się wyrazy z odpowiedzi ankietowanych zostały zebrane w chmurę słów.

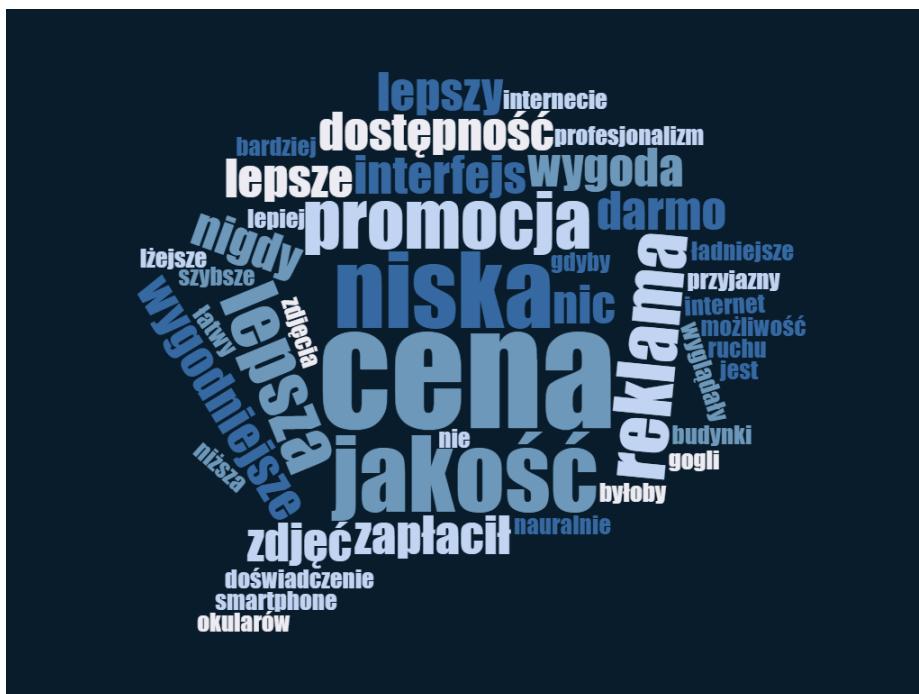
Jakie są Twoje odczucia po wirtualnym spacerze?



Rysunek 57 - chmura słów - pierwsze pytanie

Wśród badanych zwrócono uwagę na to, że aplikacja jest przyszłościowa i drzemie w niej potencjał, jednak zdania, co do komfortu z użytkowania zaprezentowanego rozwiązania, są podzielone. Niektórzy byli bardzo zadowoleni i określali doświadczenie jako ciekawą podróż do znanego świata, ale w oryginalnym wydaniu, zaś inni określili rozwiązanie jako niedopracowane i skarżyli się na mdłości. Jest to spowodowane niedoskonałością technologii, która jednak stabilnie rozwija się, aby jak najwierniej odwzorowywać rzeczywistość przy jak najmniejszej liczbie skutków ubocznych.

Co wpłynęłoby na to, że byłbyś w stanie zapłacić za takie doświadczenie?



Rysunek 58 - chmura słów - drugie pytanie

Duża część osób uzależnia się od potencjalnej ceny czy promocji, ale przede wszystkim zwracają uwagę na fakt, że potencjalny komercyjny projekt musiałby być po prostu lepszy – profesjonalne zdjęcia oraz wygodniejsze gogle to pierwsze rzeczy jakie przyszły na myśl ankietowanym.

Co Twoim zdaniem można by poprawić w prezentowanym rozwiązaniu?



Rysunek 59 - chmura słów - trzecie pytanie

Uzyskano tutaj wiele cennych wskazówek odnośnie poprawy aplikacji, oprócz wszystkiego „lepiej i bardziej” w kwestii sprzętowej ankietowani zwróciли uwagę na niezbyt intuicyjny system hotspotów w aplikacji Unity w postaci ptaków, zwróciли również uwagę, że wiarygodności wirtualnemu środowisku ujmują nie do końca dobrze sklejone zdjęcia oraz niewystarczająco ostry obraz.

8.3. Wnioski wynikające bezpośrednio z badania

Przede wszystkim przyjrzelismy sprite'om, które służyły jako znaczniki do przechodzenia do następnego obszaru. Początkowo zakładano, że pierwotne rozwiązanie (w postaci ptaków, zdjęcie poniżej) będzie eleganckie i oryginalne. Po przeprowadzeniu badań jednak spotkaliśmy się z wielką dezaprobatą i niezadowoleniem większości badanych. W myśl user experience, które mówi że doświadczenia powinny być jak najbardziej pozytywne, ptaki (Rys. 60) zostały zamienione na bardziej intuicyjne strzałki na ziemi (Rys. 61), przypominające te z Google Street View. Zamienienie znaczników na formę znaną każdemu z podobnych aplikacji spotkało się z pozytywnym odzewem i oceną.

Ponieważ, zamiarem jest rozwijanie aplikacji w przyszłości zdecydowano się uwzględnić wszystkie opinie, ale skupić się już na jednej technologii.

Wybrano Unity, ponieważ pozwala na lepsze dostosowywanie jej do potrzeb użytkowników, API od Google jest zamkniętym systemem i rozwój pewnych funkcjonalności mógłby być trudny, albo wręcz niemożliwy. Niestety, większość niedociągnięć to kwestia hardware'u, a nie software'u, co wymaga dużych nakładów finansowych i czekania na wyjście nowocześniejszych technologii.



Rysunek 60 - PRZED: Sprite'y służące do przejść jako ptaki na niebie



Rysunek 61 - PO: Sprite'y służące do przejść jako strzałki na ziemi

9. Perspektywy rozwoju i przyszłość wirtualnej/rozszerzonej rzeczywistości

Już w późnych latach 80 oraz wczesnych 90 modne były gogle VR, lecz wtedy jeszcze były toporne i mało przekonujące. To była pierwsza fala. Obecnie postęp rozwoju układów GPU i podobnych jest nieuchronny, nie tylko z powodu zapotrzebowania na wydajność gier/video, układy GPU oraz FPGA wykorzystuje się również w sieciach neuronowych oraz innych formach AI (ang. Artificial Intelligence). Google czy Facebook i ich centra danych nie są w stanie poradzić sobie ze wszystkimi zadaniami nie przenosząc większości obciążen związanych z ich przetwarzaniem na wspomniane układy. Dodajmy do tego rosnący w siłę rynek kryptowalut i wiele innych.

9.1. Rozwój wyświetlaczy

Wirtualna rzeczywistość, aby wiernie symulować środowisko spotyka się z problemem rozdzielczości – jest aktualnie za mała do tego celu. Nawet przy najlepszych układach graficznych VR wciąż wygląda zbyt nieostro. Jednak wydajność to jedno, dużą rolę odgrywają również same wyświetlacze, które mają pewne ograniczenia względem ilości pikseli, które fizycznie mogą reprezentować. Obecnie Google pracuje nad wyświetlaczem VR z 10x większą ilością pikseli niż dostępne na rynku google[34]. Współpracując z firmą Sharp chce stworzyć OLED-owy wyświetlacz VR o matrycy 20 Mpix. Nie jest to oczywiście jeszcze zagęszczenie pikseli, które pozwoli na symulowanie obrazu oka, ale jest to z pewnością krok w dobrą stronę i na pewno wniesie VR na wyższy poziom. 20 MP przy 90-120 klatkach wyświetlonego obrazu niesie za sobą oczywiście ogromne wymogi szybkości przesyłania danych w granicach 50-100 Gb/sekundę. Jedna z najbardziej aktywnych korporacji w branży – NVIDIA estymuje, że jesteśmy 2 dekady od poziomu rozdzielczości, który oszukałby nasze oczy, że symulowany świat jest, w prawdziwy[35]. NVIDIA pracuje obecnie również nad dodatkowymi technologiami jak tzw. „foveated rendering” [36], które może drastycznie zmniejszyć narzut obliczeniowy renderowania obrazu w wirtualnej rzeczywistości (Rys. 62). W skrócie jest to technika renderowania obrazu, która używa systemu śledzenia wzroku użytkownika, która renderuje tylko to, co koniecznie, zmniejszając jakość obrazu znajdującego się poza widzeniem peryferyjnym.



Rysunek 62 - foveated rendering [37]

9.2. Systemy śledzenia wzroku

W ostatnim roku japońska firma FOVE wydała pierwsze na świecie okulary VR ze wspomnianym systemem. Technologia ta zapowiedziała się na tyle obiecująco że wiele gigantów branży zakupiło start-upy zajmujące się rozwojem tej technologii, aby wyposażyc w nią swoje urządzenia - Facebook/Oculus zakupił Eye Tribe pod koniec 2016, Google wykupiło firmę zwaną Eyefluence, Apple nabyło SMI, a z kolei Intel zainwestował 4,6 miliona dolarów w AdHawk[39].

Jakie to ma znaczenie[38]? Przede wszystkim, wydajność – wspomniany foveated rendering i inne technologie mogą delegować zasoby tam, gdzie jest to faktycznie potrzebne. Takie używanie zasobów zmniejsza ogólną barierę wejścia (wydajności) w VR, a także daje twórcom większe pole manewru i możliwość tworzenia jeszcze wiarygodniejszych efektów wizualnych. Technologia taka może wiernie symulować skoncentrowanie wzroku, co bez wątpienia jest wizualnym usprawnieniem, które do tej pory było nieobecne. System śledzenie wzroku daje również nowe możliwości co do projektowania interfejsów użytkownika oraz UX. Obecnie w urządzeniach, które bazują na wyświetlaczu, kiedy wykonywana jest jakaś akcja, konieczna jest bezpośrednia interakcja. Wykonywane jest to to zazwyczaj poprzez dotyk obszaru na ekranie albo wskazując obiekty kursorem (używając np. myszki). Zanim to jednak nastąpi użytkownik patrzy na obiekt, z którym wchodzi w interakcję i w tym

momencie do akcji wchodzi wspomniana technologia. Eliminuje ona pośrednika i pozwala oddziaływać bezpośrednio. Da to nowe możliwości tworzenia interfejsów, które będą naturalne i niezwykle precyzyjne, efektywnie zastępując kursory i większość interakcji opartych na dotyku. Wzrokowa interakcja jest również dyskretna i może pozwolić na używanie immersyjnych urządzeń w ciasnych przestrzeniach publicznych itp.

Kolejną zaletą systemu śledzenia wzroku jest to, że daje twórcom dostęp do dużej ilości danych odnośnie użytkowania – będą wiedzieli na co użytkownik patrzy, co ignoruje w trakcie swojego doświadczenia, a także będą mogli zbadać zaangażowanie wskutek obserwacji i śledzenia źrenic. Rozszerzone ludzkie źrenice zdradzają m.in. psychiczne napięcie albo zaangażowanie emocjonalne. Idąc tą ścieżką być może będzie możliwość przewidywania akcji użytkownika **zanim ją wykona**. Te wszystkie informacje mogą pomóc zbudować immersyjne oprogramowanie które w 100% reaguje na emocje użytkownika.

Taki system pozwoli również na tworzenie nowych rodzajów interakcji i mechanik rozgrywki, które nie były nigdy wcześniej możliwe – wirtualne postaci, będą teraz np. wiedziały kiedy użytkownik na nie spojrzy, a kiedyś może nawet przeanalizuje gdzie patrzy i dlaczego. Użytkownicy będą mogli celować za pomocą oczu, podejmować decyzje w dialogach i znacznie wpływać na wirtualny świat wokół nich za pomocą podświadomych gestów. Otwiera to wiele możliwości dla kreatywnej narracji i projektowania interakcji (Rys. 63).



Rysunek 63 - Sterowanie za pomocą wzroku[40]

Technologia ta jest silnie rozwijana, do tej pory śledzenie wzroku polegało na wykonywaniu setek zdjęć na sekundę i określaniu pozycji źrenic. Często taki

sposób potrzebował dobrego oświetlenia, co powodowało opóźnienia. Wspomniane wcześniej AdHawk zajmuje się rozwijaniem technologii, która zastępuje aparat ultra kompaktowymi elektromechanicznymi systemami zwany MEMS, a te są tak małe, że aż niewidoczne przez ludzie oko. MEMS niweluje problem wymagającego przetwarzania obrazów, co przekłada się na szybkość, większą kompaktowość oraz energooszczędność urządzeń VR, które z nich korzystają. Podobnie jak inne technologie śledzenia wzroku AdHawk przewiduje gdzie użytkownik spojrzy do 50 milisekund wprzód. System MEMS posiada promień LED-owy, który skanuje oko i używając fotodiody wykrywa pozycję oka. Wszystko działa na CMOS-ie. Rozwiążanie pobiera bardzo niewiele energii, a projekt składa się z 2 układów i szeregowej magistrali (trójprzewodowy interfejs). Jest również bardzo lekki, co sprawia że jest idealny dla każdego urządzenia – gogli czy smartfonu. Koszty mają wynosić od 10 dolarów na oko.

9.3. Inne projekty

AdHawk ma w planach również inny projekt – trójwymiarowy sensor gestów. Sensor ten miałby projektować sferę (10 cm objętości) naokoło np. tarczy zegarowej smartwatcha. Ma rozpoznawać gesty do 25 mikronów rozdzielczości na wszystkich osiach X,Y,Z. Byłoby możliwe np. pisanie na wirtualnej klawiaturze na dowolnej powierzchni. Wiele rozwiązań zapożycza z ich technologii śledzenia oczu.

Obecnie również ciekawym projektem są bieżnie, które zapewniają nieograniczoną płaszczyznę, po której użytkownik może się poruszać, zapewniając jeszcze większą immersję i dając złudzenie faktycznego poruszania się po symulacji. Przykładem jest np. VirtuSphere[41], która jest sferą położoną na specjalnej platformie która pozwala jej się obracać we wszystkich kierunkach w zależności od ruchów użytkownika, który znajduje się wewnątrz (Rys. 64). Infinadeck to inny przykład, gdzie występują zwykła pozioma bieżnia, która porusza się we wszystkich kierunkach i wraca do środkowego położenia. Takie rozwiązania mogą również poszerzyć zakres wykorzystywania technologii VR do np. ćwiczeń czy treningów wojskowych.

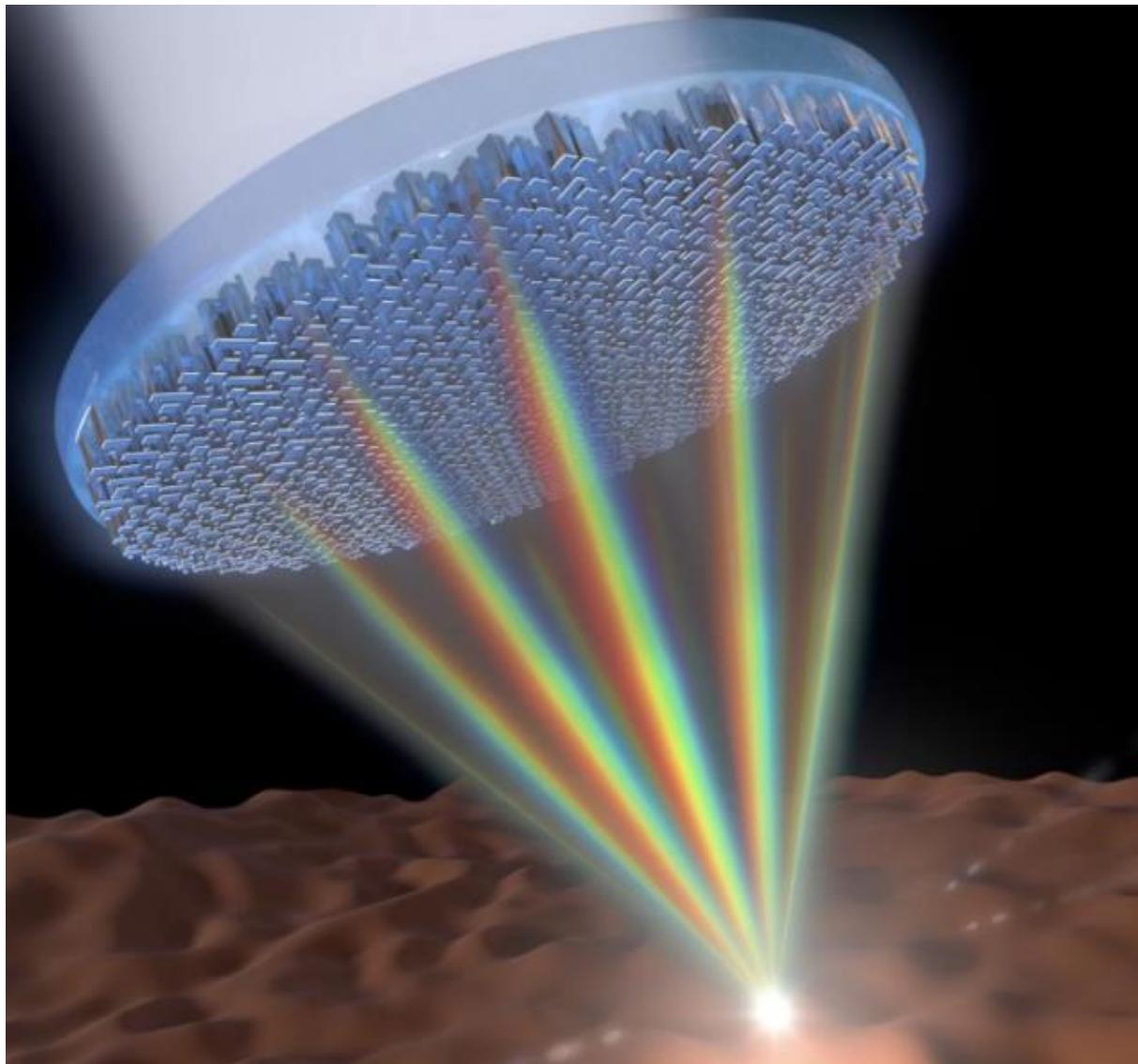


Rysunek 64 - VirtuSphere[41]

9.4. Metasoczewki

Ciekawym, i dość przełomowym ostatnim odkryciem w dziedzinie optyki było opracowanie przez grupę badaczy z Harvard John A. Paulson School od Engineering and Applied Science pierwszej pojedynczej soczewki, która potrafi skupić spektrum światła fal od 470nm do 650nm w tym samym miejscu i z wysoką rozdzielczością[42]. W konwencjonalnych soczewkach zostaje to osiągnięte przez układanie kilku soczewek obok siebie (Rys. 65). Metalenses – płaskie powierzchnie, które używają nanostruktur do skupiania światła być może zrewolucjonizują optykę zastępując wielkie zakrzywione soczewki używane w optyce. Do tej pory płaskie soczewki nie potrafiły skupiać światła spektrum zbyt dobrze, w efekcie występowało rozszczepienie światła. Nie jest to jeszcze produkt doskonały, gdyż widzialne spektrum mieści się w granicach 390 nm do 700nm (a soczewki skupiają zakres 470-650)[43], sprawność soczewek wynosi 20%, co efektywnie oznacza 80% utraty światła (soczewki fresnel w HTC Viveo charakteryzują się tą wartością na mniej niż 5%). Dodatkowo występuje wariacja pomiędzy sprawnościami dla różnych długości fal (dla 405nm 86% dla 660nm 66%) co znaczy, że niektóre kolory prezentują się jaśniej niż inne.

W kolejnym kroku badacze zamierzają zwiększyć ich średnicę do 1cm. Otworzyłoby to nowe możliwości zastosowania w aplikacjach zarówno VR jak i AR. Teoretycznie takie soczewki mogłyby pozbyć się efektu chromatycznej aberracji.



Rysunek 65 - soczewka Metalens[44]

10. Podsumowanie

Pracowano w środowisku Unity, które zawiera szereg wygodnych narzędzi, co ułatwiło proces wytwarzania oprogramowania, z drugiej strony biblioteki w języku JavaScript, również posiadają gotowe komponenty, które są standardowe dla aplikacji VR. Jednakże API dla drugiej aplikacji jest zamknięte i ma dość ograniczone możliwości w stosunku do rozwiązania napisanego dla silnika graficznego Unity.

Mimo wielu zalet podejście drugie czyli wersja w czystym JavaScript, okazała się lepsza w jakości stitchingu zdjęć, czyli łączenia ich w wirtualną sferę. Wynikało to prawdopodobnie ze złego doboru shadera dla sfery, co spowodowało lekkie zniekształcenia budynków.

Mysiąc przyszłościowo o wytwarzaniu aplikacji, które miałyby być bardziej elastyczne tzn. mające możliwość uruchomienia na różnych platformach systemowych uznano, że z pewnością lepsze byłoby środowisku Unity z racji opcji eksportu na pod 10 różnych platform.

Badanie user experience wyraźnie pokazało, że jest zainteresowanie wśród użytkowników wirtualną rzeczywistością, mimo pewnych niedoskonałości. Pozwoliło na jasne stwierdzenie, że wykorzystywane na początku modele ptaków służące do zmiany widoku, były nie intuicyjne, co od razu zmieniono. Zaprezentowany „spacer” może pozostawiać pewien niedosyt z powodu możliwości poruszania się na zasadzie zmian położenia kamery z jednej lokalizacji do drugiej oddalonych od siebie w sporej odległości. Generalnie można zaobserwować pozytywne nastawienie potencjalnych użytkowników do przedstawianego wirtualnego świata.

Po zbudowaniu dwóch aplikacji i ich porównaniu, stwierdzono, że zarówno zbudowanie aplikacji w postaci instalacyjnej na systemy Android jak i w wersji webowej, było w miarę łatwe i dające dużo satysfakcji. Testy wykazały, że opracowane rozwiązania są wydajne i zdecydowanie nadają się do użytkowania.

Podsumowując, dzięki realizowanej pracy autorzy mieli możliwość poznania środowiska Unity, które daje bardzo dużo możliwości. Uznano je za lepsze do tego typu aplikacji mimo drobnego problemu, nad którym autorzy zamierzają pracować. Twórcy nie chcą ograniczać się do spaceru po terenie AGH, w przyszłości chcą stworzyć aplikacje pozwalającą użytkownikom zwiedzić miejsca dotąd niezbadane lub niedostępne.

11. Spis rysunków i tabel

11.1. Rysunki

Rysunek 1 - park rozrywki z VR[2]	7
Rysunek 2 - aplikacja na smartphone'y o nazwie „Pokemon Go!”[4]	8
Rysunek 3 - zrzut ekranu z aplikacji Tłumacz Google, przedstawiający możliwości rozszerzonej rzeczywistości	9
Rysunek 4 - okulary Google Glass[6]	10
Rysunek 5 - Microsoft HoloLens[9]	11
Rysunek 6 - Leap One[10]	11
Rysunek 7 - Meta 2[11]	11
Rysunek 8 - koło kolorów RGB[16]	14
Rysunek 9 - tryb anaglifowy w grze Minecraft[17]	14
Rysunek 10 - skupienie soczewek 1 [18]	15
Rysunek 11 - skupienie soczewek 2 [18]	16
Rysunek 12 - barrel distortion na przykładzie zaimplementowanej aplikacji webowej	16
Rysunek 13 - naprawa chromatycznej aberracji [20]	17
Rysunek 14 - Google Cardboard [21]	20
Rysunek 15 - okulary FiiT VR 2S [22]	21
Rysunek 16 - zdjęcie sferyczne w formacie equirectangular	22
Rysunek 17 - statyw stabilizujący aparat[24]	23
Rysunek 18 - smartphone wykorzystywany do robienia zdjęć 360[25]	24
Rysunek 19 - scena w środowisku Unity 3D	26
Rysunek 20 - struktura folderów	28
Rysunek 21 - struktura sceny	29
Rysunek 22 - wirtualny statyw	30
Rysunek 23 - fader	30
Rysunek 24 - pusta sfera 3D z kamerą	31
Rysunek 25 - sfera wypełniona shaderem	32
Rysunek 26 - gotowa sfera, po wszystkich etapach budowania	32
Rysunek 27 - zdefiniowane pola w skrypcie	33
Rysunek 28 - metoda Awake()	34
Rysunek 29 - metoda ChangeSphere()	34
Rysunek 30 - metoda FadeCamera()	35
Rysunek 31 - metody FadeIn() oraz FadeOut()	35
Rysunek 32 - metoda Update()	36
Rysunek 33 - przedstawienie GazeClick	36
Rysunek 34 - okno przedstawiające ustawienia Event Trigerra	37
Rysunek 35 - okno eksportu, budowania pliku .apk	38
Rysunek 36 - przedstawienie występujących w projekcie strzałek do zmiany położenia; w lewym okienku widok sceny, po prawej widok gry	39
Rysunek 37 - przedstawienie funkcjonalności aplikacji	39
Rysunek 38 - widok z aplikacji zainstalowanej na smartphone'ie z systemem operacyjnym Android ..	39
Rysunek 39 - kod HTML z dołączonymi skryptami i zadeklarowanym kontenerem na zawartość	40
Rysunek 40 - funkcja wywołująca konstruktor po załadowaniu strony	41

Rysunek 41 - przechowywanie informacji o obiektach	41
Rysunek 42 - obsługa eventów 1/2	42
Rysunek 43 - obsługa eventów 2/2	42
Rysunek 44 - ładowanie nowej sceny.....	43
Rysunek 45 - dodawanie hotspotów do ładowanej sceny	44
Rysunek 46 - obsługa kliknięcia.....	44
Rysunek 47 - obsługa kliknięcia 2/2	44
Rysunek 48 - tryb desktopowy (poruszanie za pomocą kurSORA) – aplikacja webowa	45
Rysunek 49 - tryb VR – aplikacja webowa	45
Rysunek 50 - przedstawienie skoku prędkości sieci podczas testowania aplikacji webowej	48
Rysunek 51 - przedstawienie zasobów w czasie bezczynności smartphone'a.....	49
Rysunek 52 - wykresy zużycia zasobów przez aplikację napisaną w środowisku Unity	49
Rysunek 53 - wykresy zużycia zasobów przez aplikację napisaną w JavaScript	49
Rysunek 54 - porównanie jakości Unity - po lewej, Web - po prawej.....	50
Rysunek 55 - kompresja 1024 KB - po lewej, oraz jej brak - po prawej	51
Rysunek 56 - po lewej - Unity, po prawej - Web	51
Rysunek 57 - chmura słów - pierwsze pytanie	60
Rysunek 58 - chmura słów - drugie pytanie	61
Rysunek 59 - chmura słów - trzecie pytanie.....	62
Rysunek 60 - PRZED: Sprite'y służące do przejść jako ptaki na niebie	63
Rysunek 61 - PO: Sprite'y służące do przejść jako strzałki na ziemi	63
Rysunek 62 - foveated rendering [37]	65
Rysunek 63 - Sterowanie za pomocą wzroku[40]	66
Rysunek 64 - VirtuSphere[41].....	68
Rysunek 65 - soczewka Metalens[44]	69

11.2. Tabele

Tabela 1 - przedstawia porównanie istotnych parametrów sprzętowych.....	19
Tabela 2 - przedstawienie specyfikacji technicznej komputerów testujących	46
Tabela 3 - przedstawienie zużycia zasobów przez urządzenia w stanie spoczynku (wartości podawane w procentach, Rys. 51)	46
Tabela 4 - przedstawienie zużycia zasobów przez urządzenia podczas uruchomionej aplikacji Unity (wartości podawane w procentach; Rys. 52)	47
Tabela 5 - przedstawienie zużycia zasobów przez urządzenia podczas uruchomionej aplikacji JavaScript (wartości podawane w procentach; Rys. 53)	47

12. Bibliografia

- [1] <https://www.merriam-webster.com/dictionary/virtual%20reality>
- [2] <http://slqjvideo.com/photos/vr-theme-park>
- [3] <https://www.merriam-webster.com/dictionary/augmented%20reality>
- [4] <https://apynews.pl/friz-pierwszy-40-poziom-pokemon-go-wywiad>
- [5] <http://www.komputerswiat.pl/nowosci/sprzet/2017/29/google-glass-powracaja-jako-enterprise-edition.aspx>
- [6] <https://www.x.company/glass/>
- [7] https://en.wikipedia.org/wiki/Mixed_reality
- [8] <https://www.techopedia.com/definition/32501/mixed-reality>
- [9] <https://hololens.reality.news/how-to/buy-microsoft-hololens-0169192/>
- [10] <https://www.spiria.com/en/blog/tech-news-brief/magic-leap-one>
- [11] <https://www.roadtovr.com/meta-2-ar-glasses-available-to-pre-order-1440p-with-90-degree-fov-for-949/>
- [12] <https://pl.wikipedia.org/wiki/Stereoskopia>
- [13] <https://sjp.pwn.pl/slowniki/widzenie%20stereoskopowe.html>
- [14] <https://pl.wikipedia.org/wiki/Stereogram>
- [15] <https://pl.wikipedia.org/wiki/Anaglif>
- [16] https://upload.wikimedia.org/wikipedia/en/7/73/Simple_RGB_color_wheel.png
- [17] <http://www.minecraftforum.net/forums/minecraft-java-edition/survival-mode/215125-beta-3d-anaglyph-screenshots-videos>
- [18] <http://smus.com/vr-lens-distortion/>
- [19] <https://encyklopedia.pwn.pl/haslo/aberracja-chromatyczna;3865138.html>
- [20] <https://cdn.photographylife.com/wp-content/uploads/2011/11/Uncorrected-and-Corrected-CA.jpg>
- [21] https://vr.google.com/intl/pl_pl/cardboard/get-cardboard/
- [22] <http://www.hypergridbusiness.com/2016/08/review-fiit-vr-2s-is-my-new-favorite/>
- [23] <https://play.google.com/store/apps/details?id=com.google.android.street>
- [24] <http://images.okazje.info.pl/p/fotografia/5934/opticam-first-c-3570.jpg>
- [25] http://www.xiaomi4you.pl/wp-content/uploads/2017/04/213201714746PM_635_xiaomi_redmi_note_4x_matte_black.jpeg
- [26] <https://unity3d.com/unity>
- [27] <https://github.com/googlevr/gvr-unity-sdk/releases/tag/v1.110.0>
- [28] <https://unity3d.com/learn/tutorials/topics/graphics/gentle-introduction-shaders>
- [29] <https://docs.unity3d.com/Manual/SL-ShadingLanguage.html>
- [30] <https://docs.unity3d.com/Manual/UICanvas.html>
- [31] <https://developers.google.com/vr/develop/unity/controller-support>
- [32] <https://docs.unity3d.com/ScriptReference/MonoBehaviour.StartCoroutine.html>
- [33] <https://developers.google.com/vr/develop/web/vrview-web>
- [34] <https://www.roadtovr.com/google-developing-vr-display-10x-pixels-todays-headsets/>
- [35] <https://uploadvr.com/nvidia-estimates-20-years-away-vr-eye-quality-resolution/>
- [36] https://en.wikipedia.org/wiki/Foveated_rendering
- [37] <https://cdn.uploadvr.com/wp-content/uploads/2016/01/SMI-Foveated-Rendering.jpg>

- [38] <https://medium.com/futurepi/why-eye-tracking-is-a-huge-deal-for-vr-ar-683e971652ee>
- [39] <http://www.tomshardware.co.uk/intel-invests-4.6m-adhawk-eye-tracking,news-57098.html>
- [40] <http://www.startlr.com/wp-content/uploads/2016/12/the-eye-tribe-1.jpg>
- [41] <https://en.wikipedia.org/wiki/VirtuSphere>
- [42] <https://www.seas.harvard.edu/news/2018/01/single-metalens-focuses-all-colors-of-rainbow-in-one-point>
- [43] <https://www.ncbi.nlm.nih.gov/pubmed/27257251>
- [44] <https://www.livescience.com/61311-metalens-visible-light.html>