

**WYDZIAŁ INŻYNIERII METALI I INFROMATYKI STOSOWANEJ**

KATEDRA INFORMATYKI STOSOWANIEJ I MODELOWANIA

**PRACA INŻYNIERSKA**

Wirtualny spacer po terenie AGH z wykorzystaniem technologii VR.

*Virtual walking around the AGH using VR technology.*

Autorzy: *Paweł Brzoza, Marcin Szumlański*

Opiekun pracy: *dr inż. Tomasz Dębiński*

Kierunek studiów: *Informatyka Stosowana*

*Kraków 2018*

Oświadczam, świadomy (-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem (-am) osobiście i samodzielnie i że nie korzystałem (-am) ze źródeł innych niż wymienione w pracy.

Paweł Brzoza

Marcin Szumlański

***Spis treści:***

1. Wprowadzenie (wspólnie)
   1. Cele i zakres pracy
   2. Streszczenie kolejnych rozdziałów
2. Świat wirtualny, a rzeczywisty oraz wirtualna a rozszerzona rzeczywistość (M. Szumlański)
   1. Niezbędne definicje
   2. Historia
   3. Obszary zastosowań
   4. Przedstawiany typ wirtualnej rzeczywistości
3. Zdjęcia sferyczne – pozyskiwanie i właściwości (M. Szumlański)
4. Przegląd dostępnych rozwiązań technologicznych oraz wybór wykorzystywanego sprzętu (P. Brzoza)
   1. Przegląd rynku
   2. Inne możliwości zastosowania
   3. Charakterystyka sprzętowa wybranych urządzeń
      1. Xiaomi Redmi Note 4X
      2. Sony Xperia Z1
   4. Opis wybranych okularów VR
      1. Google Cardboard
      2. FiiT VR 2S
5. Praca z silnikiem graficznym Unity3D (P. Brzoza)
   1. Niezbędne elementy instalacyjne
      1. Unity3D
      2. Android Studio SDK
      3. Java Development Kit
   2. Poszczególne komponenty środowiska
   3. Składniki projektu
   4. Struktura sceny
   5. Elementy biblioteki Google VR SDK
   6. Implementacja mechaniki przejść i funkcji GazeClick
   7. Export aplikacji na urządzenia z systemem Android
      1. Niezbędne ustawienia
      2. Budowanie aplikacji
6. Badanie dostępnych typów teksturowania sfer (M. Szumlański)
   1. Opis Shaderów
   2. Porównanie różnych jakości teksturowania oraz ich kompresji
7. Aplikacja wykonana przy użyciu API VR View w Javascript (M. Szumlański)
8. Porównanie wydajności dwóch aplikacji (M. Szumlański)
9. Badania „user experience” (M. Szumlański)
10. Perspektywy rozwoju i przyszłość wirtualnej/rozszerzonej rzeczywistości (M. Szumlański)
11. Podsumowanie (wspólnie)
    1. Rozwiązanie podziału pracy
12. Spis ilustracji

Bibliografia

1. **Wprowadzenie**

Na przestrzeni ostatnich lat nie sposób nie zauważyć ogromnego postępu w przedstawianej dziedzinie tj. wirtualnej rzeczywistości (ang. virtual reality). Dzięki łatwości użycia i coraz większej dostępności sprzętu do obsługi VR, ludzie zaczynają eksperymentować z prostymi aplikacjami czy filmikami dającymi szansę przetestować wirtualny świat. Przedmiotem niniejszej pracy jest rozpoznanie możliwości dostępnych na rynku związanych z tematem, a następnie implementacja aplikacji do wirtualnego „spaceru” po terenie AGH na platformę Android oraz wersję przeglądarkową. Przeanalizowano różnice pomiędzy różnymi typami wirtualnej rzeczywistości, następnie przedstawiono proces budowania aplikacji w środowisku Unity jak i poza nim wykorzystując czysty JavaScript, wszystko to wspomagając się API od Google. Na koniec zaprezentowano wyniki, czyli porównanie możliwości pomiędzy dwoma rodzajami implementacjami i wyciągnięto odpowiednie wnioski dotyczące m.in. wykorzystania mocy obliczeniowej sprzętu czy wygody użycia.

* 1. **Cele i zakres pracy**

Głównym celem pracy było zapoznanie się z technologią VR, następnie rozpoznanie rozwiązań aktualnie dostępnych na rynku i wybranie możliwie niskobudżetowego zestawu, który w pełni umożliwi korzystanie z przedstawianej technologii. Umożliwiło to wybranie odpowiedniego sprzętu oraz oprogramowania do tworzenia wirtualnej rzeczywistości po czym, stworzono aplikacją.

Implementacja programu występuje w 2 wersjach. Pierwsza zaimplementowana za pomocą silnika graficznego Unity3D, a druga przy pomocy języka skryptowego JavaScript, wszystko po to aby dobrze zbadać możliwości i uzyskać jak najlepszy efekt końcowy.

* 1. **Streszczenie kolejnych rozdziałów**

1. **Świat wirtualny, a rzeczywisty oraz wirtualna a rozszerzona rzeczywistość** (M. Szumlański)

**Rzeczywistość wirtualna**

Wszystkie technologie „przerabiające” rzeczywistość zmieniają w pewnym sensie sposób w jakim postrzegamy rzeczywistość, ale rzeczywistość wirtualna (VR – Virtual Reality) całkowicie zmienia środowisko wokół nas. Merriam-Webster definiuje rzeczywistość wirtualną jako sztuczne środowisko które jest przeżywane za pomocą bodźców (jak wzrok/słuch) dostarczone przez komputer i w którym akcje użytkownika częściowo determinują co stanie się w tym otoczeniu.

To, w jaki sposób wchodzimy w interakcję z otoczeniem zależy jednak sporo od platformy której używamy. 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 przyczepiony do głowy użytkownika (trochę jak gogle) i zasłania jego dużą część pola widzenia „zanurzając” go w wirtualnym 360 stopniowym świecie.

Oculus Rift jest dobrze znany jako jeden z najbardziej popularnych zestawów VR aktualnie dostępnych masowo, 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 który zakłada się na głowę. Podobnie działa PlayStation VR oraz HTC Vive.

Na drugim końcu mamy np. Google Cardboard i (trochę od niego solidniejszy) Samsung Gear VR, które oferują podobne doświadczenia – niższej jakości, ale również niższej ceny. Plusem tych urządzeń jest z pewnością to że nie muszą być okablowane i podłączone do komputera. Zamiast tego do stworzenia doświadczenia VR używają smartfona podłączonego do gogli które zakładamy na głowę.

[[tu bedzie dokładniejszy opis gogli, jak działają, budowa itd]]

Ze względu na tą prostotę i niski koszt zdecydowaliśmy się wybrać tą technologię do stworzenia w nim naszego projektu.

Istnieją również bardziej rozbudowane technologie VR, które próbują stworzyć hiperrealistyczne środowisko które pozwala na fizyczne poruszanie się i manipulację środowiskiem, korzystają z tego np. wirtualne parki rozrywki. W takim parku ujrzymy wirtualny świat, ale ponieważ nie jesteśmy uwiązani do komputera możemy fizycznie chodzić po przestrzeni wokół nas, która często jest po prostu gołymi ścianami lub atrapami w które wdmuchiwane jest życie dopiero w VR. Użytkownik sam jest niejako kontrolerem i może zanurzyć się całkowicie w wyrenderowanym świecie, który jest mu przedstawiony, wchodząc w interakcję z otoczeniem. Często można natknąć się również np. na sztuczny descz, ciepło/zimno w takim miejscu aby gra była jeszcze bardziej interaktywna i immersyjna. Jak nietrudno się domyśleć rozwój technologii VR przyczynił się do wzrostu popularności gier itp.

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.

**Rzeczywistość rozszerzona**

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

Rzeczywistość rozszerzona może przyjąć wiele form, ale jej najbardziej powszechne użycie jest w smartfonach. Najbardziej znaną obecnie aplikacją jest chyba gra Pokemon GO! w której łapiemy wirtualne pokemony wesoło biegające sobie po naszym sąsiedztwie. Istnieją również aplikacje które na bieżąco, po skierowaniu kamery w odpowiednie miejsce pokazują informacje o pobliskiej restauracji itp. Wszystkie te aplikacje używają aparatu smartfona aby pokazywać wygląd na żywo tego co jest wokół nas, ale nakładają informacje na wyświetlacz. AR znajdziemy w wielu innych aplikacjach na smartfony, tłumacz google pozwala po skierowaniu kamery na tekst w jednym języku przetłumaczyć i zamienić go na inny język. Jest ona dostępna również na Google Glass. Google Glass było jednym z pierwszych urządzeń który wykorzystywało do korzystało z AR poza smartfonami, ale stało się źródłem kontrowersji z powodu obaw dotyczących prywatności, wysokiej ceny, małej ilości istotnych funkcjonalności. Jednak porażka google glass nie przeszkadza innym firmom w próbach popularyzacji tej bardziej pasywnej metody interakcji (względem chociażby VR) i tchnięcia nowego życia w AR – są to np LAFORGE Optical albo Optinvent. Niektórzy uważają że urządzenia AR jak Google Glass nigdy nie osiągną sukcesu, ale to pokaże już tylko czas.

Ogólnie rzecz ujmując, o rozszerzonej rzeczywistości można myśleć jak o nowej warstwie nałożonej na istniejącą rzeczywistość ale nie przemieszanej z nią. Okulary być może zdecydowanie podpadają pod tą kategorię ale to smartfony znajdują w AR największe zastosowanie – jako nakładka na pole widzenia smartfona, bez interakcji jako część większego środowiska.

**Rzeczywistość mieszana**

Rzeczywistość mieszana (MR – Mixed Reality) z drugiej strony, zabiera AR na nowy poziom i jest w zasadzie tym, czym wszyscy początkowo chcieli lub oczekiwali od AR. Zamiast

[[tu będzie jeszcze tekst o MR]]

1. **Zdjęcia sferyczne – pozyskiwanie i właściwości** (M. Szumlański)

**Zdjęcie sferyczne, media 360**

Zdjęcie sferyczne (inaczej fotosfera, zdjęcie 360) – zdjęcie uchwycające 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, lub łączyć ze sobą komputerowe efekty i fotografię. 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 oczywiście płaskie, a efekt wirtualnej rzeczywistości uzyskuje się odpowiednio je przekształcając. Przykładowe wykonane zdjęcie przez nas (format equirectangular) wygląda tak:



Istnieją również inne formaty tzw. niepełne które nie pokrywają całej sfery, jednak w naszej pracy nie są nam potrzebne, wręcz niepożądane.

**Wykorzystany sprzęt**

Do zrobienia zdjęć wykorzystano aplikację Google Street View (<https://play.google.com/store/apps/details?id=com.google.android.street>), statyw FIRST C-3570 oraz telefon Xiaomi Redmi Note X4.

Aplikacja Google Street View to aplikacja od firmy Google która m.in. pozwala na dość łatwe robienie zdjęć sferycznych oraz udostępnianie ich. Cały proces jest dość prosty, aplikacja przy użyciu żyroskopu w telefonie i po określeniu lokalizacji naznacza nam 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 zdjęcia to staje się punktem odniesienia i wokół niego pojawiają się kolejne punkty, które po skupieniu się na nich robią kolejne zdjęcia. W efekcie, jeżeli poruszamy się po osiach mając kamerę w jednym miejscu możemy zapełnić naszą wirtualną grafikę 3D 360 stopni mozaiką zdjęć. Kiedy wykonamy wszystkie zdjęcia (w przypadku naszego aparatu kilkadziesiąt) poddawane one są scaleniu do zdjęcia sferycznego.

W naszym przypadku wykorzystano telefon Xiaomi Redmi Note X4 o matrycy 13 Mpix, ogniskowej 4mm oraz z przysłoną f/2.0, a także statyw FIRST C-3570.

Statyw bardzo ułatwił umiejscowienie punktu orientacji w jednym miejscu dzięki czemu zdjęcia wyszły stabilniej i aplikacja lepiej je scaliła. Porównanie gotowej zrobionej „z ręki” oraz przy użyciu statywu poniżej.

[[tutaj będa zdjęcia porównawcze z biblioteki]]

Gdyby wybrać potencjalne ulepszenie całego procesu bez wątpienia telefon możnaby wyposażyć 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.

1. **Przegląd dostępnych rozwiązań technologicznych oraz wybór wykorzystywanego sprzętu** (P. Brzoza)
   1. **Przegląd rynku**

Research dotyczący innych okularów oraz innych środowisk tworzenia technologii VR.

* 1. **Inne możliwości zastosowania**
  2. **Charakterystyka sprzętowa wybranych urządzeń**
     1. **Xiaomi Redmi Note 4X**
     2. **Sony Xperia Z1**
  3. **Opis wybranych okularów VR**
     1. **Google Cardboard**
     2. **FiiT VR 2S**

1. **Praca z silnikiem graficznym Unity** (P. Brzoza)
   1. **Niezbędne elementy instalacyjne**
      1. **Unity3D**

Zintegrowane środowisko programistyczne (ang. IDE, Integrated, Development Environment), jakim jest Unity, pozwala tworzyć nie tylko gry komputerowe (z czym może być kojarzone), ale również pozwala na kreowanie różnych wizualizacji, animacji czy materiałów interaktywnych. Wszystko to może zostać stworzone na płaszczyźnie dwuwymiarowej 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 stacjonarne – Windows, Mac, Linux

- konsole do gier – PlayStation 4, PS Vita, Xbox One

* + 1. **Android Studio SDK**

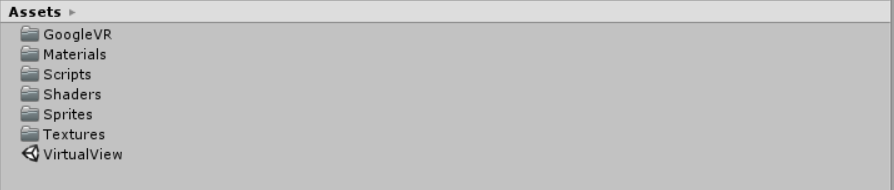
Opis ważnych punktów instalacji. Zwrócenie uwagi na instalację odrębnych wersji SDK przez . Android 4.4 oraz Android API 27. Użyteczność w Unity.

* + 1. **Java Development Kit**

Opis i użyteczność w Unity.

* 1. **Poszczególne komponenty środowiska**
  2. **Składniki projektu**

Powstanie projektu rozpoczęło się od stworzenia odpowiedniej struktury, czyli podziału na foldery zawierające odpowiednie moduły, z których powstał projekt. Przedstawiono kawałki screenów ze środowiska Unity3D:



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 – zaimportowane elementy z biblioteki GoogleVR SDK, będące dodatkowym wsparciem dla projektu, ważne skrypty to: GvrEditorEmulator, GvrControllerMain, GvrEventSystem i GvrReticlePointer, które dokładniej zostaną opisane w specjalnym pod rozdziale,

- Materials – stworzone „materiały”, czyli elementy graficzne kontrolujące modele 3D, zawierające w sobie niżej opisane Shadery

- Scripts – skrypty, odpowiadające głównie za mechanikę programu

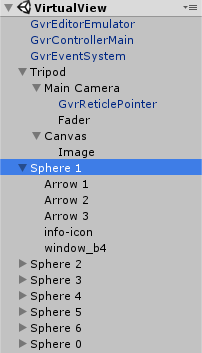
- Shaders – folder zawierający tzw. Shadery, czyli kod napisany w języku HLSL (High-Level Shading Language) lub CG (C for Graphics), które są do siebie bardzo zbliżone. Służą między innymi do opisu sposobu teksturowania modeli.

- Sprites – elementy, które rozszerzające rzeczywistość takie jak: strzałki wskazujące kierunek, w którą stronę się udamy, ikony o informacjach, okienka informacji o budynkach oraz wskaźniki w postaci okręgów ułatwiające określenie czasu w momencie wykonywania tzw. GazeClicku

- Textures – tekstury, czyli w naszym wypadku zdjęcia sferyczne

- VirtualView – scena zawierająca obiekty „gry”

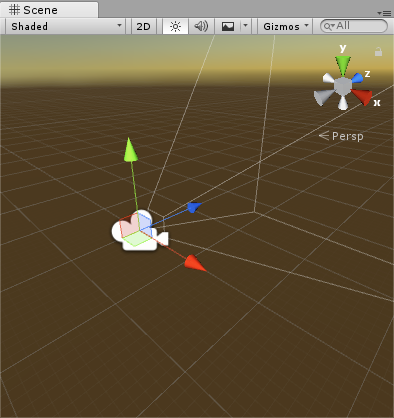
* 1. **Struktura sceny:**



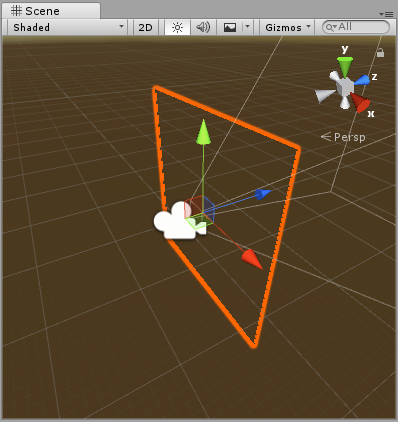
Na strukturę sceny składają się kolejno:

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

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



Ta z kolei zawiera w sobie skrypt służący do wskazywanie na inne obiekty oraz obiekt 3D typu czworokątnego (ang. Quad, Rys.x) o nazwie Fader, który posłuży do ciekawszych przejść widoków kamery.

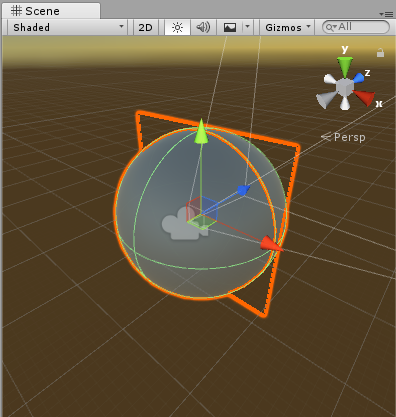


Statyw zawiera, również strefę Canvas, gdzie powinny znajdować się wszystkie elementy związane z interfejsem użytkownika (ang. UI). W naszym przypadku jest to komponent typu Image odpowiadający za tzw. GazeClick (dokładniej opisany w osobnym podrozdziale)

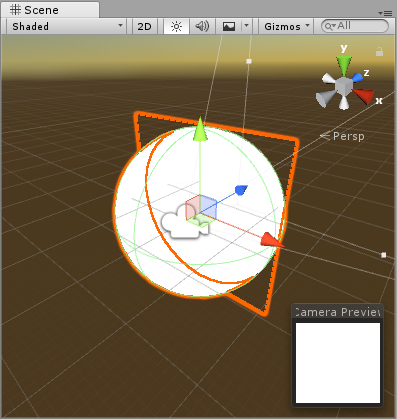
Siedem sfer – z czego każda, zawiera strzałki, oznaczające kierunki przemieszczania oraz ikony, dzięki którym, po najechaniu na nie, wyświetlana zostaje informacja o danym obiekcie.

Wszystkie sfery tworzone są analogicznie do opisanej w następujący sposób:

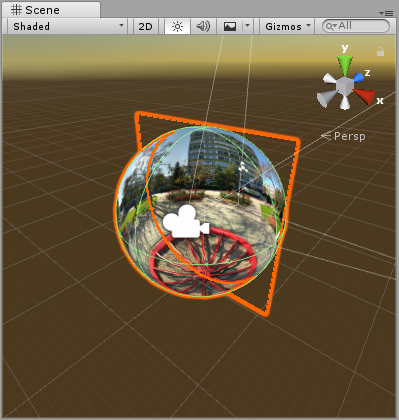
Do naszego statywu z kamerą dodatkowo utworzono obiekt 3D typu sfera (ang. Sphere, Rys.x)



Po utworzeniu sfery dodano Shader(opisany w pracy w osobnym podrozdziale) o nazwie Insideout, który umożliwi odpowiednie teksturowanie sfery oraz obiekt nie będzie wymagał dodatkowego oświetlenia, domyślne renderujący siatkę na biało (Rys.x)



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

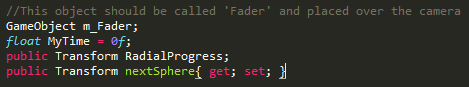


* 1. **Elementy biblioteki Google VR SDK**
* GvrEditorEmulator -
* GvrControllerMain -
* GvrEventSystem -
* GvrReticlePointer -
  1. **Implementacja mechaniki przejść i funkcji GazeClick**

Jeśli chcemy uzyskać dodatkowe efekty specjalne potrzebujemy do tego dodatkowe skrypty, które w naszym projekcie pozwolą na zaprogramowanie na przykład ciekawszych przejść między poszczególnymi sferami, czy wygodnej opcji przejścia między danymi sferami. Na potrzeby projektu stworzono skrypt obsługujący obie te możliwości. W dalszej części pracy zaprezentowane zostały kawałki kodu skryptu o nazwie „SphereChangerWithGazeClick.cs” wraz z ich opisem.

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

Wszystko zaczyna się od zdefiniowania odpowiednich pól:



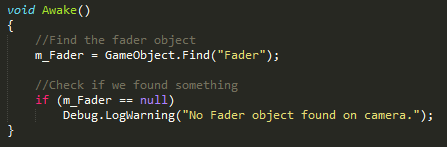
- m\_Fader obiekt klasy GameObject obsługuje wcześniej stworzony obiekt gry o nazwie Fader,

- MyTime pole typu zmiennoprzecinkowego, float, odpowiada za odmierzanie czasu

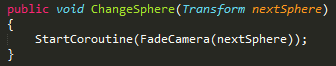
- RadialProgress obiekt klasy Transform obsługuje komponent Image będący edytowany w czasie

- nextSphere obiekt typu Transform, zawiera charakterystyczny dla języka C# getter i seter, odpowiada za zmianę sfery

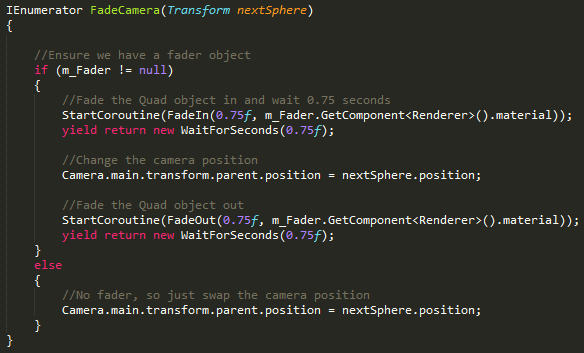
Następnie metoda typu void Awake(), tworzona w momencie tworzenia obiektu na scenie, wykonuje się gdy obiekt zostaje aktywowany. Traktowana jest jako konstruktor:



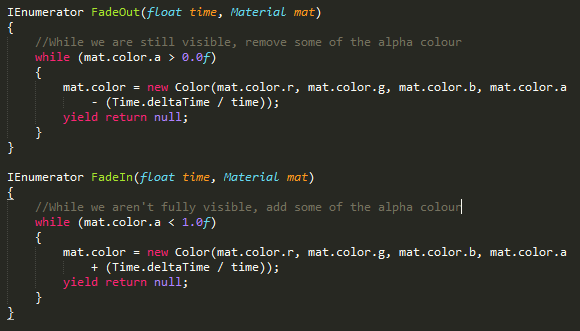
Zawiera w sobie przypisanie znalezionego obiektu Fader oraz instrukcje warunkowa, która dla przypadku nie znalezienia obiektu wypisuje w konsoli Unity ostrzeżenie.



Metoda ChangeSphere typu void zawierająca jeden argument o nazwie nextShpere klasy Transform. Jej wywołanie odbywa się w metodzie Update(dokładniej wyjaśniona przy opisie GazeClick) podczas przekroczenia czasu 3 sekund. ChangeSphere wywołuje metode StartCoroutine klasy MonoBehaviour, która zaś przyjmuje wywołanie innej metody o nazwie FadeCamera powodująca oczekiwany efekt zakrycia i odsłonięcia kamery. Między tymi zdarzeniami występuje zmiana pozycji kamery na nową nextSphere wskazaną w środowisku Unity.



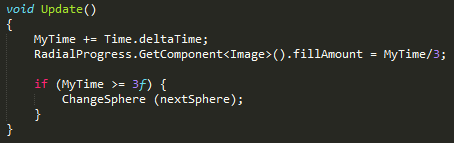
FadeCamera, metoda typu IEnumerator sprawdza czy pobranie obiektu typu Fader powidoło się jeśli tak to wywołuje kolejne współprogramy, które odpowiadają za zmianę, w przypadku nie powodzenia zwyczajnie zmienia sferą bez specjalnego przejścia.



Przedstawiony kod skryptu zawiera dwie metody typu IEnumerator o nazwach FadeOut oraz FadeIn pobierające dwa argumenty typu float i klasy Material.

Dane metody odpowiadają za usuwanie w przypadku FadeOut i dodawanie w przypadku FadeIn, koloru alfa w obiekcie Fader, co prowadzi albo do całkowitej przeźroczystości obiektu albo ukazania się go. Kolor jest zmienny w czasie.

Drugą wprowadzonym udogodnieniem jest GazeClick, który po najechaniu na dany obiekt włącza dany skrypt



Update () – metoda jest wywoływana w każdej klatce jeśli MonoBehaviour jest włączone. Jest to najczęściej używana metoda służąca do obsługi jakichkolwiek zachowań w grze: przesunięcia, obracanie, itd. Dla potrzeb uzyskania określonego zachowania z każdym wykonaniem funkcji Update inkrementowana jest zmienna MyTime o dany czas uzyskany z metody deltaTime. Następnie zwiększany jest promień naszego okręgu i gdy czas przekroczy, bądź będzie równy 3 sekundy, nastąpi wywołanie metody ChangeSphere.

* 1. **Export aplikacji na urządzenia z systemem Android**
     1. **Niezbędne ustawienia**
     2. **Budowanie aplikacji**

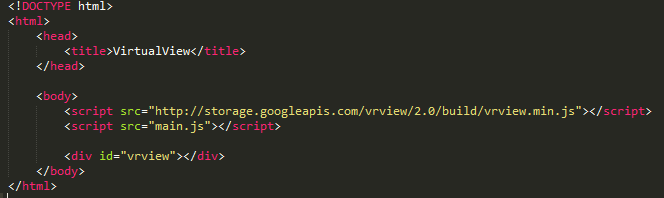
1. **Badanie dostępnych typów teksturowania sfer** (M. Szumlański)
   1. **Opis Shaderów**
   2. **Porównanie różnych jakości teksturowania oraz ich kompresji**
2. **Aplikacja wykonana przy użyciu API VR View w JavaScript** (M. Szumlański)

**Opis:**

Google udostępnia JavaScript API, które pozwala na wstawienie zdjęć oraz filmów 360 na stronę internetową tworząc i kontrolując zawartość elementu iframe lub jawnie deklarując element iframe.

**Implementacja:**

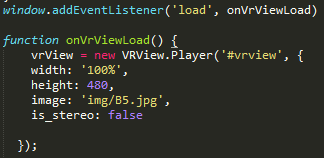
Aby w ogóle używać klasy VR View należy dołączyć źródło skryptu w kodzie HTML oraz stworzyć element div o id „vrview”, który po załadowaniu VR View zostanie zastąpiony elementem iframe o klasie „vrview”.



**VR View**

Kiedy strona się załaduje, chcemy zarejestrować to że się załadowała i wywołać funkcję która stworzy nową instancję VR View.

Instancję VR View tworzymy wołając konstruktor, VRView.Player i przekazując mu selektor który determinuje gdzie „wrzucić” VR View oraz przekazując mu kolekcję parametrów. Selektor powinien wskazywać na id elementu div którego zawarliśmy w HTML-u. Parametrami są m.in. źródło np filmu lub zdjęcia (video/image), wymiary iframe(width/height).



VR View rozgłasza 4 różne rodzaje zdarzeń, które muszą być zarejestrowane używając funkcji VRView.on(String event, function handleEvent). Funkcja on() przyjmuje dwa argumenty: nazwa wydarzenia w postaci Stringa oraz funkcja do obsługiwania go.

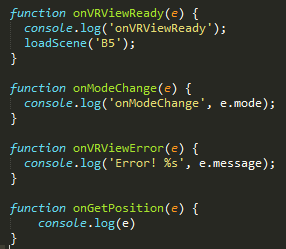
Typy wydarzeń to:

- ready : VR View załadował początkową zawartość i jest gotowy przyjmować komendy.

- error : VR View zgłasza błąd, np. kiedy zawartość nie może być załadowana .

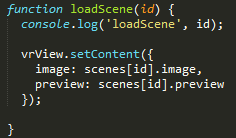
- click : VR View zarejestrował „klik” (tap na telefonie lub gaze w trybie VR). Może być użyte do przejścia do następnego slajdu w Carousel lub rejestrowania gdy użytkownik klika na hotspot.

- modechange : VR View zmienia tryb, np. kiedy użytkownik wchodzi w tryb VR lub pełny ekran. Można używać tego aby śledzić zachowanie użytkownika rejestrując wywołania zwrotne kiedykolwiek użytkownik zmienia tryby.



**Ładowanie nowej zawartości**

Nową zawartość można załadować do iframe bez przeładowywania samego iframe używając funkcji setContentInfo(). Funkcja ta przyjmuje URL do zdjęcia 360 lub filmu jao string i kolekcję innych parametrów. Zasadniczo może używać tych samych parametrów co konstruktor VR View, poza width i height.



**Hotspoty**

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

- Na desktopach, najechanie na hotspot zmienia je wizualnie a kliknięcie aktywuje.

- W trybie mobile tap na hotspoty aktywuje go

- w trybie VR, wpatrywanie się w fotosfere zawierająca hotspoty powoduje pojawienie się celownika, który zmienia stan gdy wcelujemy go bezpośrednio w hotspot, wtedy tap gdziekolwiek aktywuje hotspot.

Hotspoty to okrągłe znaczniki na sferze z nadanym ID (String), koordynatami środka, promieniem oraz odległością od kamery. Mogą być dodane poprzez funkcję vrView.addHotspot(). 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 górę. Yaw to zakres [-180, 180] a dodatnie wartości wskazują na prawo.

[[zdjecie kodu]]

Aby dodać wydarzenie, kiedy hotspoty zostanie aktywowany używam funkcji on().

[[zdjecie kodu]]

1. **Porównanie wydajności dwóch aplikacji** (M. Szumlański)

Porównanie wydajności(zasobożerności), jakości zdjęć i ogólnej wygody dla użytkownika.

1. **Badania „user experience”** (M. Szumlański)

Ankietowanie chętnych dot. Aplikacji i wygody jej użytkowania, jakości. Zbieranie najczęściej powtarzających się słów kluczowych.

Co można poprawić w aplikacji, co zostało poprawione po feedbacku od innych (m. In. zamiana ptaków na strzałki - bardziej intuicyjne)

1. **Perspektywy rozwoju i przyszłość wirtualnej/rozszerzonej rzeczywistości** (M. Szumlański)
2. **Podsumowanie**
   1. **Rozwiązanie podziału pracy**

Zarezentowana praca inżynierska została wykonana przez dwie osoby, wobec tego zastosowano podział pracy tak, aby zarówno jedna jak i druga osoba mogła w pełni zaprezentować swoje umiejętności inżynierskie.

1. **Spis ilustracji**

**Bibliografia**

Paul Neumann, [dostęp online: 19.09.2017], <http://paulneumann.blox.pl/2011/09/Swiat-realny-a-swiat-rzeczywisty.html>

<https://unity3d.com/unity>

<https://unity3d.com/learn/tutorials/topics/graphics/gentle-introduction-shaders>

<https://docs.unity3d.com/Manual/SL-ShadingLanguage.html>

<https://docs.unity3d.com/Manual/UICanvas.html>

<https://www.merriam-webster.com/dictionary/virtual%20reality>

<https://www.merriam-webster.com/dictionary/augmented%20reality>

https://en.wikipedia.org/wiki/Mixed\_reality