POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydzial Informatyki

Katedra Sieci Komputerowych
Sieci Urządzeń Moblinych

Kamil Warpechowski
Nr albumu 10709

Praca inżynierska
Promotor:
dr inż. Michał Tomaszewski

tutaj będzie zajebisty tytuł

Warszawa, 24 maja 2016

Spis treści

1 Wprowadzenie

Ludzie od lat przyzwyczaili się korzystać z elektroniki oraz internetu na standardowych urządzeniach elektronicznych. Na początku lat dziewięćdziesiątych do domów zaczęły trafiać komputery stacjonarne. Najpierw z modemami DLS, a następnie już ze stałymi łączami. Na przestrzeni lat korzystanie z ekranu w połączeniu z klawiaturą i myszą stało się dla ludzi naturalne.

Przez ostatnią dekadę na rynku pojawiły się interfejsy dotykowe. Popularność smartfonów, a następnie tabletów spowodowało, że coraz bardziej sporadyznie korzystamy z standardowej fizycznej klawiatury. Ekrany dotykowe pojawiły się nie tylko na urządzeniach telekomunikacyjnych, lecz także jako moniotory w komputerach pokładowych samochodów oraz instalowane są w zagłówkach w samolotach jako multimedialne centrum rozrywki [1].

Przez ostatnie kilka lat narasta trend poszukiwania innych metod dostępu do danych, zwłaszcza multimedialnych. Obecnie wiele przedsiębiorstw prowadzi badania nad nowymi, bardziej naturalnymi interfejsami, które nie wymagałyby użycia standarodwych urządzeń typu klawiatura czy myszka.

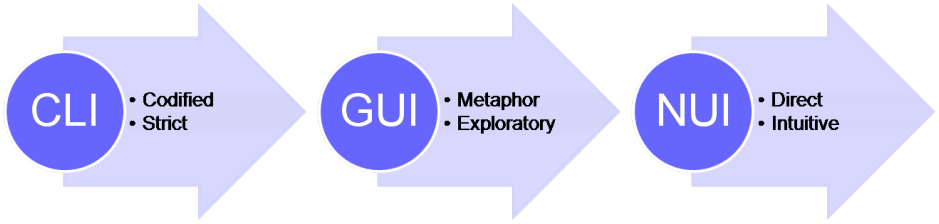
Wiele nowych urządzeń próbuje implementować sterowanie intefejsem użytkownika za pomocą gestów (np. Microsoft Kinect[2]), czy też za pomocą myśli (np. Emotiv[3]). Na rynku widać duże zainteresowanie nową formą kontroli urządzeniami, zwłaszcza tymi bardziej naturalnymi dla człowieka.

Na przestrzeni lat interfejsy urządzeń elektronicznych ewoluowały pierwotnie z aplikacji sterowanych za pomocą wiersza polecań poprzez programy z graficznym interfejsem użytkownika (Graphic Uuser Interface), aż do obecnie coraz bardziej popularnej i rozwijanej grupy interfejsów „naturalnych" (Natural User Interface).

[1] http://www.komputerswiat.pl/nowosci/wydarzenia/2012/28/boeing-z-androidem-na-pokladzie.aspx

[2] http://www.xbox.com/pl-PL/xbox-one/accessories/kinect-for-xbox-one

[3] http://emotiv.com



Rysunek 1: Ewolucja interfejsów użytkownika
źródło: https://en.wikipedia.org/wiki/Natural_user_interface

W branży filmowej oraz gier wideo narasta trend używania nowych technologii do rozszerzania doznań jakie otrzymuje odbiorca. W kinach odbywają są coraz częściej projekcje filmów stworzonych w technologii trójwymiarowej. Natomiast w ostatnim czasie pojawią się sale kinowe pozwalające na projecję filmów trójwymiarowych wraz z dodatkowymi elementami takimi jak: drganie foteli, wiatr, dym, woda [4]. Jednakże w obecnej chwili taki format rozrywki jest dość drogi, gdyż wymaga specjalnie przygotowanej sali kinowej oraz okularów, które pozwalają tworzyć iluzję przestrzenną.

1.1 Rozszerzona rzeczywistość

Coraz bardziej popularne staje się pojęcie rozszerzonej rzeczywistości (ang. augmented reality). Jest to zbiór różnych technologii pozwalającej łączyć świat rzeczywisty z wirtualnym. Jest to jezcze mało popularny sposób interakcji, lecz w ostatnim dziesięcioleciu rozwój (zarówno urządzeń jak i specjalistycznego oprogramowania) jest bardzo dynamiczny.

Pierwsze próby w tej dziedzinie odbywały się jeszcze w latach sześćdziesiątych amerykański naukowiec oraz artysta Myron Krueger prowadził badania nad wirtualną oraz rozszerzoną rzeczywistością.

[4] http://cinema-city.pl/4dx-info

dopisac dalej..

Natural Machines, Meta2, Wonderbook

Rysunek 2: Wizualizacja Microsoft HoloLens
źródło: https://www.microsoft.com/microsoft-hololens/en-us/why-hololens

2 Cel pracy

Celem niniejszej pracy jest stworzenie platformy do budowania gier oraz interaktywnych animacji prezentowanej za pomocą rozszerzonej rzeczywistości sterowaniej za pomocą zdalnego kontrolera.

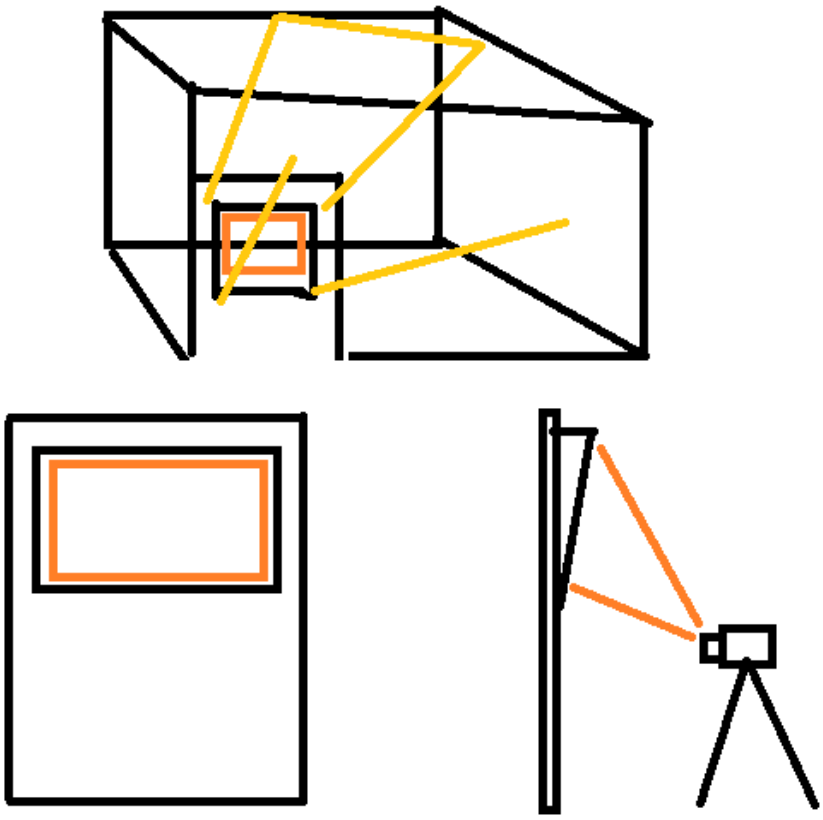
Przykłady zastosowanie zestawu aplikacji:

- Prezentacje przestrzeni architektonicznych
- Rozrywka
- Reklama miejscach użyteczności publicznych (np. centra handlowe)

3 Implementacje

3.1 Implementacja w technologii hologramu

Tutaj opisać próby i przemyślenia na temat hologramu na drzwiach.



Rysunek 3: Poglądowy schemat działania
źródło: własne

3.2 Implementacja w technologii mappingu 3d

Tutaj opisać próby i przemyślenia

4 Wykorzystane technologie

4.1 Unity

Unity jest obecnie najpopularniejszą platformą do tworzenia gier na wiele platform.

4.1.1 Dlaczego Unity

Najnowsza wersja posiada natywne wsparcie do rozszerzonej oraz wirtualnej rzeczywistości. Narzędzie te posiada prosty, ergonomiczny interfejs co ułatwia pracę.

Bardzo pomocnym dodatkiem do narzędzia jest „Assets Store". Jest to wirtualny sklep z komonentami do tworzenia gry. W projekcie zastosowałem tekstury i obiekty 3D pochodzące z tego źródła.

4.1.2 Alternatywne rozwiązania

	Unity	Unreal Engine
Wsparcie języków programowania	C#, JavaScript, Boo	c++
Obsługa wielu ekranów	Tak	Nie
Wsparcie dla Google Cardboard	Tak	Nie

Tablica 1: Porównanie silników gier

4.1.3 Wybór języka programowania

Środowisko Unity wspiera obsługę skryptów (animacje oraz logika biznesowa) w kilku językach programowania: C#, UnityScript (zmodyfikowana wersja JavaScript) oraz w przeszłości Boo. Podjęto decyzję, by w projekcie użyć język C#, gdyż ów język jest najbardziej stabilny, posiada najbardziej rozbudowaną dokumentację oraz jest to najpopularniejszy język w specjalistycznej literaturze. Dodatkowym udogodnieniem jest to, iż język posiada wiele wbudowanych klas (np. do obsługi połączeń TCP) oraz niezliczoną ilość zewnętrznych bibliotek.

4.1.4 Unity IDE

Środowisko Unity jest multiplatformowe. Aplikacje można używać na dowolnym sytemie operacyjnym. Jednakże edycja skryptów odbywa się za pomocą zewnętrznego narzędzia. W systemie Mac OS X jest to MonoDevelop, natomiast w systemie Windows jest to VisualStudio w wersji Community. Opisywana aplikacja początkowo była tworzona na systemie Mac OS X, jednakże kołopoty ze środowiskiem MonoDevelop spowodwały decyzje o przeniesieniu środowiska na system Windows. Subiektywnie mogę stwierdzić, że stabilność oraz komfort pracy jest dużo lepszy w systemie Windows. Dodatkową alternatywą dla MonoDevelop może okazać się Visual Studio Code. Jest to prosty multiplatformowy edytor posiadający obsługę języka C# .

4.1.5 Licencja i koszty

Unity jest zamkniętym, licencjonowanym oprogramowaniem. Darmowa wersja (Personal Editon) pozwala na nielimitowane użycie, jednakże jest to okrojona edycja. Szersze informacje o ograniczeniach wersji Personal zawarte są w kolejnych rozdziałach. Licencja pozwala na komercyjne użycie przy limicie zarobków na poziomie stu tysięcy dolarów. Komercyjna wersja (Professional Edition) jest płatna w modelu subskrybcyjnym (75 dolarów za miesiąc)[5]. Na potrzeby opisywanego projektu zasotosowano Unity w wersji Personal Edition.

4.2 Android

Naturalnym wyborem technologii przy tworzeniu aplikacji na urządzenie sterujące byłoby Unity, gdyż te środowisko pozwala na kompilacje kodu na urządzenia mobilne(systemy: iOS, Android, Windows Phone, Tizen itp[6]). Jednakże Unity w wersji Personal Edition nie pozwala na uruchomienie warstwy sieciowej na urządzeniach mobilnych.

Na potrzeby implementacji przykładowego urządzenia sterującego wybrano platformę Android, gdyż ma ona największy udzła w rynku.[7] Proces

[5]https://store.unity3d.com/subscribe

[6]https://unity3d.com/unity/multiplatform

[7]https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=1

tworzania aplikacji na tą platformę przebiega w języku Java.

4.2.1 Android Studio

opisac

4.2.2 Zależności

1. Butter Knife

4.3 Komunikacja sieciowa

Największym wyzwaniem było stworzenie dwukierunkowego protokołu komunikacyjnego pomiędzy serwerem (aplikacja napisana w środowisku Unity) oraz dowolnym kontrolerem lub w przyszłości innym urządzeniem wysłającym dane do aplikacji. Podstawowym założeniem było to iż, kontrolerem gry może być standardowy telefon komórkowy. Dodatkowo w przyszłości planowana jest rozbudowa o zdalne sterowanie za pomocą przeglądarki internetowej. Pierwotnie ozważane było użycie Bluetooth, jednak ograniczyłoby to zdalne sterowanie. Podjęto decyzję projektową o użyciu połączenia sieciowego. Rozważano następujące protokoły:

4.3.1 UNET

Unity wspiera natywną obsługę multiplayer - UNET, jednakże jest to zamknięty protokół. Komunikacja możliwa jest tylko pomiędzy aplikacjami stworzonymi w tym środowisku.

4.3.2 HTTP (SOAP, REST)

Komunikacja za pomocą HTTP (protokoły komunikacyjne takie jak np. SOAP, REST) są bardzo często spotykane. Jest to standard aplikacji internetowych. HTTP nadaje się do przesyłu dużych wolumenów danych, jednak niezbyt dobrze sprawdza się przy małych, lecz częstych połązeniach pomiędzy klientem, a serwerem. Duży narzut czasowy może spowodować transormacja danych do oraz z formatu JSON lub XML. Jednakże dużą zaletą wspomnianych protokołów jest prostota implementacji w większości języków programowania, gdyż są już gotowe komponenty.

4.3.3 TCP

Opisać, że TCP jest ogólnie lepsze - socket, ale to jest nadal połączeniowy, więc lepiej by było udp

4.3.4 UDP

opisać, ze to najlepsze rozwiazanie - bezpolazeniowe

5 Aplikacja główna

tutaj będą makiety

5.1 Świat gry

5.2 Aktorzy

Tablica 2: Właściwości aktora

```
public void SendInfo() {
        Network.SendMessage("hasax_"+this.hasAx);
        Network.SendMessage("hassh_"+this.hasSh);
        Network.SendMessage("hasdrabina_"+this.drabina);
        Network.SendMessage("isMove_"+this.isMove);
}
```

Listing 1: Do Poprawy

5.3 Kamery

Platforma Unity wspiera do 8 wirtualnych kamer [8]. Każda z tych kamer może być prezentowana na oddzielnym fizycznym ekranie.

[8] http://docs.unity3d.com/Manual/MultiDisplay.html

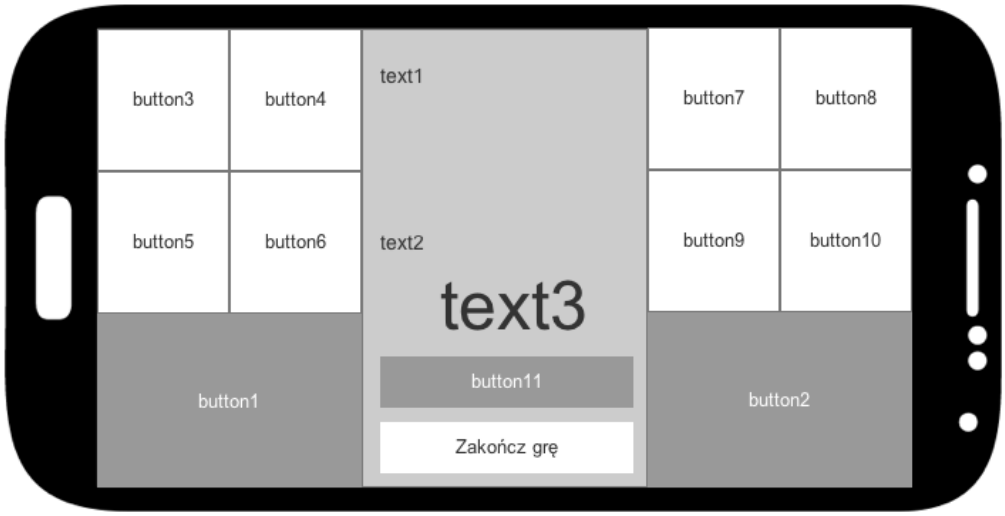
5.4 Prefabrykaty

W Unity możliwe jest używanie prefabrykatów (Prefabs [9]). Są obiekty lub grupy obiektów, które służą do wielokrotnego wykorzystywania. W Projekcie założono, że wszystkie reużwalne komponety (dziedziczone pomiędzy scenami) będą prefabrykatami.
Dodatkowo aktorzy gry (generowane dynamicznie) są również prefabrykatami. Instancje aktora są tworzone podczas działania aplikacji.
Opisać prefabrykaty

5.5 Logika biznesowa

5.6 Serwer komunikacyjny

6 Aplikacja mobilna - kontroler



Rysunek 4: Makieta - układ przycisków

Głównym założeniem było stworzenie uniwersalnego kontrolera przygotowanego pod dowolny rodzaj gry, bądź innej wizualizacji stworzonej w środowisku Unity. Podczas uruchomienia kontrolera serwer wysyła statusy przycisków oraz pól tekstowych.

[9] http://docs.unity3d.com/Manual/Prefabs.html

6.1 Przyciski

Każdy przycisk może zostać skonfigurowany poprez ustawienie tekstu. Dodatkowo można zablokować przycisk podczas gdy nie jest on potrzebny w danym czasie.

Aby ułatwić pracę nad aplikacją stworzono pole wyliczalne (ENUM) zawierającą wszystkie przyciski.

```
public enum ButtonEnum {
    BTN1, BTN2, BTN3, BTN4, BTN5, BTN6, BTN7, BTN8,
        BTN9, BTN10, BTN11
}
```

Listing 2: Enum z przyciskami

Jednakże aby powiązać elementy „Button" z warstwy widoku (defincja XML) na kod przycisku należy stworzyć listę elementów. Będzie ona służyła do wyszukiwania przycisków oraz zmiany ich właściwości. Dodatkowo do każdego przycisku należy dodać obsługę zdarzeń. Po kliknięciu wysyłany będzie odpowiedni komunikat.

```
    protected HashMap<ButtonEnum, Button> buttonsMap =
        new HashMap<ButtonEnum, Button>();

    protected void mapButton(int btnId, final
        ButtonEnum btnName) {
        final Button button = (Button)
            findViewById(btnId);
        buttonsMap.put(btnName, button);

        button.setOnClickListener(new
            View.OnClickListener() {
            public void onClick(View v) {
                sendToServer("button_" + btnName);
            }
```

```
        });
    }
```

Listing 3: Metoda obsługująca przycisk

```
@Override
    protected void onCreate(Bundle savedInstanceState)
        {
         super.onCreate(savedInstanceState);

         this.mapButton(R.id.button1, ButtonEnum.BTN1);
         this.mapButton(R.id.button2, ButtonEnum.BTN2);

    }
```

Listing 4: Przykład mapowania przycisku

6.2 Informacje tekstowe

Pola tekstowe mogą mieć ustawioną dowolną treść w dowolnym czasie.

6.3 Komunikacja sieciowa

7 Środowisko uruchomieniowe

Powyżej opisana aplikacja została uruchomiona testowo w labolatorium na Polsko-Japońskiej Akademii Technik Komputerowych.

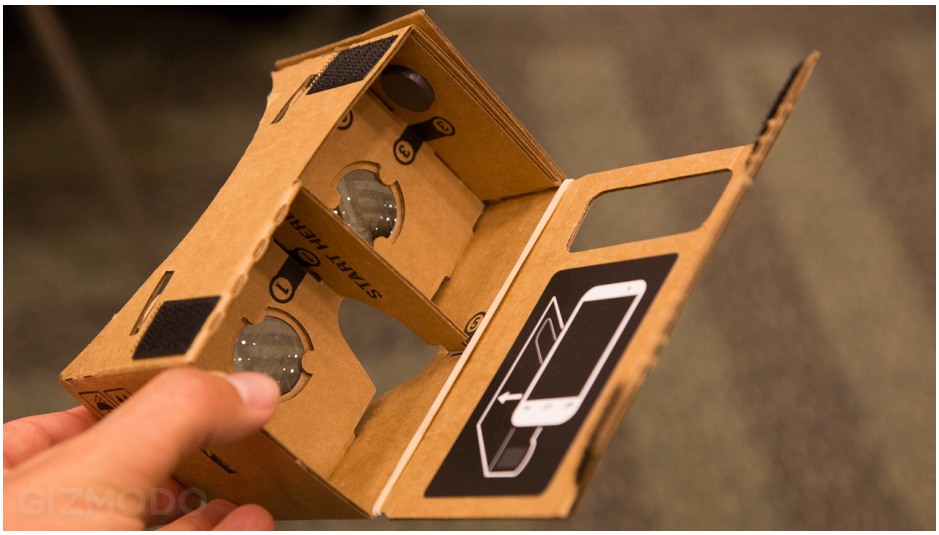
7.1 Serwer

Serwer został uruchomiony na komputerze przenośnym(laptop) posiadającym kartę graficzną umożliwiajacą podłączenie dwóch zewnętrznych ekranów - projektorów. Pierwszy z nich został połączony za pomocą złącza cyfrowego HDMI, natomiast drugi łączem DVI.

7.2 Aplikacja mobilna - kontroler

8 Implementacja na platformie Google Cardboard

Innym przykładem implementacji projektu w rozszerzonej rzeczywistości może być platforma Google Cardboard [10]. Są to niskobudżetowe okulary stworzone przez firmę Google do wyświetlania wirtualnej rzeczywistości. Kartonowe okulary powstawły w celu wyświetlania obrazu stereoskopowego. Posiadają one miejsce do umieszczenia dowolnego telefonu komórkowego typu smartphone. Do projektu wybrano wersję, która pozwala na umieszczenie telefonu w sposób taki, iż tylna kamera nie jest załonięta przez obudowę. Dzięki temu można użyć platformę Cardboard przeznaczoną pierwotnie tylko do wirtualnej rzeczywistości do stworzenia aplikacji wykorzystującą augmented reality.

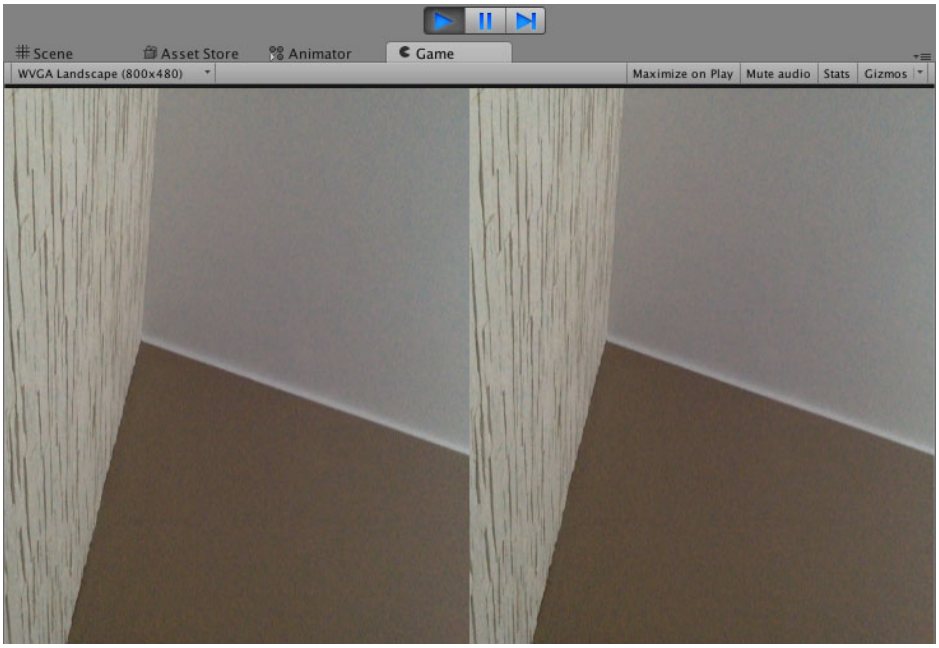


Rysunek 5: Google Cardboard w wersji do samodzielnego złożenia
źródło: http://gizmodo.com/turn-your-android-into-a-virtual-reality-headset-with-g-1596026538

Dzięki użyciu kamery użytkownik widzi obraz znajdujący się przed nim. Aby obraz był stereoskopowy należało stworzyć aplikację, która wyświetlić

[10] https://vr.google.com/cardboard/index.html

strumień danych z kamery dzieląc go na dwa obrazy (kolejno dla lewego oraz prawego oka).



Rysunek 6: Ujęcie z kamery aparatu w obrazie stereoskopowym - scena w Unity

źródło własne

8.1 ArToolkit

W celu wyświetlenia wirtualnych obiektów na obrazie z kamery rozbudowano aplikację z rodziału pierwszego uzupełić rozdział. Do platformy Unity doinstalowano zewnętrzny komponent ARToolKit [11]. Jest to biblioteka wydana przez University of Washington [12], lecz obecnie upostępniona jest na licencji GNU. Kod źródłowy jest otwarty i rozwijany przez środowisko Open Source [13].

[11] http://artoolkit.org/
[12] https://www.hitl.washington.edu/artoolkit/
[13] https://github.com/artoolkit

8.2 Markery

Za pomocą tej biblioteki możliwe jest wykrywanie w obrazie z kamery markerów, czyli specjalnie przygotowanych czarno-białych obrazków (w naiwnej implementacji - wydrukowanych na kartkach), oraz nakładanie w ich miejsce trójwymiarowych modeli lub całych scen. Dzięki bibliotece ArToolkit możliwe jest diagnozowanie pod jakim kątem padania oraz w jakiej odległości od urządzenia znajduje się marker. Umiejscowienie tagu analizowane jest w czasie rzeczywistym, co zapewni ciągłą korekcję ułożenia wirtualnych modeli względem ich realnych odpowiedników.

Rysunek 7: Przykład przygotowanego obrazka do rozpoznawania - marker

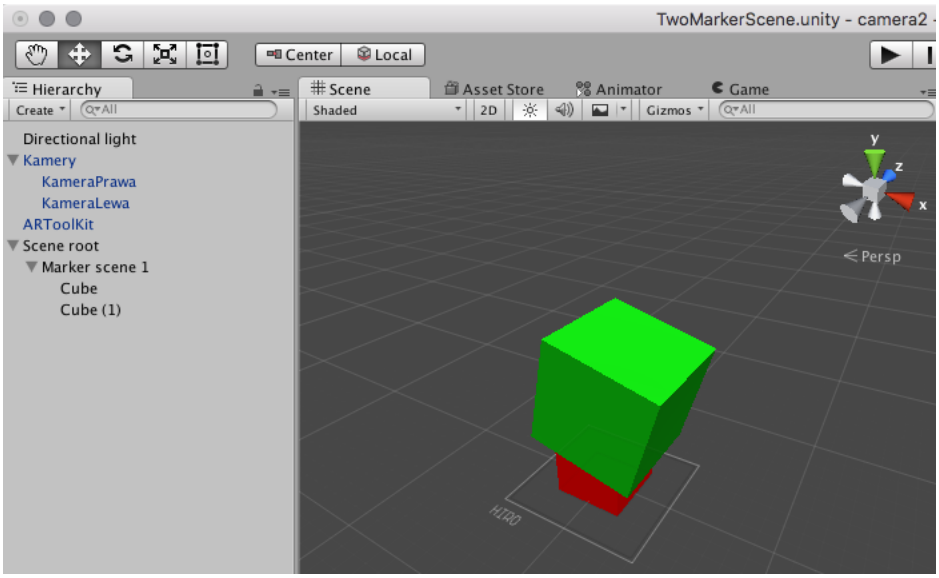
Szablony markerów można wykonywać we własnym zakresie. Aby zaimportować nowe obrazki do biblioteki ArToolkit należy przygotować specjalny plik binarny reprezentujący model markera. [14].

[14]http://bit.ly/1YU199f

Rysunek 8: Przykład zastosowania markera w ARToolkit
źródło http://arblog.inglobetechnologies.com/?p=421

8.3 Przykład implementacji

Opisać prefabrykaty



Rysunek 9: Podstawowa impementacja
źródło własne

8.4 Napotkane problemy i ograniczenia

1. Proces renderowania musi odbywać się na telefonie komórkowym, gdyż obraz kamery jest ciągle analizowany. Przy dużych scenach stworzonych w Unity moc obliczeniowa urządzenia jest niewystarczająca.

2. Analizowanie pozycji markera przy nieustannie włączonej kamerze powoduje dużą drenację baterii urządzenia. Czas pracy na baterii jest mocno ograniczony

3. Unity w wersji Personal (darmowej) skompilowanej pod platrformę mobilną (np. Android) nie udostępnia obsługi sieci (np. za pomocą połączenia TCP). Nie pozwala to na połączenie z zenętrznym kontrolerem.

4. Sterowanie za pomocą kontrolera bez fizycznych przycisków z założonymi okularami Google Cardboard jest bardzo uciążliwe. W przyszłości należałoby rozważyć połączenie telefou z zewnętrznym kontrolerem typu PAD[15].

[15]https://pl.wikipedia.org/wiki/Gamepad

9 Dalszy rozwój

Opisać

Spis rysunków

Spis tablic

Listings

Literatura

[1] Building Microservices, Sam Newman , Wydanie 4, 2016