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, 19 maja 2016

Spis treści

1 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)

2 Rozszerzona rzeczywistość

Natural Machines, Meta2

Rysunek 1: Wizualizacja Microsoft HoloLens
źródło: https://www.microsoft.com/microsoft-hololens/en-us/why-hololens

3 Wykorzystane technologie

3.1 Unity

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

3.1.1 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.

3.1.2 Alternatywne rozwiązania

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

Tablica 1: Porównanie silników gier

3.1.3 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. Podjąłem decyzję, by w projekcie użyć język C#, gdyż ów język jest najbardziej stabilny, posiada najbardziej rozbudowaną dokumentację oraz jest to najbardziej popularny język w specjalistycznej literaturze. Dodatkowym udogodnieniem jest to, iż język posiada wiele wbudowanych klas (np. do obsługi połączeń TCP) oraz niezliczoną ilość zewnętrznych bibliotek.

3.1.4 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# .

3.1.5 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)[1]. Na potrzeby opisywanego projektu zasotosowano Unity w wersji Personal Edition.

3.2 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). 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. Proces tworzania aplikacji na tą platformę przebiega w języku Java.

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

3.2.1 Android Studio

3.3 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:

3.3.1 UNET

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

3.3.2 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.

3.3.3 TCP

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

3.3.4 UDP

opisać, ze to najlepsze rozwiazanie - bezpolazeniowe

4 Aplikacja główna

tutaj będą makiety

4.1 Świat gry

4.2 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

4.3 Kamery

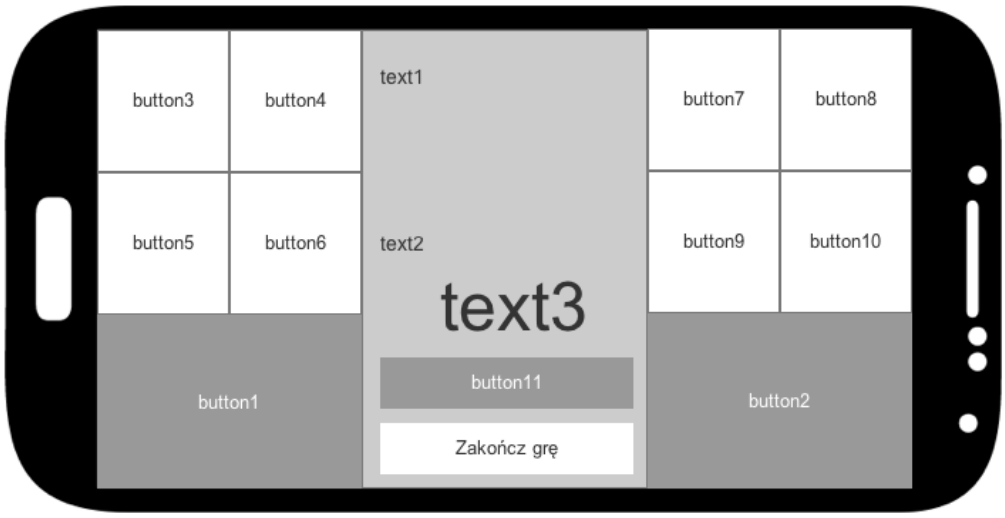
Platforma Unity wspiera do 8 wirtualnych kamer [2]. Każda z tych kamer może być prezentowana na oddzielnym fizycznym ekranie.

4.4 Logika biznesowa

4.5 Serwer komunikacyjny

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

5 Aplikacja mobilna - kontroler



Rysunek 2: 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.

5.0.1 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

5.0.2 Informacje tekstowe

Pola tekstowe mogą mieć ustawioną dowolną treść w dowolnym czasie.

5.1 Komunikacja sieciowa

6 Środowisko uruchomieniowe

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

6.1 Serwer

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

6.2 Aplikacja mobilna - kontroler

7 Dalszy rozwój

Spis rysunków

Spis tablic

Listings

Literatura

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