SPRAWOZDANIE

Web Socket - komunikacja w czasie rzeczywistym

Cloud Computing

Programowanie równoległe

13.01.2015

Karol Suwalski 125NCI B

https://github.com/SuwalskiKarol/oor.git

Uwaga pseudo-naukowy bełkot. 99.(9)% tego sprawozdania jest nie na temat.

1. **Przykładowe metody oraz narzędzia Networkingu wykorzystywane w silniku Unity3D korzystające z WebSocket(w tym chmury).**

- **Unet**(Unity networking)

http://docs.unity3d.com/Manual/UNet.html

[Dzięki tej technologii stworzyłem swój przykładowy program]
Jest oficjalnym narzędziem wbudowanym w silnik. Przed wersją 5.1tworzenie w tej technologii było skomplikowane i czasochłonne. Istniały metody oraz klasy do tworzenia serwerów oraz do nawiązywania łączności między serwerem oraz klientem, lecz były ciężkie do zaimplementowania oraz działały głównie w oparciu o protokół HTTP. A jak wiemy HTTP to droga jednokierunkowa. Od wersji 5.1 Unity zaprezentowało zupełnie odmienioneUnet. Dla ułatwienia tworzenia multi,Unet został podzielony na 3 fazy:

- **Faza I**–
Multiplayer Foundation. Najogólniej rzecz biorąc faza ta służy do pobierania feedbacku od użytkowników i przesyłania go do naszego projektu. Faza ta ma kilka charakterystycznych cech.

Wysowa wydajność warstwy transportowej bazująca na UDP w celu wspierania gier wszelkiej maści oraz wszelkiej ilości graczy.

Low Level API (LLAPI). Zapewnia pełną kontrolę za pośrednictwem socketów. Tutaj możemy wykorzystać sockety(TCP Socket, UDP Socket, **WebSocket**) Zapewnia większą kontrolę łączności w porównywaniu z HLAPI.

High Level API (HLAPI). Zapewnia prosty i bezpieczny model komunikacji client/server. Bez potrzeby pisania wielkiej ilości kodu. Unity robi to za nas.

Matchmaker Service. Zapewnia podstawową funkcjonalność dla tworzenia tzw „pokoi” oraz pomaga graczom wyszukiwanie innych graczy.

Relay Server. Rozwiązuje problemy z łącznością dla graczy próbujących się połączyć ze sobą za firewallami.

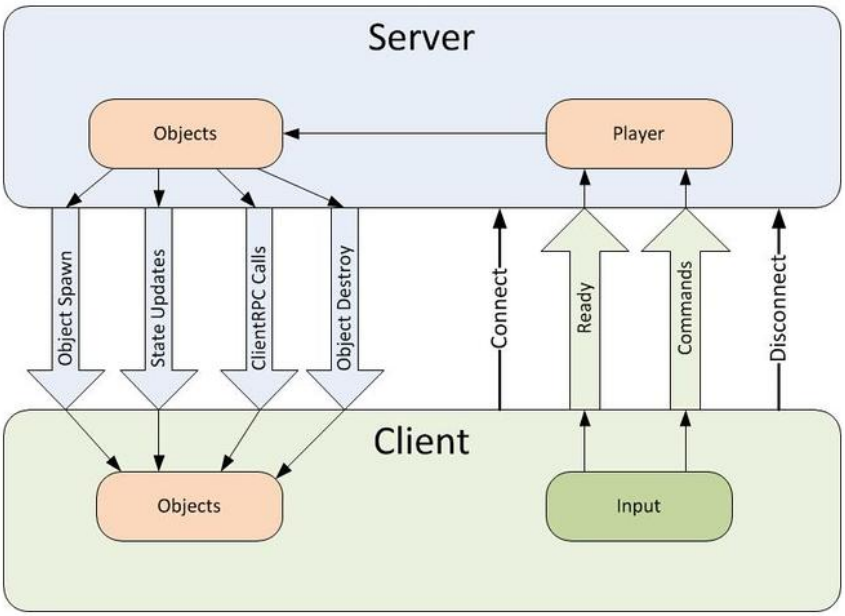
“We know from our prior experiences that making multiplayer games involves a lot of pain. So the Multiplayer Foundation is a new set of easy to use professional networking technology, tools and infrastructure for making multiplayer games without this pain.”
ERIK JUHL, Director of Unity Development

- **Faza II –**
Simulationserver. Wprowadza autorytatywny serwer gier
we’ll blog about this later.
Smuteg :<<

- **Faza III** –
Master Simulationserver. Służy do zarządzaniawielomaSimulationserverami.

- **RPC**(Remote Procedure Call)
http://docs.unity3d.com/Manual/net-RPCDetails.html

Protokół zdalnego wywoływania procedur. Istnieją dwa typy RPC.
- Commands – są wysyłane przez klienta i uruchamiane na serwerze.
- ClientRpc - sąwysyłane przez serwer i uruchamiane u klienta.
Diagram przedstawiający przykład



Deklaracja metody przeznaczonej do zdalnego wywołania wymaga jedynie użycia dekoratora [RPC]. Parametrami takiej metody mogą być standardowe typy obsługiwane przez Unity. Plusem jest, że dzięki w RPC możemy używać typy danych stworzonych tylko dla unity takie jak: Vector3, Quaternion, GameObject.

```
1. [RPC]
2. voidmyRemoteCall(stringsomeText,floatsomeNumber)
3. {
4. // akcja wywołana po stronie odbiorcy komunikatu
5. }
```

Wywołanie takiej metody może się odbywać na kilka sposobów:

```
1. // przekazanie zbuforowanego komunikatu do wszystkich (łącznie z sobą)
2. networkView.RPC("myRemoteCall",RPCMode.AllBuffered,"example text",2.5f);
3.
4. // jak wyżej, lecz z pominięciem siebie
5. networkView.RPC("myRemoteCall",RPCMode.OthersBuffered,"example text",2.5f);
6.
7. // jak wyżej, lecz bez buforowania
8. networkView.RPC("myRemoteCall",RPCMode.Others,"example text",2.5f);
9.
```

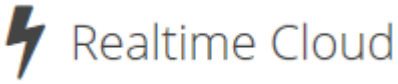
```
// jak wyżej, lecz tylko do serwera
networkView.RPC("myRemoteCall",RPCMode.Server,"example text",2.5f);

// przesłanie komunikatu do konkretnego gracza
NetworkPlayersomePlayer;// = inny gracz
networkView.RPC("myRemoteCall",somePlayer,"example text",2.5f);
```

- **PUN**(Photon Unity Networking)
  https://www.photonengine.com/en/PUN

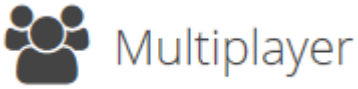
  

  Photon *is areal-time multiplayer game development framework*(poang.lepiejbrzmi☺). Jest najpopularniejszymi najbardziej rozbudowanym zewnętrznym narzędziem służącym do zarządzania naszymi grami sieciowymi stworzonymi w Unity.

  Usługi Photona:

  Gry na PUNie są hostowane na dystrybuowanej globalnie **chmurze.** Gwarantuje najniższe opóźnienia (latency) oraz czasy dostępu w obie strony dla graczy na całym świecie.

  Podłącz, ustaw i graj! PUN jest solidnym fundamentem dla każdego rodzaju pokoju bazującego na multiplayerze. My tworzymy grę w Unity, oni zajmują się jej backendem.

  *“Photon is the standard cross-platform multiplayer service and the world's #1 for Unity games”.*Szybki, łatwyiprzyjemnyeksportnaplatformymobilne, desktopoweczykonsole.

  Gry budowane na PUN są skalowane płynnie i automatycznie w **chmurze** Photon. Łatwa możliwość rozbudowy usługi od kilku do kilkinastu tysięcy użytkowników równocześnie obsługujących naszą grę.

Matchmaking API

Wyszukiwanie graczy do gry losowo lub przy odpowiednich parametrach wyszukiwania. Możliwość otwierania listy pokoi.

Customization

PUN wspiera wszystkie możliwe rodzaje gier oraz platformy (serio wszystkie, 24 platformy, nawet nie wiedziałem, że tyle istnieje)

Client To Server

Pełna obsługa oraz niezawodne działanie takich protokołów jak UDP, TCP, http czy **Webworkers.**

Unmatched Flexibility

Dzięki PUN gracze korzystający z np. IOS Game Center mogą grać z graczami używającymi Google Play Service lub z tymi, co grają przez fejsbuka.

Trochę więcej o samej chmurze.

Jest chmurąSaaS(Software as a service). Wszystkie aplikacje, jakie udostępniamy są przechowywane na chmurze. Dzięki chmurze Photon może zająć się hostingiem usług, działalnością i skalowaniem. Działa na serwerach dedykowanych, zapewnia Service Level Agreement (umowa o gwarantowanym poziomie świadczenia usług).

SLA obejmuje:
- monitorowanie usługi
-uzgodnienia
-raportowanie
-przegląd osiąganych wyników.
Można na chmurze uruchomić własny autorytatywny serwer, jako plug-in.
PhotonCloud jest usługą client-serwer, nie p2p.

Używanie Photonu nie wymaga hostingu, więc gracz, który stworzył pokój może wyjść z gry bez crashowaniaklientów.Photon jest bardzo stabilny, jego kod jest dostępny w razie indywidualnych preferencji. Jego komunikacja jest oparcia na **socketach.**

Photon vs Unet

Plusy photona

-Host model. Oby dwa działają na modelu client-server. Lecz Photon posiada dedykowane serwery, dzięki czemu zerwanie połączenia nie występuje po opuszczeniu gospodarza.

-Connectivity. Unity networking działa z NAT(translacja adresów sieciowych) dzięki czemu połączenie może się zrywać przy firewallach czy routerach. PUN dzięki serwerom dedykowanym nie potrzebuje NAT.

-Performance. Photon jest poprostu wydajniejszy. Dzięki dedykowanym serwerom latency/ ping będą zawsze na lepszym poziomie.

- Features&Maintenance. Jeśli chodzi o rozwijanie swoich produktów to Exit Games(twórcy Photona) jak i Unity Technologies się przykładają. Chociaż na przód wychodzi Photon ze względu na jego wieloletnie doświadczenie gdzie Unity swoje technologie Networking zaczął tak naprawdę rozwijać dopiero od wersji 5(wyszła gdzieś w połowie 2015 roku).

Plusy Unet

- Unet jest narzędziem wbudowanym w silnik. Co za tym idzie jego integracja z Game Objectami jest lepsza. Photon nie wykrywa takich rzeczy jak kolizje, geometria samego Unity i wiele więcej. Jeśli mamy własne preferencje i musimy zacząć pisać kod w Photonie to skrypty często zajmują ponad 1000 linii kodu, gdzie w Unet zajmują maksymalnie 100 linii.

- Price. Przy większych projektach Unity Pro(płatna wersja) i tak musimy wykupić, więc Unet mamy już w cenie Unity. Natomiast za Photona musimy dodatkowo płacić.

Wszytkie metody związane z Photonem znajdują się w bibliotece `PhotonNetwork`

Do podłączenia siędo Photon networku wystarczy kod:

```
voidStart()
{
PhotonNetwork.ConnectUsingSettings("0.1");
}
```

Do dołączenia do gry:

```
PhotonNetwork.JoinRandomRoom();
```

Do stworzenia gry:

```
public void OnConnectedToMaster()
{
PhotonNetwork.CreateRoom("Room42", true, true, 4);
}
```

Do uruchomieniamultiplayera:

```
public void OnConnectedToMaster()
{
PhotonNetwork.CreateRoom("Room42", true, true, 4);
}
```

Niestety nie jestem w stanie pokazać Photona w praktyce ze względu na brak wykupionej licencji. Chociaż miałem okazje na nim pracować, kiedy pisałem kilka skryptów, jako freelancer i musze stwierdzić, że jest kozacki.

- **uLink**

http://developer.muchdifferent.com/unitypark/uLink/uLink

Jest biblioteką do tworzenia sieciowych gier przy pomocy silnika Unity(network librarybuilt). Biblioteka ta nie działa, jako dodatek do istniejącej już Unet. Zupełnie ją zastępuje dodając do tego nowe funkcje(przynajmniej tak było przed wejściem wersji 5.1 Unity. Teraz nie wiem).
uLink makes it extremely fast and simple for developers to create network functionality.
uLink jest raczej stosowany do mniejszych projektów, gdzie nie potrzebujemy wielkich serwerów czy rozbudowanych usług np. na platformy WebGl lub Unity web player.
Wspiera **WebSocket.**

- **Tnet**(Tasharen Networking)

  http://www.tasharen.com/?page_id=4518

  Jest frameworkiemstworzonym przez kilku fanów Unity i jest dostępny na Assetsstore. Korzysta z socketów TCP oraz opcjonalnie UDP. Co go wyróżnia? No na pewno brak zajebistej strony, rzeszy inżynierów oraz góry pieniędzy :D. Mimo wszystko oferuje własny przyjazny iterface i wydaje się być dobrym narzędziem.

- **Smartfox**

  http://www.smartfoxserver.com/

  Działa podobnie do Photona. Tutaj również wstawiamy aplikacje na chmurę i smartfox robi za nas całą robotę. Jego cechą wyróżniającą jest to, że można na nim budować gry nie tylko postawione na Unity. Smartfox wsumie bardziej skupia się na grach napisanych w HTML5, Adobe Fash/Flex/Air czy C++

- **Forge Networking**

  http://epicjoin.com/ForgeNetworking

  Totalna świeżynka, chyba jeszcze w wersji beta. Jest odpowiedzią na technologie, jakie wprowadza Unity 5.3. Jako jeden z niewielu wspiera technologie **IL2CPP.** Czym jest IL2CPP? O tym później(zapomniałem wspomnieć Photon oraz Unity Networking również wspierają tę IL2CPP). Napisany tylko w C#.
  Wspiera również:
  - RUDP(Reliable User Datagram Protocol).
  - Multi-threaded, multi-processing (ni mom pojęciajak)
  -oczywiście UDP, TSP HTTP oraz HTTPS
  -EventDriven, czyli w bardzo łatwy sposób wspiera c# eventy takie jak mouseclick czy keypresses czy Unitowe Input.
  -Jako jeden z nielicznych UWP (Universal Windows Platform).
  - oraz wiele, wiele więcej rzeczy, których w ogóle nie rozumiem :D
  - gry VR(virtualreality)

- **Unity Cloud**

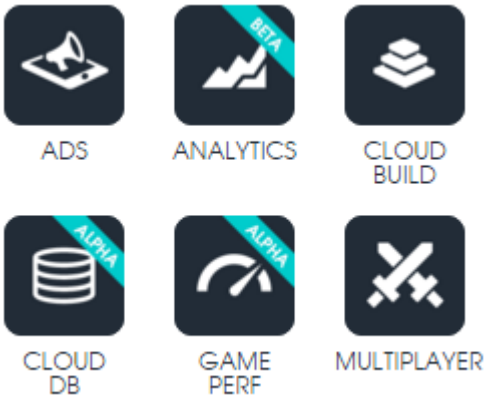
UNITY **CLOUD BUILD**

AUTOMATICALLY BUILD, INSTALL, AND TEST YOUR APPS

https://build.cloud.unity3d.com/landing/

Unity po wprowadzeniu wersji 5.1 orazUnetu zaczęło rozwijać własnego frameworka służącego nie tylko do hostowania gier, ale również, jako system zarządzania projektami.

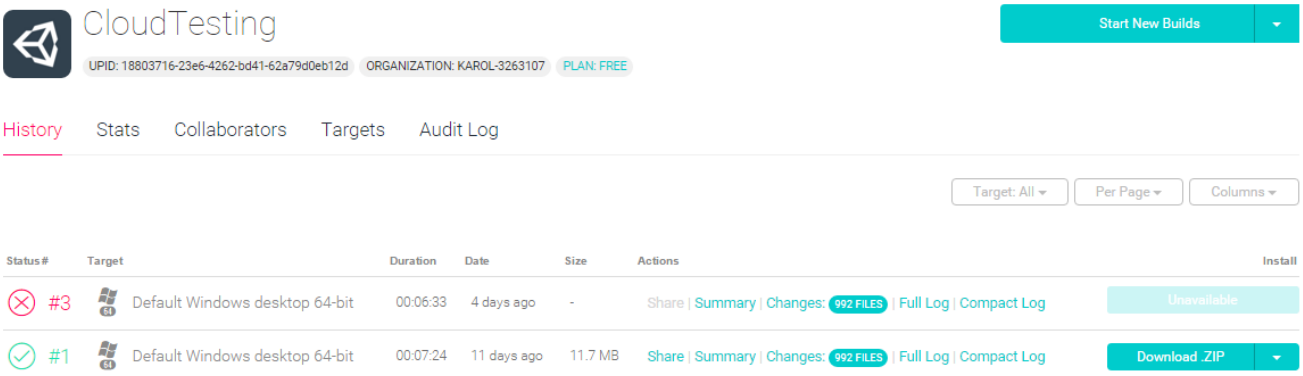
Usługi Unity Cloud:



ADS - coraz popularniejsza sieć mobilnych reklam wpierana przez Unity.
Analytics - usługa do analizy danych z gry, wkrótce poszerzona o nowe funkcje.
CloudBuild - automatyczne eksportowanie aplikacji na wiele platform, wyeksportowane aplikacje można pobrać z chmury oraz udostępnić innym użytkownikom.
Multiplayer - usługa wspomagająca prostsze tworzenie gier sieciowych na wiele platform.
Cloudc DB oraz Game PERF – póki, co nie doczekały się bezpośredniego wsparcia bezpośrednio w edytorze.

Skupmy się trochę na CloudBuild.

Jak to działa?



Gry przeznaczone na wiele platform będą łączone z jednym procesem tworzenia.Jeśli tworzymy grę i wrzucimy kod źródłowy na gita, chmura automatycznie to zauważy, i utworzy nam projekty na interesujące nas platformy. Nie musimy ręcznie tworzyć wielu projektów pod konkretne platformy. Jeżeli któryś współpracownik wprowadzi zmiany w projekcie zostaniemy automatycznie poinformowani o tym mailowo. Samą chmurę możemy odpalić na innych platformach i zarządzać projektami np. na telefonie.

'CloudTesting' (Default Windows desktop 64-bit) #3 failed to build for Windows x86_64!

CHANGES: https://build.cloud.unity3d.com/orgs/karol-3263107/projects/cloudtesting/buildtargets/default-windows-desktop-64-bit/builds/3/changes

THE LOG: https://build.cloud.unity3d.com/orgs/karol-3263107/projects/cloudtesting/buildtargets/default-windows-desktop-64-bit/builds/3/log

Summary: 22 warnings, 33 errors:

```
[Unity] Initialize engine version: 5.1.0f3 (ec70b008569d)
[Unity] -----CompilerOutput:-stdout--exitcode: 1--compilationhadfailure: True--outfile: Temp/Assembly-CSharp-
firstpass.dll
[Unity] Compilation failed: 2 error(s), 0 warnings
[Unity] Assets/Standard Assets/Utility/ForcedReset.cs(3,19): error CS0234: The type or namespace name `SceneManagement'
does not exist in the namespace `UnityEngine'. Are you missing an assembly reference?
[Unity] Assets/Standard Assets/Utility/TimedObjectActivator.cs(4,19): error CS0234: The type or namespace name
`SceneManagement' does not exist in the namespace `UnityEngine'. Are you missing an assembly reference?
[Unity] Assets/Standard Assets/Utility/ForcedReset.cs(3,19): error CS0234: The type or namespace name `SceneManagement'
```

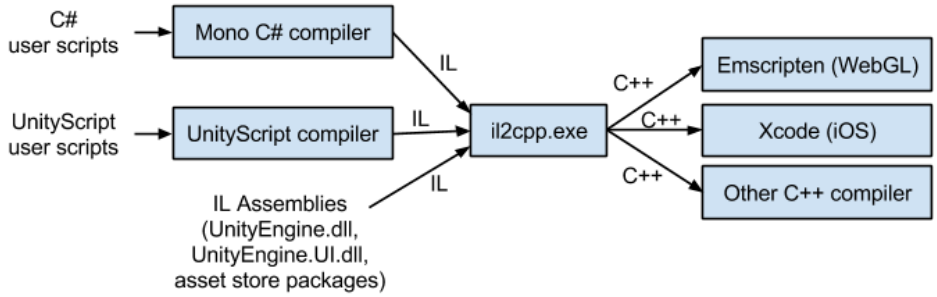
2. Przykład programu

Będzie to prosta gra przeglądarkowa wykorzystująca takie technologie jak Unity Networking(LLAPI), Websocket, WebGL, NodeJS, MYSQL, JSON. (Przy poważnym projekcie zamiast NodeJS oraz XAMPP użyłbym PhotonClouda).

Dlaczego WebGL a nie Unity web player?

Największy problem Web playera jest fakt, że do jego działania potrzebny jest Plugin. Google Chrome od jakiegoś czasu nie wspiera NPAPI także już sobie nie pogramy w gry postawione na web playerze w chromie. WebGL używa IL2CPP.
Po drugie web player przestał już być po prostu wspierany na rzecz WebGla.

IL2CPP składa się z dwóch rzeczy.
-KompilatorAhead of time. AOT jest generacją kodu maszynowego przed rozpoczęciem wykonania programu. Zamienia on Intermediate Language(taki .netowy odpowiednik assemblera. Język pośredni wyrażający kod w C# i 40 innych języków, tłumaczony na kod binarny) na kod C++. W przypadku WebGL zamienia on potem C++ na JS.



-runtimelibrary do wspierania wirtualnej maszyny.

Właśnie dzięki temu narzędziu WebGL jest wstanie użyć o wiele więcej technologii aniżeli webplayer. WebGl jest wydajniejszy, wspiera OpenGL ES i w ogólne jest fajny.

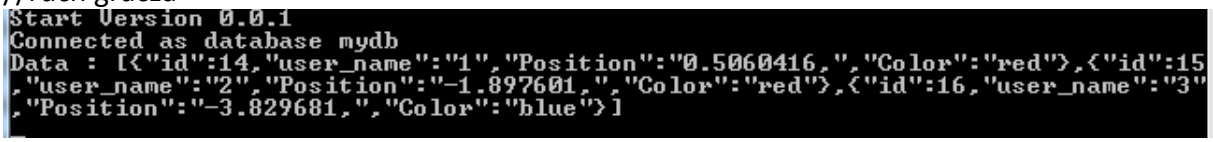
Tworzymy bazę danych na XAMPPIE

	id	user_name	Position	Color
Edytuj Kopiuj Usuń	14	1	0.5060416,	red
Edytuj Kopiuj Usuń	15	2	-1.897601,	red
Edytuj Kopiuj Usuń	16	3	-3.829681,	blue

Piszemy skrypt będący naszym serwerem. Oczywiście będziemy go setki razy poprawiać ze względu na dane, jakie będziemy mu dopisywali w trakcie tworzenia gry, ale sam szkielet można postawić.

```
var io = require('socket.io')(1234);
var shortId = require('shortid');
var datasetup = require('./Database/dataSetup');

datasetup.connect(function (err_connect) {

    datasetup.loadallUser(function (err, data) {
      //kod odpowiadający za wyszukiwanie bazy danych
      }}

io.on('connection', function (socket) {
      socket.on('LoadMap', function () {
      // wczytywanie użytkowników
      }}
```

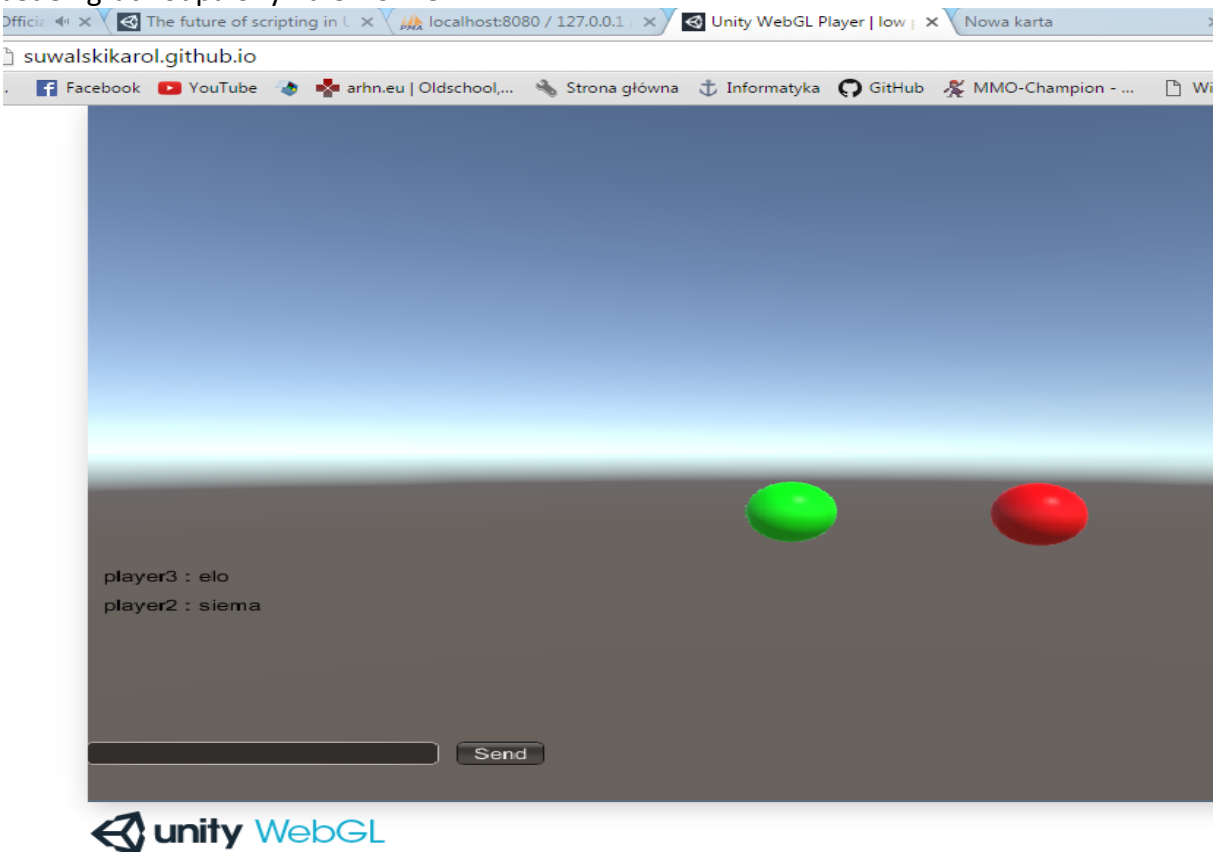
```
socket.on('talk', function (data) {
	// kodpotrzebny do czatu
	}

socket.on('disconnect', function (){
	// odłączeniegracza
	}
socket.on('register', function (data){
	//rejestracjagracza
	}
socket.on('login', function (data) {
	//logowaniegracza
	}
socket.on('UpdatePosition', function (data) {
	//zczyytywanie pozycji gracza
	}
socket.on('move', function (data){
	//ruch gracza
```



Odpalony serwer w NodeJS. Dane wyświetlone z bazy danych za pomocą JSON.

Po stronie klienta mamy skrypty Odpowiadające za:
- Obsługę czatu
- Przekazywanie danych o pozycji gracza do serwera
- Wczytywanie mapy poziomu
- Łączność z serwerem. Jest to skrypt pobrany z paczki WebSocket-Sharp.
- Przekazywanie informacji o statusie gracza
- Przekazywanie informacji o pozostałych graczach znajdujących się na mapie do gracza.

Przetestujmy aplikację.
Jeden gracz odpalony na Chromie



Drugi na Firefoxie