VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Strategická hra pro více hráčů v prostředí Internetu s klientem v HTML5

Multiplayer Strategic Game over Internet with Client in HTML5

2015 Dominik Sypula

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dominik Sypula**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Strategická hra pro více hráčů v prostředí Internetu s klientem v HTML5
Multiplayer Strategic Game over Internet with Client in HTML5

Zásady pro vypracování:

Cílem práce je vytvořit strategickou hru v reálném čase v prostředí Internetu. Hra bude tématicky zasazena do současnosti a umožní výstavbu budov, vytváření více druhů bojových jednotek, zlepšování zbraňových systémů, hru více hráčů.

Hra bude umožňovat:
1. Editaci herních plánů.
2. Výstavbu budov, zlepšování zbraňových systémů.
3. Vytváření více druhů bojových jednotek (vzdušné, pozemní) s možností nastavení několika parametrů v průběhu výroby.
4. Hru více hráčů v prostředí Internetu v kooperativním a kompetitivním režimu.
5. Bojové jednotky budou podporovat jednoduché autonomní příkazy (útoč, hlídkuj, ...).
6. Hru proti jednoduché umělé inteligenci.

Práce bude obsahovat:
1. Implementaci výše popsané strategické hry.
2. Programátorskou dokumentaci řešení s využitím diagramů jazyka UML.
3. Uživatelskou dokumentaci aplikace.

Seznam doporučené odborné literatury:

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four): Návrh programů pomocí vzorů. Grada. Praha 2003. ISBN 8024703025
[2] CLARK, Richard. Beginning HTML5 and CSS3: the Web evolved : next generation Web standards. New York: Distributed to the book trade worldwide by Springer Science Business Media, c2012, xx, 600 p. Expert's voice in Web development. ISBN 978-1-4302-2874-5.

Dále podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2015 .....

Rád bych na tomto místě poděkoval vedoucímu své práce, Ing. Davidu Ježekovi, Ph.D.
za rady a připomínky, které mi pomohly vypracovat tuto bakalářskou práci.

Abstrakt

V první části bakalařské práce je seznámení se s novou technologií v HTML5 pro vykreslování 2D a 3D grafiky, vybrání správneho enginu a následně implementace online 3D strategické hry pro více hráčů. Jako další část práce je vytvoření serverové části pro tohoto herního klienta v jazyce Java, která reprezentuje celou logiku hry, včetně jednoduché umělé inteligence.

Klíčová slova: HTML, canvas, WebGl, 3d grafika, Java, hra, strategie

Abstract

In first part of the bachelor thesis is familiarization with new technology in the HTML5 for render 2D and 3D graphics, select one of the game engine and implement online 3D strategy multiplayer game. In a second part of work is create server for game client which implement main function of game including low artificial intelligence.

Keywords: HTML, canvas, WebGl, 3D graphics, Java, game, strategy

Seznam použitých zkratek a symbolů

HTML	–	HyperText Markup Language
WebGl	–	Web Graphics Library
API	–	Application Programming Interface
OpenGl ES	–	Open Graphics Library for Embedded Systems
AJAX	–	Asynchronous JavaScript and XML
IP	–	Internet Protocol
JSON	–	JavaScript Object Notation
XML	–	Extensible Markup Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Na začátku této bakalářské práce bych rád uvedl, co mě vedlo k výběru tohoto tématu. Od začátku studia na vysoké škole jsem se zabýval vývojem her pro Android a tímto způsobem jsem si zlepšoval své znalosti v oblasti programovacích jazyků. Jelikož jsem vyvíjel hry v jazyce Java, tak se tento jazyk stal mým nejoblíbenějším. Vývoj her mě vždycky bavil a stále baví, jelikož oproti informačním systémům nebo aplikacím je výstup programu v grafické podobě a zabaví nejen mě ale i další lidi.

Cílem práce je vytvořit prototyp hry s využitím technologií HTML5 (klient) a Java (server). Jelikož HTML5 přišlo s novými technologiemi, které dokáži vykreslování v reálnem čase bez obnovení stránky, je tuto technologií možno využít při tvorbě her. Tímto vznikly nové možnosti využití HTML, o kterých se zmíním.

Dokument je rozdělen do několika částí. V první části bych rád seznámil čtenáře nejen s vyvíjenou hrou ale i s popisem her obecně. Ve druhé části jsou zmíněny základní požadavky na server a klienta a od toho se odvíjí popis použitých technologií ve třetí části. Ve čtvrté části se nachází programatorská dokumentace, ve které popisuji jednotlivé algoritmy a ukázky kódu, které slouží k lepšímu vysvětlení problematiky. Další část je uživatelská dokumentace, ve které Vás seznámím s principy ovládaní hry a serveru. V závěru shrnu, čeho jsem dosáhl a uvedu informace, které jsem za dobu vývoje získal.

2 Úvod ke hrám

Počítačové hry jsou druh softwaru, kde výstupem pro uživatele, pokud nepočítám textové hry, je 2D nebo 3D grafika a nějaký zvuk. Jsou už mnoho let velice populární a na jejich popularitě se nepodepsal ani fakt, že mnoho dnešních herních titulů jsou předělané starší verze do nového grafického kabátu. V dnešní době je dokonce i možné, se hraním počítačových her živit. Pořádají se různé turnaje, kde si mohou týmy či jednotlivci z celého světa poměřit své schopnosti a vyhrát nemalé odměny. Počítačové hry slouží především pro zábavu nebo relaxaci a dokážou, a to může potvrdit nejeden hráč, zabavit na dlouhé hodiny. Ale najdou se i vzdělávací hry, ve kterých se autoři snaží zábavnou formou obohatit vědomosti hráčů. V mnohých případech jsou to hry pro menší děti, ale existují také pro starší či dospělé osoby. Existuje několik hlavních herních žánrů: akční, strategie, simulátor, arkáda, advetura a hra na hrdiny.

Dnešní počítačové hry jsou práci nikoli jednotlivců ale týmů, které jsou rozděleny do několika skupin. Nejdůležitější dvě skupiny dle mého názoru jsou grafici a programátoři. Grafik se stará o grafickou část, mnohdy má na starost i animace modelů a programátor vytváří funkční část celé hry. Stále větší popularity získávají indie hry, to jsou hry vytvářené jednotlivci nebo malými skupinami, které nemají finanční podporu. Jejich popularita se zakládá na tom, že přinášejí mnoho zajímavých neohraných nápadů. Jako nejlepší příklad indie hry je Minecraft, který se stal oblíbený po celém světě, jelikož nabídnul hráčům možnost, vytvořit si svůj vlastní svět složený z kostek. Dnes je už tato hra vyvíjená pod společnosti Microsoft. Nejlepší místo pro indie hry se stal obchod pro mobilní zařízení. Smartphone nebo tablet má dnes už skoro každý a jelikož jejich výkon, který je již dnes poměrně vysoký, se rok od roku zvyšuje, tak pro taková to zařízení není problém rozběhnout i náročnější 3D hru. Existují tři nejpoužívanější operační systémy pro mobilní zařízení Android, iOS, Windows a každý má svůj obchod, kde si mohou vyvojáři vložit své hry.

2.1 Popis strategické hry

Žánr strategie je velice populární mezi hráči, nejen že slouží pro zabavení, ale některé dokonce rozvíjí myšlení hráče, jelikož musí logicky uvažovat a dopředu promýšlet tahy. Pohled u většiny strategických her je z ptačí perspektivy. Ovládaní, ať už jednotek nebo kamery, se provádí u většiny strategických her pomocí myši. Základní rozdělení strategických her:

- tahové strategie
- realtimové strategie
- budovatelské strategie

U tahové strategie hraje několik hráčů proti sobě, ale jak už název napovídá, každý hráč táhne až na něj přijde řada. Příklad tahové hry Civilization. Realtimová strategie je založena na tom, že hráč může svobodně ovládat své jednotky či budovy, jednoduše řečeno

každý je na tahu. Jako známého představitele bych uvedl StarCraft. Budovatelské strategie nejsou zaměřené na boj, ale spíše na ekonomiku a rozvoj města jako například u Sim City.

2.2 Popis vyvíjené hry

Vyvíjená hra je 3D strategická, realtimová v prostředí HTML pro více hráčů. Jako vzor pro tuto hru posloužil už několik let velice oblíbený Stronghold crusader nebo Command and Conquer Red Alert. Ve hře můžete stavět budovy, přičemž jednotlivé budovy mají své úkoly, nebo vyrábět bojové jednotky. Aby bylo možné stavět budovy nebo vytvářet jednotky, je potřeba dostatek zlata. Zlato lze získat v rafineriích, které prodávají ropu. Aby mohly rafinerie prodávat ropu, musí ji odněkud získat a k tomu slouží těžba ropy a cisterna, která danou ropu dováží do rafinerie. Hra nabízí možnost vybrat mapu a počet hráčů na mapě. Je zde i možnost vytvoření své vlastní mapy, kde se po uložení zobrazí všem hráčům ve výběru map. Cílem celé hry je vyhrát nad protivníkem tím, že zničíte jeho základnu.

3 Analýza požadavků

3.1 Požadavky na klienta

Při tvorbě klienta v HTML5 je nejdůležitější, aby aplikace správně fungovala v moderních prohlížečích, tedy v prohlížečích, které mají nativně implementovány technologie přinášející HTML5. Jelikož HTML5 je stále ve vývoji a prohlížeče nemají všechny novinky implementovány, je potřeba se ujistit, zda při vývoji hry nenastanou problémy se spuštěním.

Jak je vidět v tabulce č. 1 většina prohlížečů tyto, pro mě důležité, technologie podporují a tedy při spuštění by neměl nastat problém. V tabulce jsou zmíněny i prohlížeče pro mobilní zařízení Android a iOS, jelikož jsem chtěl poukázat na to, že je možné takto vytvářet hry i pro tyto platformy, ale vyvíjená hra je optimalizováná pouze pro desktopová zařízení.

Jako základní požadavky pro klienta po softwarové stránce bych určil operační systém podporující moderní prohlížeče jako například Windows, Linux Max OS. Po hardwarové stránce bych určil standartní sestavu: RAM 2GB, CPU 2 x 2,5GHz, HDD 100GB a pochopitelně přípojku k internetu s co nejmenší odezvou, aby hra jela plynule a nedocházelo k nepříjemným skokům jednotek při hraní.

	Canvas	WebGl	Web Sockets
IE ver. 11	Ano	Ano	Ano
Firefox ver. 36	Ano	Částečně	Ano
Chrome ver. 41	Ano	Ano	Ano
Safari ver. 8	Ano	Částečně	Ano
Opera ver. 27	Ano	Částečně	Ano
iOS Safari ver. 8.1	Ano	Ano	Ano
Opera Mini ver. 8	Částečně	Ne	Ne
Chrome pro Android ver. 41	Ano	Částečně	Ano

Tabulka 1: Podpora HTML5 technologií

3.2 Požadavky na Sever

Nejdůležitějším požadavkem na server je podpora Javy. Jelikož Java podporuje platformy jako Windows, Linux Mac OS, tak by tady problém nastat neměl. Jelikož bude server zpracovávat logiku hry, tak bych jako hardware navrhl : RAM 8GB, CPU 4 x 3.2GHz, HDD 500GB a připojení k internetu.

4 Použité technologie

4.1 HTML5

Vynálezce HTML je Tim Berners-Lee [2, 1], který roku 1989 pracoval pro CERN na informačním systému, na němž by mohli vědci z celeho světa sdílet své informace. Jelikož Berners-Lee věděl, že potřebuje něco jednoduššího, vytvořil HTML a roku 1991 CERN zprovoznil svůj web. Roku 1993 vznil prohlížeč Mosaic, který slavil veliký úspěch a následně na to se začal web rychle rozvíjet. V průběhu let 1990 až 1999 vzniklo několik verzí HTML, ta poslední byla verze 4.01. Trvalo téměř deset let než přišla aktuální verze HTML5. Podle „Plánu 2014“ by měla vycházet nová verze každé dva roky, tedy v roce 2016 by měla vyjít HTML5.1

Od roku 1990 do roku 2008, kdy byla publikováná první verze HTML5, byl HTML používán převážně k vytváření webových stránek. S příchodem HTML5, který nabídl vyvojářům mnoho nových funkcí, jako například canvas, který slouží jako plátno pro vykreslovaní 2D a 3D grafiky, se tento jazyk stává stále více populární a oblíbený. Jelikož lze HTML5 spustit jak na mobilních, tak na desktopových zařízeních, stal se výhodný k tvorbě nejen webových stránek ale i aplikací nebo her.

Nyní zmínim 3 nové technologie HTML5, které byly pro vývoj herního klienta důležité a ve zkratce je popíšu.

4.1.1 Canvas

Canvas je prvek pro vykreslování jednoduchých i složitějších animací nebo statických obrázků. Pro vykreslování je zapotřebí znát jazyk JavaScript. Více informací naleznete zde [7]

4.1.2 WebGl

WebGl [3] je nativní JavaScript API sloužící pro vykreslování 3D grafiky s hardwarovou akcelerací. Vychází z OpenGl ES 2.0 a verze 1.0 byla vydána v roce 2011. Aktuálně je v přípravě OpenGl 2.0, jenž je založeno na OpenGl ES 3.0 a tuto verzi začínají také podporavat některé herní enginy, takže se hráči mohou v budoucnu dočkat propracované 3D hry v prostředí HTML.

4.1.3 Web Sockets

Další velice užitečnou technologií jsou Web Sockets. Tato technologie umožňuje oboustranné spojení se serverem a výměnu dat v reálnem čase. Díky tomu mohou vyvojáři vytvářet "živější"aplikace a nemusí se zabývat například komunikaci se servem za pomocí technologie AJAX. Více informací naleznete zde [4].

4.2 Výběr enginu

V dnešní době existuje již několik HTML5 herních enginů [9] a jejich možnosti jsou různé. Před výběrem je potřeba si nejprve promyslet, co všechno vyvíjená hra bude obsahovat a podle toho určit základní kritéria pro herní engine. Jako dalším impulsem k výběru poslouží již hotové ukázky, které mě osobně velice potěšily. V následujících částech se pokusím popsat alespoň tři nejpopůlárnější HTML5 herní enginy, a proč jsem si vybral právě Three.js.

4.2.1 Construct 2

Construct 1 byl vydán v roce 2007, verze 2 v roce 2011 a už dnes se pracuje na verzi 3. Tento engine patří mezi nejpopulárnější, jelikož pro tvorbu her nepotřebujete žádné programatorské znalosti. Celý vývoj hry probíjá v editoru a jednotlivé objekty se jednoduše přidávají vložením pomocí myši. Vývoje her jsou tímto způsobem mnohem rychlejší. Construct 2 je vybaven 2D grafikou, fyzikou, podporou pro různé velikosti obrazovky, vizuální efekty, zvukové efekty, implementované algorimty pro hledání cest, hru pro více hráčů a mnoho dalšího. Podporuje publikování pro Web, iOS, Windows jak mobilní, tak stolní zařízení, Max, Linux, Anroid, Tizen, Facebook, Wii U a další. Nabízí několik druhů licení: zdarma, osobní licenci za €99.99 a firemní licenci za €329.99. Tento engine nebyl vhodnou volbou kvůli absence programatorské části a 3D grafiky.

4.2.2 Pixi.js

Pixi.js je 2D engine, který je zcela zdarma. Podporuje WebGl a Canvas vykreslování, více dotykové interakce, filtry a další. Oproti Construct 2 je jeho výbava podstatně chudší. Programování probíhá v JavaScriptu. Ani tento engine nebyl vhodnou volbou, jelikož tak jako Construct 2 i tento engine podporuje pouze 2D vykreslování.

4.2.3 Three.js

Jako jednoho ze zástupců 3D enginu bych rád zmínil Three.js. Je zcela zdarma, ale jeho výbava není nějak ohromující. Neobsahuje fyziku, ale je možno použít knihovnu Oimo.js. Rovněž zde chybí 3D audio. Tyto nedostatky lze nahradit pomocí externích knihoven. Nabízí ale mnoho věcí jako několik druhů kamer a osvětlení, různé formáty modelů pro načítaní, vykreslování pomocí Canvas nebo WebGl, podpora Oculus Rift a další. Tento engine jsem si vybral, jelikož obsahuje pouze to, co jsem opravdu potřeboval a ukázky, které mají na svých stránkách, mě osobně přesvědčily.

	Construct 2	Pixi.js	Three.js
2D	Ano	Ano	Ne
3D	Ne	Ne	Ano
Fyzika	Ano	Ne	Ne
Cena	Zdarma – omezená verze, Placená – plná verze	Zdarma	Zdarma
WebGl	Ano	Ano	Ano
Zvuky	Ano	Ne	Ne

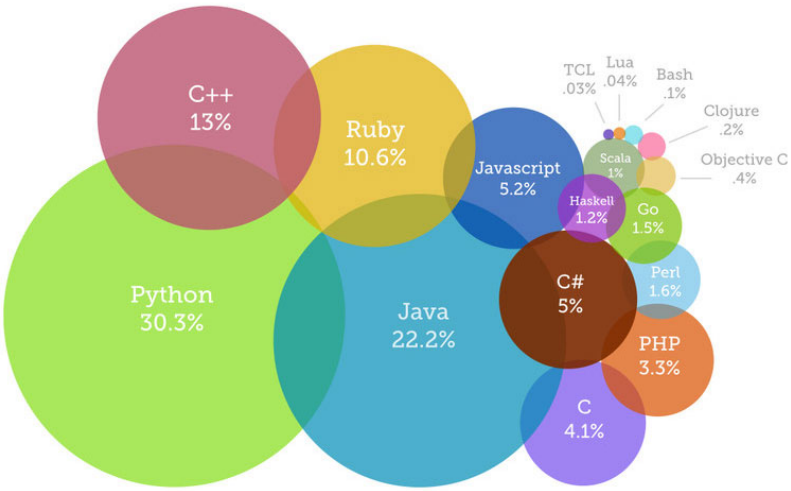
Tabulka 2: Základní popis HTML5 enginu

4.3 Server

Serverová část je psána v jazyce Java a implemetuje celou logiku hry. Je naimplementován tak, že je možné na jednom serveru spustit hned několik her. Spuštění a ukončování hry ovládá automaticky server. Rovněž se na serveru nacházejí uložené mapy, které jsou před spuštěním hry poslány na klienta pro nastavení scény. Pro komunikaci s klientem pomocí WebSocket je použita knihovna Jetty.

4.3.1 Java

Java patří mezi nejpopulárnější programovací jazyky hned po Pythonu, jak je ukázano na obrázku č. 1, který je převzat ze stránky [8] a tuto popularitu si hájí už několik let. Svou oblibu si získal díky tomu, že vytvořené programy lze spouštět takřka na všech zařízeních. Java je objektově orientovaný jazyk a vývoj pomocí něj je jednoduchý.



Obrázek 1: Popularita programovacích jazyků za rok 2014

5 Programatorská dokumentace

Pro orientaci bych rád shrnul základní vlastnosti vyvíjené hry. Celou hru tvoří herní klient a server. Klient má na starosti ovládaní hry a vykreslování na základě dat, která získaká ze serveru. Struktura herního klienta je popsána v kapitole 5.2. Data mezi klientem a serverem jsou posílana jako řetězce. Server obsahuje celou logiku hry a jelikož je možno spustit hned několik her na jednom serveru, tak ovládá i spouštění a zastavování her. Struktura serveru je popsána v kapitole 5.7. Na serveru se nachází i umělá inteligence, která je popsána v kapitole 5.14.

5.1 Úvod do Three.js

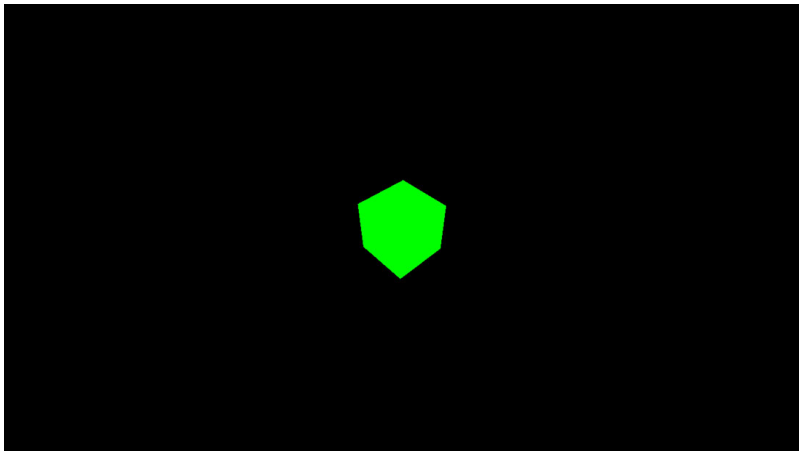
Abychom porozuměli v následujících částech implementaci vyvíjené hry, která je rozsáhlejší, je potřeba si nejprve ukázat na jednoduchém příkladu implementaci herního enginu Three.js (kapitola 4.2.3)

```
<body>
    <script src="js/three.min.js"></script>
    <script>
        var scene = new THREE.Scene();
        var camera = new THREE.PerspectiveCamera( 75, window.innerWidth/window.
            innerHeight, 0.1, 1000 );

        var renderer = new THREE.WebGLRenderer();
        renderer.setSize( window.innerWidth, window.innerHeight );
        document.body.appendChild( renderer.domElement );

        var geometry = new THREE.BoxGeometry( 1, 1, 1 );
        var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
        var cube = new THREE.Mesh( geometry, material );
        scene.add( cube );

        camera.position.z = 5;

        var render = function () {
            requestAnimationFrame( render );

            cube.rotation.x += 0.1;
            cube.rotation.y += 0.1;

            renderer.render(scene, camera);
        };

        render();
    </script>
</body>
```

Výpis 1: Three.js – ukázka kódu

Následující popis se vztahuje k výpisu kódu č. 1. Abychom mohli pracovat s enginem je třeba jej nejprve "připnout"v hlavičce souboru index.html. Všechen kód se píše mezi tagy

<script> a je psán v jazyce JavaScript. Pomocí THREE.Scene() vytvoříme scénu, která reprezentuje námi vytvářený svět a do které se vkládají jednotlivé části, které chceme vykreslovat. Další řádek vytváří kameru, konkrétně se jedná o perspektivní kameru, která je jedna z několika druhů co engine nabízí. Této kameře se nastaví atributy: vertikalní úhel zorného pole, poměr stran a další dva parametry jsou vzdálenosti viditelnosti oddo. THREE.WebGLRenderer() vytvoří render, který bude vykreslovat pomocí WebGl. Je zde i možnost použít THREE.CanvasRenderer(), ale Canvas nepodporuje hardwarovou akceleraci a tedy celé vykreslování neprobíhá na grafické kartě, tudíž je pomalejší. Poté se nastaví velikost rendereru a vytvoří se element, který slouží jako plátno pro vykreslování. Pro vytvoření zelené kostky, která je zobrazena na obrázku č. 4, je potřeba tří částí. Nejprve je potřeba vytvořit geometrický objekt (THREE.BoxGeometry(1, 1, 1)). Poté materiál pomocí THREE.MeshBasicMaterial(color: 0x00ff00), který nepotřebuje osvětlení, aby mohl být vykreslen. Three.js nabízí několik druhů materiálů. Ve třetí části spojíme geometrický objekt a materiál v jeden objekt zvaný Mesh a vložíme ho do scény. Ve funkci render probíhá vykreslování všech objektů ve scéně a otáčení zelenou kostkou. Tato funkce se zavolá jednou a poté je automaticky volána. Three.js nabízí vykreslovací frekveci standartních 60 snímků za sekundu.



Obrázek 2: Three.js – výstup z ukázkového kódu

5.2 Struktura HTML klienta

HTML klient je rozdělen do několika částí. Tyto části bych rozdělil do dvou skupin. První skupina jsou JavaScriptové knihovny, které mají už implementovanou určitou logiku a druhá skupina je mnou vyvíjená hra.

První skupina:

- **three.js** je javascriptová knihovna implementující celý engine, který je popsán v kapitole 4.2.3.

- **ColladaLoader.js** je knihovna, která je navržena pro Three.js engine a pomáhá s načítaním objektů s koncovkou dae.
- **reconnecting-websocket.js** automaticky obnovuje spojení webových soketů. Tato knihovna je velice užitečná, protože občas dochází k nečekaným výpadkům spojení a opětovné připojení by bylo možné provést až po znovu načtení stránky.

Druhá skupina:

- **modelLoader.js** je skript, ve kterém probíhá načítaní modelů a po úspěšném načtení spustí hlavní funkci celé hry.
- V **Scene.js** se nastavujé celá hlavní scéna. Nastavuje se kamera, světla, zem, obloha a jsou zde funkce na přidávání objektů do scény.
- **mouse.js**. Jelikož je ovládaní celé hry za pomocí myši, tak tento script obsahuje funkce, které implementují jednotlivé akce pro ovládaní. Například stavění budov nebo přidávání dekorací při tvorbě mapy.
- **MyModel.js** je soubor, ve kterém se nachází pět funkcí reprezentující třídy. Jsou zde třídy pro uložení a ovládaní 3D modelů, třída pro hráče, třída pro vykreslování gridu.
- **syle.css** je soubor, který obsahuje stylování HTML elementů hry.
- **index.html** obsahuje hlavní funkce celé hry. Je to nejdůležitější a nejrozsáhlejší část, kterou budu ještě popisovat.

5.3 Načítaní 3D modelů

Jelikož je potřeba než začne hra načíst všechny potřebné modely a až potom spustit hlavní část hry, tak jsem k tomu vytvořil funkci **myModelLoader**, která implemetuje další funkce, jako je v ukázce kódu č. 2.

```
var loader = new THREE.ColladaLoader();
    loader.options.convertUpAxis = true;
    loader.load('models/planeAttack/plane/plane.dae', function(result) {
        plane = result.scene;
        plane.scale.x = plane.scale.y = plane.scale.z = 20;
        plane.updateMatrix();
        planeLoaded = true;
        plane.position.y = 400;
        plane.position.x = 100;
        plane.position.z = 100;
        updateProgress();
    });
```

Výpis 2: Načtení 3D modelů letadla

Nejprve se vytvoří objekt loader pomocí **THREE.ColladaLoader()**. Tímto vytvořeným objektem zavolám funkci se dvěma parametry. První parametr je cesta k objektu, který chci načíst (musí být s příponou dae). Druhý parametr je funkce, která se chová jako listener. Tento listener se spustí až bude objekt plně načten. Nejprve vložíme do proměnné plane načtený model, poté mu nastavíme velikost a provedeme aktualizací pomocí funkce **updateMatrix()**. Dále nastavíme výchozí pozice objektu a zavolám funkci **updateProgress()**, která provede aktualizaci grafického stavu načítaní a zkontroluje, zda jsou už všechny modely načteny. Jestliže jsou všechny modely připraveny, může být spuštěná hlavní funkce v souboru index.html s názvem **initScene**.

5.4 WebSockets na klientovi

Jelikož jsou WebSockets implementované ve webových prohlížečích (Tabulka č. 1), tak pro jejich použití není potřeba importovat žádnou další knihovnu. Data mezi serverem a klientem jsou zasílaná ve formátu String, ale na zařízeních jsou zpravovavána ve formátu JSON. Jako návod pro implementaci kódu mi posloužil zdroj [5].

5.4.1 Ukázka implementace

```
var socket = new WebSocket("ws://127.0.0.1:2005");

  socket.onopen = function () {
      console.log('Connection␣Established!');
  };

  socket.onclose = function () {
      console.log('Connection␣Closed!');
  };

  socket.onerror = function (error) {
      console.log('Error␣Occured:␣' + error);
  };

  socket.onmessage = function (e) {
          console.log('e.data');
      };
      socket.send("Hello␣WebSocket!");
  socket.close();
```

Výpis 3: Ukázka použití WebSockets v HTML5

Jako první věc je potřeba nastavit IP adresu a port, na kterém komunikuje server. K dispozici je několik metod, které se automaticky spouštějí jakmile dojde k situaci, na kterou mají reagovat.

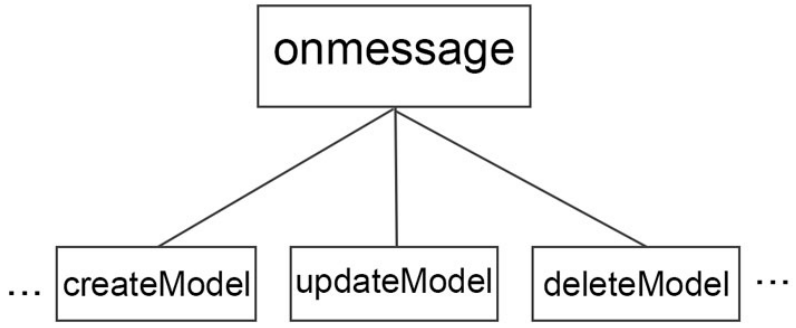
- Funkce **onopen** se spustí pokaždé, když je úspěšně navázáno spojení.
- Funkce **onclose** reaguje na uzavření spojení.

- Funkce **onerror** je spuštěna v případě, že se spojení ukončí bez pokynu od serveru nebo klienta a zachytává chyby ve spojení.
- Funkce **onmessage** je spuštěna příjde-li ze serveru zpráva.

Jestliže je potřeba zaslat na server zprávu, zavolá se funkce **send** a jako parametr se předá zpráva. Pro ukončení spojení se zavolá funkce **close()**.

5.4.2 Zpracování zpráv

Jelikož se mezi klientem a serverem posílají zprávy o různých informacíh, je potřeba tyto zprávy rozlišit. Pro rozlišování zprávy jsem určil, že bude na prvním místě příslušné zprávy umístěn řetězec nesoucí název akce. Jelikož jsou zasílaná data ve formátu String, ale pro práci se používá formát JSON, kde jich může být hned několik najednou, tak se ve funkci **onmessage** projdou postupně všechny příchozí zprávy a pomocí přepínače se rozdělí. Rozdělená data jsou pak předána funkci (obrázek č. 3), která nese odpovědnost za jejich zpracování. Pro ukázku popíší zpracování dat ve funkci updateModel (kapitola 5.4.2.1).



Obrázek 3: Princip zpracování zpráv na klientovi

5.4.2.1 updateModel Pro jednoduchost mám rozdělené hráče na tři různé objekty. Objekt hráče **me** je ten, kdo hraje na klientovi a patří mu jednotky a budovy, se kterýma může klient manipulovat. **Enemy** jsou nepřátelští hráči, proti kterým hráč bojuje a **ally** je hráč, který hraje v alianci s hráčem **me**. Jak je už asi zřejmé, nejprve musí funkce rozdělit, komu tato data přišla. Rozdělení probíhá naprosto stejně jako ve funkci **onmessage**, a to za pomocí přepínače. Po rozdělení se projdou objekty hráče a podle identifikačního čísla se najde objekt pro aktualizaci. Server neposílá zbytečně data, která jsou stále aktuální, a proto je potřeba profiltrovat, která data jsou zasílana a ty nechat aktualizovat.

5.4.3 Zasílaní zpráv

```
var dataToSend = [];
dataToSend.push(new createTank(type));
if (dataToSend.length > 0) {
    server.send(JSON.stringify(dataToSend));
}
function createTank(type){
    this.action = "createTank";
    this.type = type;
    this.buildingId = me.getSelectedModels()[0].getId();
    this.option = document.getElementById("tankCreateOption").value;
}
```

Výpis 4: Zasílaní zpráv z HTML klienta

Při zasílaní zpráv z klienta na server, je opět třeba rozlišit, o jakou zprávu se jedná. Každá zpráva má svou funkci, ve které se nastavují potřebné parametry, jako je v ukázce kódu č. 4, který popisuje zasílaní zprávy o vytvoření vojenské jednotky typu tank. Opět je zde možnost zaslat více zpráv najednou, ale z klienta se zasílají data pouze po nějaké akci od uživatele.

5.5 Výběr jednotek

Ve hře je možné vybrat jednotku nebo budovu dvěmi způsoby. První způsob je kliknutím a vybráním pouze jednoho modelu. Druhý je táhnutím myši a vybrání více jednotek zároveň. Pro tyto způsoby vybíraní jsem vytvořil třídu **boundBox**, která implementuje funkce pro vykreslování čtverce znázorňující oblast výběru a funkce, které řeší samotné vybíraní 3D modelů.

```
this.setSelected = function(player) {
    if (this.isPressed) {
        if (this.clickStart.x == this.clickEnd.x && this.clickStart.y == this.clickEnd.y) {

            //implementace prvniho zpusobu vyberu

            var projector = new THREE.Projector();
            var vector = new THREE.Vector3((this.clickStart.x / window.innerWidth) * 2 - 1, -(this.
                clickStart.y / window.innerHeight) * 2 + 1, 0.5);

            projector.unprojectVector(vector, scena.getCamera());
            var ray = new THREE.Raycaster(scena.getCamera().position, vector.sub(scena.
                getCamera().position).normalize());
            var models = [];
            var selects = ray.intersectObjects(player.getAllDynamicModels(models), true);

            if (selects.length > 0) {
                //vyhledani nakliknute jednotky
            }
        } else {

            //implementace druheho zpusobu vyberu
```

```
        }
    }
};
```

Výpis 5: Ukázka výběru jednotek a budov

Následující popis se vztahuje ke kódu č. 5. V případě, že hráč pouze klikne (pozice při stisknutí tlačítka na myši se rovná pozici při uvolnění tlačítka), provede se část kódu označená poznámkou: "implementace prvniho zpusobu vyberu". V této části se provedou složité funkce, které vypočítají směr (ray - paprsek) kliku ve 3D scéně. Na tomto paprsku zavolám funkci **intersectObjects** a jako parametr předám 3D modely. Tato funkce zkontroluje, jestli nedošlo ke střetu paprsku s 3D modelem a pokud ano, vrátí pole s objekty typu Mesh. Jesliže je nějaký výsledek, zkontroluje se, kterému modelu Mesh patří a uloží se do pole vybraných jednotek. Implementace druhého způsobu je jednodušší, jelikož se zde kontrolují pozice modelů s pozicemi oblasti výběru. Modely, které se nacházejí uprostřed výběru, se uloží do pole vybraných modelů.

5.6 Ovládaní klienta

Pro ovládaní klienta a celé hry jsou velice důležité listenery v JavaScruptu, které se dají nastavit na HTML elementy. V mém případě je tento element canvas. V kódu č. 6 je ukázáno, jaké události jsou v projektu použité. Všechny události, až na změnu velikosti okna ('resize'), jsou nastaveny na element canvas. Pro nastavení listeneru se zavolá funkce **addEventListener** se třemi parametry. První parametr je název, jaký typ listeneru má být použit. Jako druhý parametr se nastaví funkce, která vykoná potřebnou funkcionalitu a třetí parametr se může vynechat, v mém případě není potřebný a jeho výchozí hodnota je False.

```
window.addEventListener('resize', function onWindowResize() {}, false);

canvas.addEventListener('mousewheel', function(event) {}, false);

canvas.addEventListener('mouseup', function(event) {}, false);

document.addEventListener('mousemove', function(event) {}, false);

canvas.addEventListener('mousedown', function(event) {
    switch(event.which) {
        case 1:

        break;
        case 3:

        break;
    }
}, false);
```

Výpis 6: Výpis použitých listenerů

- **resize** – Tento listener se spustí pokaždé, když je elementu změněna velikost.

- **mousewheel** – Tento listener se pustí pokaždé, když je kurzor myši na elementu (canvas) a zároveň je aktivované (otočené) kolečko myši.
- **mouseup** – Listener spouštějící funkci, až po uvolnění stlačeného tlačítka myši.
- **mousedown** – Tento listener se aktivuje pokaždé, když je stisknuté tlačítko myši. V ukázce kódu č. 6 je znázorněno, jak se pomocí přepínače rozlišují stisknutá tlačítka.
- **mousemove** – Listener, který se spouští po pohybu myši nad elementem.

Nyní jsem vysvětlil ovládání klienta ve 3D scéně, ale ještě chybí popsat, jak funguje ovládání herního panelu. Toto ovládání je natolik lehké, že ukázka kódu je zbytečná. V elementu, který má reagovat na kliknutí myši, se nastaví atribut **onclick** a přiřadí se mu do uvozovek funkce, která má vykonat potřebnou funkcionalitu.

5.7 Struktura serveru

Pro přehlednost je server rozdělen do několika balíčků. Jelikož jsem autorem tohoto serveru, tak rozdělení tříd do jednotlivých částí je dle mého uvážení a může se někomu zdát zvláštní nebo nesrozumitelné, ale podotýkám, že rozdělení nemá vliv na funkci.

- **artificialIntelligence** – Tento balíček obsahuje třídu pro umělou inteligenci.
- **GameController** – V tomto balíčku se nacházejí všechny důležité třídy, které implementují jádro celé hry.
- **Models** – Balíček obsahující všechny třídy, které reprezentují 3D modely ve hře.
- **PathFind** – V tomto balíčku se nachází pouze jedná třída, která implementuje algoritmy pro hledání cesty.
- **ServerSide** – Balíček obsahuje hlavní spouštěcí třídu a třídu, která řídí spouštění her a obsluhuje příchozí zprávy.

Toto rozdělení už poodhalilo vnitřní rozdělení serveru, ale aby bylo možné pochopit celou logiku serveru, je zapotřebí si jednotlivé třídy popsat.

artificialIntelligence:

- **AI** – Třída obsahující implementaci umělé inteligence.

GameController:

- **Controller** – Třída, ve které je řešena smyčka celé hry za pomocí vlákna.
- **Grid** – Tato třída uchovává data o hrané mapě a obsahuje i funkce, které s těmito daty úzce souvisí. Například kontrola překážek pro čistou cestu střely, kontrola místa pro stavbu budovy a další.

- **MapData** – Zpracovává data mapy. Nacházejí se zde metody pro čtení a ukládání dat ze souboru XML.
- **Message** – Třída, ve které se vytvářejí zprávy pro klienta.
- **Player** – jak už název napovídá, objekty této třídy reprezentují reálné hráče ve hře nebo pokud se hráč odpojí, tak ho začne ovládat umělá inteligence.
- **ThreadCompleteListener** – Interface sloužící pro komunikaci z herní smyčky (Controller) do třídy, která ovládá celý server (ServerMethods2).

Models:

- **Army** – Třída reprezentující vojenské jednotky, jako je tank nebo voják.
- **ArmyBase** Třída, která nenese implementaci, ale slouží pro jednoduché rozpoznání zničené hlavní budovy (cíl hry).
- **AskForBuild** – Jednoduchá třída, která uchovává data pro hráče, který chce stavět budovu.
- **Barracks** – Třída reprezentující kasárny, ve kterých se vytváří jednotka typu voják.
- **BaseModel** – Hlavní rodič všech 3D modelů. Nachází se v něm základní proměnné, které mají všechny budovy a jednotky stejné.
- **Bomb** – Už podle názvu je zde jasné, že se jedná o třídu která reprezentuje bombu svrženou letadlem.
- **Building** – Rodič pro všechny budovy.
- **Bullet** – Střelivo vojenských jednotek. Střetem tohoto objektu s nepřátelskou budovou nebo jednotkou vznikne poškození o síle, jakou měla útočící jednotka.
- **Laboratory** – tato třída reprezentuje laboratoř, kde se provádějí vylepšení vojenských jednotek.
- **Model** – Rodič pro všechny vojenské i nevojenské pozemní jednotky.
- **ModelData** – Zde se necházejí statické proměnné, které nesou data pro nastavení jednotek (ceny, palebná síla, životy).
- **ModelState** – Enum pro Model, ve kterém se nacházejí popisy stavů (pohyb, útok, stůj).
- **OilExtraction** – Tato třída reprezentuje těžbu ropy.
- **OilRafinery** – Třída, která reprezentuje ropnou rafinerii. Zde se převádí ropa na peníze.

- **OilTruck** – Nevojenská jednotka, která převáží ropu.
- **Plane** – Vojenská vzdušná jednotka, která se nedá zničit. Implementuje metody pro svrhávaní bomb.
- **PlaneBuilding** – Tato třída implementuje metody pro radiovou věž, která ovládá letadlo.
- **TankFactory** – Třída reprezentující továrnu, ve které se vytvářejí tanky.

PathFind:

- **PathFinding** – Třída implementující vlákno, které ovládá algoritmus pro hledaní cesty na mapě (data v gridu).

ServerSide:

- **Main** – Tato třída spouští a nastavuje celý server.
- **ServerMethods** – Tak jako Controller ovládá celou hru, tak tato třída ovládá celý server a spouští nebo zastavuje jednotlivé hry.

5.8 WebScokets na serveru

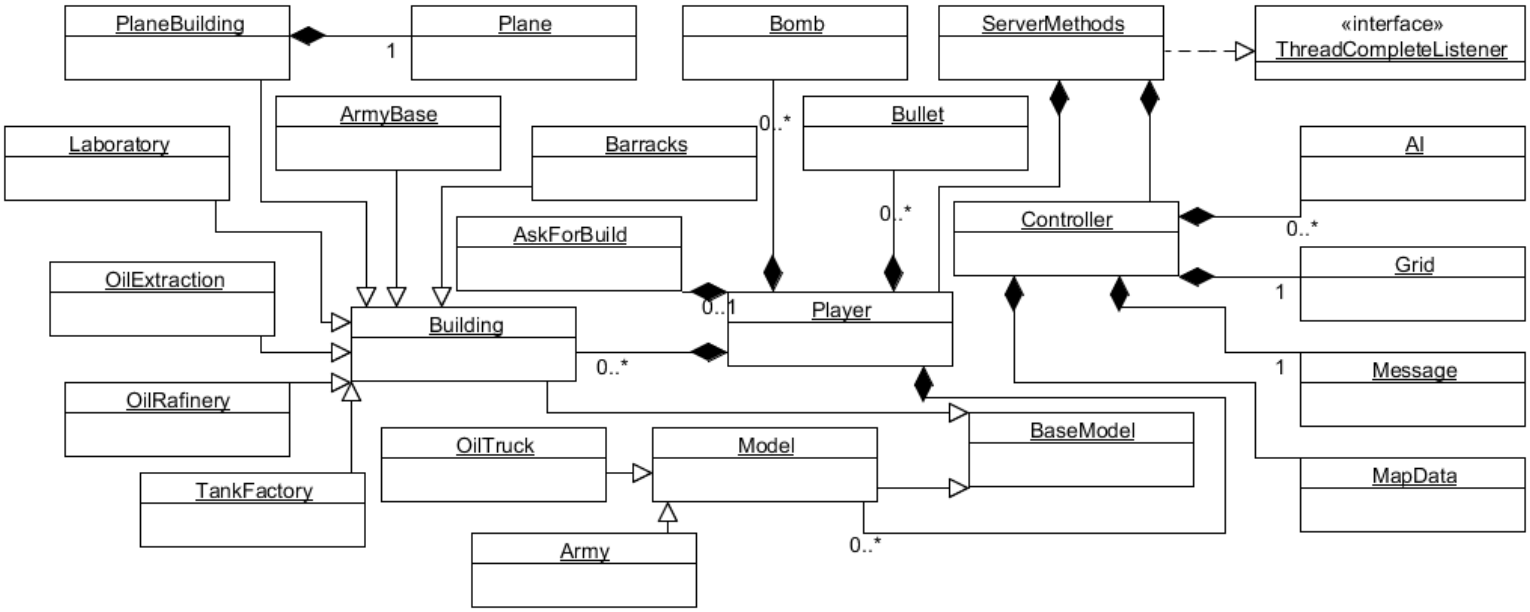
Aby bylo možné komunikovat s klientem, který pro komunikaci využívá WebSockets, je potřeba stejnou technologii implementovat na server. Tuto implementaci nabízí knihovna **Jetty**. Zdroj [6] mi posloužil jako návod pro implementaci webových soketů na serveru.

```
server = new Server(new InetSocketAddress(InetAddress.getByName("127.0.0.1"), port));

WebSocketHandler wsHandler = new WebSocketHandler() {
    @Override
  public void configure(WebSocketServletFactory factory) {
        factory.register(ServerMethods.class);
    }
};
server.setHandler(wsHandler);

server.join();
server.start();
```

Výpis 7: Spuštění a nastavení serveru pro komunikaci s Websockets

Nejprve je třeba nastavit na jaké adrese a portu bude server komunikovat. V mém případě je to adresa localhostu. Jako další věc je zapotřebí registrovat třídu, která bude komunikaci řídit (ServerMethod). V podstatě se v registrované třídě budou nacházet základní metody pro komunikaci jako u klienta, který je popsaný v kapitole 5.4.1. Je dobré zavolat funkci **join()**, která zabrzdí celé spouštění do doby, než bude server nastaven a připraven na spuštění. Spuštění celého serveru má na starosti funkce **start()**.

To že je třída **ServerMethod** registrována neznamená, že je práce hotová a může se vesele komunikovat. Proto, aby vše fungovalo, je zapotřebí správně nastavit anotace.



Obrázek 4: server –UML diagram tříd

Potřebné anotace:

- **@WebSocket** – Anotace pro registrovanou třídu.
- **@OnWebSocketClose** – Anotace patřící před funkci, která bude zachytávat odpojené hráče.
- **@OnWebSocketError** – Anotace pro funkci zachytávající chyby ve spojení.
- **@OnWebSocketConnect** – Tato anotace patří pro funkci, která bude zaznamenávat nové připojení.
- **@OnWebSocketMessage** – Pro funkci zachytávající příchozí zprávy.

Důležitý parametr těchto funkcí je **Session**, ve kterém jsou uložené všechny potřebné informace patřící klientovi. Tento parametr uložím do hráče a díky němu pak dokáži rozlišit, kdo posílá data nebo který hráč se odpojil. Jelikož je zpracování přijatých zpráv totožný se zpracováním v klientovi (kapitola 5.4.2), tak v další části popíši jen odsilání zpráv.

5.8.1 Odesilání zpráv

Aby nedocházelo ke zbytečnému zahlcování komunikačního kanálu, jsou zprávy odesilány na konci každého cyklu hlavní smyčky (třída Controller). Při průchodu se vytvářejí zprávy, které se ukládají ve třídě **Player** a každému hráči je vytvořena jeho vlastní zpráva. Všechny funkce na vytváření zpráv se nacházejí ve třídě **Message**.

```
for (Player player : players) {
    JSONObject data = new JSONObject();
    try {
        data.put("actionObject", "deletePlayer");
        data.put("playerId", removePlayer.getId());
    } catch (JSONException e) {
        e.printStackTrace();
    }
player.addToMessage(data);
}
```

Výpis 8: Vytvoření zprávy o prohře hráče

Kód č. 8 znázorňuje vytvoření zprávy pro všechny stále hrající hráče, která oznamuje, že hráč s tímto identifikačním číslem prohrál a klient musí smazat všechny jeho jednotky a budovy. Jelikož se zprávy předávají ve formátu String, ale pro práci s daty je lepší formát JSON, tak je nejprve vytvořen objekt data, do kterého se pak jednoduše vkládají informace. Jako první parametr je klíč ukazující na informaci. Jelikož klient i server musí rozeznat typ zprávy, je u každé nastaven klíč **actionObject**, který ukazuje na název akce, která se má na klientovi provést. Pokud je už zpráva nastavena, uloží se k hráči.

```
if (messages.length() == 0){return;}
if  (isDisconect()) {
    clearMessages();
    return;
}

try {
    session.getRemote().sendString(messages.toString());
} catch (IOException e) {
    e.printStackTrace();
}
clearMessages();
```

Výpis 9: Odesilání zprávy ze serveru

V kódu č. 9 je ukázáno, jakým způsobem server odesílá zprávy. Konkrétně je tato metoda, která implementuje tento kód, uložena ve třídě Player. Jelikož se při vytváření zpráv kontroluje, zda posílaná data jsou stejná s daty, které byly už hráči zaslány, tak může dojít k tomu, že hráč nemá zprávu, kterou by poslal na klienta. Proto je důležitý první if. Další blok if kontroluje, zda je hráč opravdovým hráčem nebo je odpojen a za něj hraje umělá inteligence. V případě, že je odpojen dojde k vymazání uložených zpráv a konečně je zde samotné odesílání. Každý hráč má proměnnou session a nejen, že se díky této proměnné kontroluje totožnost hráče, ale jde přímo pomocí této proměnné posílat zprávy.

5.9 Řízení hry

5.9.1 Spouštění hry

Jakmile se připojí nový hráč k serveru, zařadí se do listu čekajících hráčů a pošlou se mu všehny mapy pro výběr. Hráč na klientovi vybere mapu, typ hry a pošle se zpráva na server. Server tuto zprávu zpracuje a vyhledá ve frontě čekajících hráče, kteří mají stejné požadavky pro hru. Jakmile je nalezeno dostatečný počet hráčů, vytvoří se vlákno pro hru, které je řidáno do listu hrajících her a předají se mu objekty hráčů. Tito hráči jsou pak z čekající fronty odstraněni. V tomtéž okamžiku se pošlou vybraným hráčům data pro nastavení mapy a hra začíná.

5.9.2 Ukončení hry

Jakmile dojde k ukončení hry, které je popsáno v kapitole 6.1.6, tak se pomocí listeneru zavolá funkce **finishGame** a do parametru se předá vlákno (controller), které je nutno ukončit. Listener je naimplementovaný ve třídě **ServerMethods** a zde se vyhledá hra, která má být ukončená a ještě před tím se ukončí všechny běžící umělé inteligence. Hráči z končené hry se opět vloží do listu čekajících, vymažou se jim paramatry hledané hry a pošle se opět zpráva s daty pro výběr mapy.

5.10 Hlavní smyčka na serveru

Jak jsem se už zmínil, tak hlavní smyčka ovládající hru se nachází ve třídě Controller. Jelikož se zde ovládá celá hra, tak je zapotřebí, aby celé vlákno fungovalo s určitou frekvencí. Pokud budu uvažovat, že jeden cyklus je jako jeden snímek na obrazovku, tak potřebuji, aby cykly byly stálé a s určitou frekvencí. Aby bylo možné vypočítat měnnou dobu uspání vlákna, je potřeba na začátku každého cyklu zaznamenat čas. Čas uspání se vypočítá tak, že se nejprve vypočítá počet milisekund pro danou frekvenci a od toho se odečte doba průchodu, což je čas na začátku a aktuální čas při výpočtu.

V tomto vlákně se spíše volají potřebné funkce na jednotlivé modely jako například pohyb jednotek, útok jednotek, vytváření a posílání zpráv, vytváření jednotek a další.

5.11 Algoritmus pro hledání cest

Důležitou části pro pohyb jednotek na mapě je algoritmus řešící tuto problematiku. Existuje několik algoritmů, ale já jsem si vybral algorimus s trochu podivným názvem **A*** [10], který je naimplementovaný ve třídě PathFinding. Každá hledaná cesta se provádí ve svém vlastním vlákně, jelikož může celý proces podle mých testů trvat i přes minutu.

Než začnu vysvětlovat tento algoritmus, je potřeba si nejprve vysvětlit pár pojmů, které budu při vysvětlování používat.

- **Uzavřený list** – List uzlů, které už byly zpracovány.
- **Otevřený list** – List volných uzlů, které je potřeba zpracovat.
- **uzel** – Jednotlivé čtverce, které tvoří celou mapu. Každý uzel nese informace, jako je indentifikační číslo, cenu a souseda pro zpětné nalezení cesty.
- **cena** – Metrika vzdáleností.

Jako první je potřeba nastavit cíl, start, data mapy a jednotku, pro kterou se hledá cesta. Na začátku se vloží startovní uzel do otevřeného listu a začne se smyčka, která běží dokud aktuální uzel není ten cílový nebo otevřený list není prázdný. Ve smyčce se seřadí otevřený list od nejnižší ceny a ten s nejnižší cenou se nastaví jako aktuální uzel. Vybraný aktuální uzel se vymaže z otevřeného listu a vloží se do listu uzavřeného. Hned po tom je potřeba zkontrolovat aktuální uzel, jestli to není cílový a pokud není, začnou se kontrolovat jeho sousední. Kontrola sousedů probíhá v několika krocích. V prvním kroce je potřeba zjistit, jestli se nenachází v uzavřeném listu. V dalším kroce je potřeba zkontrolovat, jestli je na kontrolovaném místě volno (kontrola dat na mapě). Tady bych se rád zastavil a podotknul, že pokud hledám cestu k budově, kterou chci napadnout, tak cílový bod se nachází uprostřed budovy, což je místo, kam by se algorimus nedostal. Proto zde kontroluji identifikační číslo budovy s daty na mapě (5.12) a pokud se rovnají, tak se kontrola nezruší. Pokud uzel neprojde přes tyto dvě kontroly, tak přeskočí na dalšího souseda. Pokud ale prošel vypočítá se mu cena. Cena se vypočítá tak, že se sečte hodnota souseda (pokud je soused ve vertikální nebo horizontální rovině s aktuálním uzlem, tak se mu přidělí hodnota 10, ale pokud je šikmo, tak hodnota 14) s hodnotou aktuálního

uzlu a hodnotou k cílovému uzlu (počet řádků k cílovému uzlu plus počet sloupců k cílovému uzlu). Po vypočtení ceny se zkontroluje otevřený list, jestli se soused v něm už nenachází. Jestliže se nachází a má větší cenu, tak se mu nastaví nově vypočtená a jako cesta k němu se nastaví aktuální uzel. Pokud nebyl nalezen v otevřeném listu, tak se vytvoří nový uzel a vloží se do něj a pokračuje se na dalšího souseda. Jakmile je smyčka ukončena, tak se vytvoří zásobník s daty, který se vloží jednotce. Jelikož má každý uzel uložen cestu k předchozímu uzlu, tak se zásobník plní od cílového uzlu postupně až k startovnímu. Jakmile se předají data je vlákno ukončeno.

5.12 Data na mapě

Aby bylo možné pracovat s mapou, je potřeba ukládat její data do dvourozměrného pole a každému uzlu přiřadit informaci o tom, co se na jeho místě nachází. Toto pole a funkce k práci s ním se nacházejí ve třídě **Grid**.

Označení dat na mapě:

- **-** – Volné ničím neobsazené místo.
- **x** – Místo zabrané stromem nebo kamenem.
- **#** – Označení ropy.

Označení pro budovy se kvůli pozdějším pracím značí složitěji.

A1:2

- **A** – První znak z typu budovy.
- **1** – Identifikační číslo hráče.
- **2** – Identifikační číslo budovy.
- **:** – Znak pro rozdělení dvou čísel.

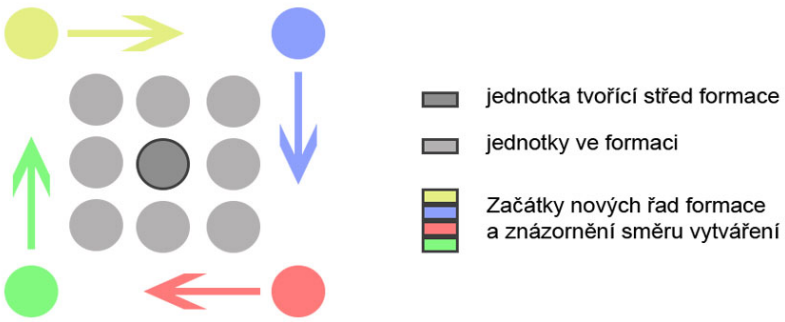
Ve třídě Grid se nacházejí i důležité funkce jako je **checkPlaceNextToBuilding**, která hledá nejbližší volné místo kolem budovy, **checkFormationPosition** funkce řešící formaci jednotek nebo také **freeWayForShoot**, která kontroluje volnou trasu pro střelu.

5.13 Formace jednotek

Jelikož hra nabízí i manipulaci s více jednotkama najednou, tak je zapotřebí, aby při přesunu nedocházelo k tomu, že se všechny jednotky postaví na to samé místo. K tomu jsem vytvořil algoritmus, který dříve než se začne jednotce vyhledávat cesta, změní cílový bod.

Algoritmus funguje tak, že se vezme první jednotka a nastaví ji jako střed. Kolem tohoto středu se budou stavět další jednotky a to tak, že se postaví horizontálně a vertikálně vedle něj a šikmo k němu. Tady jde jednoduše vypozorovat, že každý další "obal"je

o dvě jednotky větší než ten předchozí na každé straně, jak je zobrazeno na obrázku č. 5. Při vytváření nového obalu formace se střídají strany, na kterou se jednotka postaví, tím dojde k zaplnění celého obalu. Jakmile jsou řady zaplněné, algoritmus zvětší rozsah o dvě jednotky a začne se opět přídávat na každou stranu. Pochopitelně řeší algoritmus i zda jsou místa volná k postavení pro jednotku.



Obrázek 5: Princip vytváření formace

5.14 Umělá inteligence

Jelikož bylo v zadání i vytvoření jednoduché umělé inteligence, tak jsem vytvořil algoritmus, který reprezentuje tuto umělou inteligenci a nachází se ve třídě **AI**. Tento algoritmus je spuštěn ve vlákně a v případě, že hrají čtyři hráči, může být v jedné hře spuštěno hned několik těchto vláken. Tomuto vláknu se předá objekt hráče, kterého ma algoritmus zastoupit. Algoritmus je rozdělen do několika části a tyto části jsou volány příslušnými funkcemi.

Jako první je volána funkce **setData()**. Tato funkce nastaví všechny potřebné data a na základě těchto nastavených dat se bude v dalších částech rozhodovat.

Ve druhé fázi se spouští funce **makeChoice_building()**, ve které se staví budovy. V této funkci jsou seřazeny podle priority kontroly postavených budov a pokud nějaká chybí a je dostatek peněz, tak se najde místo pro budovu a postaví se.

V třetí fázi se spouští funkce **upgrade()**, která nejprve kontroluje, jestli je postavena laboratoř a pokud ano, tak kontroluje, jestli je možné vylepšovat určité věci ve hře, a to podle množství jednotek a množství provedených vylepšení. Pokud projdou tyto kontroly pozitivně, tak se náhodně vybere jedna z možností vylepšení, která se má provést.

Čtvrtá fáze spouští funkci **makeChoice_army()**, která má za úkol vytvářet vojenské jednotky. Opět se kontroluje, jestli je již postavena továrna a kasárna. Pokud ano, tak podle počtu peněz se rozhodne, jestli se budou vytvářet jednotky. Čím více má jednotek, tím více peněz musí mít pro vytváření, aby bylo možné provádět i jiné úkoly, které stojí peníze.

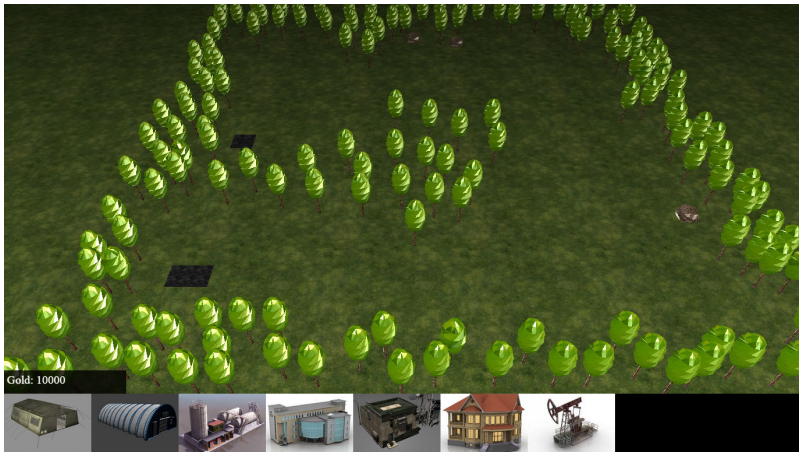
Pátou fázi má na starosti funkce **defense()**, ve které se řeší obrana. Jestliže je nějaká budova pod útokem a má volné jednotky (neútočící), tak pošle tyto jednotky na napadající jednotku.

Šestá a poslední fáze řídí funkce **attack()**, kde dochází k napadání nepřítele. V této části se kontroluje, zda může zaútočit letadlem a nebo i armádou. Jelikož je ropa velice důležitá, tak nejprve napadá těžbu ropy a až poté útočí na základnu nepřítele.

Na konci dojde k vymazání dat pro další cyklus a k uspání vlákna na dvě sekundy.

6 Uživatelská příručka

6.1 HTML klient



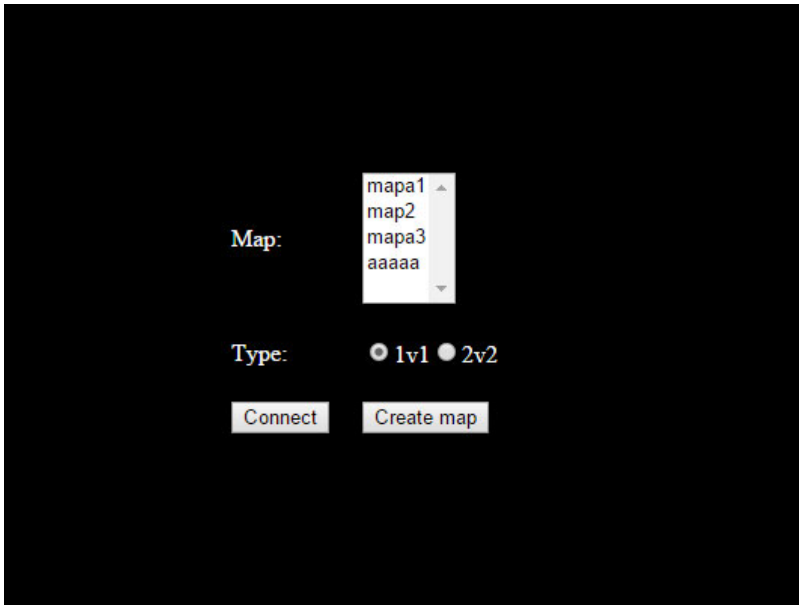
Obrázek 6: Klient po připojení do hry

6.1.1 Spuštění hry

HTML klient se jako každá jiná webová stránka spouští po zadání příslušné adresy do prohlížeče, jelikož je hlavní soubor pojmenován jako index.html. Pro otestování je v příloze složka obsahující server pro operační systém Windows bez nutnosti instalace. Jediné, co je potřeba udělat, je uložit tento server do kořenového adresáře disku C a zkopírovat celého klienta do složky htdocs. Spuštění serveru má na starosti soubor httpd.exe, který je umístěn ve složce bin.

Po načtení stránky se objeví v dolní části obrazovky červený pruh znázorňující průběh načítaní 3D modelů. Po načtení všech potřebných modelů do hry se objeví obrazovka, jako je na obrázku 7, která nabídne hráči hned několik možností.

- Výběr mapy, kde je hráči zobrazen seznam všech dostupných map. V případě, že nejsou zobrazené mapy na výběr, je s největší pravděpodobností server mimo provoz.
- Je zde možnost vybrat dva typy her. 1V1 jsou dva hráči proti sobě a hra končí až jeden z hráčů příjde o základnu. 2V2, jak je už asi zřejmé, jsou čtyři hráči rozdělení do dvou skupin, ve kterých mohou spolupracovat. V tomto případě vyhrává ta skupina, která zničí nepříteli obě základny.
- Jakmile je zvolená mapa a typ hry, může hráč stisknout tlačítko připojit. Hra se nesputí okamžitě, ale zobrazí se text, který hráče informuje o tom, že se vyhledává hráč nebo v případě 2V2 hráči, kteří vybrali stejné parametry pro hru.



Obrázek 7: Klient HTML – připojení do hry

- Hra nabízí i možnost vytvoření své vlastní mapy, která po uložení bude dostupná všem hráčům.

Jakmile server vyhledá hráče, spustí se hra, jako je na obrázku č.6

6.1.2 Ovládaní hry

Pro ovládaní celé hry stačí používat myš. Pro pohyb kamery, tak jako u většiny her tohoto žánru, stačí přesunout kurzor myši na okraj obrazovky a rychlost pohybu závisí na míře přiblížení kamery. Pokud je kamera oddálená, tak pohyb je rychlejší, v opačném případě je pohyb pomalejší, aby nedocházelo k nechtěným přílišným posunutím. Pro přibližování pak poslouží kolečko myši. V dolní částí obrazovky se nachází panel, na kterém se zobrazují všechny potřebné informace. Po spuštění hry je zobrazen hlavní panel, který je zobrazen na obrázku 8. Tento hlavní panel zobrazuje všechny budovy pro stavbu. Jako první budovu je potřeba postavit Základnu. Všimněte si, že po spuštění hry se Vaše kamera nachází nad určitou části mapy. Není podmínkou zde začít stavět, ale pro zachování bezpečné vzdálenosti od protihráče je toto místo vhodná volba. Při vybíraní jednotek nebo budov slouží levé tlačítko myši a zobrazí se příslušný panel zobrazující informace nebo akce, které může jednotka nebo budova provést. V případě potřeby vybrání většího počtu jednotek, stačí stisknout levé tlačítko myši a táhnout přes ně. Pokud jsou vybrány jednotky a je potřeba je přesunout na jinou část mapy, je potřeba stisknout pravé tlačítko myši. Toto tlačítko rovněž slouží pro útok na nepřátelskou jednotku. Nad panelem se rovněž nachází množství zlata. Pokud máme vybrané jednotky zobrazí se

panel, na kterém je i možnost nastavit jednotky na hlídkovaní. Stačí vybrat políčko a kliknout na místo na mapě, jednotky se budou pohybovat sem a tam.

Obrázek 8: Klient HTML – hlavní panel

6.1.3 Stavba budov

Důležitou části hry je stavba budov. Po vybrání jakékoli budovy se zobrazí na mapě bílé mřížkování, které znázorňuje po jakých částech se může budova při stavbě pohybovat. Po vybrání se objeví budova na mapě s červenou nebo zelenou plochou pod ní. Tato plocha slouží jako informace, zda je možno na daném místě budovu postavit. V případě, že plocha svítí zeleně je dovoleno na místě stavět a po kliknutí levého tlačítka se budova postaví. V opačném případě, tedy pokud svítí červeně, tak se po stisknutí levého tlačítka žádná akce nestane. Pravé tlačítko zruší stavění a je možné vybrat jinou budovu. Pro otočení budovy slouží kolečko na myši. Při stavění se nemusí na nic čekat, budova se postaví okamžitě.

6.1.4 Popis budov

Ve hře se nachází několik druhů budov. Každá ma svou cenu, funkci a některé se musí postavit na správné místo. Všechny 3D modely jsou k získaní zdarma na stránce [11].

6.1.4.1 Základna Jak už jsem se zmínil v odstavci 6.1.2, tuto budovu je potřeba postavit jako první. Jakákoli jiná budova nepůjde jako první postavit. Tato budova slouží pouze jako cíl pro nepřítele a nemá žádné další funkce ani informace. Tuto budovu lze postavit pouze jednou. V hlavním panelu na obrázku č. 8 se nachází jako šestá v pořadí.

6.1.4.2 Kasárna Tato budova slouží k vojenským účelům pro vytváření vojáků. Po kliknutí na budovu se zobrazí panel. Nachází se na něm tlačítko, na kterém je v pozadí voják. Po stisknutí se na tlačítku objeví číslo a při opětovném stisknutí se toto číslo zvětší. Toto číslo znázorňuje, kolik vojáků bude postupně vytvořeno. Další věc, která se po stisknutí zobrazí, je průhledně zelené pozadí překrývající obrázek vojáka na tlačítku. Tento zvětšující se pruh je průběh vytváření vojáka. Jakmile je voják vytvořen, stoupne si vedle kasáren. V hlavním panelu na obrázku č. 8 se nachází na první pozici.

6.1.4.3 Továrna na tanky Tato budova je téměř stejná jako kasárna, ale místo vojáků se zde vyrábějí stejným způsobem tanky. Jednu odlišnost to ale má. Vedle tlačítka pro výrobů tanků se nachází posuvné tlačítko. Toto tlačítko slouží k úpravě vyvíjeného tanku.

V případě, že ho například posuneme směrem doleva, snížíme na určité procento maximalní životy, ale zvýšíme poškození. Ve výchozím stavu je procento poškození a životu na 100%. Toto nastavení je ale pouze pro tanky, které teprve budou přidané do výrobní fronty. V hlavním panelu na obrázku č. 8 se nachází na druhé pozici.

6.1.4.4 Rafinérie Rafinérie je druh budovy pro ekonomické účely. Jejím hlavním úkolem je převést ropu na palivo a prodat ho. Tím získá hráč potřebné zlato. Tento proces probíhá pomalu po určitém množství. Po kliknutí na rafinérii se objeví panel, který podobně jako u kasárny či továrny na tanky má funkci vytvořit cisternu. Jako další věc je zde i informace o množství ropy v zásobníku. V hlavním panelu na obrázku č. 8 se nachází na třetí pozici.

6.1.4.5 Laboratoř Tato budova je efektivnější až v pozdější herní době, jelikož slouží pouze ke vylepšování maximalních životů tanků i vojáků a ke zvýšování palebné síly. Po vybrání laboratoře se objeví panel, který se skládá ze čtyř již zmíněných částí. Každá tato část má tlačítko pro provedení vylepšení a zelený pruh znázorňující průběh vyvoje s nápisem informující hráče o připravenosti k vylepšení. Tuto budovu může mít hráč postavenou pouze jednu, ale v případě že je zničená, má možnost postavit další. V hlavním panelu na obrázku č. 8 se nachází na čtvrté pozici.

6.1.4.6 Rádiová věž Tato věž je velice užitečná v bojích, jelikož díky ní je možno provádět letecké bombardování. Po vybrání této věže je zobrazen panel, který informuje, zda je nálet připraven. Podobně jako u laboratoře, je zde zelený pruh s textem, který informuje hráče o stavu připravenosti k leteckému úderu. Jakmile je nálet připraven, stisknutím pravého tlačítka myši na mapě dojde k leteckému útoku. V hlavním panelu na obrázku č. 8 se nachází na páté pozici.

6.1.4.7 Těžební věž Tuto věž lze postavit pouze na místo, kde se nachází ropa. Ropu na mapě poznáme podle tmavě zbarvených míst jako na obrázku č. 6. Jedinou věc, kterou po vybrání věže zobrazí panel, je informace o množství vytěžené ropy a uložené pro převoz do rafinérie. V hlavním panelu na obrázku č. 8 se nachází na posledním místě.

6.1.5 Popis jednotek

Důležitou části bojových strategických her jsou vojenské jednotky. Každá vojenská jednotka má dva rozsahy. V tom širším rozsahu pozoruje okolí a v případě, že nemá nastaven cíl, sám si ho automaticky vyhledává. Druhý rozsah je pro palbu. Každá jednotka má palebný dosah jiný. Jestliže je mezi jednotkou a cílem postavena nějaká budova, automaticky hledá volnou cestu pro střelu. Ve hře se nevyskytují pouze vojenské jednotky, ale i cisterna pro převoz ropy. I tyhle 3D modely jsou všechny k získaní zdarma na stránce [11].

6.1.5.1 Voják Voják je nejlevnější vojenská jednotka. Její palebná síla a maximalní množství životů nejsou silnou stránkou, ale alespoň nízká cena a rychlá výroba vynahradí tyto nedostatky. Voják se vyrábí v kasárnách.

6.1.5.2 Tank Tato jednotka bude ve hře nejpoužívanější, jelikož má velkou palebnou sílu a dostatek životů k delšímu boji. Jeho výroba je oproti vojákovi delší a za větší cenu, ale v pozdější části hry se investice a čas věnovaný této jednotce vyplatí. Vyrábí se v továrnách.

6.1.5.3 Cisterna Pro převoz ropy z těžební věže do rafinérie je potřeba cisterny. Cisterna není vojenská jednotka, proto se může stát snadným terčem. Nepotřebuje od uživatele žádné akce, sama si dokáže najít těžební věž a rafinérii. Vyrábí se v rafinérii.

6.1.5.4 Bombardér Toto letadlo patří k vojenským jednotkám, ale nedá se zničit. Je ovládáno z radiové věže a v případě vypuštění bomby, zničí nebo poškodí jakoukoli jednotku nebo budovu v okolí, proto je nutné, aby hráč dával pozor, jelikož může dojít k nechtěným ztrátam svých jednotek. Jediná možnost, jak zabránit bombardování, je zničit nepříteli radiovou věž.

6.1.6 Konec hry

V případě, že dojde ke zničení základny nepřítelem, hra se ukončí a informuje hráče o vítězství nebo porážce. Hráči je nabídnuta možnost vrátit se na nabídku, která je popsána v odstavci 6.1.1. Jesliže při hraní dojde k odpojení jednoho z hráčů, server okamžitě nahradí odpojeného hráče umělou inteligencí a tím nedojde ke zrušení rozehrané hry.

6.1.7 Vytvoření mapy

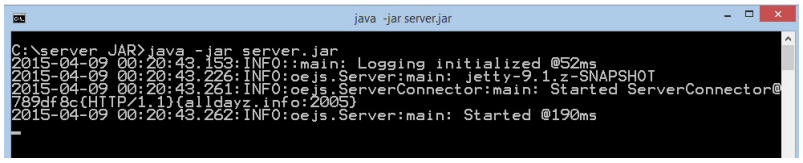
Možnost vytvoření vlastní mapy se nachází na nabídce vedle připojení do hry, obrázek č. 7. Po stisknutí tlačítka se objeví prázdná mapa s panelem, na kterém jsou dekorace jako strom, kámen nebo také důležitá věc, která by měla být na každé hratelné mapě a tou je ropa. Na rozdíl od stavění budov je možné objekty stavět na sebe. Před uložením mapy na server je třeba mapu pojmenovat.

6.2 Sever

6.2.1 Ovládání serveru

Jelikož je server psán v jazyce Java, je potřeba mít tuto platformu na svém počítači nainstalovanou. Spouštění serveru probíhá v příkazovém řádku a to příkazem java -jar server.jar. Po zadání tohoto příkazu se vypíše informace o spouštěném serveru, jako je na obrázku č. 9. Výchozí port je nastaven 2005, ale jestli je využíván jinou službou, je možné spustit server s jiným portem, například 2000, příkazem java -jar server.jar 2000, ale tento

nový port se musí nastavit u klienta v index.html souboru. Vypnutí serveru provedete opět v příkazovém řádku stisknutím kláves ctrl+c.



Obrázek 9: Spuštění serveru

7 Závěr

Cílem práce bylo vytvořit 3D strategickou hru v prostředí HTML a server. Všechny zadané cíle jsem splnil. V průběhu vývoje jsem sám na sobě pozoroval, že kód na konci vývoje je přehlednější a srozumitelnější než kód, který jsem psal na začátku práce. Můžu říci, že tato práce mi obohatila zkušenosti a je určitým krokem k mému zlepšení v programování. Jelikož jsem při práci používal dva programovací jazyky Javu a JavaScript, tak pokud bych si měl vybrat, v jakém jazyce se mi vyvíjelo lépe, tak jednoznačně v Java. Ze začátku, jelikož jsem byl v JavaScriptu nováčkem, mi dělalo problém se v tomto jazyce zorientovat. Při vývoji jsem vyzkoušel několik nástrojů pro vývoj v JavaScriptu, ale nedokázal jsem najít nástroj s funkčním našeptávaním, tak jako v jazyce Java, ale na tento nedostatek jsem si v pozdější fázi vývoje zvyknul a nedělal mi problémy.

Při vývoji jsem nenarazil na žádné větší problémy, se kterými bych si nevěděl rady. Jen prohlížeče občas nefungovaly, tak jak by měly. Občas padaly nebo se zasekly. U konce vývoje mi začal dělat problémy webový prohlížeč Chrome, který například nedokázal spustit čtyři herní klienty najednou. Tento problém jsem vyřešil testováním na prohlížeči Opera, na kterém jsem nezaznamenal problémy. Tyto problémy mi ukázaly, že webové prohlížeče, ačkoli mají implementovanou potřebnou technologii, tak nejsou ještě plně stabilní, ale jako protiklad k mému tvrzení bych rád uvedl, že již dnes existuje mnoho velice hezky propracovaných her, které ukazují, že pokud je hra dobře optimalizovaná, tak si s ní webový prohlížeč hravě poradí.

Při vývoji mě napadlo mnoho vylepšení, například chatové okno, kde by si mohli hráči mezi sebou psát nebo kde by přicházely informace ze serveru. Chatové okno bylo jedno z vylepšení, které jsem chtěl udělat, ale jelikož jsem musel řešit vzniklé problémy, tak mi na to nevyšel čas a to mě velice mrzí.

I když se o vývoj her zajímám a ve volném čase vytvářím hry na mobilní zařízení, tak ve vývoji tohoto projektu nehodlám zatím pokračovat, i když bych moc rád, ale pro jednoho je obtížné vyvíjet jak server tak klienta, jde o rozsáhlejší projekt pro více lidí.

8 Reference

[1] Htmlguru: Historie a vývoj HTML. [online]. [cit. 2015-04-16]. Dostupné z: http://htmlguru.cz/uvod-historie.html

[2] Wikipedia: HyperText Markup Language. [online]. [cit. 2015-04-16]. Dostupné z: https://en.wikipedia.org/wiki/HTML

[3] Khronos: WebGL. [online]. [cit. 2015-04-22]. Dostupné z: https://www.khronos.org/webgl/

[4] Zdrojak: Web Sockets. [online]. [cit. 2015-04-22]. Dostupné z: http://www.zdrojak.cz/clanky/web-sockets/

[5] Tutorialspoint: HTML5 - WebSockets Tutorial. [online]. [cit. 2015-04-16]. Dostupné z: http://www.tutorialspoint.com/html5/html5_websocket.html

[6] Jetty: Jetty WebSocket Client API. [online]. [cit. 2015-04-16]. Dostupné z: http://eclipse.org/jetty/documentation/current/jetty-websocket-client-api.html

[7] Kniha.html5: canvas. [online]. [cit. 2015-04-22]. Dostupné z: http://kniha.html5.cz/canvas.html

[8] Codeeval: Most Popular Programming Languages. [online]. [cit. 2015-04-22]. Dostupné z: http://blog.codeeval.com/codeevalblog/2014#.VTfRsiHtmko=

[9] Html5 game engine. [online]. [cit. 2015-04-22]. Dostupné z: https://html5gameengine.com/

[10] Policyalmanac: A* Pathfinding. [online]. [cit. 2015-04-22]. Dostupné z: http://www.policyalmanac.org/games/aStarTutorial.htm

[11] Tf3dm: 3D modely. [online]. [cit. 2015-04-24]. Dostupné z: http://tf3dm.com

9 Seznam příloh

- bakalarska_prace.pdf –Tento dokument v digitalní podobě
- Apache24 – Složka ve které je server na otestování klienta
- Klient – V této složce se nachází celý vytvořený klient
- Server_jar – Zde se nachází vytvořený server v podobě souboru s příponou jar a soubor xml kde si server ukálá mapy.
- Server_workspace – Zde je zdrojový kód serverové části.