Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Interaktivní 3D grafika s WebGL

Interactive 3D graphics with WebGL

Bakalářská práce

Vypracoval: Lukáš Máče

Vedoucí práce: PaedDr. Petr Pexa, Ph.D.

České Budějovice 2017

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lukáš MÁČE**
Osobní číslo:	**P12081**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**Interaktivní 3D grafika s WebGL**

Zadávající katedra: **Katedra informatiky**

Zásady pro vypracování:

Cílem bakalářské práce bude zpracovat možnosti vytváření interaktivní 3D grafiky pomocí JavaScriptového rozhraní WebGL. V rámci práce budou popsány jednotlivé metody vytváření interaktivní 3D grafiky s WebGL, kdy se v prvním případě bude jednat o ukázkové postupy tvorby pomocí JavaScriptových příkazů s využitím knihovny Three.js. V druhém případě bude analyzována metoda vytváření interaktivní 3D grafiky bez znalosti programovacích jazyků pomocí grafických nástrojů, které umožňují export do WebGL, jako je Blender či Coppercube 3D. Nástroje, pomocí nichž lze interaktivní grafiku 3D grafiku vytvářet bez znalosti programovacích jazyků, budou následně porovnány a budou zanalyzovány jejich funkce a možnosti, které nabízejí. Výstupem práce bude sada praktických příkladů, které budou vytvořeny pomocí obou výše zmíněných metod.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **Chrome experiments: WebGL Experiments. [cit. 2016-04-07]. Dostupné z: http://www.chromeexperiments.com/webgl/**
2. **Khronos Group: Getting Started with WebGL. [cit. 2016-04-07]. Dostupné z: http://www.khronos.org/webgl/wiki/Getting_Started.**
3. **ŽÁRA, Ondřej. Začínáme s WebGL. In: Zdroják.cz [online]. 2013. Dostupné z: https://www.zdrojak.cz/serialy/zaciname-s-webgl/**
4. **Learning WebGL [online]. Dostupné také z: http://learningwebgl.com/**
5. **WebGL academy. WebGL academy [online]. Dostupné z: http://www.webglacademy.com/courses.php**
6. **MATSUDA, Kouchi a Rodger LEA. WebGL programming guide: interactive 3D graphics programming with WebGL. Upper Saddle River, NJ: Addison-Wesley, 2013. Always learning. ISBN 0321902920.**

Vedoucí bakalářské práce: **PaedDr. Petr Pexa, Ph.D.**
Katedra informatiky

Datum zadání bakalářské práce: **19. dubna 2016**

Termín odevzdání bakalářské práce: **28. dubna 2017**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 19. dubna 2016

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 10. července 2017.

Lukáš Máče

Abstrakt/Anotace

Cílem bakalářské práce je zpracovat možnosti vytváření a zobrazování interaktivní 3D grafiky pomocí JavaScriptového rozhraní WebGL. V rámci práce popíšu základní metody a postupy, kterých lze využít při vytváření interaktivního 3D obsahu s využitím rozhraní WebGL.

Jednou z možností, kterou se bude tato práce zabývat, je tvorba 3D grafiky prostřednictvím JavaScriptových příkazů, současně s nimi bude u této metody využita Javascriptová 3D knihovna Three.js, která byla vytvořena právě za účelem usnadnění vytváření webového obsahu ve 3D. Druhou z možností, které bude věnována část této práce, je vytváření interaktivní 3D grafiky bez znalosti programovacích jazyků, kterým může být právě například výše zmíněný JavaScript. Tuto možnost představují grafické nástroje, jako je Blender či Coppercube 3D, ve kterých lze 3D obsah za pomoci nejrůznějších technik vytvářet a následně exportovat do WebGL.

U obou metod bude provedena analýza funkcí a možností, které při vytváření 3D grafiky nabízejí a následně budou na základě této analýzy vzájemně porovnány. Výstupem práce bude sada praktických příkladů, vytvořených za pomoci obou výše zmíněných metod.

Klíčová slova

WebGL, JavaScript, 3D grafika, Three.js, Blender, Coppercube 3D

Abstract

The aim of this bachelor thesis is to prepare options for creating and displaying interactive 3D graphics by using JavaScript environment WebGL. In this work, I will describe the basic methods and techniques which can be used to create interactive 3D content using WebGL.

One of the options, which this bachelor thesis will deal with, is creating 3D graphics through Javascript commands, simultaneously using Javascript 3D library Three.js. Second option, which will be described with is creating interactive 3D graphics without knowledge of programming languages. This option is represented by graphics tools like Bledner or Coppercube 3D, which enables creating interactive 3D graphics and export them to the WebGL.

Both methods will be analyzed and compared with each other. The outcome of the work will be set of practical examples, created by both methods.

Keywords

WebGL, JavaScript, 3D graphics, Three.js, Blender, Coppercube 3D

Poděkování

Mé poděkování patří panu PaedDr. Petru Pexovi, Ph.D. za ochotu, vstřícnost a čas, který mi během psaní této práce věnoval.

Obsah

1 Úvod

1.1 Cíle

Cílem bakalářské práce je dopodrobna zpracovat možnosti vytváření interaktivní 3D grafiky pomocí JavaScriptového rozhraní WebGL, které slouží k zobrazování interaktivní 3D grafiky bez použití zásuvných modelů. Detailně popsat funkčnost WebGL a možnosti, pomocí nichž lze obsah pro WebGL vytvářet.

V teoretické části práce se budu zabývat možnostmi, které jsou aktuálně využitelné při tvorbě interaktivního 3D obsahu, ať už se jedná o metodu vytváření grafiky pomocí Javascriptových příkazů s využitím knihovny Three.js, nebo tvorbu bez nutnosti znalosti programovacích jazyků prostřednictvím grafických 3D nástrojů jako je Blender či Coppercube 3D. U obou možností budou detailně zanalyzovány možnosti, které při tvorbě grafiky uživatelům nabízejí a následně budou vzájemně porovnány na základě několika stanovených kritérií.

V praktické části bude vytvořena sada několika praktických příkladů v podobě interaktivních 3D objektů, vytvořených za pomoci obou metod, kterými se bude práce zabývat.

1.2 Metody

V úvodu práce popíšu co je to WebGL, k čemu slouží a jaká je jeho aktuální pozici na trhu a také zmíním projekty, které tuto technologii momentálně využívají.

Krátce představím knihovnu Three.js a prostřednictvím základních příkazů a metod nastíním možnosti vytváření interaktivní 3D grafiky pomocí Javascriptu. Obdobným způsobem se zaměřím na vytváření interaktivního 3D obsahu bez znalosti programovacích jazyků, tedy v grafických programech Coppercube 3D a Blender. Obě metody následně vzájemně porovnám.

V praktické části bakalářské práce vytvořím za pomoci obou metod několik interaktivních 3D objektů, které pomocí WebGL zobrazím na webové stránce, které pro tuto práci vytvořím.

1.3 Východiska

Internet a s ním spojený webový obsah se neustále vyvíjí a uživatelům je poskytována široká škála zobrazovacích metod, pomocí nichž lze tento obsah prezentovat. Mezi takové metody patří také interaktivní 3D vizualizace, která se v současnosti díky dostatečnému výpočetnímu výkonu, jehož absence v minulých letech bránila větší rozšířenosti této metody, stává stále rozšířenější. Interaktivní 3D grafika nám umožňuje aktivně pracovat se zobrazovaným objektem, kdy ho na rozdíl od statické grafiky můžeme např. otáčet kolem své osy o 360 stupňů.

Interaktivní 3D grafika může být vytvářena prostřednictvím široké škály nástrojů a metod s nimi spojených. Vytvořit a následně zobrazit 3D objekty na webových stránkách je v současné době možné realizovat bez znalostí programovacích jazyků, tato skutečnost je příslibem širšího rozšíření 3D obsahu na webových stránkách. Příkladem takové realizace může být vytvoření 3D modelu v grafickém programu Coppercube 3D a jeho následném exportu do WebGL.

2 Základní pojmy

2.1 OpenGL a OpenGL ES

OpenGL (Open Graphics Library) je programové rozhraní, pomocí nějž můžeme vytvářet 2D a 3D obsah, ať už se jedná o počítačové hry, CAD programy nebo různé interaktivní aplikace. Společně s OpenGL spadá do kategorie programů, sloužících k vytváření multimediálního a interaktivního obsahu například Microsoft DirectX, který můžeme nalézt v zařízeních, spojených s dodávaným softwarem firmy Microsoft. Nejčastěji se tedy jedná o desktopové počítače a notebooky s operačním systémem Microsoft Windows. Rozdíl mezi rozhraním OpenGl a Microsoft DirectX je v tom, že OpenGL je rozhraní multiplatformové, lze ho tedy využít na různých verzích Unixových systémů, včetně Linuxu, IRIXU a také na platformách Microsoft Windows. [1]

Z výše zmíněného OpenGL vychází standart s názvem OpenGL ES (OpenGL for Embedded Systems), kdy se jedná o část rozhraní OpenGL, které slouží k vykreslování 2D a 3D grafiky v počítačových hrách, jejichž akceleraci zajišťuje grafický procesor počítače. OpenGL ES je zaměřen na využití v v malých vestavěných systémech, jako jsou smartphony, tablety nebo herní konzole. Aktuální verze OpenGL ES nese označení OpenGL ES 3.1, která je podporována operačním systémem Windows, Linux a Andorid (od verze 5.0).

2.2 WebGL

WebGL (Web Graphics Library) je technologie, která umožňuje kreslení, zobrazování a interaktivitu s 3D počítačovou grafikou ve webovém prohlížeči. Struktura WebGL vychází z technologie OpenGL ES 2.0., která byla jako celek okrajově popsána v předchozí kapitole této bakalářské práce. WebGL je jakýmsi „rasterizačním motorem“, který vykresluje body, linky a trojúhelníky na základě kódu, který dodáme. [1] K vykreslování grafických prvků ve webových prohlížečích nevyužívá WebGL dalších zásuvných prvků, jedná se tedy o rozhraní určené pro nativní zobrazování

interaktivní webové grafiky. [2]

Zásuvné moduly není navíc potřeba instalovat ani během samotného vytváření grafického obsahu. Rozhraní WebGL je plně integrované se všemi webovými prohlížeči, které podporují GPU akceleraci 3D grafiky a technologii HTML5, tyto podmínky momentálně splňují všechny moderní prohlížeče, včetně několika mobilních. [2] První specifikovaná verze WebGL byla spuštěna v březnu 2011. Nejnovější verze nese označení 2.0. a stále je ve stádiu vývoje a vylepšování.

K vytváření WebGL aplikací je využíván jazyk JavaScript, aplikace ale lze vytvářet také bez znalosti programovacích jazyků, řešením jsou 3D grafické nástroje jako Blender či CopperCube 3D, z nichž je možné hotové aplikace do WebGL vyexportovat.

3 Technologie a funkčnost WebGL

3.1 Základní funkce

Struktura WebGL programů se skládá z kontrolního kódu, který je psaný v hojně využívaném jazyce JavaScript a kódu shaderů, který je psán v jazyce GLSL, svou strukturou podobný jazyku C a C++. WebGL může být využito k modelování fyzikálních modelů, 3D prostoru, znázornění matematických těles a funkcí nebo například k tvorbě již zmíněných počítačových her. Hlavní výhodu WebGL je přenositelnost vytvořeného díla, kdy k jeho prezentaci postačí pouze internetové připojení a vhodný webový prohlížeč, který umožní zobrazení dané aplikace. [1] Nevýhodou WebGL může být dostupnost kódu na webové stránce na které je aplikace nahrána ostatním uživatelům a z toho plynoucí možné zneužití ze strany ostatních návštěvníků a uživatelů dané webové stránky. Tato problematika se týká obecně kódů a aplikací tvořených prostřednictvím JavaScriptu, zamezit čitelnosti kódu lze pomocí tzv. Javascript Obfuscatorů. [1]

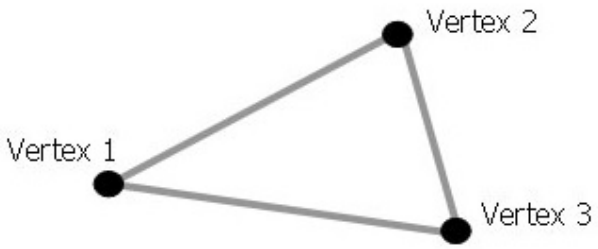
K zobrazování vlastního obsahu využívá WebGL HTML elementu <canvas>, počet těchto prvků v HTML dokumentu může být libovolný. Objekt <canvas> poskytuje metodu getContext(ID), která se využívá k získání kreslícího kontextu příslušného prvku. Parametr (ID) může nabývat hodnoty 2d,v případě jednoduchého dvojrozměrného obsahu, nebo webgl v případě zobrazování obsahu WebGL.

Prvky WebGL lze volně kombinovat s dalšími prvky jazyka HTML nacházejícími se na cílové webové stránce, k usnadnění vytváření aplikací vzniklo pro WebGL mnoho různorodých frameworků, které mohou vývojáři při tvorbě využít. V této práci bude zmíněn framework s názvem Three.js, který kromě standardních prvků, jenž nabízejí také ostatní frameworky, poskytuje navíc širokou škálu ukázkových demo příkladů. Framework Three.js byl využit také v praktické části této práce.

3.2 Geometrie a 3D modely

Stejně jako u ostatních 3D grafických rozhraní, také ve WebGL se tvary a výsledné modely skládají z trojúhelníků (triangles), které jsou dále definovány pomocí bodů v 3D prostoru, tyto body se nazývají vrcholy (vertex). Kromě informací o poloze mohou obsahovat hodnoty jako je barva, textura nebo průhlednost.[3]

Běžně využívané 3D WebGL modely se často skládají z tisíců takových bodů, které jsou definovány v polích JavaScriptu a ukládány do paměti grafické karty pomocí vyrovnávací paměti. Pomocí trojúhelníků můžeme vymodelovat jakýkoliv složitější 3D model, na který je následně pomocí vykreslovacích programů(shaderů) možné aplikovat textury, čímž se model stává z pohledu uživatele realistickým.



Obrázek 1: Trojúhelník (triangle) složený z vrcholů (vertexů)

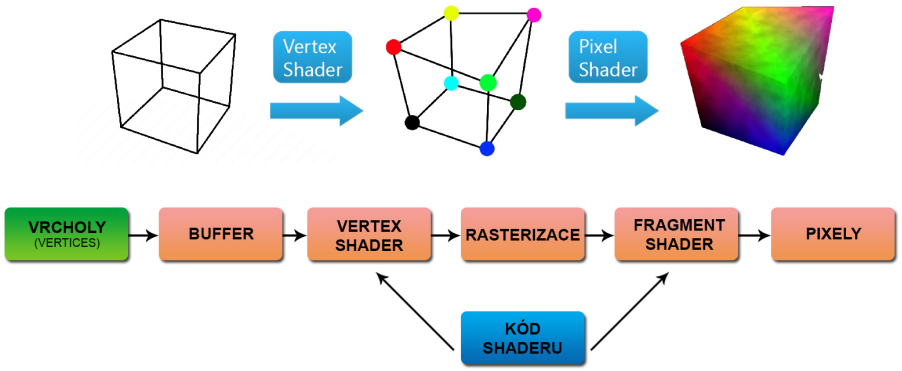
3.3 Grafická pipeline WebGL

Na začátek této podkapitoly je nutné vysvětlit samotný pojem „grafická pipeline“. Grafická pipeline by se dala definovat jako proces postupných kroků, které je potřebné absolvovat k úspěšnému zobrazení 3D obsahu na monitoru počítače.

Grafická pipeline rozhraní OpenGL ES 2.0, ze kterého vychází WebGL obsahuje následující kroky:

- Vertex shader - objekt, složený z polí vrcholů je v prvním kroku předán Vertex Shaderu, který pro každý vrchol vypočítá jeho pozici a přiřazuje mu další jeho vlastnosti.

- Rasterizace objektu - převede scénu na 2D obrázek a odstraní se všechny neviditelné části objektu. Během tohoto kroku se většinou taktéž provádí anti-aliasing.
- Fragment shader - pomocí Fragment shaderu je každému pixelu přiřazena barva, hloubka nebo textura.
- Frame buffer – v závěru procesu je obraz uložen do Frame bufferu grafické karty, odkud je následně odeslán na monitor uživatele. V případě WebGL se jedná o element <canvas>, který je následně zakomponován do frame bufferu webového prohlížeče.



Obrázek 2: Grafická 3D pipeline [3]

Praktické použití této pipeline v rozhraní WebGL je následující: Buffer, vytvořený metodou createBuffer se naplní polem vrcholů, které obsahují nějaký objekt, spojení s daným typem bufferu obstará metoda bindBuffer a naplnění daty zase funkce bufferData. Dále je potřeba definovat, jak bude náš objekt vypadat pomocí vertex a fragment shaderu. Pomocí nám již známým vertex a fragment shaderům nadefinujeme, jak bude objekt vypadat, následné vykreslení proběhne pomocí funkce drawArrays. [4]

3.4 Shadery

Shadery, jejichž koncept je společný pro všechny 3D grafické rozhraní, jsou podprogramy, které umožňují zápis programů, prováděných přímo na grafické kartě a jsou tedy velmi efektivní. Díky své schopnosti dynamicky měnit každý aspekt scény během renderování (např. polohy a barvy vrcholů, textur a obrazových bodů), nabízejí vývojářům velkou flexibilitu. V současné době patří mezi nejdůležitější vertex, pixel (fragment) a geometry shader. V případě WebGL jsou vždy vyžadovány dva shadery – vertex shader a fragment shader, přičemž každý z nich je vlastně samostatná funkce. Oba shadery se následně slučují do jednoho programu, typická WebGL aplikace obsahuje nespočet takových shaderových programů. Úkolem vertex shaderu je spočítat pozice vertexů (vrcholů) na základě těchto pozic poté může proběhnout jejich rasterizace, fragment shader následně přiřadí každému pixelu barvu. [5]

WebGL shadery jsou psány v jazyce GLSL, který byl představen společně s OpenGL 2.0. GLSL kód může být definován buďto jak řetězec v kódu jazyka JavaScript nebo může být zahrnut do HTML pomocí standardních <skript> tagů, které musí mít definovaný buď x-shader/X-vrchol, nebo x-shader/X-fragment, v uvedeném pořadí, jako atribut type. GLSL kód musí být následně zkompilován pomocí JavaScriptu za pomoci vestavěných metod, jako je createShader a compileShader jenž poskytuje WebGL kontext.

4 Kompatibilita a podpora WebGL

4.1 Kompatibilita webových prohlížečů

Technologie WebGL se na trhu webových prohlížečů těší velké podpoře, jedním z důvodů je také fakt, že většina firem, zabývajících se vývojem renomovaných prohlížečů je členem Khoronos Group, která stojí za vývinem standartu technologie WebGL(WebGL Woking Group): Google (Chrome), Mozzila (Firefox), Apple (Safari) a Opera. Oficiální webová stránka WebGL nabízí jednoduchou možnost ověření, zda uživatelův webový prohlížeč technologii WebGL podporuje. [1]

Prohlížeč	**Podpora WebGL**	**Od verze**
Internet Explorer	Ano	11
Microsoft Edge	Ano	12
Mozilla Firefox	Ano	4
Google Chrome	Ano	8
Safari	Ano	5.1
Opera	Ano	12.1

Tabulka 1: Kompatibilita WebGL 1.0 ve webových prohlížečích

Nejnovější verze WebGL, která nese označení 2.0 přináší řadu novinek, jako jsou nové matematické operace nebo výrazné posílení práce s texturami, jelikož WebGL 2.0 stále prochází vývojem a inovacemi, není zde tak výrazná podpora prohlížečů, jako u první verze, i přesto drtivá část prohlížečů již WebGL 2.0 podporuje.

[1] https://get.webgl.org/

Prohlížeč	Podpora WebGL 2.0	Od verze
Internet Explorer	Ne	-
Microsoft Edge	Ne	-
Mozilla Firefox	Ano	52
Google Chrome	Ano	56
Safari	Ne	-
Opera	Ano	44

Tabulka 2: Kompatibilita WebGL 2.0 ve webových prohlížečích

4.2 Kompatibilita mobilních webových prohlížečů

Popularita a využití mobilních zařízení se stále zvyšují a stávají se tak každým rokem mocnějším nástrojem, významnou roli na těchto platformách tak hraje také 3D grafika. Oba nejvyužívanější systémy na těchto zařízeních Android a Apple iOS nativně podporují technologii OpenGL ES 2, tedy základní požadavek pro WebGL. Celkový přehled jednotlivých prohlížečů a jejich podpory WebGL je vyznačen v tabulce níže.

Prohlížeč	Podpora WebGL	Od verze
Safari iOS	Ano	8
Android Browser	Ano	56
Samsung Internet	Ano	4
Internet Explorer mobile	Ano	11
Opera mobile	Ano	12
Firefox for Android	Ano	52
BlackBerry browser	Ano	10
Chrome for Android	Ano	57

Tabulka 3: Kompatibilita WebGL v mobilních webových prohlížečích

5 Interaktivní 3D grafika pomocí jazyku JavaScript s využitím knihovny Three.js

5.1 Představení knihovny Three.js

Vzhledem k tomu, že WebGL je nízkoúrovňová technologie, u které je k vytvoření i primitivní aplikace potřeba provést spoustu úkonů a napsat plno řádků kódu, je k dispozici řada knihoven, usnadňujících vytváření 3D obsahu. Jednou z těchto knihoven je JavaScriptový framework Three.js, který je zaměřen na jednoduchost řešení a poskytuje spoustu hotových demo příkladů včetně připojené dokumentace. Knihovna Three.js taktéž obsahuje předpřipravené komponenty a pomocné metody, čímž se stává vhodnou pro začátečníky v tomto oboru.[6]

Knihovnu je možné využít ve dvou dostupných verzích:

- Three.min.js - jedná se o jakousi ořezanou verzi knihovny, vhodnou k využití při tvorbě jednodušších projektů nebo webových stránek, obsahuje přibližně čtvrtinu obsahu kompletní knihovny Three.js
- Three.js - klasická verze knihovny, jejíž komponenty budou využity také v rámci této práce

Three.js samozřejmě není jedinou osamocenou knihovnou, která ulehčuje práci s WebGL. Dalšími v pořadí jsou například Greensock, PlayCanvas a Pixi.js, které všechny nabízejí výhody WebGL v jednodušším provedení. Pro již zmíněnou hojnou zásobu demo příkladů včetně dokumentace jsem se ale rozhodl v této práci využít služeb právě knihovny Three.js.

5.2 Funkcionalita knihovny Three.js

- Efekty – anaglyf, paralaxní bariéra a „cross-eyed“
- Scény – přidávání a odebírání předmětů za běhu, mlha
- Kamery – perspektivní a ortografická

- Animace – armatury, kinematiky, inverzní kinematiky, morfing a klíčový snímek
- Světla – okolní a bodová světla
- Materiály – Lambert, Phong, hladké stínování, textury a další
- Shadery – plný přístup k OpenGL Shading jazyku (GLSL)
- Objekty – meshe, částice, sprajty, linky, stuhy
- Geometrie – roviny, krychle, obloun koule, 3D text
- Načítání dat – binární, obrazové, JSON a scéna
- Utility – časové a matematické funkce
- Export a import – nástroje pro vytváření Three.js kompatibilních JSON souborů
- Podpora – dokumentace, veřejné fórum
- Ukázkové příklady – více než 150 ukázek kódů, navíc modely, textury, zvuky a další podpůrné soubory

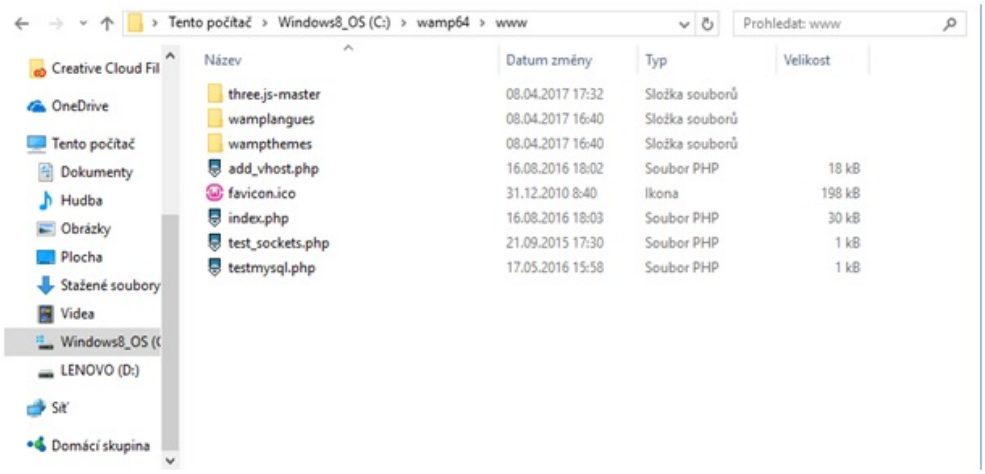
5.3 Vytvoření lokálního webového serveru a stažení knihovny Three.js

Programovací jazyk JavaScript obsahuje bezpečnostní funkci politiky stejného původu (same-origin policy), která přikazuje, že skripty v jedné doméně nesmí mít přístup k prostředkům jiné domény, ani nemohou ovlivňovat skripty a data v jiné doméně.[7] V případě knihovny Three.js představuje tato podmínka menší překážku z toho důvodu, že potřebuje načítat geometrie, textury a další soubory.

K vyřešení tohoto problému budeme potřebovat lokální http server, který zajistí stejný původ všech souborů, prosté zobrazení souboru index.html by jinak nebylo

funkční. Tuto drobnou překážku odstraníme vytvořením lokálního webového serveru pomocí balíčku WampServer[2], který obsahuje vše potřebné.

Po klasické instalaci balíčku WampServer se na námi zvoleném disku a základě typu systému vytvoří složka s názvem wamp32 nebo wamp64. V této složce nalezneme pod složku www, do které uložíme složku s knihovnou Three.js, kterou lze stáhnout na oficiální webové stránce frameworku.[3]

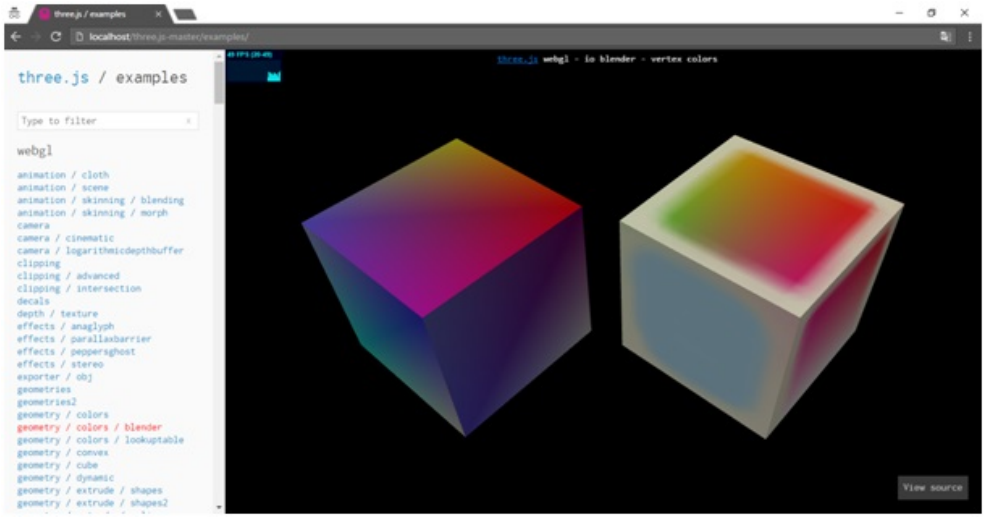


Obrázek 3: webový server se složkou knihovny Three.js

Funkčnost lokálního webového serveru lze ověřit zobrazením ukázkových příkladů, které knihovna Three.js nabízí. Tyto příklady nalezneme v pod složce knihovny s označením examples. Abychom mohli praktické příklady zobrazit ve webovém prohlížeči, je nutné do adresního řádku prohlížeče zadat přesnou cestu k této složce.

[2]http://www.wampserver.com/en/

[3]https://github.com/mrdoob/three.js/archive/master.zip



Obrázek 4: Zobrazení ukázkového příkladu knihovny Three.js na lokálním webovém serveru

5.4 Základní scéna, kamera a render

Jako praktický úvod do světa WebGL a 3D grafiky vytvořené pomocí JavaScriptu s využitím frameworku Three.js bude vytvořena základní scéna, obsahující kameru, světla a jednoduchý geometrický tvar, jemuž bude následně přiřazeno interaktivní ovládání. Celá tato scéna bude nakonec vyrenderována a výsledek celého snažení vyobrazen jako obsah webové stránky.

Prvním krokem bude vytvoření HTML souboru, pomocí něhož zobrazíme výsledný projekt ve webovém prohlížeči. První část kódu je věnována CSS stylu stránky a canvasu, ve kterém bude výsledná animace zobrazena, samotné načtení externí knihovny Three.js je zajištěno v části kódu s značením <body>, kde elementu <script> přiřadíme cestu k souboru three.js, který obsahuje samostatnou knihovnu, druhým skriptem bude priklad.js, v němž budeme obsah pro WebGL vytvářet. Pokud takto připravený soubor nyní zobrazíme, výsledkem, bude pouze pozadí canvasu. Celý kód má pak následující podobu:

```
1  <!DOCTYPE html>
2  <head>
3    <meta charset="UTF-8" />
4    <title>Moje prvni WebGL</title>
5    <style>
6      body { margin: 0; }
7      canvas { width: 100%; height: 100% }
8    </style>
9    <script src="three.js"></script>
10 </head>
11 <body>
12   <script src="priklad.js"></script>
13 </body>
14 </html>
```

Příklad 1: Základní HTML kostra ukázkového projektu

K zobrazení obsahu vytvořeného pomocí knihovny Three.js je potřeba pomocí kódu nadefinovat několik objektů, ze kterých se bude výsledná scéna skládat. Těmito objekty jsou samotná, scéna, kamera, osvětlení a render. Z nabízených možností kamer, které Three.js poskytuje, byl u tohoto příkladu zvolen typ PerspectiveCamera, tedy kamera z perspektivy. U kamery je nutné k správnému chování nastavit několik základních atributů. První atribut, kterému byla přiřazena hodnota 75, představuje hodnotu zorného pole. Následující hodnota určuje poměr stran, k ideálnímu zobrazení a zachování vhodného poměru je nejčastější hodnotou šířka elementu vydělená jeho výškou. Poslední dvě hodnoty slouží k nastavení bližší a vzdálenější ořezové roviny, které určují to, že bližší nebo vzdálenější objekty od kamery v nastavených mezích nebudou zobrazeny. Tyto hodnoty najdou své využití především ve větších projektech, kde mohou vylepšit jejich výkon během vykreslování. [8]

Osvětlení zde zastupuje světlo typu AmbientLight, které osvětluje rovnoměrně

všechny objekty na scéně, a stačí u něj pouze definovat barvu osvětlení. Dalšími typu osvětlení, které lze v knihovně využít jsou tyto:

- DirectionalLight – světlo, které se míří v určitém směru a chová se tak, jako by bylo nekonečně daleko. Obvyklým případem použití je simulace denního světla.
- HemisphereLight – zdroj světla umístěný přímo nad scénou, taktéž nepodporuje stíny.
- PointLight – světlo, které se vyzařuje paprsky z jednoho místa do všech směrech, jako např. žárovka.
- SpotLight – paprsek světla, vyzařující z určitého bodu a postupně se rozšiřující směrem od tohoto bodu.

Po nadefinování scény, kamery a osvětlení přichází na řadu již zmíněný render. Instanci renderer byla přiřazena hodnota typu WebGLRenderer, která jak už z názvu vyplývá, zajišťuje zobrazení pomocí WebGL. Kromě vytvoření této samotné instance je potřeba nastavit velikost, ve které chceme naší aplikaci renderovat. Vhodným řešením je použít šířku a výšku oblasti, kterou chceme vyplnit naší aplikací, v tomto případě se jedná o šířku a výšku webového prohlížeče. Jako poslední přiřadíme rendereru domElement, který renderer využívá k vyobrazení scény.

```
1   scene = new THREE.Scene();
2   camera = new THREE.PerspectiveCamera
3   (75, window.innerWidth / window.innerHeight, 0.1, 1000);
4   camera.position.set( 20, 50, 40 );
5
6     var light = new THREE.AmbientLight( 0xffffff );
7     scene.add( light );
8
9   renderer = new THREE.WebGLRenderer();
10  renderer.setSize( window.innerWidth, window.innerHeight );
11  document.body.appendChild( renderer.domElement );
12
13  renderer();
```

Příklad 2: Nadefinování scény, kamery, světel a renderu pomocí knihovny Three.js

5.5 Geometrické útvary

V takto připravené scéně je samozřejmě potřeba zobrazit nějaký obsah, v tomto případě se bude jednat o geometrický útvar v podobě krychle. K vytvoření krychle, nebo nepravidelného čtyřúhelníku nabízí Three.js objekt typu <BoxGeometry>, v jehož parametrech lze nastavit výšku, šířku a hloubku výsledného tvaru. Takovýchto objektů, sloužících k vytvoření požadovaného tvaru je v knihovně Three.js několik, mezi nejdůležitější, nabízející základní 3D útvary patří tyto:

- <BoxGeometry> - geometrická třída, sloužící k vytváření krychlí a nepravidelných čtyřúhelníků, jako základní parametry se v tomto případě nastavuje šířka,výška a hloubka

  ```
  new THREE.BoxGeometry(10,10,10);
  ```

- <SphereGeometry> - třída generující tvar koule

  - Základní parametry: poloměr, šířka segmentů, výška segmentů

```
new THREE.SphereGeometry( 5, 32, 32 );
```

- <CylinderGeometry> - válcovité tvary
  - Základní parametry: horní poloměr, spodní poloměr, výška, počet ploch tvořící plášť válce

```
new THREE.CylinderGeometry( 5, 5, 20, 32 );
```

- <TorusGeometry> - torus[4],
  - Základní parametry: poloměr, průměr trubky, úhlové segmenty, počet trubicovitých segmentů

```
var geometry = new THREE.TorusGeometry( 10, 3, 16, 100 );
```

- <TextGeometry> - třída generující text jako jako geometrický útvary
  - Parametry, definovatelné v konstruktoru textu: typ fontu, velikost textu, výška, počet křivkových bodů, nastavení úkosu písma (viditelnost, tloušťka, velikost, počet segmentů)

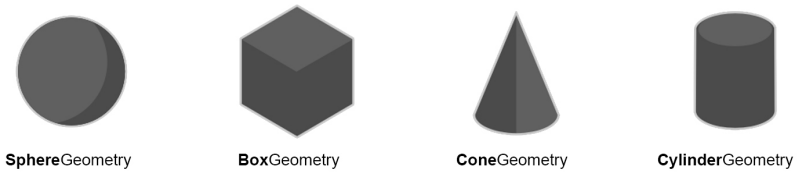
```
new THREE.TextGeometry("Text", parametry textu));
```

V ukázkovém případě bude využit typ objektu <BoxGeometry>,u kterého je potřeba definovat již zmíněné parametry šířky, výšky a hloubky, kompletně nadefinovaný objekt pak může vypadat například takto:

```
1 var geometry = new THREE.BoxGeometry( 30, 30, 30 );
```

Příklad 3: Objekt typu BoxGeometry s nastavenými parametry

[4]Torus (též anuloid) je rotační plocha, která vznikne otáčením kružnice kolem osy, která leží ve stejné rovině a nemá s ní společné body. Tento tvar má například vzdušnice (duše) pneumatiky nebo nafukovací kruh.



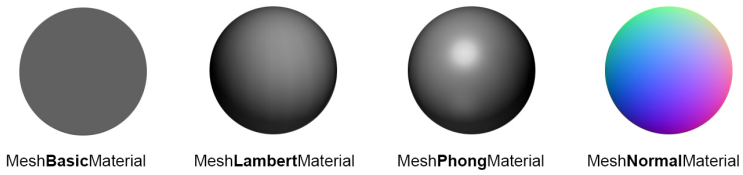
Obrázek 5: Základní 3D geometrické útvary v knihovně Three.js

5.6 Materiály

Zvolený tvar je potřeba pokrýt nějakým materiálem, díky čemuž můžeme objektu určit jeho barvu, texturu, a to jak se na něj odráží světlo. I zde knihovna nabízí několik možností a typů materiálů, ze kterých lze volit.

- <MeshBasicMaterial> – nejzákladnější typ, u kterého můžeme kromě již zmíněných vlastností jako parametr nastavit barevnou hodnotu nebo průhlednost, objekt lze taktéž zobrazit pomocí drátového modelu(wireframu).
  - Další vlastnosti: stínování(Shading), barva vrcholu(vertexColors), mlha (fog), tloušťka linky wireframu(Wireframelinewidth), zakončení linky wireframu (Wireframelinecap), spoje linek wireframu(wireframeLinejoin)
- <MeshNormalMaterial> – materiál podobný výše popsanému BasicMaterialu, tento druh materiálu navíc umožňuje obarvit každou část sítě objektu (meshe) jiným barevným odstínem.
  - Další vlastnosti: drátový model(Wireframe), stínování(Shading), tloušťka linky wireframu(Wireframelinewidth)
- THREE.MeshLambertMaterial - tento materiál lze použít k vytváření matných, ne-lesklých povrchů. Jedná se o velmi snadno použitelný materiál, který reaguje na zdroje osvětlení ve scéně.
  - Další vlastnosti: Okolní barva materiálu(ambient), vyzařování(emissive), obalení tělesa osvětlením(wrapAround), rychlost dopadu světla (WrapRGB)

- THREE.MeshPhongMaterial - materiál s podobnými vlastnostmi jako <MeshPhongMaterial>, u něhož navíc přibývá vlastnost odlesků, díky kterým se stává výsledný objekt realističtějším.
  - Další vlastnosti: Okolní barva materiálu(ambient), vyzařování(emissive), zrcadlení(specular), lesk(shininess), efekt kovu(metal), obalení tělesa osvětlením(wrapAround), rychlost dopadu světla (WrapRGB)



Obrázek 6: Základní materiály v knihovně Three.js

Knihovna Three.js poskytuje třídu s názvem THREE.Material, které obsahuje všechny společné vlastnosti, které lze roztřídit do následujících kategorií:

- Základní vlastnosti - díky těmto vlastnostem lze například kontrolovat a nastavovat průhlednost objektu, přiřazovat ID a jméno objektu, patří mezi ty, jenž jsou při práci s materiály nejpoužívanější.
- Vlastnosti prolínání - materiály mají několik obecných vlastností, týkajících se mísení barev (blendingu). Právě tyto vlastnosti, jako například kombinace barvy objektu vzájemně k jeho pozadí, jsou obsaženy v této kategorii.
- Pokročilé vlastnosti - tyto vlastnosti se vztahují k samotné interní funkčnosti WebGL, podrobněji se o těchto vlastnostech lze dočíst v oficiální specifikaci technologie OpenGL.

V ukázkovém projektu přiřadíme již definovanému objektu v podobě krychle materiál typu <MeshBasicMaterial> a jako vlastnost definujeme libovolnou barvu v hexadecimálním kódu.

```
1    var geometry = new THREE.CubeGeometry(30, 30, 30);
2    var material = new THREE.MeshBasicMaterial
3    ({color: 0x00ff00});
4    mesh = new THREE.Mesh(geometry, material );
5    mesh.position.z = -50;
6    scene.add( mesh );
```

Příklad 4: Přiřazení materiálu MeshBasicMaterial na daný objekt

5.7 Textury

Nezbytnou součástí 3D aplikací jsou textury, které pomocí pokrývání plochy realistickými fotografiemi a materiáy dodávají modelům na realističnosti. Knihovna Three.js využívá textury několika různými způsoby, můžeme je využít nejen k definování barev meshe či pokrytí objektu obrázkem, ale také například k definování odlesků a nárazů nebo odrazů objektů. [9]

Three.js nabízí širokou škálu formátů, které se dají využít jako textura, konkrétně se jedná o PNG, JPG, GIF, TGA, DDS, nebo PVR formát.bV případě WebGL jsou textury načítány asynchronně pomocí JavaScriptu, ke každé textuře musí být přiřazena funkce zpětného volání, která zajistí nahrání obrázku, který má daný objekt pokrýt. Textura nemusí být pouze v podobě statického obsahu, na objekty lze pomocí canvas, nebo video elementu aplikovat také dynamické video textury, podporované formáty v tomto případě jsou MP4 a OGG/OGV. Jako texturu můžeme využít téměř jakéhokoliv obrázku, nejlepší výsledek je dosažen, pokud použijeme čtvercový formát, jehož rozměry jsou násobkem čísla dvě, tedy např. 256 x 256px, 512 x 512px nebo 1024 x 1024px. [1]

K nahrání textury v nejběžněji využívaných grafických formátech využijeme funkci THREE.ImageUtils.loadTexture, která po přesném zadání cesty umožňuje nahrání souboru ve formátech JPG, PNG A GIF. Pokud bychom jako texturu chtěli využít formát DDS(DirectDraw Surface format) řešením je funkce THREE.DDSLoader,

v případě formátu PVR (Power VR) se pak jedná o funkci THREE.PVRLoader. Jako poslední možnost lze využít formátu TGA(Targa), nahrání této textury zajišťuje funkce THREE.TGALoader.

Textury lze využít i k jinému účelu, jakým je například tzv. Bump mapping. Jedná se o techniku, kdy je pomocí texturování vytvořena iluze nerovnosti povrchu bez změny jeho geometrie. Iluze nerovnosti povrchu se dosahuje úpravou normály v každém pixelu plochy. Modifikovaná normála pak ovlivní výpočet osvětlení plochy. [10]

V ukázkovém příkladu bude využita funkce THREE.ImageUtils.loadTexture, pomocí které na náš objekt naneseme texturu dřevěné bedny, která nahradí dosud zelenou barvu objektu.

```
1  var material = new THREE.MeshPhongMaterial
2      ({ map: THREE.ImageUtils.loadTexture('box.jpg') });
```

Příklad 5: Nahrání textury pomocí funkce THREE.ImageUtils.loadTexture

5.8 Interaktivita scény

5.8.1 Animace objektů

Díky animacím ožívá jinak statická scéna k životu, můžeme pomocí nich poměrně lehce měnit rotaci objektů, jejich měřítko, pozice a spoustu dalších věcí. I když samotné WebGL nemá předdefinované možnosti animací, díky výkonu a rychlosti tohoto rozhraní máme možnost renderovat vytvořenou grafiku až na 60 snímků za sekundu, v kombinaci s vylepšenou architekturou moderních prohlížečů to znamená možnost animovat objekty plynule bez jakéhokoliv trhání či jiných nežádoucích efektů. [11]

Animace lze v rámci WebGl využít k úpravě vlastností téměř všech objektů na scéně, lze je transformovat, můžeme měnit jejich geometrii a textury, animovat lze také světla a kamery. Objekty se mohou pohybovat, otáčet a měnit nebo následovat předdefinované cesty. Geometrické tvary můžeme ohýbat, převracet, případně je

měnit do zcela jiných tvarů. Textury mohou být přemísťovány, zmenšovány, otáčeny a posouvány a jejich pixely lze modifikovat na každý snímek. Světla mohou blikat, pohybovat se a měnit barvy, kamery lze přemístit a otáčet tak, aby vytvářely filmové efekty. Možnosti jsou v podstatě neomezené. [1]

K animování obsahu WebGL využívá knihovna Three.js standardní HTML5 funkce requestAnimationFrame, pomocí které knihovně přidělíme informaci o tom, jak renderovat scénu v průběhu času a která zajistí, že funkce render() bude volána ve správném intervalu (obvykle přibližně 60krát za sekundu).[11]

Pro přidání jednoduché animace k našemu vytvořenému objektu v podobě barevné krychle je potřeba definovat funkci renderer(), ve které přiřadíme osám X,Y a Z rotaci o hodnotě 0.015, výsledkem je rotující objekt okolo všech os.

```
1 function animate() {
2     mesh.rotation.x += 0.015;
3     mesh.rotation.y += 0.015;
4
5     render();
6     requestAnimationFrame( animate );
7 }
```

Příklad 6: Přiřazení animace v podobě rotace objektu

5.8.2 Ovládání kamery

Three.js nabízí několik několik možností, jak přidat scéně na interaktivitě a probudit jí tak ze statického stavu. Kromě animací objektů lze pomocí knihovny bohatě pracovat také s kamerou a různými způsoby se tak pomocí její ovládání pohybovat po scéně. Nejpoužívanější typy ovládání kamery, které knihovna poskytuje jsou tyto:

- FirstPersonControls – Ovládání kamery z pohledu první osoby, známé například z FPS her. Pohyb po scéně pomocí klávesnice, rozhlížení po scéně je zajištěno myší.

- FlyControls – Jak už z názvu vyplývá, ovládání vychází z efektu vznášení a létání nad scénou, pohyb i rozhlížení zajišťuje klávesnice společně s myší.
- RollControls – Zjednodušené ovládání, vycházející z FlyControls, které umožňuje pohyb po ose Z.
- TrackBallControls – Pohyb kamery po scéně pouze pomocí myši nebo Trackballu, patří mezi nejvyužívanější druh ovládání.
- OrbitControls – Simulace družice na oběžné dráze kolem konkrétní scény, umožňuje pohyb s myší a klávesnicí.

Na našem ukázkovém příkladu bude použito ovládání OrbitControls, které je nejprve potřeba importovat do HTML souboru stejně jako samotnou knihovnu Three.js. Import provedeme tímto příkazem:

```
1 <script src="js/controls/OrbitControls.js"></script>
```

Příklad 7: Import ovládaní OrbitControls

5.8.3 Ovládací prvky

Klávesa / tlačítko myši	**Přiřazená akce**
Šipka nahoru nebo W	Pohyb po scéně vpřed
Šipka dolů nebo S	Pohyb po scéně vzad
Šipka vlevo nebo A	Pohyb po scéně směrem vlevo
Šipka vpravo nebo D	Pohyb po scéně směrem vpravo
R	Pohyb po scéně směrem vzhůru
F	Pohyb po scéně směrem dolů
Q	Zastavení pohybu
Pohyb myší	Rozhlížení po scéně

Tabulka 4: Ovládací prvky FirstPersonControls

Klávesa / tlačítko myši	**Přiřazená akce**
Levé nebo prostřední tlačítko myši	Zahájení pohybu vpřed
Pravé tlačítko myši	Pohyb vzad
Pohyb myší	Rozhlížení po scéně
W	Zahájení pohybu vpřed
S	Pohyb vzad
A	Pohyb po scéně směrem vlevo
D	Pohyb po scéně směrem vpravo
R	Pohyb po scéně směrem vzhůru
F	Pohyb po scéně směrem dolů
G	Rotace směrem vlevo
E	Rotace směre vpravo

Tabulka 5: Ovládací prvky FlyControls

Klávesa / tlačítko myši	**Přiřazená akce**
W nebo levé tlačítko myši nebo šipka nahoru	Zahájení pohybu vpřed
S nebo pravé tlačítko myši neb šipka dolů	Pohyb vzad
A nebo šipka vlevo	Pohyb po scéně směrem vlevo
D nebo šipka vpravo	Pohyb po scéně směrem vpravo
R	Pohyb po scéně směrem vzhůru
F	Pohyb po scéně směrem dolů
Q	Rotace směrem vlevo
E	Rotace směre vpravo

Tabulka 6: Ovládací prvky RollControls

Klávesa / tlačítko myši	**Přiřazená akce**
Levé tlačítko myši	Rotace a rolování kamery po scéně
Kolečko myši	Zoom kamery
Pravé tlačítko myši	Posouvání scény

Tabulka 7: Ovládací prvky TrackBallControls

Klávesa / tlačítko myši	**Přiřazená akce**
Levé tlačítko myši + pohyb	Rotace kamery okolo středu scény
Kolečko myši	Zoom kamery
Pravé tlačítko myši	Posouvání scény

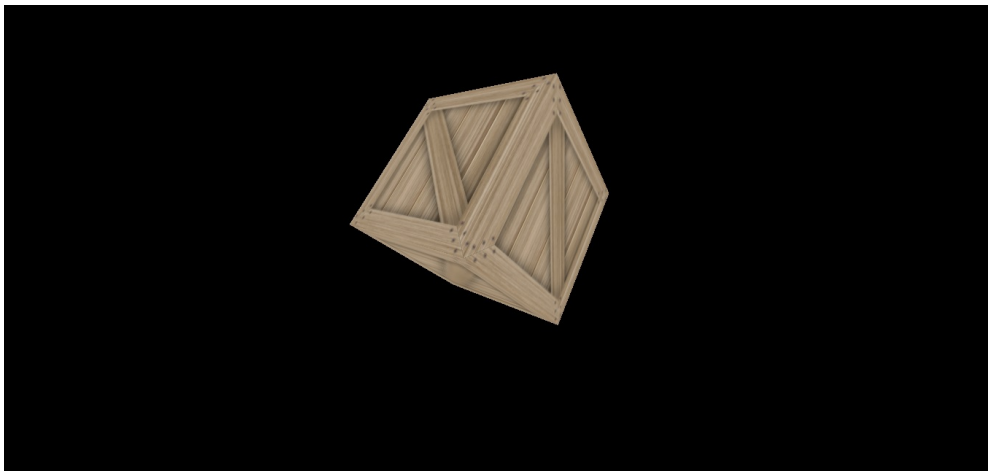
Tabulka 8: Ovládací prvky OrbitControls

Nyní už lze plně využívat využívat funkce tohoto ovládání, samotné ovládání vytvořené scény se provede jednoduše pomocí jednořádkového příkazu, kdy proměnné controls deklarujeme hodnotu new THREE.OrbitControls(), parametrem funkce je pak aktuální kamera scény.

```
1 controls = new THREE.OrbitControls( camera );
```

Příklad 8: Ovládací prvek OrbitControls

Tímto krokem je tvorba ukázkové scény, která byla vytvořena pouze za pomoci knihovny Three.js u konce. Jedná se o vcelku jednoduché řešení, na kterém bylo cílem ukázat základní prvky a jejich vlastnosti v prostředí WebGL a knihovny Three.js. Pro rozsáhlejší projekty je samozřejmě potřeba využít dalších grafických programů a nástrojů, právě tomuto tématu se věnují následující kapitoly.



Obrázek 7: Výsledná scéna vytvořená pomocí knihovny Three.js

6 Vytváření modelů v grafických nástrojích a jejich export do WebGL

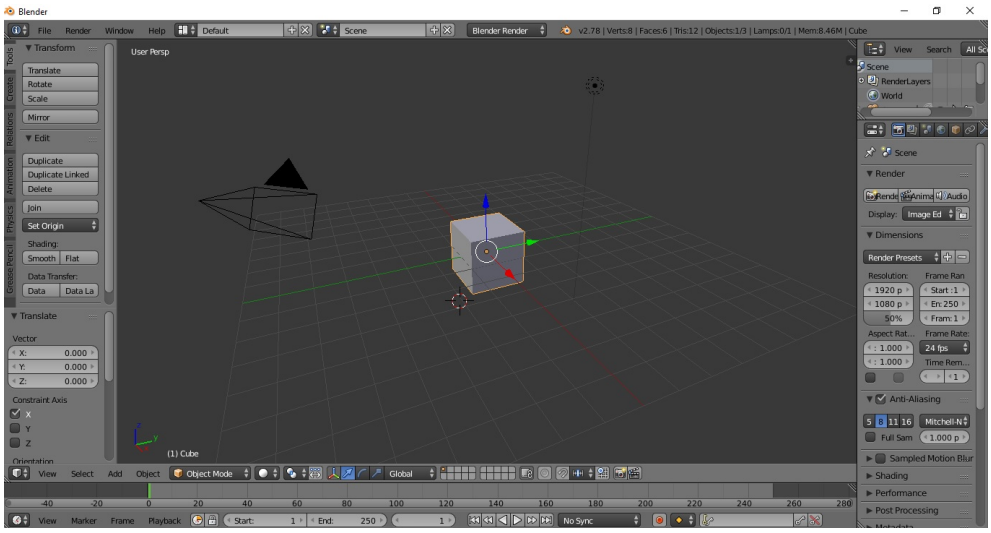
V této práci dosud popsané možnosti vytváření interaktivních 3D objektů pro WebGL využívali pouze možností jazyku JavaScript a podpůrné knihovny Three.js, v rámci možností této metody lze vytvářet spíše jednodušší objekty a scény, skládající se ze základních geometrických útvarů. Svět 3D grafiky ale samozřejmě nabízí mnohem více možností, k vytváření pestřejšího obsahu pro WebGL a složitějších a reálněji vypadajících objektů je potřeba využít dalších programů, ve kterých se dané objekty vymodelují a následně exportují do daného prostředí, kde jim lze přiřadit stejné vlastnosti, jako objektům vytvořeným pouze pomocí knihovny Three.js

V rámci bakalářské práce byly k vytváření propracovanějších 3D modelů využity dva grafické programy, prvním z nich je opensourcový Blender od vývojářů z Blender Foundation, druhým použitým programem je CopperCube od vývojářů z Ambiery, u něhož byla využita čtrnáctidenní trial verze.

Vzhledem k velké komplexnosti a nespočtu nabízených funkcí obou programů, jejichž detailnější popis by pokryl možná několik set stránek se v této části práce budu věnovat pouze základní modelaci 3D objektů a následných možností uložení a exportu do prostředí WebGL.

6.1 Blender

Blender je bezplatný a opensourcový nástroj pro vytváření 3D obsahu, který poskytuje celou škálu nástrojů od modelování, animaci až po renderování celých scén. Jedná se o multiplatformní aplikaci, kromě systému Windows lze tedy Blender spustit také na linuxových systémech nebo na Mac OS. Díky dlouholetému vývoji, široké fanouškovské základně a častým aktualizacím lze v současné době Blender kvalitativně srovnávat s komerčními 3D aplikacemi střední třídy, aplikaci navíc lze doplnit celou řadou rozšíření. [12] V případě WebGL se například jedná o framework s názvem Blend4Web, který slouží k převodu celých scén vytvořených v Blenderu, obsahujících grafiku, animace, zvuky i fyzikální vlastnosti objektů do samostatné webové stránky. Aktuální verze Blenderu nese označení 2.78c a lze jí stáhnout na oficiálním webu aplikace. [5]



Obrázek 8: Základní prostředí programu Blender 2.78c

[5] https://www.blender.org/download/

Platforma	Windows, Macintosh, Linux
Podporované jazyky	Python
Licence	GNU GPL
Velikost na disku	71.7 – 144.1 MiB
Cena programu	Zdarma

Tabulka 9: Specifikace programu Blender 2.78c

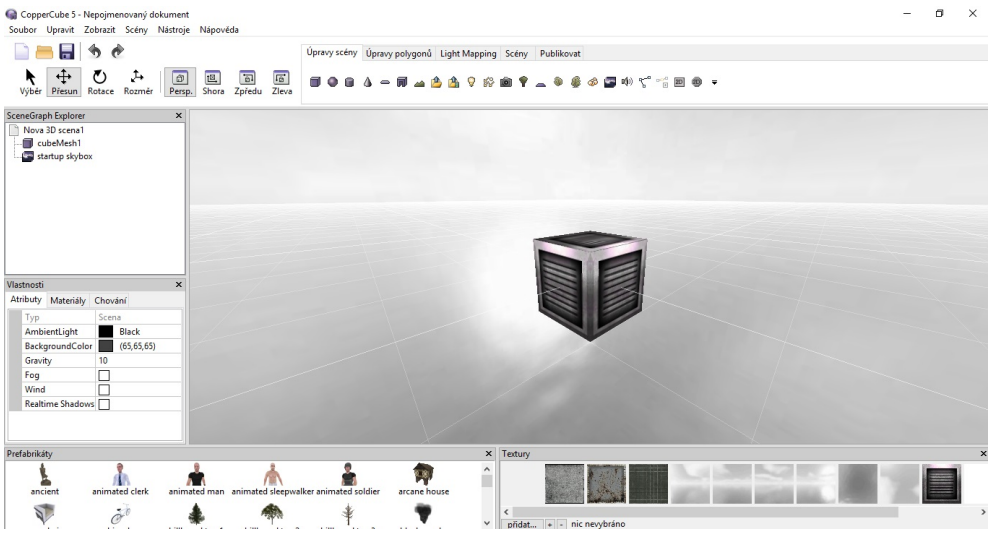
6.2 CooperCube

Dalším programem, umožňujícím vytváření 3D obsahu pro prostředí WebGL je CopperCube. Narozdíl od Blenderu se v tomto případě jedná o komerční software s uzavřeným kódem, program lze bezplatně využít po dobu čtrnácti dnů, po uplynutí této doby je k jeho plnému využití potřeba zaplatit částku stanovenou vývojáři. CopperCube je taktéž multiplatformní software, pomocí kterého lze vytvářet obsah pro Windows, Mac nebo mobilní Android. Za hlavní výhody lze označit jednoduchost ovládání programu, kdy lze bez znalosti programovacích jazyků vytvářet interaktivní 3D obsah s přímým exportem do prostředí WebGL. Coppercube v aktuální verzi 5.7 lze stáhnout na stránce vývojářské firmy Ambiera [6] nebo také v aplikaci Steam.

Platforma	Microsoft Windows, macOS, Android, IOS
Podporované jazyky	Javascript, ActionScript 3
Licence	Proprietární software
Velikost na disku	65 MiB
Cena programu	69€(základní verze)

Tabulka 10: Specifikace programu CopperCube 5.7

[6] http://www.ambiera.com/coppercube/index.html



Obrázek 9: Základní prostředí programu CopperCube 5.7

6.3 Porovnání programů Blender 2.78c a CopperCube 5.7

CopperCube i Blender poskytují dostatek nástrojů a možností pro vytváření 3D obsahu, každý ale trochu jiným způsobem. Při pohledu na základní specifikace obou programů je u druhého jmenovaného velkým plusem GNU GPL licence a s ní spojená možnost program využívat zcela zdarma. Tato skutečnost je vykoupena oproti CopperCube, především u začátečníků poněkud vyšší náročností při vytváření 3D obsahu, díky široké komunitě a dostatku dostupných materiálů a tutoriálů lze ale i ovládání a prostředí Blenderu dostat pod kůži poměrně rychle a zdarma tak získat mocný nástroj na poli 3D grafiky.

CopperCube lze v základní verzi zdarma využívat po dobu čtrnácti dnů, během nichž se uživatel může rozhodnout, zda v užívání programu pokračovat po zaplacení částky 69 euro v případě základní verze nebo 358 euro za profesionální verzi programu, nabízející rozšířené funkce. Právě cenová politika může hrát rozhodující roli při výběru mezi těmito dvěma programy, kdy například na komunitním webu Slant [7], který slouží k vzájemnému porovnávání dvou produktů či programů vítězí v oblíbenosti Blender nad CopperCube právě na základě parametru nulové ceny. V případě CopperCube je vhodné využít případného nákupu pomocí platformy Steam,

[7] https://www.slant.com

kde lze jednotlivé licence zakoupit s výraznými slevami.

Co se týče podpory českého jazyka, CopperCube je nastaven do češtiny ihned po instalaci a prvním spuštěním automaticky, ale ani v případě Blenderu není nutné češtinu dodatečně stahovat. Defaultně ale nastavena není a tak je potřeba program v uživatelském rozhraní do češtiny přepnout. Nicméně využití českého jazyka vzhledem k tomu, že většina materiálů a tutoriálů je v anglickém jazyce slouží spíše k seznámení se s jednotlivými nabídkami a funkcemi programu.

Samotné vytváření vlastního obsahu je poměrně jednoznačnou záležitostí Blenderu, pomocí kterého lze vymodelovat téměř cokoliv, ať už pomocí 2D předlohy, či zcela čistě na základě uživatelovo kreativity. CopperCube naopak lze označit spíše za nástroj, pomocí kterého můžeme již předpřipraveným komponentám jednoduše přiřazovat fyzikální vlastnosti a jejich interaktivitu s následným jednoduchým exportem do různých prostředí, to vše bez jakýchkoliv programovacích znalostí. Vývojáři CopperCube svůj software označují jako "pravděpodobně nejlepší engine pro práci s WebGL", což potvrzuje například možnost přímého exportu do prostředí WebGL pomocí publikování projektu do formátu HTML, bez nutnosti jakýchkoliv doplňků či knihoven.[13] Pokud bychom chtěli stejnou cestou postupovat v případě Blenderu, je nutné externě doinstalovat framework s názvem Blend4Web, který také umožňuje export do formátu HTML. Dalším formát, který lze následně využít při exportu do knihovny Three.js je v obou případech formát COLLADA(.dae) Blender navíc po instalaci frameworku umožňuje export do formátu JSON. Naopak v případě importu souborů do obou programů je jednoznačná výhoda na straně CopperCube, kdy nabízí širokou škálu možností importu v podobě 22 formátů, Blender bez zásuvných modulů nabízí necelou polovinu tohoto čísla.

Celkově lze oba programy označit jako silné nástroje pro vytváření interaktivního 3D obsahu, přičemž nejlepších výsledků lze dosáhnout vzájemným využitím obou z nich. V případě Blenderu je výrazným plusem fakt, že program je zcela zdarma a oponuje bohatou základnou uživatelů, materiálů a frameworků, pomocí nichž lze nabídku funkcí vylepšovat. CopperCube sází na jednoduchost ovládání a postupů při vytváření obsahu, kdy lze bez znalosti programování vytvářet celé hry a vytvo-

řený obsah exportovat do široké škály platforem, od klasicky spustitelné počítačové aplikace až po webový obsah prostředí WebGL bez nutnosti dalších frameworků, omezen je ale v případě vytváření vlastních modelů, které je ve většině případů nutné vytvořit v jiných programech a do CopperCube je následně importovat.

	Blender 2.78c	**CopperCube 5.7**
Velikost na disku	71.7 – 144.1 MiB	65 MiB
Licence	GNU GPL	Proprietární software
Cena	Zdarma	69 euro + 14 dní trial verze
Podpora češtiny	Ano	Ano
Přímý export do WebGL	Ne	Ano
Instalace frameworků	Ano	Ne
Dostupný na Steamu	Ne	Ano
Podporovaný programovací jazyk	Python	Javascript, ActionScript 3
Pravidelné aktualizace programu	Ano	Ano

Tabulka 11: Základní parametry programů Blender 2.78c a CopperCube 5.7

	CopperCube 5.7	**Blender 2.78c**		**CopperCube 5.7**	**Blender 2.78c**
.dae	✓	✓	**.ply**	✗	✓
.stl	✓	✓	**.3ds**	✗	✓
.fbx	✓	✗	**.obj**	✓	✓
.x3d	✓	✗	**.xml**	✗	✓
.irrmesh	✓	✗	**.app**	✓	✗
.swf	✓	✗	**.exe**	✓	✗
.html	✓	✗	**.apk**	✓	✗

Tabulka 12: Formáty exportu programů Blender 2.78c (bez frameworků) a CopperCube 5.7

	CopperCube 5.7	Blender 2.78c		CopperCube 5.7	Blender 2.78c
.dxf	✓	✗	**.oct**	✓	✗
.abc	✗	✓	**.fbx**	✓	✗
.ase	✓	✗	**.irrmesh**	✓	✗
.fdx	✗	✓	**.Lwo**	✓	✗
.3ds	✓	✓	**.x**	✓	✗
.obj	✓	✓	**.ms3d**	✓	✗
.bvh	✗	✓	**.my3D**	✓	✗
.svg	✗	✓	**.mesh**	✓	✗
.b3d	✓	✗	**.lmts**	✓	✗
.x3d	✗	✓	**.bsp**	✓	✗
.blend	✓	✗	**.md2**	✓	✗
.csm	✓	✗	**.raw**	✓	✗
.xml	✓	✗	**.r16**	✓	✗
.dae	✓	✗	**.stl**	✓	✓
.dmf	✓	✗	**.ply**	✓	✓
.cob	✓	✗	**.scn**	✓	✗

Tabulka 13: Formáty importu programů Blender 2.78c (bez frameworků) a CopperCube 5.7

6.4 Převod 2D objektu na 3D objekt v programu Blender 2.78c a jeho export do WebGL pomocí knihovny Three.js

6.4.1 Import svg souboru a tvorba 3D objektu

Postupů, jakými lze v grafických nástrojích vytvářet 3D obsah je nespočet, v případě, kdy potřebujeme vymodelovat objekt na základě hotové jeho 2D předlohy, nabízí Blender poměrně jednoduché řešení.

Základní ovládací prvky programu Blender 2.78c:

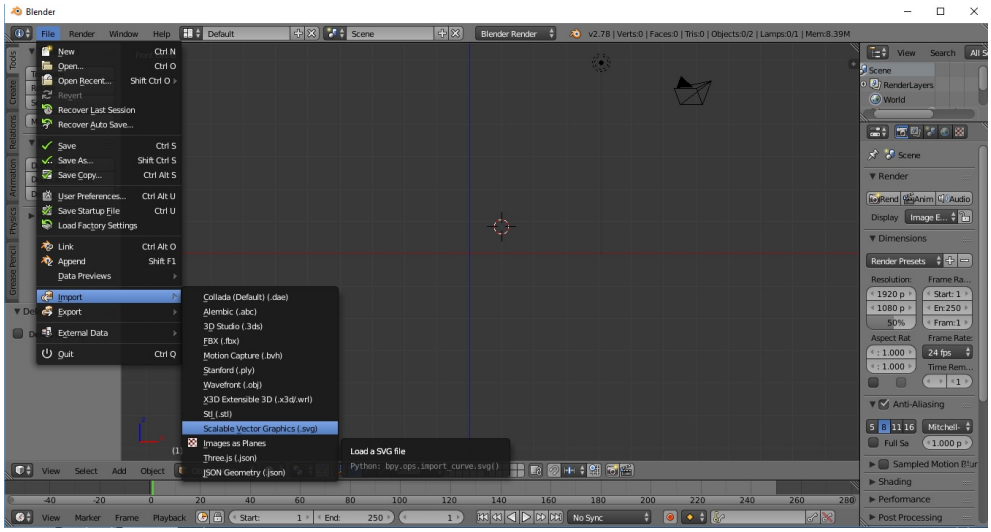
- Stisknuté kolečko myši - rozhlížení po scéně
- Shift + stisknuté kolečko myši - posouvání
- Pravé tlačítko myši - výběr objektů a jejich přesun
- Kolečko myši - přibližování/oddalování scény

Prvně je potřeba zajistit 2D předlohu, v případě této práce se jedná o znak hokejového klubu, jenž je volně využitelný k nekomerčním účelům. Znak byl původně v rastrovém formátu .jpg, pro práci v programu Blender a jeho následném modelování je nutné převést znak do vektorového formátu .svg. Tento převod zajistí vektorizace objektu, tedy převod veškerých pixelů na jednotlivé geometrické tvary, jejichž kvalita zůstává neměnná i při změně velikosti objektu. Vektorizaci lze provést zcela automaticky ve vektorových grafických nástrojích jako je Corel Draw či Adobe Illustrator pomocí metody trasování, převod dokáže zajistit také online aplikace s názvem Vector Magic [8].

[8] http://https://vectormagic.com

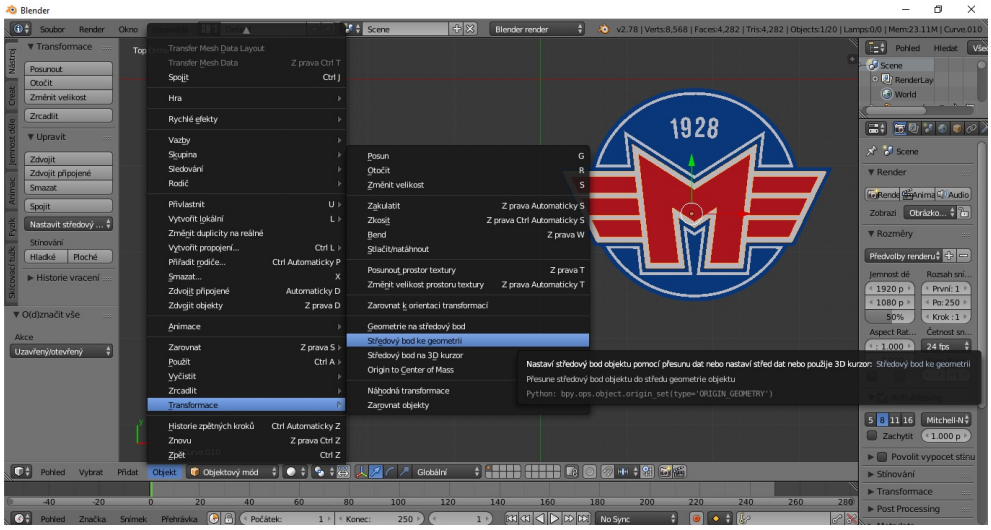
Obrázek 10: Detail rozdílů rastrového a vektorového objektu

Vektorizovaný soubor můžeme nyní importovat do prostředí Blenderu pomocí tomu určené nabídky, kterou najdeme pod položkou File v hlavní horní liště programu. Import probíhá klasickým zadáním přesné cesty k souboru, který chceme do scény vložit. Před vložením samotného souboru je nutné vytvořit novou scénu a následně z ní odstranit veškeré objekty, které se na scéně nacházejí, to provedeme dvojitým stisknutím klávesy A, pomocí níž označíme všechny objekty na scéně, které následně vymažeme kláveso delete, nyní je scéna připravena na import požadovaného souboru.



Obrázek 11: Import svg souboru do prostředí Blenderu

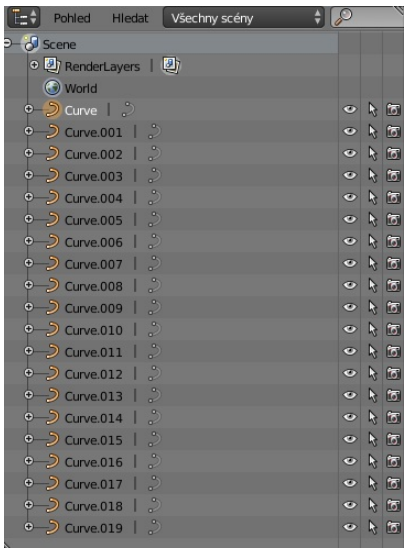
Po úspěšném naimportování se lze klávesou 7 přepnout do vrchního pohledu na scénu a importovaný objekt by tak měl být vidět ve stejné podobě, v jaké byl uložen po vektorizaci. Dalším krokem bude přenesení těžiště našeho objektu na jeho přesný střed, toho docílíme umístěním středového bodu (těžiště) na střed geometrie daného objektu. Tento přesun středového bodu je ukryt v nabídce objekt > transformace > středový bod ke geometrii, nebo vyvoláním klávesové zkratky shift+ctrl+alt+c, v oboou případech je nutné mít aktivně označen celý objekt.



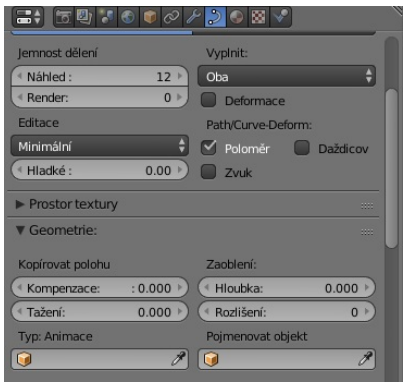
Obrázek 12: Umístění středového bodu ke geometrii objektu

Znak je nyní připraven na převod do 3D objektu, ještě předtím je potřeba změnit jeho polohu tak, ale měl kolmou pozici směrem k ose X. Zde využijeme na označený objekt klávesu R (rotace), po jejímž stisknutí označíme klávesou X osu, podle níž chceme objekt rotovat a zadáním hodnoty 90 na numerické klávesnici objekt obrátíme o 90°, tedy kolmo právě k ose X. Pro vymodelování znaku do 3D objektu využijeme funkci vytažení/vytlačení jednotlivých křivek, ze kterých se daný objekt skládá. Zde platí, že čím pečlivější a detailnější byla metoda trasování objektu, která ho z rastrového objektu převedla právě na vektorový, tím více křivek bude výsledný objekt obsahovat. [12] V prostředí Blenderu vidíme seznam všech křivek, ze kterých se objekt skládá v horní pravé části v okně s hierarchií aktuální scény. Označením jakékoliv z těchto křivek se nám otevře možnost měnit její data a vlastnosti v položce

Data objektu, která se nachází právě pod dialogovým oknem se seznamem křivek.

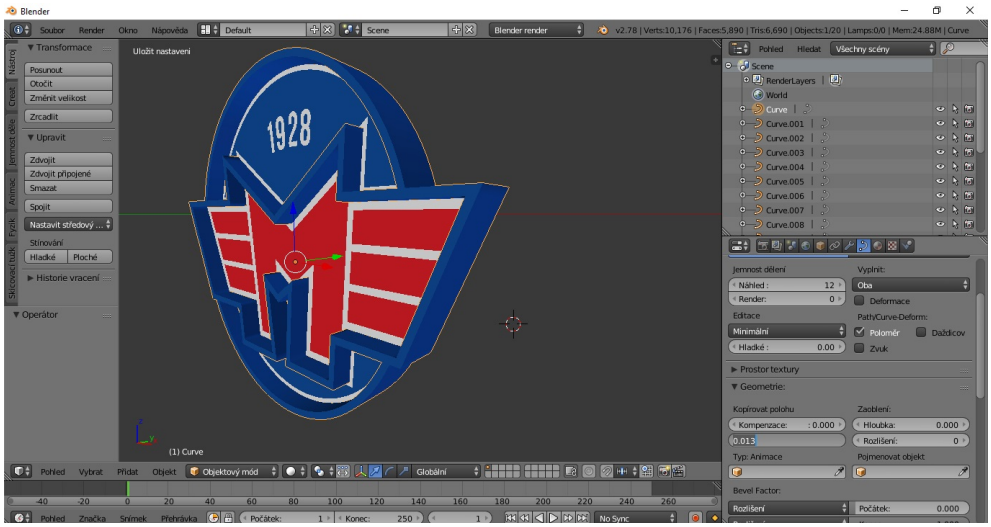


Obrázek 13: Hierarchie scény

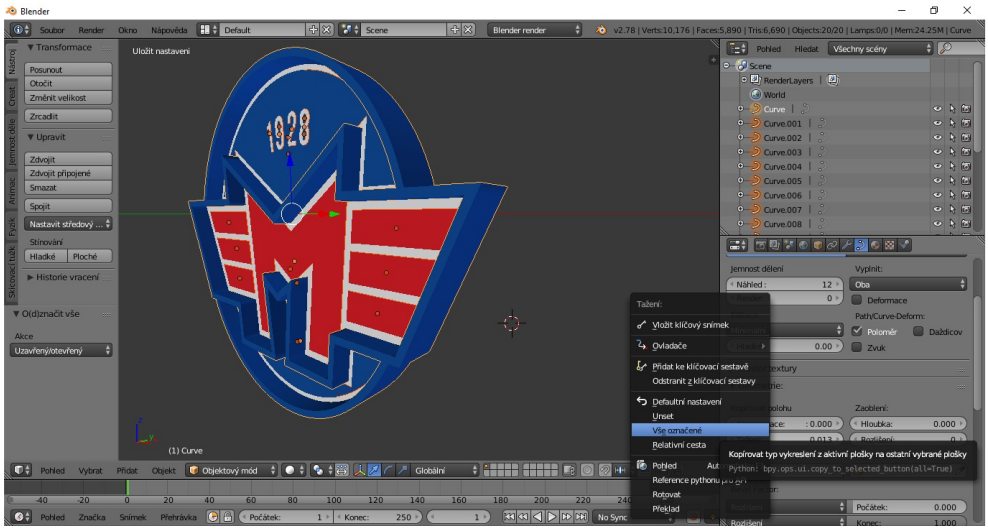


Obrázek 14: Data objektu

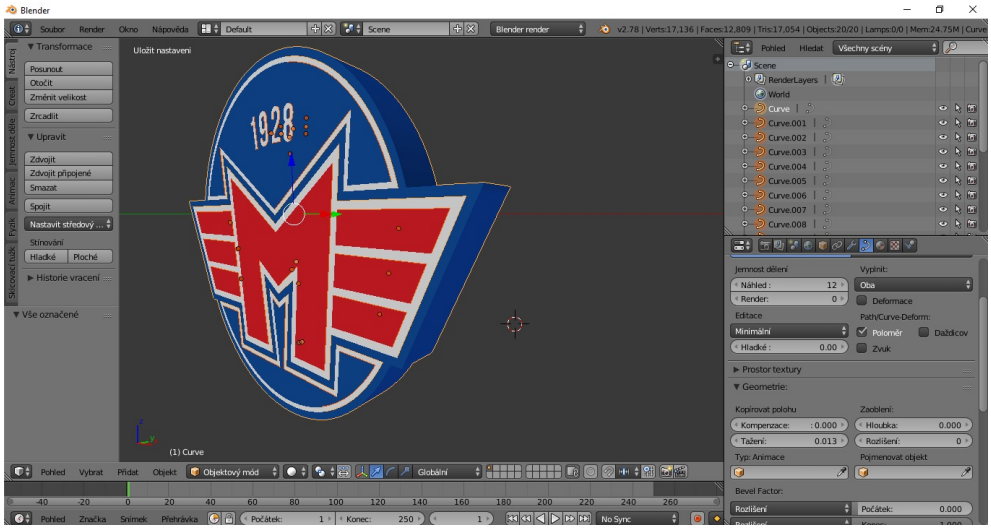
V této nabídce se pod panelem Geometrie nachází posuvník Tažení (extrude), změnou hodnoty z počáteční nulové na hodnoty plusové dochází k vytlačení označené křivky směrem ven a vzniká tak kýžený 3D efekt objektu. Změnou hodnoty se mění vytlačení pouze u aktuální křivky, pro aplikaci aktuální hodnoty na všechny křivky objektu je nejprve potřeba je značit klávesou A a následně pomocí pravého tlačítka myši u posuvníku tažení zvolit možnost Vše označené, díky čemuž se aktuální hodnota vytlačení aplikuje na všechny křivky. Změnou hodnot vytažení jednotlivých lze u daného objektu zvýraznit některé detaily, v případě našeho znaku bylo zvýrazněno červené písmeno M společně s bílou obrysovou linkou, stejná hodnota byla aplikována na letopočet ve znaku. Výsledkem je efekt vystupujícího písmene z modrého pozadí.



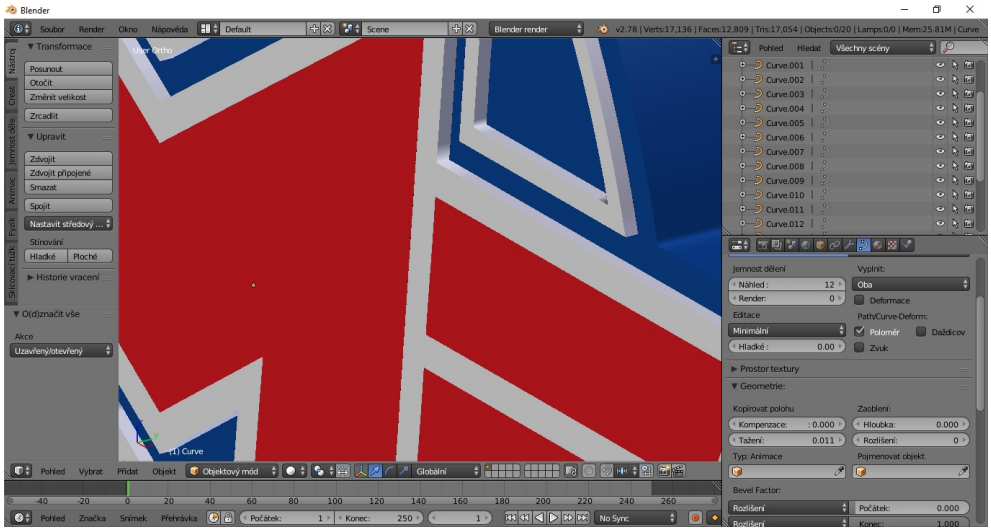
Obrázek 15: Vytlačení aktuálně zvolené křivky



Obrázek 16: Aplikace hodnoty na všechny křivky objektu



Obrázek 17: Objekt po aplikaci hodnot



Obrázek 18: Detail objektu se zvýrazněnými hranami

6.4.2 Export souboru do formátu JSON a import do prostředí WebGL

Knihovna Three.js, které už byla věnována první část práce umožňuje import modelů, vytvořených v externích grafických programech v několika různých formátech, přičemž jsou i po importu zachovány geometrické i vzhledové vlastnosti meshe. [1] Pro naše potřeby nahrání vymodelovaného objektu v programu Blender do prostředí WebGL pomocí knihovny Three.js lze využít tyto formáty:

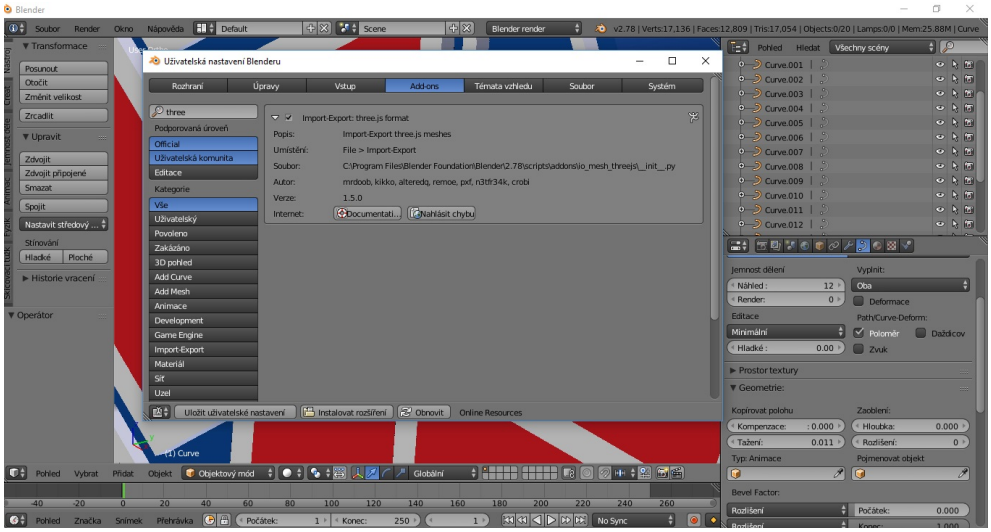
- JSON - vlastní formát knihovny Three.js, pomocí kterého lze definovat geometrii objektů, ale i celé scény.
- OBJ - formát souborů používaný pro trojrozměrný objekt, který obsahuje 3D souřadnice (polygon čar a bodů), textur a další informace o objektu.
- Collada - formát pro ukládání 3D objektů a animací. Je založena na otevřeném XML schématu, tj. můžeme ji snadno přečíst, vytvářet a editovat v libovolném textovém editoru. Obvykle používá koncovku .dae.

Pro import našeho 3D modelu do prostředí WebGL bude využita možnost uložit soubor do formátu JSON, tuto funkci ale samotný program Blender nenabízí a je nutné doinstalovat rozšíření s názvem Blender-JSON-Exporter. [9]. Instalace rozšíření probíhá v následujících krocích:

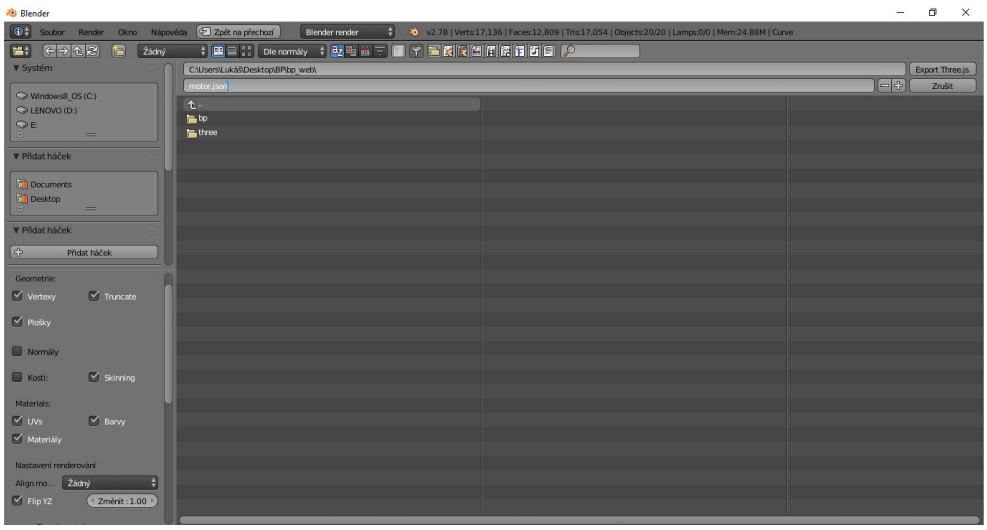
1. Stažení pluginu Blender-JSON-Exporter
2. Zkopírování celé složky s rozšířením do složky addons v kořenové složce programu
3. Aktivace pluginu v uživatelském rozhraní Blenderu, pod panelem Add-ons

Nyní už nic nebrání exportu souboru do požadovaného formátu, ten provedeme v nabídce Soubor > Exportovat > Three.js(.json), kdy ponecháme všechny pole zaškrtnuté tak, jak jsou a uložíme soubor do libovolné složky.

[9] https://github.com/nathanfaucett/Blender-JSON-Exporter



Obrázek 19: Aktivace pluginu Blender-JSON-Exporter



Obrázek 20: Export objektu do formátu JSON

Obsah výsledného souboru s koncovkou .json lze zobrazit v jakémkoliv textovém editoru, po bližším prozkoumání lze vypozorovat, jaká data v sobě ukrývá. Jsou to data obsahující veškeré informace o 3D objektu, nesoucí informace o jednotlivých křivkách, tvarech a jejich obarvení. Pokud bychom chtěli exportovat nejen samotný objekt, ale také kameru a osvětlení, také tyto údaje by se v tomto souboru nacházely. [1]

```
1       "DbgColor": 15597568,
2       "DbgIndex": 1,
3       "DbgName": "SVGMat.001",
4       "blending": "NormalBlending",
5       "colorAmbient": [0.0036765073891729116,
6                   0.07421357184648514, 0.3613067865371704],
7       "colorDiffuse": [0.0036765073891729116,
8                   0.07421357184648514, 0.3613067865371704],
9       "colorEmissive": [0.0, 0.0, 0.0],
10      "colorSpecular": [0.5, 0.5, 0.5],
11      "depthTest": true,
12      "depthWrite": true,
13      "shading": "Lambert",
14      "specularCoef": 50,
15      "transparency": 1.0,
16      "transparent": false,
17      "vertexColors": false
```

Příklad 9: Ukázka části kódu souboru typu JSON

K nahrání 3D modelu do prostředí WebGL využijeme interní nahrávač knihovny Three.js souborů typu JSON, který se nazývá JSONLoader. Na nahrání modelů není nic složitého, loaderu pouze přiřadíme cestu k danému souboru ve formátu JSON. Nahraný soubor je potřeba přidat na scénu, která obsahuje kameru, osvětlení

a rendrer, podrobnější příprava této scény byla popsána v první části bakalářské práce. Přidání objektu do scény zajistí funkce pridejModel(), v níž můžeme navíc určit počáteční pozice X a Y objektu na scéně.

```
1  var loader = new THREE.JSONLoader();
2  loader.load( "models/motor.json", pridejModel );
3
4  function pridejModel( geometry, materials ) {
5    var material = new THREE.MeshFaceMaterial(materials);
6    model = new THREE.Mesh( geometry, material );
7    model.position.x = -0.75;
8    model.position.y = -1.2;
9    scene.add( model );
10 }
```

Příklad 10: Nahrání 3D objektu a jeho přidání na scénu

K interaktivnímu prohlížení objektu ze všech stran využijeme ovládání kamery typu OrbitControls, která umožňuje objektem natáčet okolo své osy, pohybovat s ním a přibližovat či oddalovat ho. K nasvícení objektu je využito dvou bodových světel, které nasvěcují objekt ze dvou směrů. Do ukázkového příkladu bylo také vložen obrázek na pozadí, tuto funkci zajišťuje textureLoader, kterému opět přiřadíme přesnou cestu k souboru, který má plnit roli pozadí scény.

```
1  var control = new THREE.OrbitControls(camera,canv);
```

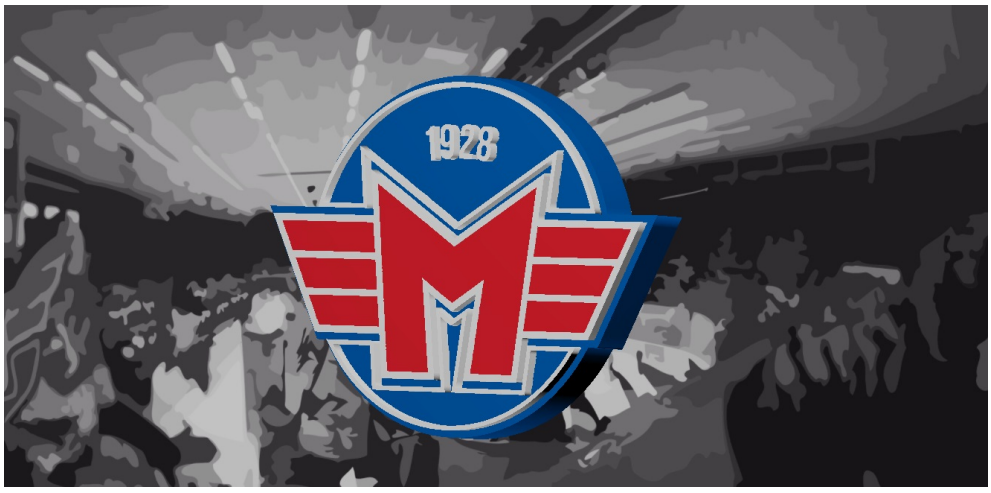
Příklad 11: Interaktivní ovládání objektu pomocí OrbitControls

```
1  var light = new THREE.PointLight( 0xFFFFFF );
2   light.position.set( 5,5, 5 );
3   scene.add( light );
4
5   var light = new THREE.PointLight( 0xFFFFFF );
6   light.position.set( -5,-5, -5 );
7   scene.add( light );
```

Příklad 12: Osvětlení scény bodovými světly

```
1      var textureLoader = new THREE.TextureLoader();
2       var pozadi = textureLoader.load( "textures/mtr.jpg" );
3          scene = new THREE.Scene();
4          scene.background = pozadi;
```

Příklad 13: Soubor typu .jpg jako pozadí scény



Obrázek 21: Interaktivní 3D objekt zobrazený prostřednictvím WebGL

6.5 Interaktivní 3D objekt v programu CopperCube a jeho přímý export do WebGL

Grafický program CopperCube jako svou hlavní výhodu udává jednoduchost ovládání a možnost vytvářet celé scény a hry bez znalosti jakýchkoliv programovacích jazyků. Jako ukázkový příklad v této části práce byl tedy zvolen postup tvorby v programu CopperCube pouze za pomoci nabízených nástrojů bez využití jakéhokoliv programování a výsledný přímý export projektu do prostředí WebGL, jenž CopperCube nabízí jako možnost publikování ve formátu HTML. [13]Využita tedy nebude ani knihovna Three.js, jako tomu bylo v případě modelování objektu v programu Blender.

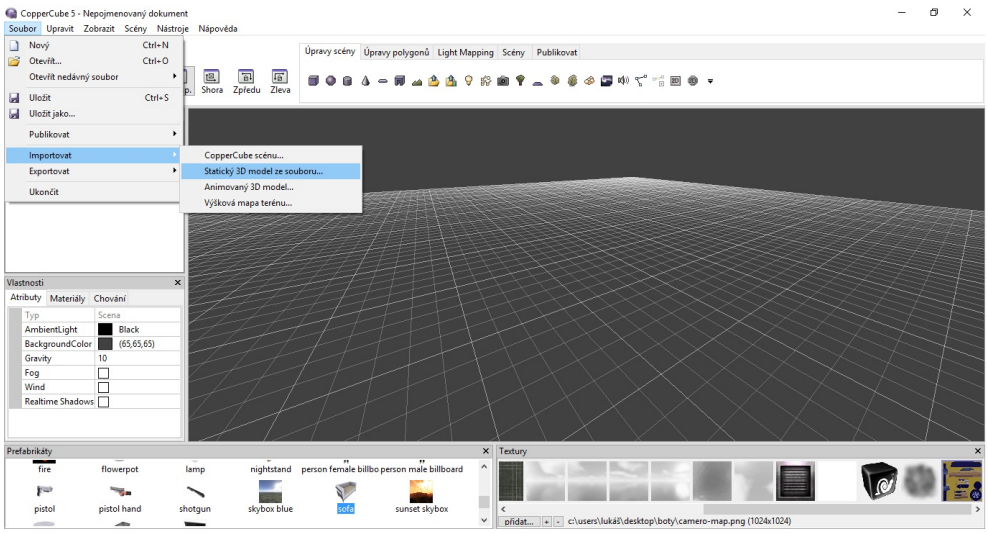
Základní ovládací prvky programu CopperCube 5.7.:

- Levé tlačítko myši - rozhlížení po scéně
- Stisknuté kolečko myši - posouvání po scéně
- Kolečko myši - přibližování/oddalování scény

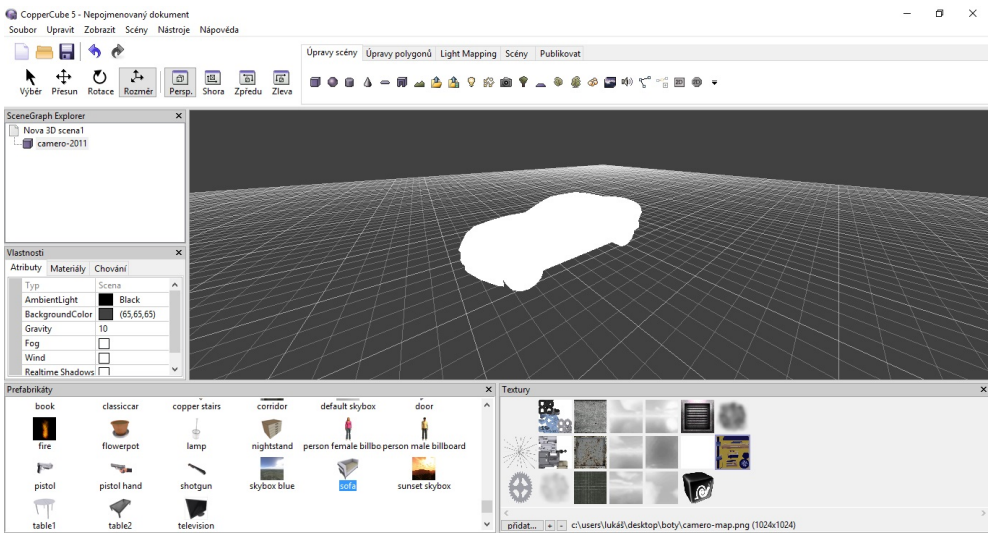
Na rozdíl od Blenderu nabízí CopperCube pouze omezené možnosti při vytváření vlastního 3D obsahu, pro ukázkový příklad, kterým bude jednoduchý konfigurátor 3D objektu, tak byl využit již hotový model [10],spadající pod licenci BY-NC-SA, která umožňuje po uvedení původu nekomerční využití včetně změny atributů daného modelu.

Import tohoto modelu do prostředí CopperCube provedeme přes nabídku Soubor > Importovat > Statický 3D model ze souboru. V případě, že se soubor nahrál pouze jako bílý objekt bez dodané textury, je nutné texturu dostatečně na model načíst. Samotnou texturu nejprve přidáme do seznamu již připravených textur, pomocí nabídky Přidat v dialogové nabídce s texturami. Po nahrání se námi zvolená textura objeví v seznamu textur, na označený objekt jí aplikujeme dvojitým kliknutím levým tlačítkem myši.

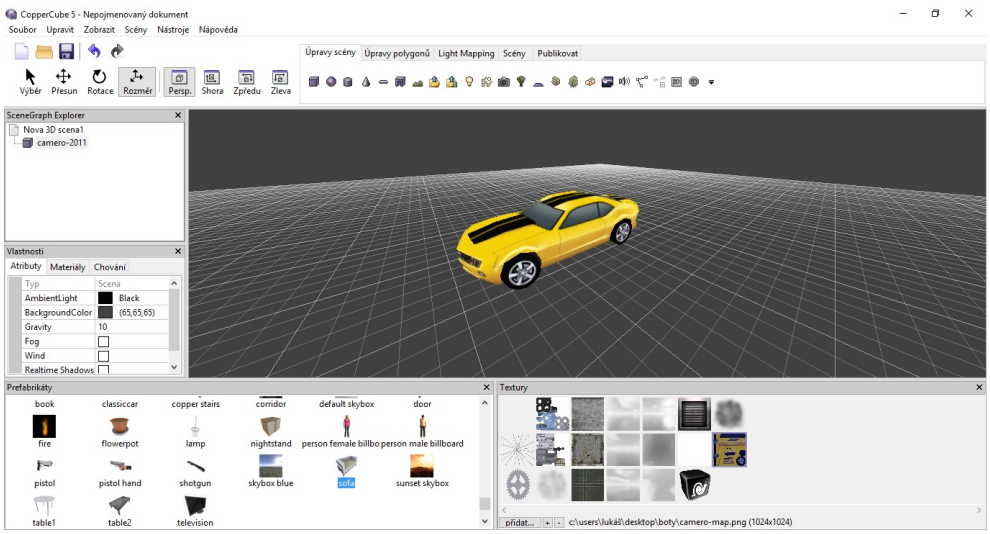
[10]https://https://clara.io/view/2aafff64-2305-4d66-98ff-ab51cb51a3b9



Obrázek 22: Import 3D modelu do prostředí CopperCube

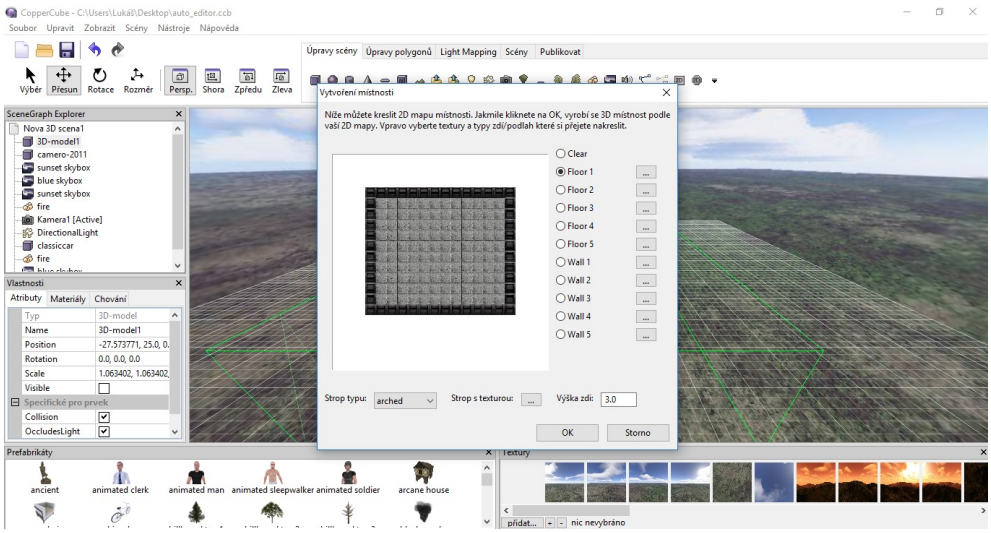


Obrázek 23: Importovaný 3D model bez zobrazené textury

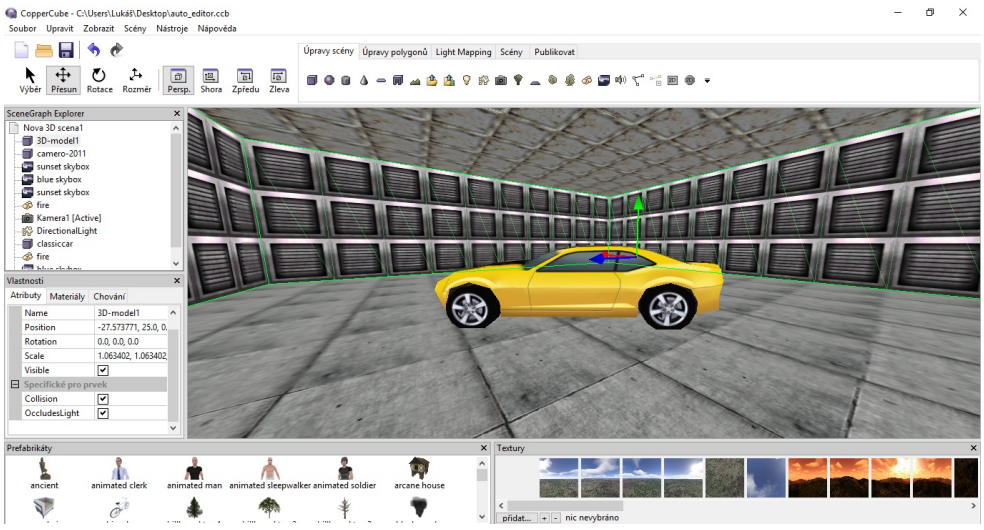


Obrázek 24: 3D model s aplikovanou texturou

Po úspěšném nahrání 3D modelu je další částí projektu vytvoření modelu místnosti, k jehož modelaci v CopperCubu slouží nástroj Vytvořit model místnosti z 2D mapy, který nalezneme v nabídce Úprava scény. V nabídce tohoto nástroje lze pomocí umisťování čtverců na připravenou plochu vytvářet 2D model místnosti z pohledu ze shora. K využití je v pravé části okna celkem 5 typů podlah, každé z nich lze přiřadit specifickou texturu, to samé platí v případě zdí, u které jde navíc nadefinovat její výška. Poslední možností je uzavření místnosti stropem, zde je na výběr plochý strop, šikmý strop, nebo lze zvolit možnost zcela bez zastřešení. Po zvolení a potvrzení všech těchto položek je 2D scéna automaticky převedena na 3D objekt.

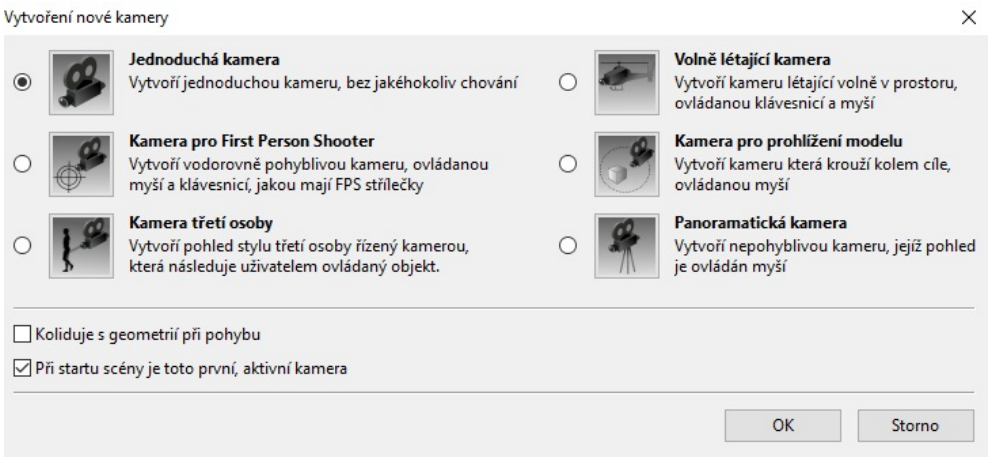


Obrázek 25: Modelace místnosti pomocí 2D mapy

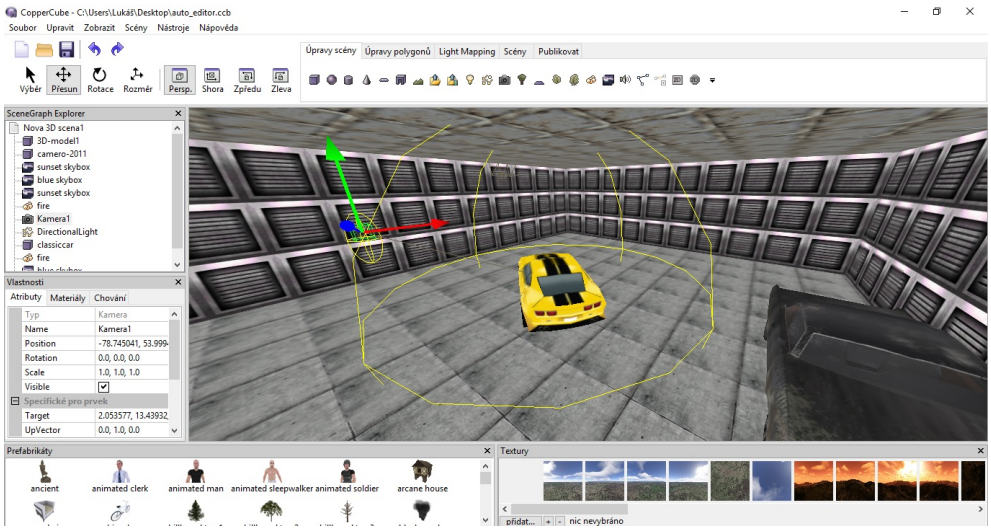


Obrázek 26: Výsledný 3D model místnosti

Dalším nezbytným prvkem, který do rozpracované scény přidáme je kamera, pomocí které lze importovaný 3D objekt prohlížet z různých úhlů. Prvek kamery se nachází ve stejné nabídce jako předešlé modelování scény, tedy v nabídce Úprava scény. Po rozkliknutí položky Vytvořit kameru nám program nabídne šest možností, společně s popisem, k jakým účelům kterou kameru zvolit. V případě interaktivního konfigurátoru 3D objektu byl zvolen typ kamery pro prohlížení modelu. Kameru v následujícím kroku pomocí nástroje pro přesun umístíme směrem k našemu objektu, dolní objekt kamery, kterou představuje šedé krychle, od níž vede přímka směrem ke kameře umístíme doprostřed objektu, který chceme kamerou sledovat. Tím se určí bod, okolo kterého kamera rotuje. Oblast, kterou kamera svým záběrem pokrývá je na scéně vyznačena pomocí žlutých linek, tento rozsah lze zvětšovat či zmenšovat ve vlastnostech kamery pod položkou Ovládání stylem prohlížeče modelu > Radius. Nyní scéna obsahuje všechny potřebné 3D objekty, rozpracovaný projekt lze už nyní otestovat pomocí klávesové zkratky ctrl+f11, po jejímž stisknutí je scéna vykreslena přímo v prohlížeči v prostředí WebGL.

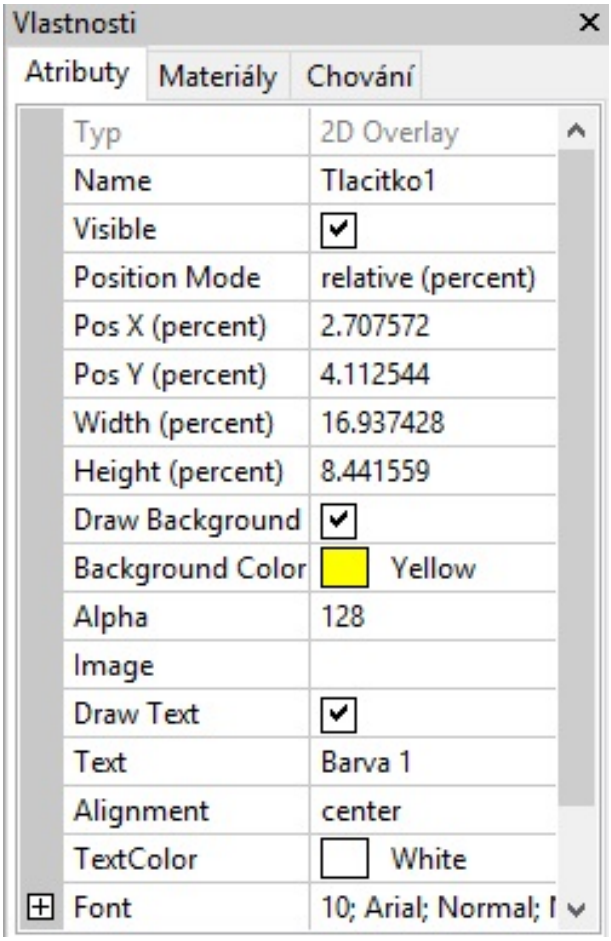


Obrázek 27: Nabídka kamer v programu CopperCube 5.7

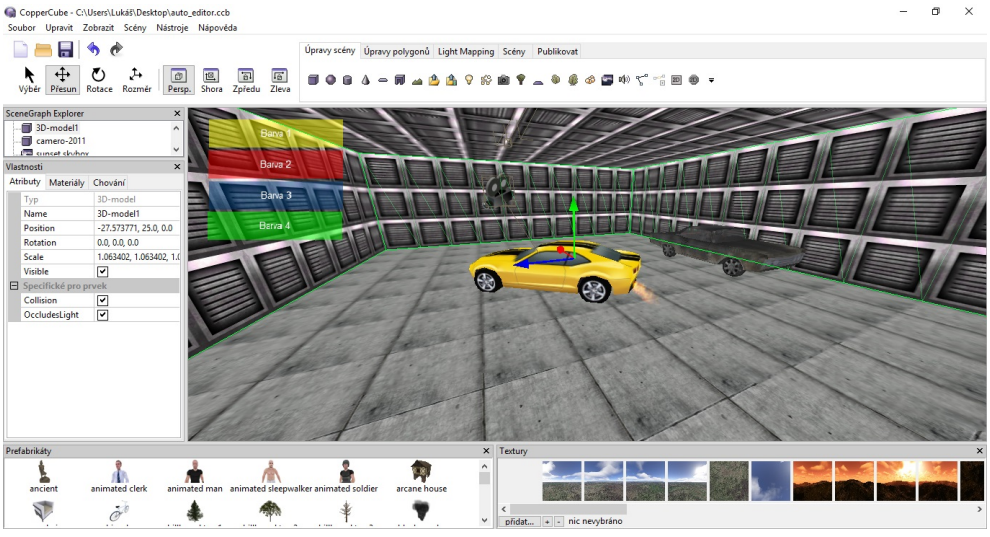


Obrázek 28: Kamera namířená na střed pozorovaného objektu

Posledními prvky, které budou do scény přidány jsou 2D objekty opět z nabídky Úprava scény > Vytvořit položku 2D overlay. Tyto objekty budou plnit funkci tlačítek, po jejichž stisknutí se změní textura importovaného objektu. V tomto praktickém příkladu byly importovány celkem čtyři tlačítka, u každého z nich lze pomocí panelu vlastnosti definovat základní atributy jako je barva, průhlednost, text a jeho styl či animace po stisknutí. [13]

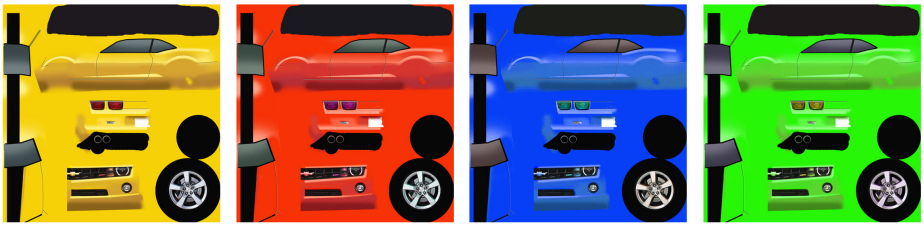


Obrázek 29: Nastavení atributů jednotlivých tlačítek



Obrázek 30: Tlačítka umístěné do 3D scény

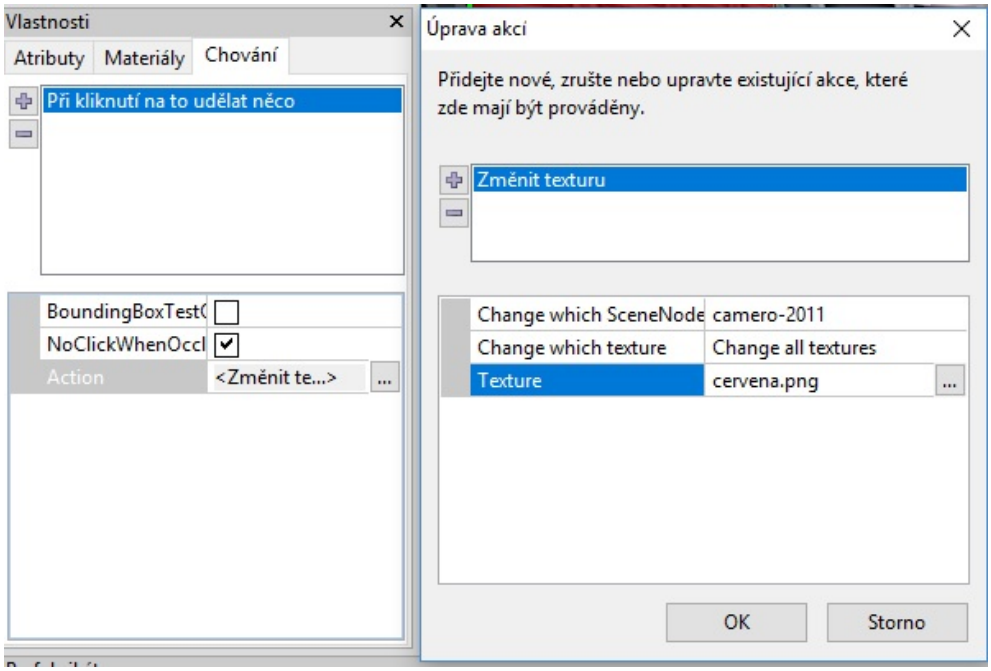
Takto vytvořenému tlačítku je třeba přiřadit akci, která se po jeho stisknutí provede, v našem případě to bude změna textury/barvy objektu automobilu. Tyto textury je tedy nutné předem připravit, změna původní barvy textury byla v tomto případě jednoduše upravena v externím grafickém programu pomocí změny barevných hodnot, čímž došlo k obarvení textury na tři další možné odstíny.



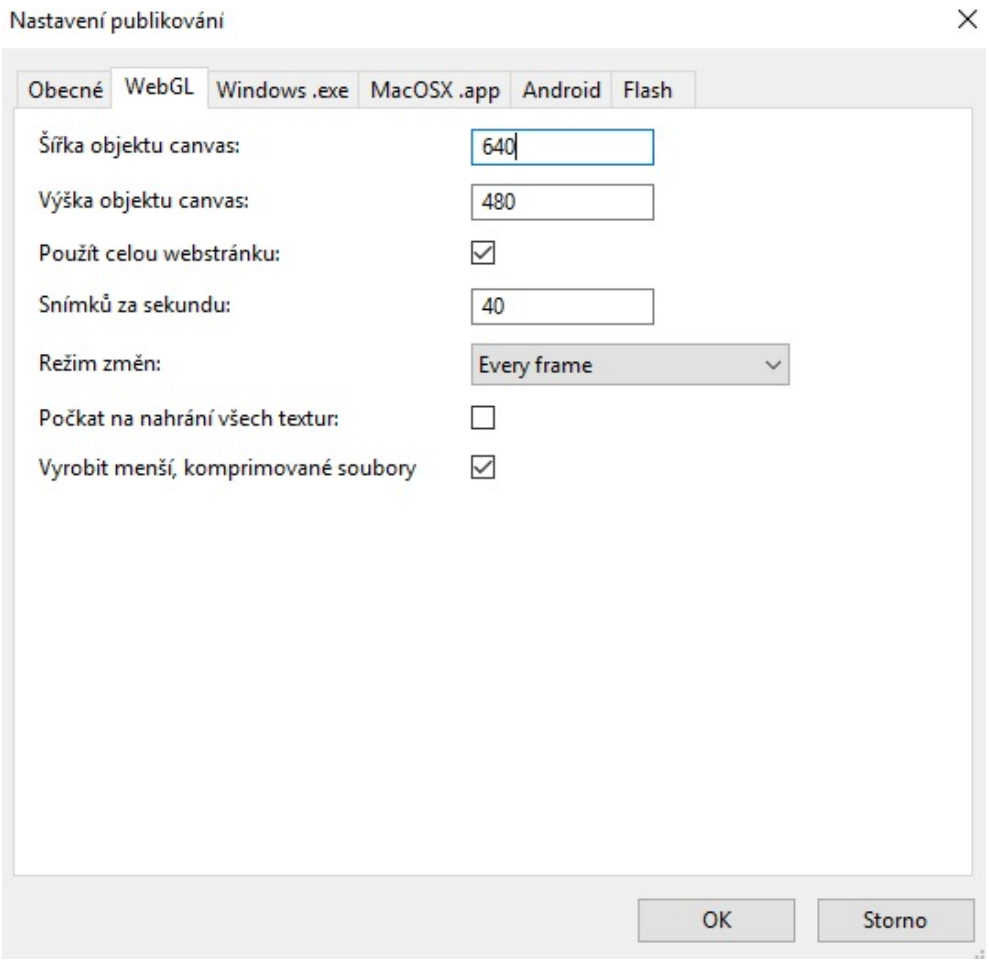
Obrázek 31: Upravené barevné odstíny původní textury

Požadovanou akci, tj. změna textury přiřadíme tlačítku pomocí položky Chování v nabídce vlastností vybraného tlačítka. Zde přidáním možnosti Při kliknutí na to udělat něco nadefinujeme akci Změnit texturu. V první položce této nabídky vybere ze seznamu všech objektů objekt, u něhož chceme změnu textury provést a nadefinujeme přesnou cestu k souboru s požadovanou texturou, po otestování v prostředí WebGL proběhne po stisknutí příslušného tlačítka změna barvy automobilu. Stejný postup je potřeba zopakovat také u zbývajících tlačítek, kdy každému z nich samozřejmě přiřadíme jinou barvu textury tak, aby při kliknutí na dané tlačítko proběhla změna na požadovanou barvu.

Tím je proces vytváření interaktivní 3D scény pomocí programu CopperCube 5.7 u konce,finálním krokem je už pouze publikace vytvořené aplikace. Zde přichází na řadu silná stránka CopperCube, kdy celý proces spočívá pouze ve výběru volby Soubor > Publikovat > Publikovat jako WebGL. Před samotnou publikací, kdy se projekt uloží do souboru HTML, lze ještě nastavit atributy projektu v prostředí WebGL, kde je možné nastavit rozměr canvasu, možnost zobrazení ve fullscreen módu či počet fps.



Obrázek 32: Přiřazení akce tlačítku



Obrázek 33: Vlastnosti publikovaného projektu v prostředí WebGL

7 Přehled praktických ukázek

Výstupem bakalářské práce je sada praktických příkladů, které byly vytvořeny pro prostředí WebGL pomocí metod, popsaných v rámci této práce. Pro prezentaci příkladů byla vytvořena webová stránka www.lukasmace.wz.cz/bp, na které jsou všechny příklady popsané detailněji.

7.1 Konfigurátor 3D objektu

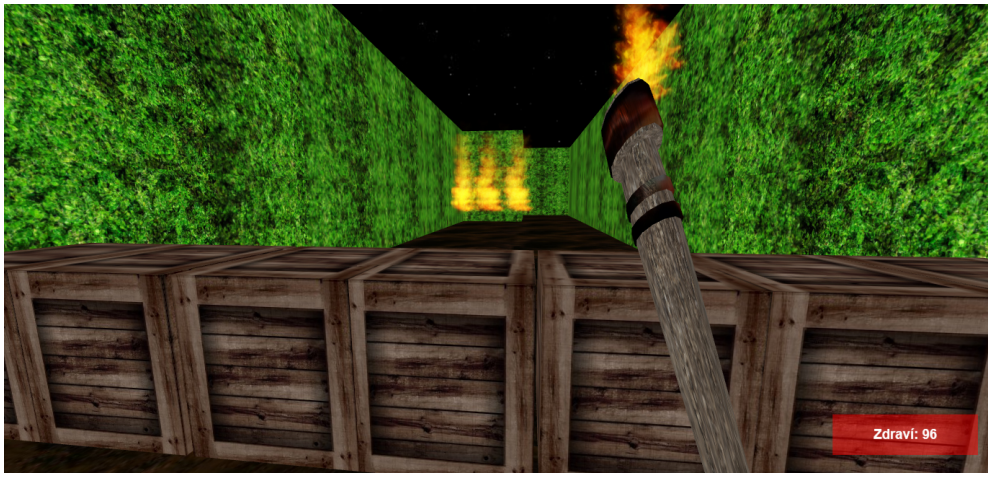
Projekt, jehož postup tvorby byl popsán v předešlé kapitole této práce. Interaktivní 3D scéna vytvořená v programu CopperCube umožňuje pomocí akčních tlačítek měnit barvu 3D modelu automobilu, model lze za pomoci myši otáčet okolo své osy a prohlížet ho tak z různých úhlů. Tlačítka v dolní části umožňují změnu prostředí scény.

Obrázek 34: Konfigurátor 3D objektu

7.2 Únik z bludiště

3D úniková hra zasazená do prostředí tajemného bludiště. Cílem hráče je najít východ z bludiště, ve kterém mu brání nejen slepé uličky, ale také nástrahy ve formě překážek, jejichž nevyhnutím ztrácí část svého zdraví. Samotné zdraví klesá automaticky po určitém časovém intervalu, pokud jeho hodnota klesne na hodnotu 0, je

hra ukončena. Úspěšně dokončení hry je oznámeno po nalezení jediného možného východu.



Obrázek 35: Únik z bludiště

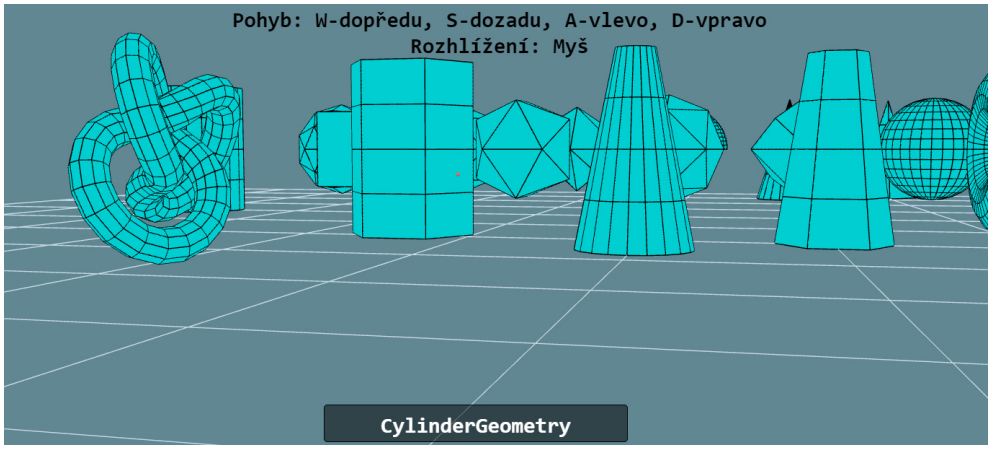
7.3 Interaktivní 3D znak HC Motor

Také u tohoto příkladu byl postup tvorby popsán v jedné z předešlých kapitol. Jedná se o 3D objekt, který byl na základě předlohy vymodelován v programu Bledner a následně ve formátu JSON importován přes knihovnu Three.js do prostředí WebGL. Interaktivita objektu spočívá v možnosti se znakem otáčet v rozmezí 360° ve všech směrech.

7.4 Interaktivní ukázka geometrických tvarů knihovny Three.js

Ukázka vytvořená bez zásahu jakéhokoliv grafického programu, tedy čistě pouze za pomoci knihovny Three.js, v níž byla vytvořena interaktivní scéna, po které se může uživatel pohybovat z pohledu první osoby do všech směrů. Objekty v podobě geometrických tvarů knihovny Three.js jsou umístěny na přesné souřadnice a lze je označit levým tlačítkem myši. Po kliknutí se v dolní části obrazovky zobrazí název daného tvaru knihovny Three.js.

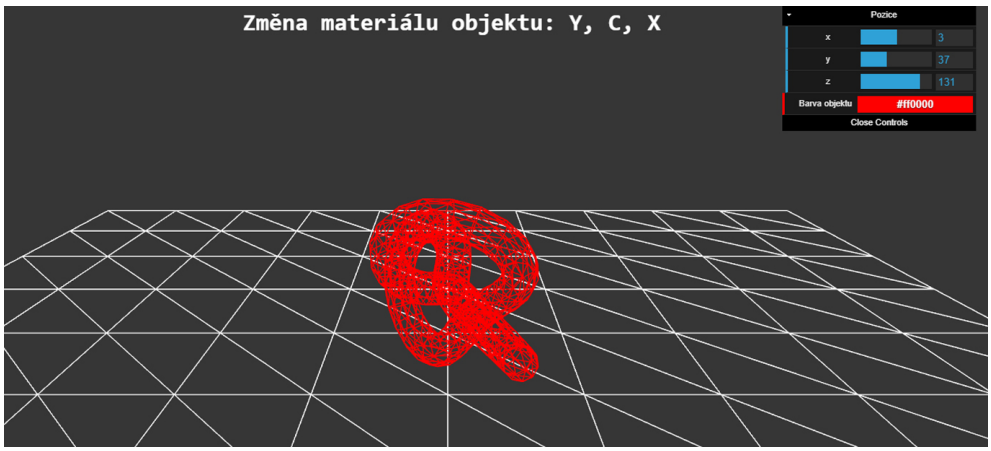
Obrázek 36: Interaktivní 3D znak HC Motor



Obrázek 37: Interaktivní ukázka geometrických tvarů knihovny Three.js

7.5 Interaktivní ukázka změny vlastností objektu knihovny Three.js

Pomocí knihovny Three.js byla vytvořena interaktivní scéna s rotujícím 3D objektem, jehož vlastnosti lze v reálném čase měnit. Pomocí GUI nabídky lze měnit barvu a pozici objektu. Vstupy z klávesnice slouží ke změně materiálu objektu.



Obrázek 38: Interaktivní ukázka změny vlastností objektu knihovny Three.js

8 Závěr

Bakalářská práce zanalyzovala možnosti a postupy, pomocí nichž lze vytvářet interaktivní 3D obsah pro prostředí WebGL. V rámci práce byla představena samotná technologie WebGL s následným popisem funkcí a vlastností knihovny Three.js, kdy byl detailně popsán postup tvorby interaktivní 3D scény pouze s využitím jazyka JavaScript a pomocných metod, které právě tato knihovna nabízí.

Další část práce se zabývala rozborem dvou grafických programů Blender 2.78c a CopperCube 5.7, které nabízí prostor pro vytváření 3D obsahu pro WebGL. Oba programy byly vzájemně porovnány na základě nabízených možnosti při vytváření a následně byl pro každý z programů popsán postup vytváření 3D obsahu pro WebGL. Výsledkem analýzy obou programů je fakt, že při vytváření komplexních 3D projektů je vhodné vzájemné využití obou z nich, kdy Blender je výborným nástrojem pro samotnou modelaci objektů a následná interaktivita a export do WebGL je zase silnou stránkou programu CopperCube.

Praktickým výstupem práce je sada příkladů, které byly vytvořeny pomocí všech popsaných metod. Tyto příklady byly umístěny na vytvořené webové stránky, kde je k nalezení jejich detailnější popis včetně zdrojových kódů.

Seznam použité literatury a zdrojů

[1] MATSUDA, Kouchi a Rodger LEA. WebGL programming guide: interactive 3D graphics programming with WebGL. Upper Saddle River, NJ: Addison-Wesley, 2013. Always learning. ISBN 0321902920.

[2] Khronos Group: Getting Started with WebGL. [cit. 2016-04-07]. Dostupné z: <http://www.khronos.org/webgl/wiki/Getting_Started/>

[3] À la découverte de WebGL [online]. Dostupné z: <http://daureg.free.fr/ta_webit/fonctionnement.html>

[4] Khronos Group: WebGL specification [online]. Dostupné z: <https://www.khronos.org/registry/webgl/specs//latest/1.0>

[5] ŽÁRA, Ondřej. Začínáme s WebGL. In: Zdroják.cz [online]. 2013. Dostupné z: <https://www.zdrojak.cz/serialy/zaciname-s-webgl/>

[6] threejs.org[online]. [cit. 2017-06-26]. Dostupné z: <https://threejs.org>

[7] PEHLIVANIAN, Ara a Don NGUYEN. JavaScript okamžitě. 1. vyd. Brno: Computer Press, 2014. ISBN 978-80-251-4163-2.

[8] Getting started with Three.js. Aerotwist [online]. Dostupné z: <https://aerotwist.com/tutorials/getting-started-with-three-js>

[9] Learning WebGL [online]. Dostupné z: <https://www.learningwebgl.com/>

[10] Bump mapping. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-06-01]. Dostupné z: <https://cs.wikipedia.org/wiki/Bump_mapping>

[11] Learning Three.js the JavaScript 3D Library for WebGL. Birmingham: Packt Publishing, 2013. ISBN 9781782166290.

[12] Blender3D.cz [online]. Dostupné z: <http://www.blender3d.cz/>

[13] CopperCube documentation[online]. [cit. 2017-06-26]. Dostupné z: <`http://www.ambiera.com/coppercube/doc/index.html`>

Seznam obrázků

Seznam tabulek

A Přílohy

1. CD: plné znění bakalářské práce ve formátu PDF, zdrojové kódy ukázkových příkladů a jejich spustitelné verze ve formátu HTML

2. Web: `www.lukasmace.wz.cz/bp`