Mendelova univerzita v Brně
Provozně ekonomická fakulta

Webová aplikácia pre zobrazenie 3D modelov

Bakalárska práca

Vedúci práce:
Ing. Karel Zídek

Dušan Mikoláš

Brno 2016

Týmto by som chcel poďakovať vedúcemu mojej bakalárskej práce Ing. Karlovi Zídekovi za vedenie práce a cenné rady pri jej tvorení. Firme Mikoláš-Kamenár, s. r. o. za možnosť vytvorenia aplikácie a v neposlednom rade priateľom a rodine za podporu

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Webová aplikácia pre zobrazenie 3D modelov** vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

Brno 23.5.2016 .....

Abstract

MIKOLÁŠ, DUŠAN. *Web-based 3D models viewer.* Brno, 2016

This bachelor thesis deals with design and implementation of display possibilities of 3D models within web browser using WebGL technology. In this thesis application for the company Mikoláš - Kamenár, s. r. o. was created, which usage is expected to improve presentation of stone industry goods and speed up communication with customer. Various computer systems were used for performance and availability tests of finished application.

Keywords

Blender, JavaScript, WebGL, Three.js

Abstrakt

MIKOLÁŠ, DUŠAN. *Webová aplikace pro zobrazení 3D modelů.* Brno, 2016

Táto bakalárska práca sa zaoberá návrhom a implementáciou možností zobrazovania 3D modelov v prostredí webových prehliadačov pomocou technológie WebGL. V rámci práce vznikla aplikácia pre firmu Mikoláš - Kamenár, s. r. o., ktorej použitie má zlepšiť prezentáciu kamenárskych výrobkov a zrýchliť komunikáciu so zákazníkom. Táto aplikácia bola testovaná na rôznych počítačových zostavách pre zistenie náročnosti a použiteľnosti.

Klúčové slová

Blender, JavaScript, WebGL, Three.js

Obsah

Zoznam obrázkov

1 Úvod a cieľ práce

1.1 Úvod

V súčasnej dobe má mnoho firiem problém so spôsobom prezentácie svojich výrobkov. Zákazníci hľadajúci vhodnú firmu sa tak nemusia dozvedieť o všetkých produktoch a detailoch, ak nie je dostatočne prepracovaný a jednoduchý prístup k požadovaným informáciám a ich sprostredkovanie. Takisto sa takéto firmy nedostanú do užšieho výberu zákazníka mimo región kde priamo pôsobia a kde sa ich referencie šíria len ústne, lebo nie je možnosť reálneho prehliadania výrobkov a poskytnuté fotografie sú často nedostačujúce.

Ak sa pridá možnosť upravovať vzhľad, materiál a konštrukcia tovaru podľa prianí zákazníka, rozdielna ponuka firiem a nemožnosť vyhovieť každej požiadavke, tak je pre zákazníka veľmi ťažké vybrať správnu zhotoviteľskú firmu spomedzi dostupných možností.

Čoraz bežnejšie je zároveň aj porovnávanie ponúk z dostupných zdrojov a zaobstarávanie približných cenových relácií. Obtiažnosť ich získania môže byť v niektorých prípadoch rozhodujúca, to už záleží na chovaní kupujúceho, ale všeobecne je čas venovaný zhodnotením možností, prezretím referencií a vyhľadaním všetkých dostupných informácií čoraz dlhší.

Ak si nakoniec zákazník vyberie a dohaduje sa výsledný produkt, dochádza k nepochopeniu zo strany zhotoviteľa čo vlastne bolo požadované, alebo zo strany zákazníka, ktorý bežne nepozná aké sú možnosti výroby a materiálu, pracovné postupov, a má častokrát mylné predstavy alebo nevie popísať čo presne požaduje. Preto je potrebná dlhá konzultácia najlepšie s názornými ukážkami, aby sa zistili potrebné špecifikácie výsledného výrobku. Ak sa tak nestane a vytvorí sa výrobok, síce vopred dohodnutý ale nevyhovujúci, dochádza k nespokojnosti oboch strán.

Riešením môže byť prezentácia výrobkov firmy pomocou moderných technológií. S dôrazom na efektívnosť a dostupnosť čo najvyššiemu počtu ľudí ale zároveň aj na možnosť interakcie a selekcie je internet ako zvolené médium najrozumnejšia voľba. Vďaka svojej obľube a rozšírenosti sú neustále vyvíjané možnosti webových stránok a už dávno neponúkajú len statické zobrazovanie fotiek, nákresov, prípadne videa. Napriek tomu mnohé firmy nevyužívajú všetok ponúkaný a dostupný potenciál. V súčastnosti môžme prezentovať výrobky prostredníctvom webového prehliadača ako budú reálne zhotovené, v priestore. Nie sme odkázaní len na pohľady a uhly, z ktorých je výrobok odfotený a zdokumentovaný ale môžme prezerať celý výrobok a všetky jeho časti podľa potreby a záujmu.

1.2 Cieľ práce

Cieľom práce je vytvorenie webovej aplikácie, ktorá bude umožňovať zobrazovanie a prácu s 3D modelmi formou dostupnou na bežnom počítači. Pre dosiahnutie uvedeného cieľa je nutné splniť niekoľko čiastkových úloh.

- Preštudovanie odbornej literatúry o dostupných možnostiach zobrazovania 3D modelov prostredníctvom webových prehliadačov. Pritom sa obzvlášť zamerať na schopnosti vykresľovania knižnice WebGL.
- Využiť nadobudnuté znalosti na návrh a implementáciu aplikácie pre prezentáciu 3D modelov výrobkov firmy Mikoláš - Kamenár, s. r. o. zákazníkom. Použiť pri tom technológiu najvhodnejšiu vzhľadom na požiadavky firmy.
- Výslednú aplikáciu otestovať, zhodnotiť prínos pre firmu a diskutovať budúci rozvoj a rozširovanie projektu.

2 Literárna rešerš

V tejto kapitole sa zameriame na zmapovanie a predstavenie existujúcich technológií pre zobrazovanie 3D grafiky v prostredí webových prehliadačov a stručne si popíšeme ich funkcie, možnosti a prostriedky využiteľné pre splnenie cieľa tejto práce.

2.1 Technológie pre 3D obsah na webe

V počiatkoch internetu boli webové stránky statické a schopné prezentovať svoj hlavne textový obsah neinteraktívnou formou. Avšak postupom času vyvíjanie webových stránok dosiahlo bodu, kedy sú schopnou platformou pre komplexné aplikácie a grafiku. (Dirksen, 2015) Podľa Anyuru (2012) sú vďaka nasledujúcim výhodám výbornou alternatívou klasických desktopových aplikácií:

- Jednoduchá distribúcia, kde nie je potrebná inštalácia ale len navštívenie stránky. Požiadavkou je len kompatibilný webový prehliadač. Odpadá problém s dodávaním softvéru alebo neželaným využívaním, ak je potrebné prihlásenie.
- Správa a servis takýchto aplikácií je jednoduchý, rýchly a centralizovaný. V prípade objavenia chýb, pridávania nových funkcií alebo zmeny dizajnu stačí len updatovať na jednom mieste a všetky vykonané zmeny sa ihneď prejavia u každého používateľa takejto aplikácie.
- Je jednoduchšie mať podporu pre viaceré platformy, keďže sa nerieši samostatne základ, na ktorom bude takáto aplikácia na konkrétnej platforme fungovať ale jediné kritérium je webový prehliadač interpretujúci stránku pre zariadenie, na ktorom je spustený.

Webové aplikácie mali v minulosti rôzne hranice a obmedzenia oproti bežným aplikáciám, ako napríklad dlhé načítavanie komplexnejších stránok alebo neschopnosť využívať hardvérové zdroje zariadenia. Nevýhodou ale naďalej zostáva nutnosť internetového pripojenia pre načítanie takejto aplikácie. Táto nevýhoda však začína byť čoraz viac zanedbateľná pri dostupnosti internetového pripojenia takmer na každom osobnom počítači alebo inom zariadení.

V súčasnosti je viac dostupných riešení po nástupe jazyka HTML5, vďaka ktorému je webový prehliadač schopný zobrazovania prepracovaných aplikácií a spolupráce s novými možnosťami JavaScriptu, podporou multimédií a využitia potenciálu mobilných zariadení. Takisto vzniklo ale aj mnoho technických riešení, ktoré využívajú novo poskytnuté možnosti inovovaného jazyka. Každá má iné výhody a nevýhody, rozdielne schopnosti a rôznu podporu vo webových prehliadačoch.

2.1.1 WebGL

WebGL (webový grafická knižnica) je technológia umožňujúca vytvárať 3D grafiku v prostredí webového prehliadača. Využíva k tomu JavaScript API, ktoré spolupracuje s GPU zariadenia. (Danchilla, 2012)

Svoje základy má WebGL v OpenGL, ktorého verzia 1.0 bola vyvinutá ako alternatíva k Iris GL v roku 1992 pre použitie v 2D a 3D grafike v oblastiach ako napríklad videohry, vedecké vizualizácie alebo simulácie. Prehľad niektorých verzií a nimi zavedených zmien a updatov z histórie OpenGL (2015b):

- 1.1 (1997) Vrcholy ukladané ako pole a manipulácia pomocou jedného príkazu, rôzne zlepšenia práce s textúrami, možnosť ukladať stav textúry do objektu.
- 1.3 (2001) Ďalšie textúrovo orientované zlepšenia, pridanie podpory cube-mapping[1]
- 1.4 (2002) Automatické generovanie mip map[2], podpora textúr obsahujúcich hĺbku
- 1.5 (2003) Nový typ objektu - VBO, ktorý funguje ako vyrovnávacia pamäť na ukladanie informácií o vrcholoch v rýchlej pamäti grafickej karty (OpenGL, 2015a)
- 2.0 (2004) Oficiálne pridane OpenGL Language – (GLSL)[3]
- 3.0 (2008) Plánované prepracovanie, zjednodušenie a revízia OpenGL skončila len ako možnosť vypínať niektoré funkcie
- 4.0 (2010) Prepracovaná podpora moderného hardvéru schopného používať technológiu Direct3D 11
- 4.5 (2014) Súčasná verzia

Khronos začal v roku 2014 s vývojom nového API Vulkan, ktorého využitie je podobné ako pri OpenGL, ale s menšou záťažou procesora, lepším využitím viacerých procesorov a väčšou kontrolau vďaka novému jazyku. Avšak pre Vulkan sú potrebné nové grafické ovládače a preto aj zariadenia, ktoré by mali Vulkan podporovať ešte nemusia. (Khronos, 2016b)

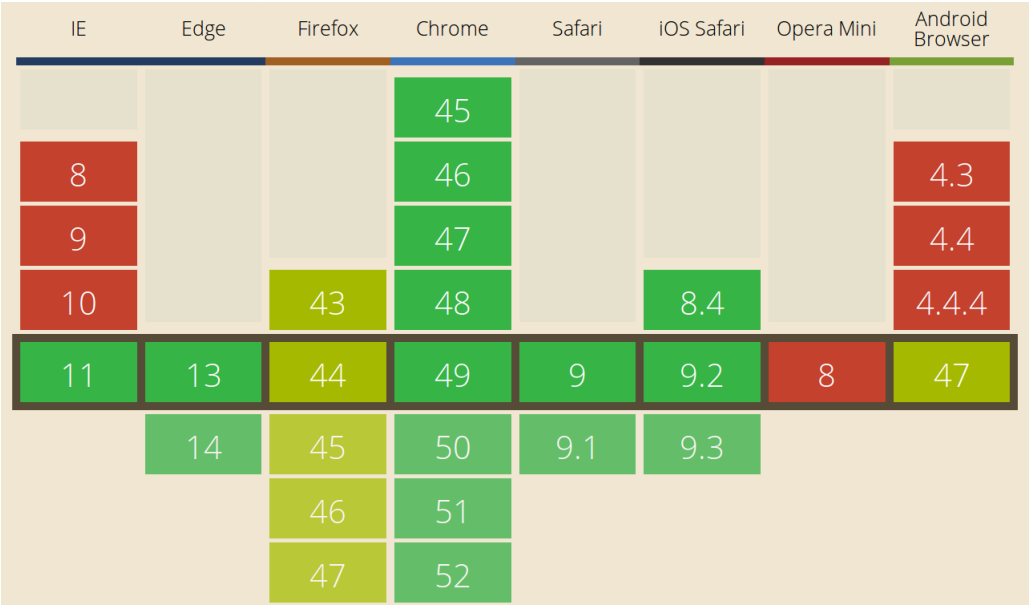
WebGL je založené na OpenGL ES 2.0 z roku 2007. ES (Embedded Systems) znamená rozšírenú podporu pre odlišné systémy, ako napríklad konzoly, telefóny alebo aj vozidlá. (Khronos, 2016a) Od roku 2009 je aktívna skupina pracujúca na WebGL, ktorá zaisťuje blízkosť implementácie v prehliadačoch a aktuálnosť dokumentácie. Súčasťou tejto skupiny sú aj tvorcovia väčšiny prehliadačov ako Apple, Google, Mozilla a Opera. (Danchilla, 2012; Khronos, 2013a)

[1] https://www.opengl.org/wiki/Cubemap_Texture
[2] https://www.opengl.org/wiki/Texture#Mip_maps
[3] https://www.opengl.org/wiki/OpenGL_Shading_Language

WebGL je API, teda rozhranie pre programovanie aplikácií s definovanou sadou funkcií a premenných. Je prístupné cez JavaScript a existujúci prvok Canvas v HTML5. Vďaka tomu je ho možné kombinovať s iným obsahom na webovej stránke, prekrývať, priradiť len určitú veľkosť, percentá zo stránky alebo umiestniť do určitého oddielu (`div`) na stránke. Toto umožňuje celú aplikáciu vytvoriť v HTML jazyku a len vyhradiť oblasť pre zobrazovanie modelov.

OpenGL ES nie je plnohodnotné a tak výkonné ako OpenGL, keďže je určené aj pre mobilné zariadenia s nižšími schopnosťami vykresľovania, preto má jednoduchšie API alebo používa len techniku VBO (pozri OpenGL verziu 1.5) na vykresľovanie bodov, ktorá je najmenej náročná. Medzi OpenGL a WebGL sú však už minimálne rozdiely.

Ako bolo spomenuté, WebGL na vykresľovanie využíva hardvér zariadenia, na ktorom je spustený. Hranice sú teda dané aj technikou zariadenia, hlavne grafickou kartou alebo čipom. Preto aj keď webový prehliadač môže podporovať technológiu WebGL, scéna sa nemusí vykresľovať správne, v niektorých prípadoch vôbec.(Parisi, 2014; Khronos, 2013b) Prehľad podpory verzií webových prehliadačov je zobrazený na obrázku 1.



Obr. 1: Webové prehliadače podporujúce WebGL. Zelená je plná podpora, žltozelená čiastočná a žiadna podpora je znázornená červenou farbou (caniuse, 2016a)

2.1.2 Three.js

Samotné programovanie WebGL cez JavaScript pre vytvorenie a animovanie scény je zdĺhavé, pracné a náchylné k chybám. K dispozícií sú ale rôzne knižnice, ktoré uľahčujú prácu od bežných procesov ako príprava scény až po špecializované doplnky. Spolupracujú a rozvíjajú WebGL a preto neobmedzujú použiteľnosť a rozšírenú podporu v prehliadačoch. Väčšina bežných úkonov v takýchto knižniciach je preddefinovaná a je vytvorené množstvo funkcií priamo dostupných po pripojení. Všeobecné úkony sú zjednodušené ale ďalší postup sa líši podľa požadovanej funkcionality a pripájame skripty, ktoré sú potrebné pre našu prácu a používame ich funkcie v našom kóde. Pri špeciálnych požiadavkách je potrebné vytvoriť aj vlastné funkcie.

Knižnica prvý krát zverejnená v roku 2010 na GitHube (webhosting pre open source projekty) Ricardom Cabellom. Cieľom bolo vytvoriť 3D grafiku bez nutnosti kompilovania a bez obmedzenia platformy. Začiatky projektu však siahajú o pár rokov skôr, kedy táto knižnica existovala a na rozdiel od podobných nemala renderer priamou súčasťou jadra, ale ako ďalší prvok, ktorému je možné priradiť parametre. Vďaka tomu sa po príchode WebGL tento renderer objavil ako súčasť knižnice ale nie je to jediný renderer v tejto knižnici.

Podľa Dirksena (2015) Three.js nasledujúce úkony výrazne zjednodušuje a celý proces tvorenia požadovaného výsledku výrazne zrýchľuje.

- Vytvorenie scény
- Aplikovanie textúr a materiálov
- Načítanie objektov z modelovacích programov
- Animovanie a pohyb objektov na scéne
- Použitie viacerých odlišných zdrojov svetla
- Pridávanie efektov do scény

Pomocou Three.js na vytvorenie scény sú potrebné len tri základné prvky, kamera, scéna a renderer. Vytvoríme scénu, v ktorej môžeme potom umiestňovať a zobrazovať objekty, pridávať zdroje svetla a všetko sa bude zobrazovať vo webovom prehliadači z pozície kamery. (ThreeJS, 2013)

Three.js umožňuje načítavať objekty v uvedených formátoch: OBJ a MTL, Collada, JSON, STL, CTM, VTK, AWD, Assimop, VRML, Babylon a mnoho ďalších. (Dirksen, 2015)

Podporu Three.js poskytujú všetky prehliadače podporujúce prvok canvas ak sa použije canvas renderer. Obdobne to platí pre WebGL renderer. Podpora knižnice teda závisí na podpore použitej vykresľovacej technológie.

2.1.3 Babylon.js

Je to knižnica veľmi podobná Three.js, s podobnými príkazmi a logikou. Hlavným tvorcom je David Catuhe, ktorý sa ako zamestnanec Microsoftu rozhodol vytvoriť 3D engine po pridaní podpory WebGL do Internet Exploreru 11.

Na vytvorenie scény postačuje len samotná scéna a kamera. Vytváranie objektov a pridávanie svetelných zdrojov funguje podobne s Three.js. Avšak Babylon.js sa špecializuje na hry, ako webový game-engine na rozdiel od Three.js, ktorý je len renderer, teda rozhranie na vykresľovanie grafiky určené hlavne na prezentáciu. Babylon.js teda zvláda viac, má možnosť podpory fyzikálnych zákonov v scéne a podobne.

Babylon.js umožňuje načítavať objekty v uvedených formátoch: glTF, OBJ a STL.

Podpora webových prehliadačov knižnice Babylon.js je zhodná z WebGL, keďže rozširuje len túto technológiu. (BabylonJS, 2013)

2.1.4 PlayCanvas

Ďalšia WebGL knižnica. Pôsobí na trhu od roku 2011 a od roku 2014 je kód open-source ale nadštandardné možnosti grafického rozhrania sú spoplatnené. Ide o game-engine, ako v prípade Babylon.js, schopný vytvárania online hier na základe HTML5 a WebGL. Samotná podstata knižnice ako game-enginu nie je zlá, samozrejme zvláda aj jednoduchú scénu pre prezentáciu výrobku, len kód môže byť zbytočne náročnejší.

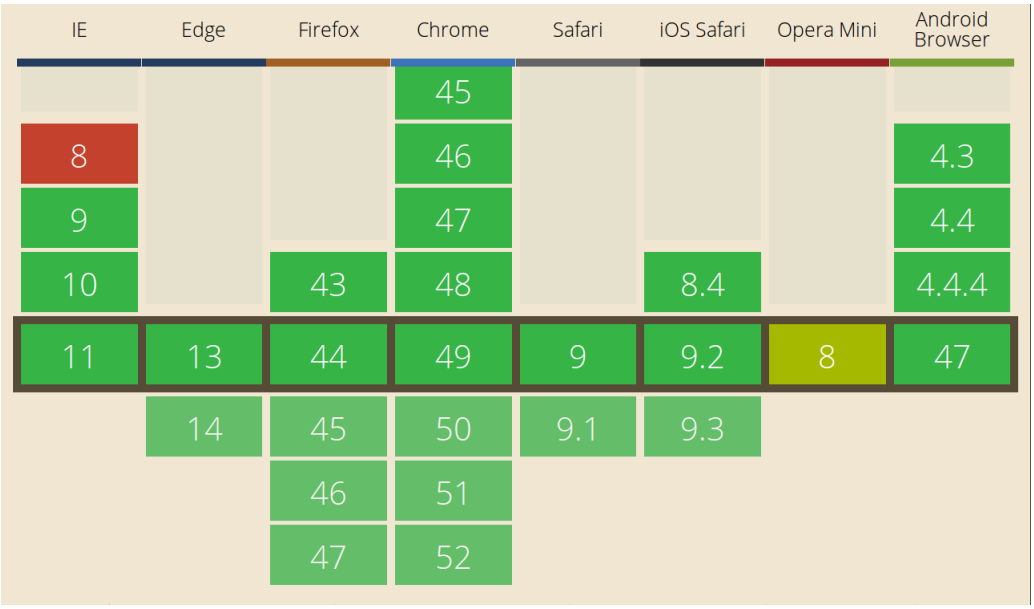
PlayCanvas však ponúka aj grafické rozhranie na tvorbu aplikácií, kde je možné vidieť už pri vytváraní objekty, umiestňovať ich a dovoľuje pracovať viacerým osobám zároveň na jednom projekte. PlayCanvas taktiež ponúka možnosť pridať fyzikálne zákony do scény, podporuje simulácie aj podporu 3D zvuku v scéne.

Babylon.js umožňuje načítavať objekty v uvedených formátoch: FBX, OBJ, Collada, DXJ a 3DS.

Podpora aj v prípade PlayCanvas je závislá na podpore WebGL zo strany webového prehliadača a grafickej karty. (PlayCanvas, 2015)

2.1.5 Canvas

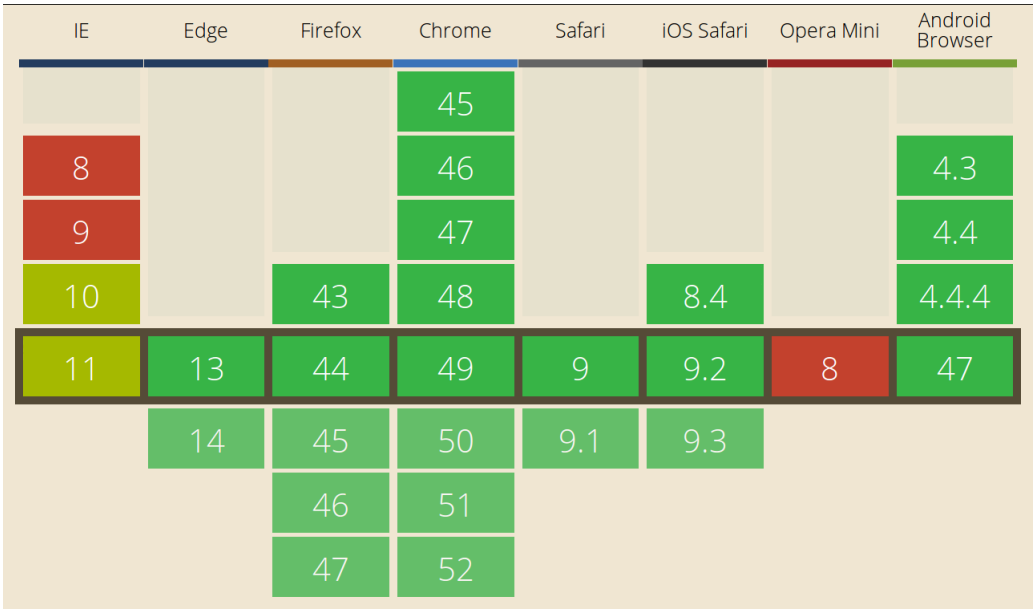
Prvok Canvas je súčasťou samotného HTML5 a je univerzálnym prvkom podporovaným v prehliadačoch a umožňuje vykreslenie ľubovoľnej grafiky do prvku DOM. Aj keď je Canvas určený pre 2D grafiku, pri použití JavaScriptových knižníc môže byť použitý pre 3D renderovanie pre platformy, na ktorých CSS 3D a WebGL nie sú podporované, ako vidíme na obrázku 2, kde je väčšina verzií prehliadačov plne podporovaná. Faktorom pre výber môže byť aj softvérové renderovanie, ktoré sa väčšinou pokladá za mínus do chvíle, kedy je potrebné vytvoriť 3D grafiku pre prenosné zariadenia a výdrž batérie pokladáme za dôležitú. (Parisi, 2014)



Obr. 2: Webové prehliadače podporujúce Canvas. Zelená je plná podpora, žltozelená čiastočná a žiadna podpora je znázornená červenou farbou (caniuse, 2016c)

2.1.6 CSS3 3D

Neposkytuje možnosť zobrazovať načítané 3D modely ale obsahuje rôzne efekty, ktoré umožňujú napríklad transformácie alebo prechody. Ponúka a podporuje hardvérom akcelerované 3D renderovanie a funkcie animácií dostupné cez CSS jazyk. Na rozdiel od WebGL ponúka možnosť pristupovať k častiam HTML stránky alebo kódu ako ku 3D objektom a tak aplikovať na tento obsah spomínané transformácie, prechody, zväčšovanie alebo animácie. Tvorba je jednoduchá ale je to daň za možnosti použitia a flexibilitu, pretože používa len geometrické útvary ako objekty. Transformácie sú podporované vo všetkých prehliadačoch a mobilných platformách. CSS Custom Filters umožňuje rôzne efekty s obsahom, ako napríklad farbenie, otáčanie, približovanie, ale je podporovaný iba experimentálne a aj to nie vo všetkých verziách webových prehliadačoch, ako je uvedené na obrázku 3. (Parisi, 2014)



Obr. 3: Webové prehliadače podporujúce CSS3 3D. Zelená je plná podpora, žltozelená čiastočná a žiadna podpora je znázornená červenou farbou (caniuse, 2016b)

2.2 Nástroje pre 3D modelovanie

3D grafika zažíva v ostatných dekádach veľký rozvoj a využíva sa v množstve oblastí od architektúry, tvorby hier cez medicínu až po filmový priemysel. Na tvorbu takejto grafiky sú potrebné špecializované programy, ktorých je dnes na trhu dostatok práve vďaka narastajúcemu záujmu o tento odbor. Existujúca konkurencia a neustály pokrok počítačových technológií podporuje zlepšovanie, zrýchľovanie a pridávanie funkcií. Zvyšuje sa kvalita výsledných scén a renderov, ich náročnosť, komplexnosť a je čoraz ťažšie rozoznať čo je virtuálne a čo reálne.

2.2.1 3ds Max

Profesionálny program pre tvorbu 3D grafiky využívaný pri tvorbe najznámejších filmov a hier s komplikovanými a rozsiahlymi scénami. Podporuje polygonálne modelovanie, NURBS (Non-Uniform Rational B-Splines) modelovanie a Patch objekty.

Vďaka dlhému pôsobeniu na trhu je k dispozícií množstvo návodov, článkov a videí. Takisto oficiálna podpora je rozsiahla a obsahuje demonštráciu funkcií programu, spolu s kompletnou dokumentáciou. Pomocou vlastného skriptovacieho jazyka MAXScript je možné vytvoriť programy a rozšírenia pre 3ds Max. Momentálna cena je 1350€ na rok a s podporou operačného systému Windows.

Umožňuje import a export veľkého počtu formátov, z ktorých najhlavnejšie sú: FBX, 3DS, DAE, DWG, DXF, OBJ, STL, XML.

Ponúka množstvo funkcií a pluginov ako Character Studio (animovanie postáv), Autodesk Vault (správa modelov) alebo Scene Explorer (prehľadávač scény). Pre renderovanie sú to napríklad V-Ray, Brazil, RenderMan.

3ds Max bol použitý vo filmoch ako napríklad Matrix, Transformers alebo Iron Man. (Autodesk, 2016)

2.2.2 Rhinoceros

Špecializuje sa na prácu a modelovanie pomocou NURBS kriviek, presne definovaných matematických tvarov a prienikov. Je vhodný pre začiatočníkov práve vďaka rýchlemu modelovaniu pomocou jednoduchých tvarov. Ale vďaka množstvu funkcií, ktoré ponúka je stále vhodným nástrojom aj pre pokročilých. Cena za tento modelovací program je 995€ na Windows a 495€ na Mac.

Podporuje viac ako 30 CAD súborových formátov bez použitia pluginov. Medzi nimi sú napríklad: DWG, DXF, 3DS, STL, OBJ, VRML. Rhinoceros podporuje dva skriptovacie jazyky, jeden vlastný Rhinoscipt a Python. Taktiež má možnosť pridávania množstva pluginov, z ktorých najznámejší pre Rhinoceros je Grasshopper. Pre renderovanie je možnosť doplnenia o Brazil, V-Ray, Maxwell a ďalšie. (Rhino3D, 2016)

2.2.3 Blender

Tento software je vytváraný stovkami ľudí po celom svete keďže je to softvér s GNU licenciou (dovoľuje koncovým používateľom užívať, študovať a upravovať program). Obsahuje funkcie pre vytváranie 3D modelov, simuláciu, mapovanie pohybu, úpravu videa a tvorbu hier. Je čoraz viac využívaný na tvorbu hier alebo napríklad filmov. Modelovanie je možné polygonálne aj pomocou NURBS kriviek.

Blender je kompatibilný s operačnými systémami Windows, OS X aj Linux. Dokáže pracovať s formátmi súborov ako FBX, DXF, OBJ, DAE, VRML, 3DS, DEC a ďalšie. Python je použitý ako skriptovací jazyk pre vytváranie nástrojov alebo hernej logiky. Medzi použiteľné externé renderery patria Mitsuba, LuxRender, Aqsis a POV-Ray. (Blender, 2016)

2.3 Technológie 3D grafiky

2.3.1 Osvetlenie

Osvetlenie scény je jedným zo základných prvkov a jeho správne nastavenie je najdôležitejšie pre dosiahnutie uveriteľného a realistického vzhľadu. Prítomnosť svetla a jeho pôsobenie, odrážanie, lámanie lúčov berieme ako samozrejmosť v každodennom živote, ale aplikácia do scény s uspokojivým výsledkom už nie je taká jednoduchá.

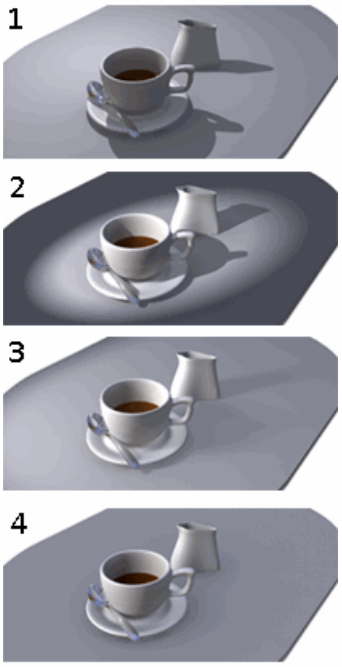
Základom je použitie vhodného typu svetla použitého v scéne. Každý typ svetla je určený pre simuláciu iného zdroja svetla z nášho okolia. Rôzne typy sú znázornené na obrázku 4 a ich popis podľa Chopine (2004) je nasledovný:

1 Bodové svetlo: najjednoduchší typ svetla, ide o bod, ktorý vyžaruje svetlo do všetkých strán

2 Reflektor: vyžaruje svetlo v tvare kužeľa z jedného bodu a umožňuje osvetliť objekt alebo určité miesto scény podobným uhlom dopadu svetla. Zanecháva kruhovú alebo elipsovú osvetlenú plochu, ktorej je možné nastavovať rýchlosť stmavovania okrajov a spôsob prechodu.

3 Plošné svetlo: svetlo je vyžarované z povrchu objektu, väčšinou ide o štvoruholník. Svetelné lúče putujú jedným smerom a sú rovnobežné a nevrhajú drsné a ostré tiene. Tento typ svetla je ale náročnejší na renderovanie. Plošným svetlom môže byť aj ktorýkoľvek objekt v scéne.

4 Smerové svetlo: aj keď je slnko bodový zdroj svetla, pre svoju vzdialenosť a jasnosť by bolo nepraktické ho úspešne simulovať. Riešením je smerové svetlo. Svetelné lúče sa pohybujú rovnobežne, ako je to pri plošnom svetle, ale ak by sme chceli vytvoriť podobný efekt, aký prináša smerové svetlo pomocou plošného svetla, muselo by byť nekonečne veľké a umiestnené nekonečne ďaleko. Smerové svetlo je teda jednoduchšie pre použitie a okrem farieb má ako parameter už len smer.



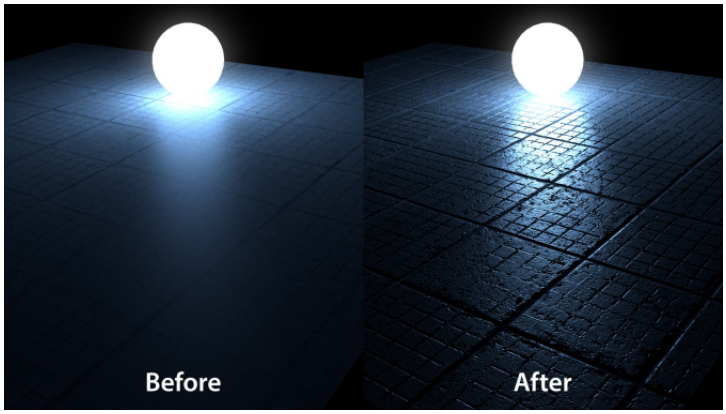
Obr. 4: Rôzne druhy zdrojov svetla (Intel, 2011)

Toto sú najbežnejšie typy svetelných zdrojov a väčšina prostredí či už pre tvorbu 3D grafiky alebo jej zobrazovania má naviac svoje vlastné typy svetiel, ktoré môžu prinášať lepšie výsledky alebo sú jednoduchšie spracovateľné pri renderovaní.

2.3.2 Textúry

Používanie textúr v počítačovej grafike je dnes veľmi rozšírené a obľúbené, ich prvé použitie siaha do roku 1974. Zaujímavé je, že sa táto technika prakticky nezmenila.

Ide o priradenie každého bodu povrchu 3D objektu ku konkrétne prislúchajúcemu 2D bodu v charakteristickom priestore so súradnicami u a v. Toto 2D ku 3D prirovnanie nám umožňuje následne zobraziť dvojrozmernú textúru na 3D povrch a $[u, v]$ parametre môžu byť kedykoľvek použité pre určenie konkrétneho bodu pre pixel z obrázka využitého na textúrovanie. Toto umožnilo vytvoriť zaujímavý a detailný vzhľad namiesto nudného jednofarebného povrchu relatívne rýchlym a jednoduchým spôsobom nenáročným pre počítač. Sú k dispozícii aj rôzne variácie a vylepšenia zobrazovania textúr. (Ebert, 2003) Vyššie popísane presné určenie miesta pre textúru sa nazýva mapovanie, pri tomto procese vznikajú mapy objektu. Základná mapa obsahuje farby, ktoré sa nanesú na presné miesta objektu a takýmto spôsobom je možné dosiahnuť zafarbenie modelu, ktorý nemá homogénnu farbu.



Obr. 5: Objekt pred a po aplikácii máp (blenderguru, 2015)

Pomocou mapy sa dá zmeniť aj geometria objektu, či už s reálnym upravovaním vrcholov alebo len vizuálna ilúzia. Ak by to nebolo možné a objekt by sa modeloval do najmenších detailov, mal by príliš veľa bodov. Na obrázku 5 môžeme vidieť objekt len po aplikácii máp. Existuje viacero druhov pre konkrétne využitie, ktoré popisuje Chopine (2014) takto:

- **Bump mapa:** čiernobiela mapa, ktorá simuluje výšku alebo hĺbku povrchu objektu. Bežné renderovanie objektu určuje farbu určitého bodu objektu pomocou normály, ktorá nám hovorí o tom, aký je uhol povrchu, kam smeruje a ako odráža svetlo. Bump mapa nám upravuje tieto normály objektu a poskytujú simuláciu hĺbky, ale nemenia geometriu objektu, takže silueta je nezmenená a tiene pracujú s pôvodným povrchom. Táto ilúzia ale funguje len z jedného miesta v scéne.
- **Normal mapa:** rozšírená verzia bump mapy, simuluje hĺbku na troch osách a teda ilúzia hĺbky povrchu sa zobrazuje vo všetkých smeroch, nie iba priamo na kameru. Normal mapa využíva RGB farebné obrázky pre ukladanie troch kanálov pre každú osu X,Y,Z.

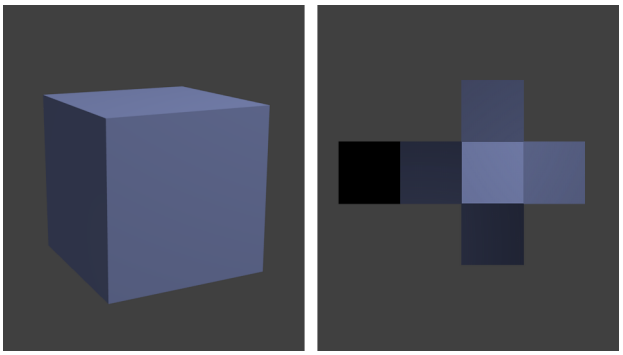
- **Displacement mapa:** ak chceme použitím textúry zmeniť samotnú geometriu objektu, musíme použiť tento druh mapy. Je to opäť čiernobiely obrázok znázorňujúci hĺbku povrchu objektu. Podľa stupňov šedej sa mení hĺbka povrchu a pri renderovaní sa tento povrch zobrazuje so zmenenou geometriou.
- **Alpha/Transparency mapa:** čiernobiela mapa používaná na určenie priehľadnosti textúry. Takto môžu byť časti objektu úplne priehľadné. Čierna v takejto textúre označuje priehľadnosť a biela označuje miesta určené pre zobrazenie textúry.

Žára (2004) uvádza aj také mapy, ktoré upravujú vlastnosti objektu a teda ho približujú k reálnej predlohe.

- **Specular:** odlesková mapa, ktorá určuje množstvo svetla odrazené od objektu, takže je možné vytvoriť lesklé aj matné objekty.
- **Reflection:** odrazová mapa neodráža len svetlo na povrchu objektu, ale aj odraz na okolie pre najlesklejšie povrchy.
- **Refraction:** mapa určená pre materiály, ktoré lámu svetlo.

2.3.3 Light Mapping

Mapovanie svetla je metóda uchovávania výsledkov renderovania pre budúce použitie. Najbežnejšie je uchovávanie osvetlenia v textúre, ktorú je možné použiť na objekt bežným spôsobom. Takúto textúru je možné vidieť na obrázku 6. Výhodou je získanie uložených hodnôt rýchlo namiesto dlhého renderovania, čo je obzvlášť výhodné pri nepriamom osvetlení. Tieto textúry sú bežne používané v počítačových hrách. Ich vytváranie je však úplne odlišné pretože sa na disk ukladajú, na rozdiel od bežných, čítaných z disku. Využíva špeciálne funkcie v rámci rozhrania shaderu pre získanie povolenia k zápisu do textúry. Mapa svetla je spracovávaná počiatočne vo vertex shadery a potom finálne pre celý objekt kde upravuje svetlosť všetkých pixelov. (Dirksen, 2015)

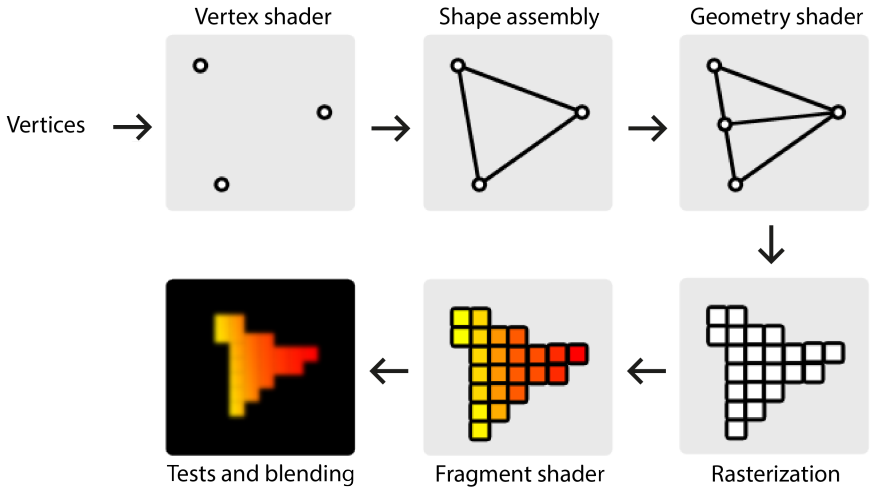


Obr. 6: Osvetlenie a tiene aplikovné na textúru kocky (Wikipedia, 2011)

2.3.4 Shadery

Dopĺňa informácie o materiály pre konkrétny objekt. Spolu s textúrami tak môžme dosiahnuť simuláciu reálneho materiálu, z ktorého má byť nami vymodelovaný objekt zhotovený. Textúry nám môžu hovoriť o vzhľade a povrchu. Shader je určený na počítanie svetelných efektov v súvislosti s materiálom a spracováva ako bude reagovať na svetlo. Ale na rozdiel od textúr rieši shader pôsobenie svetla na samotný objekt a vypočítava nerovnomerné osvetlenie ako napríklad odlesky alebo tiene na povrchu. (Kříž, 2010)

Shading funguje vo viacerých krokoch a pre každý z týchto krokov existuje mnoho typov shaderov, ktoré je treba vhodne zvoliť aby sme dostali výsledný efekt. Tieto kroky spracovania a shadery v nich je možné rozdeliť podľa metódy akou pracujú. Tieto kroky sú súčasťou tzv. pipeline (obrázok 7), postupnosti príkazov pri konečnom renderovaní. Táto pipeline sa však môže meniť v závislosti od prostredia tvorenia a použitého jazyka (DirectX HLSL (High Level Shading Language), OpenGL GLSL (OpenGL Shading Language) a podobne). Microsoft (2014) a OpenGL (2015c) jednotlivé kroky na svojich stránkach popisujú takto:



Obr. 7: Pipeline (open.gl, 2012)

- **Vertex shader:** prvým, základným a povinným krokom je shader spracovávajúci vrcholy zo vstupu, zaisťuje funkcie pre jednotlivé vrcholy ako napríklad transformácie, prechody a osvetľovanie. Vertex shader vždy načíta jeden vrchol, spracuje ho a vráti jeden vrchol na výstupe. Ak tento shader nemá nijakým spôsobom upravovať vstupy, každopádne musí prebehnúť aj bez zmeny vrcholov ako príprava pre budúce operácie.
- **Geometry shader:** druhý nepovinný krok je kód shadera, ktorý prijíma na vstupe vrcholy a má schopnosť generovať nové vrcholy na výstupe. Na rozdiel on vertex shadera, ktorý vždy pracuje len s jedným vrcholom, vstupom pre

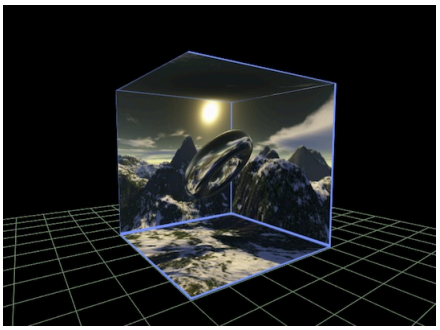
geometry shader sú všetky vrcholy daného útvaru (jeden pre bod, dva pre čiaru alebo tri pre trojuholník).

- **Pixel/Fragment shader:** jedným zo záverečných krokov spracovania a vytvárania obrazu je shader, ktorý produkuje pixely medzi vrcholmi prijímaných zo vstupu. Priraďuje im farbu, odtieň a osvetlenie. Generuje pixely na základe dát z textúr, interpolovaných hodnôt zo zadaných vrcholov a ďalších pomocných dát pre generovanie hodnôt pixelov.

Osobitným prístupom sa vyznačuje tzv. Unified Shader, ktorý je prvkom GPU zariadenia a spracováva všetky kroky shaderov v pipeline pomocou jednej jednotky, čo umožňuje úsporu miesta a menšiu hardvérovú náročnosť. (Moya, 2005)

2.3.5 Skybox

Skybox sa používa na vytvorenie 3D pozadia scény. Ide väčšinou o vzdialené objekty ako napríklad okolitá krajina, horizont, obloha a oblaky, hviezdy a iné nedosiahnuteľné prvky. Tieto vzdialené objekty sú reprezentované ako kocka s príslušnými textúrami na svojich vnútorných stranách ako je možné vidieť na obrázku 8. Záber kamery aj celá scéna sa nachádza vnútri skybox kocky. Skybox zostáva nehybný aj pri pohybe kamery a zmene záberu.



Obr. 8: Skybox (Away3D, 2012)

Po pridaní takejto kocky do scény je potrebných 6 samostatných textúr pre vrchnú, spodnú, prednú, zadnú časť a dve bočné. Keď už máme takéto okolie, chceme aby sa na lesklých povrchoch objektov v scéne objavoval jeho odraz. To je dosiahnuté takzvanou Environment mapou, ktorá je súčasťou materiálu použitého pre lesklý objekt a je v nej nastavená textúra skyboxu.

Takéto riešenie odrazivosti materiálu je veľmi rýchle, nenáročné a jednoduché. Avšak nevýhodou je, že v odraze sa neobjavujú iné objekty v scéne keďže odraz sa vytvára jedine z Environment mapy. Dynamický výpočet odrazov iných objektov v scéne je oveľa náročnejšia úloha pre spracúvajúce zariadenie. (Parisi, 2014)

3 Analýza a metodika spracovania

3.1 Požadovaná funkčnosť

Aplikácia je tvorená pre firmu Mikoláš - Kamenár, s. r. o., ktorá sa zaoberá návrhom, výrobou a predajom kamenárskych výrobkov. Je potrebné splniť jej potreby a zahrnúť žiadané funkcie do riešenia.

Pri tvorbe stránky a pri jej návrhu je nutné myslieť na jej používateľov a zameranie. Najväčšiu časť predaja firmy tvoria pomníky a sú to zároveň jediné výrobky, ktoré je možné zovšeobecniť a kombinovať do určitých tvarov preto bude táto aplikácia zameraná práve na kombinácie hrobových zostáv. Iné výrobky ako napríklad parapety nie je potrebné vizualizovať a ďalšie je nemožné, lebo sú vždy na mieru podľa miesta určenia, ako napríklad schody alebo krby. Veková štruktúra zákazníkov, ktorí žiadajú hrobové zostavy je vyššia, preto musí byť stránka jednoducho ovládateľná a prehľadná bez zbytočností, ktoré môžu viesť k ďalším otázkam, nedorozumeniam alebo vyššej náročnosti samotnej aplikácie.

Firma požaduje zobrazovanie rôznych druhov hrobového miesta s kombináciou s odlišnými modelmi a tvarmi náhrobného kameňa. Toto je potrebné kombinovať a prezentovať v rôznych druhoch ponúkaného kameňa. Musí byť možné tieto možnosti nezávisle meniť aby zákazník získal predstavu ako konkrétne zostavy spolu vyzerajú alebo len ako prehľad možných kombinácií.

V prezentácii je nutné aj umožnenie pridávania doplnkov: písmo, lampáš a váza. Avšak všetky doplnky budú mať problém s umiestnením v scéne lebo ich poloha závisí od konkrétneho typu vybratej zostavy. Toto by bolo možné vyriešiť preddefinovaním ich polohy v závislosti na vybratých ostatných možnostiach, avšak zákazníci majú často individuálne predstavy, ako napríklad váza v strede platne, ktoré by nebolo možné obsiahnuť. Preto je potrebné objekty v scéne voľne presúvať na požadované miesto.

Po analýze potrebnej funkcionality a technologických kritérií je možné vybrať z možností uvedených v kapitole Technológie pre 3D obsah na webe.

3.2 Vývoj 3D aplikácie vo WebGL

Pre vyššie popísanú prácu s modelmi, interakciu a dynamickú aplikáciu textúr je potrebné vytvorené modely načítavať a zobrazovať na stránke podľa zvolených atribútov. Teda je jasné, že nestačí statická prezentácia s pár 3D efektami ale je potrebná komplexná 3D scéna s vytvorenými modelmi. CSS 3D je v takom prípade nevyhovujúce. Zložitosť takejto aplikácie už vedie k použitiu WebGL a tento výber potvrdzuje aj určenie aplikácie pre lepšiu predstavu zákazníkov či už pri konzultácii, ktorá prebieha v kancelárii, alebo pri zisťovaní vhodnej firmy. V takom prípade je orientácia na prenosné zariadenia zbytočná.

Pred porovnateľným výberom sa ocitol aj (Mazoch, 2014) vo svojej diplomovej práci. Dôvody pre výber WebGL uviedol napríklad otvorenosť štandardu, množstvo

príkladov použitia s dostupným kódom, využívanosť v rôznych odvetviach. Ďalej je WebGL úzko prepojené s HTML 5 a JavaScriptom a z hľadiska budúceho vývoja je to najvhodnejšia voľba, ktorej znalosť bude vysoko žiadaná aj mimo tvorby interaktívnej webovej grafiky.

Získame tak aplikáciu, ktorá bude dostupná bez nutnosti inštalácie prostredníctvom internetu. Oproti bežným stránkam obdobnej tematiky, kde sú k dispozícii iba fotografie ponúkaných modelov alebo hotových prác, teda neponúkajú celistvý náhľad a je možné, že fotografie neobsahujú všetky kombinácie alebo detail, ktorý zákazníka zaujíma.

WebGL ponúka oveľa viac, takže v budúcnosti, pri zistení ďalších potrieb zákazníka nebude problém pridať nové funkcie alebo pri rozšírení ponuky zaradiť nové výrobky do prezentácie.

3.2.1 Modelovenie pomocou Blendera

Z programov uvedených v kapitole Nástroje pre 3D modelovanie je potrebné vybrať vhodný pre spoluprácu s WebGL. Na trhu je programov oveľa viac a konkurencia je veľká. Uvedení sú najrozdielnejší a najpoužívanejší zástupcovia no aj tak nie je nič podstatné čo by jeden z nich vôbec nezvládal a bránilo by to vo vytvorení modelov pre našu aplikáciu. Existencia veľkého počtu programov naznačuje miernu špecializáciu každého z nich, ktorá však nehrá pri základnej tvorbe modelov.

Pri riešení podobného problému uviedol Brunner(2015) vo svojej práci, že pri krátkodobom projekte je možné využiť trialové verzie uvedených programov a teda voľná licencia nehrá rolu. Ak je však potrebné dopĺňať portfólio ponúkaných výrobkov a aktualizovať databázu, je nutné mať 3D program k dispozícii neustále a nie len po dobu tridsiatich dní.

V takom prípade je z finančného hľadiska jasnou výhodou voľná licencia kvôli nezanedbateľným nákladom na zaobstaranie si plnej verzie programov.

Ako linuxový užívateľ pokladám za plus, keď je program dostupný aj pre hlavný operačný systém a nie je potrebné prepínať sa napr. na Windows alebo využívať VirtualBox, ktorý obmedzí dostupný výkon pre modelovací program.

Z dôvodu potreby mať program k dispozícii aj v budúcnosti a možnosti naďalej využívať primárny operačný systém padla voľba na program Blender.

3.2.2 Voľba vhodných technológií 3D grafiky

Keď už vieme čo chceme dosiahnuť potrebujeme vybrať prostriedky, ktorými docielime požadovaný vzhľad a funkcie scény. Kľúčovým prvkom je čo najrealistickejší vzhľad modelu v kombinácii s nízkou náročnosťou aplikácie potrebnou pre fungovanie na väčšine zariadení a neobmedzovanie užívateľského zážitku z aplikácie. Preto je veľmi dôležitá správna kombinácia technológií, kedy sa treba zamerať na časti, ktoré chceme prezentovať a kvôli ktorým tvoríme aplikáciu a zvyšok vynechať.

Osvetlenie scény je treba navrhnúť tak, aby nebolo príliš tmavé, ani svetlé. Tmavá scéna môže byť ťažko čitateľná a skrývať detaily, naopak svetlá je neprirodzená

a taktiež môže skrývať detaily, pretože sa presvetlené objekty zlievajú do seba nie je možné ich vizuálne oddeliť. Správne osvetlenie ukáže výrobok v reálnej podobe.

Z osvetlením prichádza do scény aj tieň, ktorý výrazne zvyšuje celkovú reálnosť. Tiene budú vrhané všetkými objektami a je potrebné ich vytvárať naživo, teda nepoužijeme Light mapy. Dôvod je taký, že sa v scéne nachádzajú objekty, ktoré sú dynamické a je ich možné voľne presúvať, preto by bolo využitie Light máp nevyhovujúce.

Využijeme dva druhy svetla, jedno smerové svetlo, ktoré bude osvetľovať celú scénu a pôsobiť dojmom reálneho dňa a druhé typu reflektor pre generovanie tieňov a zvýraznenie osvetlenia hlavných častí.

Tieto svetlá budú nakonfigurované tak, aby dotvárali a spolupracovali s použitým skyboxom. Ten využijeme pre jeho nenáročnosť a vysokú efektivitu pri simulácii prostredia, ktoré nemá obsahovať zbytočné detaily odvádzajúce pozornosť od produktov ale len vytvárať atmosféru, základ pre osvetlenia a odlesky.

Textúry na modeloch sa musia dynamicky meniť, podľa toho, aký materiál bude vybratý. Preto bude v modeloch uložené samotné UV mapping bez textúry. Po nahratí textúry sa použije na správne miesto. Ďalej bude použitá Bump mapa na miesta modelu z nepravidelného materiálu, ktorého zmenená povrchová geometria by nebola viditeľná pri použití Displacement mapy. Všetky lesklé povrchy budú mať ako Environment mapu nastavený Skybox, dosiahneme tak rýchlu reálnu podobu a simuláciu lešteného povrchu.

3.2.3 Programovanie pomocou Three.js

Pri výbere najvhodnejšej knižnice z uvedených v kapitole 2.1 pre použitie v tejto práci je dobré sa riadiť heslom menej je viac vzhľadom na požadovanú širokú použiteľnosť. Teda vybrať knižnicu, ktorá ponúka všetky potrebné funkcie a výhody, ktoré očakávame, a nie je zbytočne zložitá a prepracovaná.

Je potrebné aby knižnica podporovala všetky plánované technológie 3D grafiky a nebol následne problém z ich implementáciou. Nakoľko sú to ale základné prvky, tak výber knižnice nimi nie je ovplyvnený.

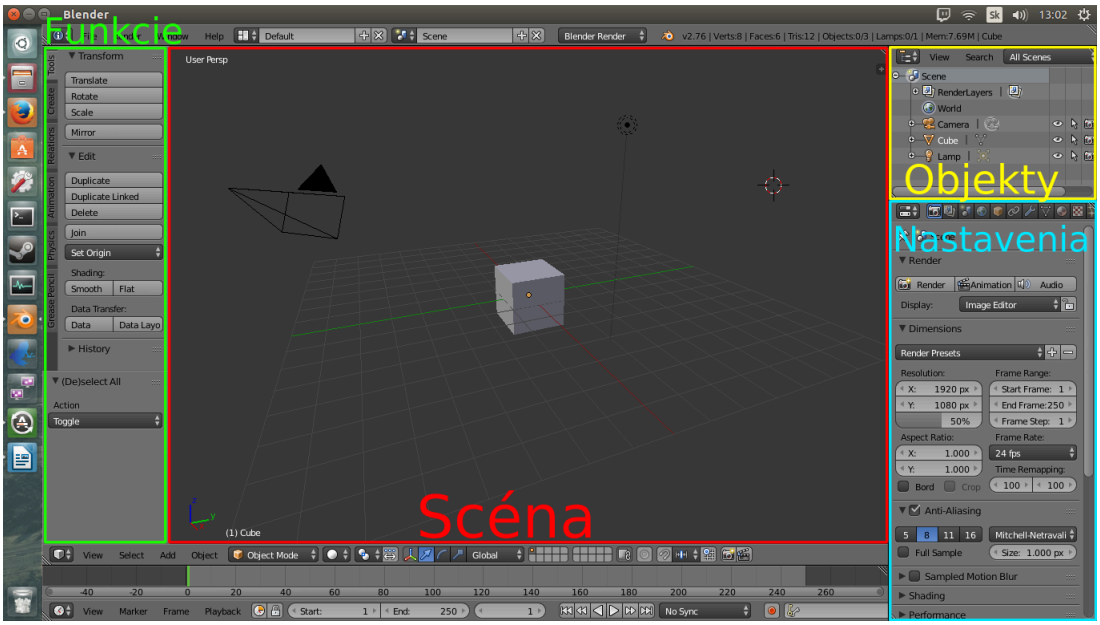
Voľba z úvodného dôvodu padla na Three.js a výber podporujú aj argumenty širokej využívanosti a veľa vyriešených problémov z rôznych oblastí, dobrá dokumentácia a množstvo demonštrácií funkcií na konkrétnych príkladoch s voľne dostupným kódom. Veľká používateľská základňa bola dôvod, prečo si aj (Jelínek, 2014) vybral Three.js vo svojej práci.

4 Vlastná práca

Keď je jasné zadanie, identifikované potreby a forma splnenia, analyzované dostupné technologické možnosti, navrhnuté adekvátne riešenie a metodika spracovania, je možné pristúpiť k samotnému tvoreniu výslednej aplikácie a jej súčastí. To bude obsahovať vymodelovanie objektov, ktoré sa budú zobrazovať, návrh a vytvorenie HTML stránky pre umiestnenie scény s objektami a naprogramovanie aplikácie pre zobrazovanie 3D scény.

4.1 Modelovanie objektov

Základné usporiadanie programu Blender po zapnutí programu je znázornené na obrázku 9.



Obr. 9: Základná obrazovka

Scéna: hlavné okno pre prácu s objektami, vytvárajú sa tu tvary, umiestnenie, orientácia, veľkosť a základná geometria. Okrem nami vytváraných objektov v scéne alebo základných preddefinovaných objektov (v obrázku 9 kocka) sú tu umiestnené v priestore objekty ako napríklad kamera alebo svetlá. Takto si umiestnime kameru na určité miesto z ktorého chceme urobiť render našej scény. Svetlá zase zo svojej pozície vnášajú svetlo do scény a môže sa nám vytvárať tieň.

Funkcie: táto ponuka obsahuje funkcie pre prácu s objektami v scéne alebo vytváranie nových objektov. Sú tu funkcie ako napríklad otoč, zväčši, duplikuj, označ šev, pridaj tvar. Tieto funkcie sa líšia podľa módu, v ktorom sa momentálne program nachádza. K väčšine funkcií existujú klávesové skratky, preto sa užívateľ obvykle zdržuje myšou v oblasti scény.

Objekty: zoznam všetkých objektov, kde je možné s nimi pracovať zoradiť ich podľa rôznych kritérií, podľa príslušnosti k skupine, nastaviť viditeľnosť alebo uzamknúť pri práci.

Nastavenia: obsahuje nastavenia scény a objektov. Najhlavnejšie položky sú nastavenie renderovania, modifikátory, textúry a materiály. Kvalitu výsledného renderu je možné ovplyvňovať podľa potreby, či ide len o kontrolu a teda je dôležitá rýchlosť alebo záverečný produkt, kedy je prvoradá kvalita. Modifikátory umožňujú napríklad zrkadliť objekty, pridávať fyzikálne zákony a podobne. Textúry a materiál upravujú vizuálnu stránku objektu, a simulujú reálne materiály, kde sa môžu nastavovať odlesky, priehľadnosť, drsnosť a iné.

4.1.1 Základné princípy

V scéne budú rôzne druhy objektov, ktoré je treba vytvoriť buď podľa zadaných rozmerov, obrazových predlôh, náčrtov alebo fotografií. Hrobová zostava sa skladá obvykle z obkladu, rámu, platní, podložky, pomníka a doplnkov ako lampáš a váza. Pre lepšiu odolnosť aplikácie je vhodné vkladať obklad, rám a platňu ako jeden objekt, aby sa predišlo nesprávnym kombináciám a tieto objekty sú väčšinou aj tak súčasťou jedného celku. Pri tomto objekte sa bude rozlišovať len typ (jednohrob alebo dvojhrob) a počet platní (žiadna, jedna alebo na dvojhrobe dve). Tento základ je na vymodelovanie relatívne jednoduchý, stačia rozmery a znalosť konštrukcie, no aj tak je treba myslieť na niektoré vecí.

Pri absencii platne alebo pri použití jednej platne na dvojhrob je potrebné pridať zeminu, ktorú je možné vidieť. Je to len rovný štvoruholník, ktorý je ale potrebné presne pomenovať pre budúcu identifikáciu, aby sa mu priradila špeciálna textúra, iná ako zvyšku objektu, keďže má mať iný materiál.

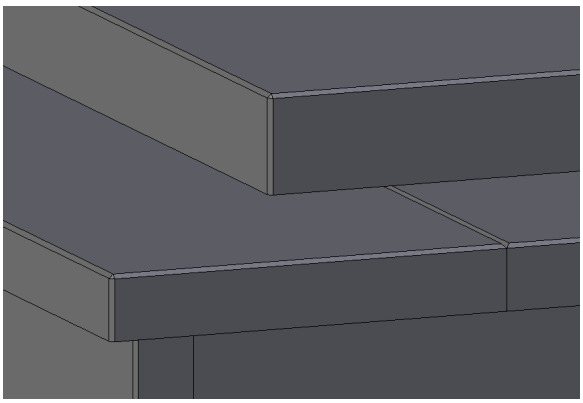
Hrany kameňa sú veľmi precízne opracované a tento prvok je treba zachovať pre dosiahnutie reálneho vzhľadu. Aj keď by si to bežný človek možno hneď nevšimol, ostré hrany by aj tak mohli pôsobiť rušivo. Bližší pohľad na vymodelované hrany je na obrázku 10.

Dodržiavať jednotnú mierku a umiestňovať objekty na ich správne miesto, aby bolo zobrazovanie a nastavovanie vo WebGL čo najjednoduchšie.

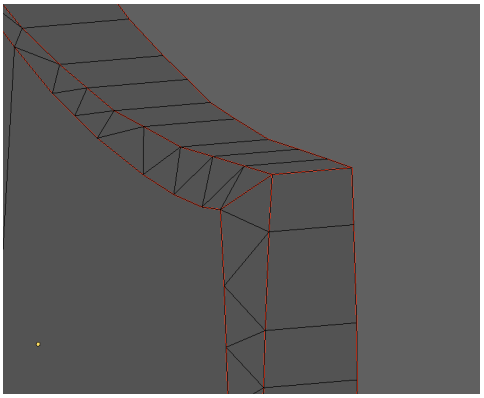
4.1.2 Pomník

Modelovanie pomníka už je zložitejšie hlavne kvôli zachovaniu správnych proporcií pri tvaroch, kedy nie sú k dispozícii rozmery. V takom prípade je potrebné modelovať podľa predlohy, ktorou môže byť fotka alebo náčrtok. Pri zložitejších tvaroch sa samozrejme hodia skúsenosti a znalosť týchto pomníkov, lebo z fotky nie je dobre poznať detaily. Pomník je dobré modelovať spolu s podložkou, na ktorej je umiestnený spolu s lampášom a vázou. Tu tiež je výhodné umiestňovať model na jeho správne miesto v scéne už v modelovacom programe.

Všetky modely je potrebné pripravovať na budúce dynamické aplikovanie textúr, ako bolo spomínané. Teda je treba pripraviť UV mapy, aby sa textúry mapovali



Obr. 10: Hrany modelu

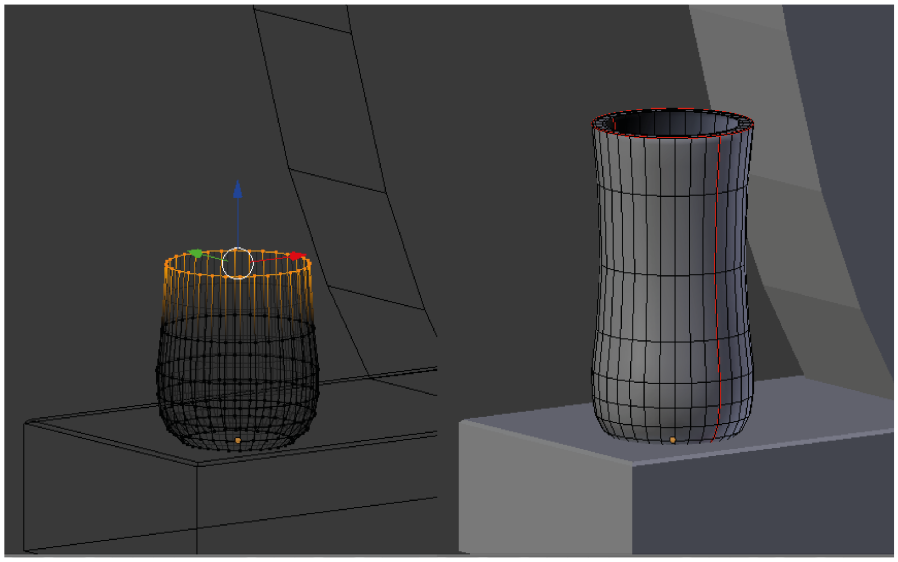


Obr. 11: Šev

správne a na presne určené miesta. Predtým je však nutné označiť na povrchoch objektoch hrany, ktoré majú byť rozrezané aby rozloženie 3D objektu na 2D povrch nebolo deformované a rozbité. Označenie švov (seams) je vidieť na obrázku 11, kde sú zvýraznené červenou farbou.

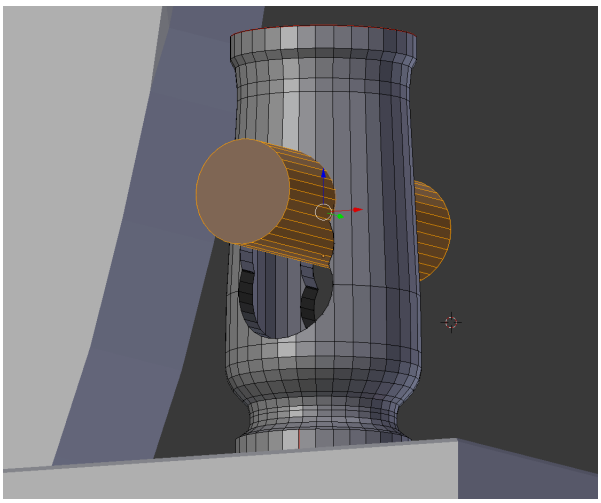
4.1.3 Lampáš a váza

Tieto dva predmety vychádzajú zo spoločného základu, keďže sú väčšinou v sade a teda majú podobný dizajn. Vymodelovať vázu je pomerne jednoduché, začne sa s kruhom a ten sa postupne pomocou nástrojov Extrude a Scale modeluje do požadovaného tvaru. Proces modelovania vázy je na obrázku 12.



Obr. 12: Model vázy

Z vymodelovanej vázy je možné vytvoriť lampáš po miernych úpravách. Zmena je hlavne v tom, že je potrebný otvor, cez ktorý je vidno sviečku a vrchný poklop proti vode, ktorý na váze nie je. Vymodelovanie poklopu je opäť pomocou kružnice a jej postupným upravovaním pomocou rovnakých nástrojov. Otvor vznikne po pridaní valca do scény na správne miesto a prerezaní vázy v miestach prieniku. Po odstránení všetkých nepotrebných zvyškov zostane hotový model. Na obrázku 13 je možné vidieť umiestnenie tretieho valca.



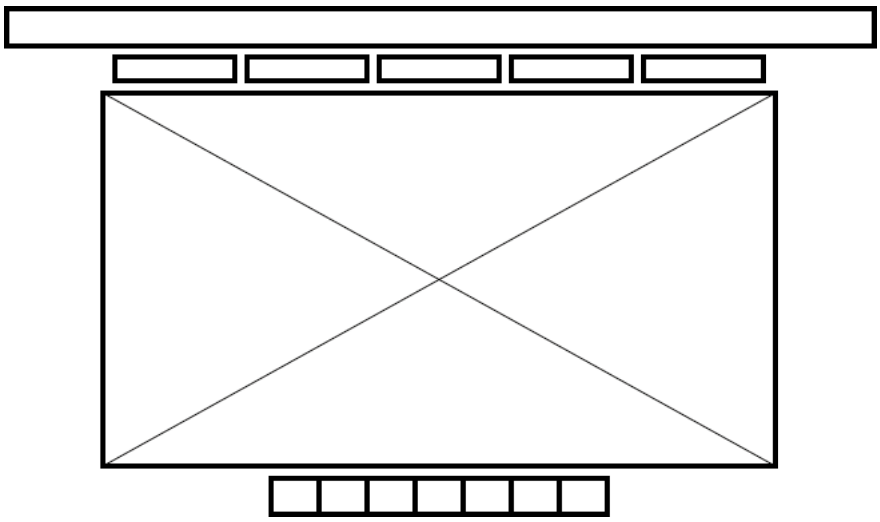
Obr. 13: Modelovanie lampáša

4.2 Uživateľské rozhranie

4.2.1 Layout stránky

Pri návrhu rozloženia stránky je dobré brať do úvahy spôsob práce s aplikáciou. Ovládanie vo WebGL závisí od požiadaviek, obvyklé spôsoby sú myšou alebo tlačidlami na klávesnici. Ovládanie myšou je pohodlné, umožňuje voľný pohľad na scénu a prezeranie si objektu zo všetkých strán a uhlov, preto je dobré ho využiť v aplikácii. Avšak toto ovládanie je ťažšie ak je užívateľ na notebooku alebo mobile. Keďže táto aplikácia nie je závislá na platforme tak aj toto je potrebné premyslieť. Na iných zariadeniach ako PC sa už zo scénou pracuje ťažšie, preto by bolo vhodné pridať ovládanie pomocou tlačidiel, ktoré sa budu nachádzať na samotnej stránke a prehliadanie, otáčanie, približovanie/vzďaľovanie bude jednoduché a rýchle.

Horná lišta by mala slúžiť pre spoluprácu so stránkou mikolas.sk a prípadne v budúcnosti pre výber vizualizácií ďalších výrobkov. Pod tým bude nasledovať sada vysúvateľných ponúk, kde si užívateľ bude vyberať čo sa bude v scéne zobrazovať, aký model, typ kameňa, doplnky a ich farba. Najviac miesta na stránke dostane najdôležitejší prvok, kvôli ktorému sa táto aplikácia tvorí. Scéna sa bude dynamicky meniť podľa výberu vrchných tlačidiel a s pohľadom sa bude pracovať pomocou spodných tlačidiel. Navrhovaný layout stránky je možné vidieť na obrázku 14.



Obr. 14: Layout stránky

Takéto formátovanie stránky potrebuje určité funkčné celky a úpravy, aby sa docielil požadovaný vzhľad. Takisto základné prvky jazyka HTML nepôsobia príliš profesionálne a slúžia len ako základ. Ak sa niekto nevenuje tvorbe stránok a potrebuje použiteľný dizajn bez extrémne originálnych prvkov, tak sú k dispozícii už hotové štýly, ktoré sa použijú a pridajú k html stránke pomocou css súboru. Jedným z najpoužívanejších je Bootstrap, ktorý je vhodný lebo poskytuje pevnú sieť pre vkladanie prvkov a komponenty stránky sú vzhľadovo upravené a pôsobia krajším dojmom.

4.2.2 Dizajn stránky

Dizajn stránky bude jednoduchý, čistý a nebude obsahovať žiadne obrázky ani farby. Dominantou bude prvok pre zobrazovanie WebGL, ktorý poskytuje hlavnú funkcionalitu.

Vrchná navigácia pre spoluprácu s mikolas.sk bude čiernej farby a bude vizuálne predstavovať oddelený celok od zvyšku aplikácie. Dôvod je, že táto lišta nebude spolupracovať s aplikáciu ale bude sa odkazovať mimo aktuálnu aplikáciu alebo na inú stránku.

Vo vysúvateľných ponukách pre volenie konfigurácie bude krátka identifikácia zvolených prvkov, ktoré sa majú pridať do scény aby užívateľ vedel čo zvolil. Ako prvá a základne zvolená bude najpredávanejšia konfigurácia, ktorá sa aj načíta pri spustení stránky.

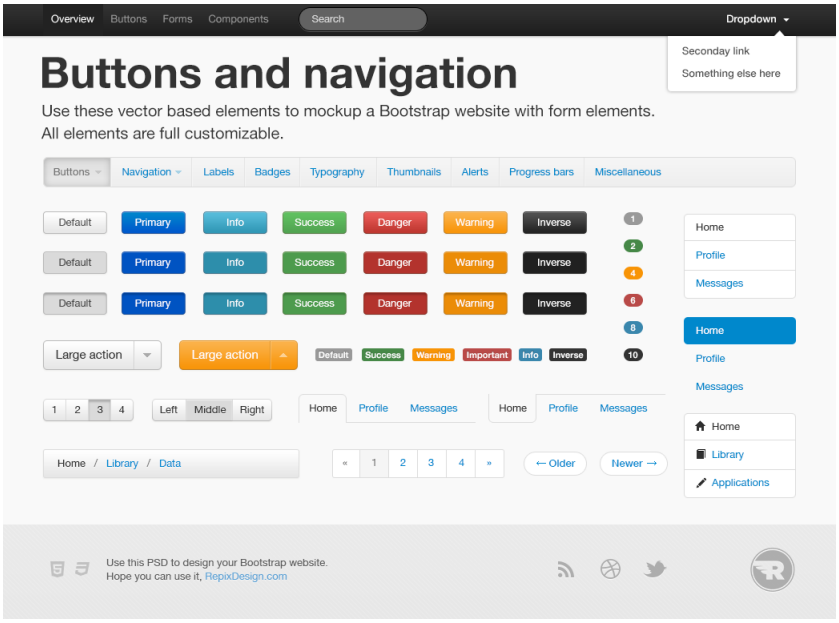
Spodné tlačidlá budú predstavovať jeden celok rozdelený na potrebný počet tlačidiel. Pre prezentáciu funkcionality budú použité piktogramy vystihujúce ich účel. Presný popis funkcií bude uvedený v nápovede.

4.2.3 Použité technológie

Pre stránku, ktorej vzhľad nie je prímárnou náplňou práce, môžeme použiť framework Bootstrap, ktorý výrazne urýchli a zjednoduchší tvorbu GUI.

Je to najpoužívanejší framework, veľmi jednoduchý a rýchlo použiteľný aj pre začiatočníkov. Bol vyvinutý pôvodne pre Twitter ako spôsob pre udržanie jednotného vnútropodnikového štýlu a zjednotenie vzorov ale bol natoľko obľúbený a rýchlo expandujúci, že sa tranformoval na open-source projekt momentálne vyvíjaný malou skupinou ľudí s pomocou obrovského množstva prispievateľov práve pre jeho rozšírenosť a populárnosť. (Bootstrap, 2012)

Základom práce a polohovania prvkov je pevná mriežka rozdelená na 12 časti, pričom je oddielom možné priraďovať potrebný počet pre dosiahnutie výsledného vzhľadu. Mnoho prvkov stránok je predpripravených ako napríklad posuvná galéria alebo sú k dispozícii celé šablóny stránok určené pre blogy, štatistické stránky, portfólio umelcov a podobne. Niektoré hotové prvky je možne vidieť na obrázku 15.



Obr. 15: Bootstrap prvky (bootstrapcss, 2012)

4.2.4 Tvorba užívateľského rozhrania

Používateľ aplikácie bude vidieť a pracovať s webovou stránkou napísanou v jazyku HTML, ktorej jednotlivé prvky budú ovládať a spolupracovať s 3D scénou umiestnenou na tejto stránke.

Navrhnutý layout je teraz potrebné preniesť na stránku. Pomocou Bootstrap frameworku nie je vytvorenie takéhoto usporiadania stránky náročné. Ako v každom HTML kóde sa píšu potrebné prvky v poradí zhora nadol a zoskupenie v riadku sa

vytvorí pomocou `container`. Samotný priestor pre vykresľovanie scény sa pripraví ako oddiel, ktorý sa vyplní prvkom `canvas`. To bude umožňovať aby následne WebGL renderer vykresľoval na tento canvas jednotlivé framy.

4.2.5 Selecty a tlačidlá

Tieto prvky musia prenášať hodnotu do scény podľa toho, čo si užívateľ vybral alebo kam klikol alebo vyvolať v scéne určitú akciu. Pre vysúvateľné ponuky je možné aby vykonali určitú akciu, funkciu zo skriptu na stránke a potom podľa id prvku preniesť jeho hodnotu a pracovať s ňou. Ukážku takejto ponuky je možné vidieť v kóde 1.

Kód 1: Vytvorený HTML select

```
<!--
prvok ma svoje id, pri zmene sa prevedie funkcia change; ako vybratu
    hodnotu, ktoru je mozne zistit ma ponuka cestu k ulozenemu modelu
-->
<select id="model" class="selectpicker" onchange="change('model')">
  <option value="" selected="selected"> </option>
  <option value="models/model1.obj">model 1</option>
  <option value="models/model2.obj">model 2</option>
  <option value="models/model3.obj">model 3</option>
  <option value="models/model4.obj">model 4</option>
  <option value="models/model5.obj">model 5</option>
</select>
```

V skripte sa potom získa jeho hodnota pomocou kódu 2 a je možné napríklad nahrať vybratý model alebo textúru.

Kód 2: Získanie hodnoty zo selectu

```
var e = document.getElementById("model"); //najde sa prvok s id model
var ChosenModel = e.options[e.selectedIndex].value; //hodnota prvku sa
    ulozi do premennej pre dalsie spracovanie
```

Pre jednoduché tlačidlá slúži `EventListener`. Tento prvok vykoná funkciu, keď sa prevedie s určitým tlačidlom vybratá akcia. V kóde 3 je uvedený `EventListener`, ktorý sa aktivuje po kliknutí na tlačidlo s id `out`.

Kód 3: Event Listener

```
document.getElementById("out").addEventListener("click", function(){
  //telo funkcie
});
```

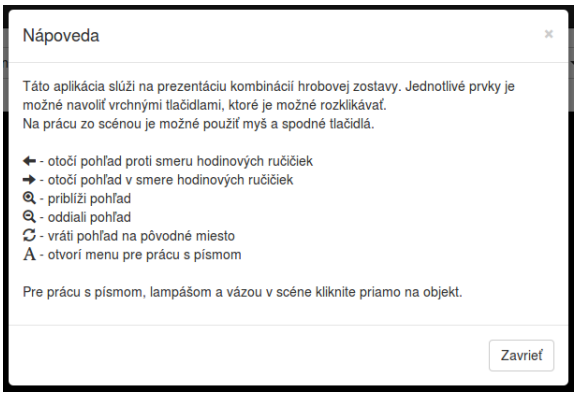
Takéto riešenie je využité pri tlačidlách, ktoré ovládajú scénu, kde je potrebné aby sa vykonala funkcia špecifická pre dané tlačidlo. Pri zmene hodnoty vo vysúvateľných ponukách sa vykoná funkcia, ktorá môže byť použitá aj inde, napríklad pri inicializácii programu, ak chceme mať pri spustení načítaný určitý model, alebo rovnakú funkciu využíva viac prvkov. Pri tlačidlách ovládajúcich scénu je náplň tlačidiel špecifická, ako napríklad otočenie scény alebo priblíženie a táto funkcia sa nikde inde nevyužíva.

Ďalšiu skupinu tlačidiel je potrebné zobrazovať po kliknutí na objekt v scéne, preto začnú na začiatku definované štýlom `document.getElementById('showControls').style.display = 'none';`. To zaručí, že kým nebudú zobrazené tak nezaberajú miesto a po vybratí určitého objektu sa display zmení na `'block'`.

4.2.6 Menu písma a nápoveda

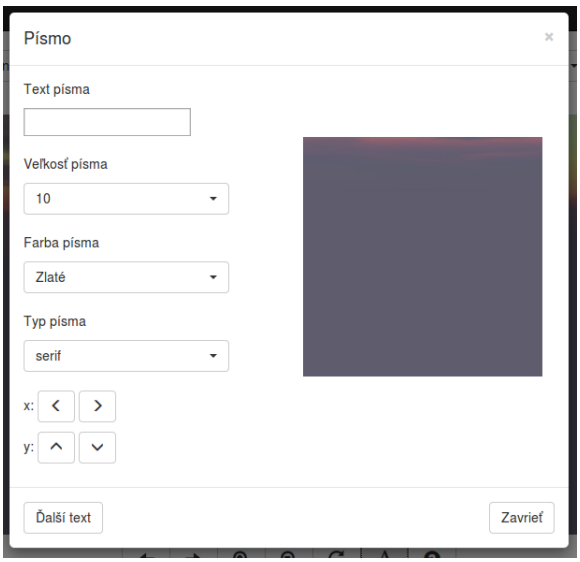
Dve špeciálne menu sa budú zobrazovať po kliknutí na príslušné tlačidlo, inak by bola stránka neprehľadná a zbytočne preplnená.

Jednoduchšie z dvojice je menu nápovedy, ktoré sa len zobrazí a informuje používateľa o funkciách jednotlivých tlačidiel pre ovládanie scény a základného určenia programu. Takéto menu je v Boostrap označené ako `modal` a tlačidlu pre jeho vyvolanie sa priradia atribúty `data-toggle="modal"` a `data-target="#help"`. Vďaka tomu sa následne vyvolá menu s id `help`, ktoré je definované vo vnútri `<div class="modal-dialog"></div>`. Hotové menu je možné vidieť na obrázku 16.



Obr. 16: Menu nápovedy

Prepracovanejšie menu je pre prácu s písmom, ktoré vyžaduje aj zásahy do scény, keďže sa písmo musí objaviť na pomníku. Menu umožňuje pridať text a upravovať jeho veľkosť, farbu, font a umiestnenie. Takéto riešenie bolo dosiahnuté vďaka použitiu ďalšieho prvku canvas, do ktorého sa vykresľuje zvolené písmo. Tento canvas je následne použitý ako textúra pre obdĺžnikový objekt, ktorý sa umiestni do scény na správne miesto. Toto menu obsahuje aj reálny náhľad ako vyzerá zvolený text už aplikovaný na mieste.



Obr. 17: Menu písma

4.3 3D scéna

Posledným a najdôležitejším prvkom aplikácie je 3D scéna, ktorá bude zobrazovať vytvorené modely podľa pokynov stránky. Je naprogramovaná formou skriptu, ktorý je umiestnený priamo v HTML kóde pre stránku.

4.3.1 Vytvorenie scény

Scéna sa bude zobrazovať na prvok canvas vytvorený pomocou HTML aby sa dosiahlo vykreslenie na správne miesto na stránke. Pripravenie miesta na vykresľovanie je uvedené v kóde 4. Šírka a výška tohto prvku bude totožná s rozlíšením vykresľovanej snímky. Tieto rozmery sú uchovávané v premenných WIDTH a HEIGHT a ich veľkosť je priradená podľa zariadenia, na ktorom beží aplikácia. Tým sa zaisťuje správne zobrazovanie scény na rôznych zariadeniach.

Kód 4: Využitie prvku canvas na vykreslenie scény

```
container = document.getElementById( 'webgl-canvas' ); //prirad
    webgl-canvas do premennej container
container.appendChild( renderer.domElement ); //nastav hierarchiu dom
     prvku

//nastav sirku a vysku prvku webgl canvas, zavolaj po funkcii init()
document.getElementById("webgl-canvas").width = WIDTH;
document.getElementById("webgl-canvas").height = HEIGHT;
```

Ako prvá sa následne volá funkcia init(), uvedená v kóde 5, ktorej úlohou je vytvorenie a príprava scény. Pripravenie scény obsahuje kroky a nastavenia potrebné

pre správne zobrazenie a možnosť používania scény ako napríklad pridávanie objektov, animovanie a podobne. Je teda nutné pridať už popísané osvetlenie, vytvoriť a nastaviť renderer, kameru, skybox, spustiť tiene a deklarovať globálne premenné.

Kód 5: Ukážka inicializačnej funkcie

```
function init() {

  scene = new THREE.Scene(); //nova scena pre zobrazovanie objektov

  //nastav sirku a vysku podla uzivatelovho zariadenia
  var WIDTH = window.innerWidth;
  var HEIGHT =window.innerHeight/1.5;

  //vytvor novy WebGL renderer, rozlisenie renderovania rovnake ako pri
      canvas
  renderer = new THREE.WebGLRenderer({alfa:true});
  renderer.setClearColor( 0xffffff );
  renderer.setPixelRatio( window.devicePixelRatio );
  renderer.setSize(WIDTH, HEIGHT);
  renderer.shadowMap.enabled = true; //povol tiene v scene
  renderer.shadowMap.type = THREE.PCFShadowMap;

  //vytvor kameru, pomer stran podla rozlisenia
  camera = new THREE.PerspectiveCamera(30, WIDTH / HEIGHT, 0.1, 3000);
  camera.position.set(50,10,50);

  //vytvor svetla
  var light = new THREE.HemisphereLight( 0xffffff, 0xffffff, 1 );
  var light2 = new THREE.SpotLight(0xffffff, 1);

  //smerove svetlo nema poziciu, iba reflektor
  light2.position.set(-100,40,100);

  //v scene vrha tiene len reflektor
  light2.castShadow = true;
  light2.shadow.camera.Near = 8;
  light2.shadow.camera.Far = 5;
  light2.shadow.camera.Fov = 5;

  //rozlisenie tienov, urcuje kvalitu vykreslenia, moze ovplyvnit vykon
  light2.shadow.mapSize.width = 2048;
  light2.shadow.mapSize.height = 2048;

  //prida svetla do sceny
  scene.add(light);
  scene.add(light2);
}
```

Skybox je teraz možné pridať do scény ako objekt s potrebnou textúrou. Túto akciu vykonáva kód 6.

Kód 6: Vytvorenie skyboxu

```
var r = "img/skybox/"; //cesta do zlozky so skybox texturami

//pridaj meno textury na koniec cesty; 6 textur pre 6 stien kocky
var urls = [
r + "nz.png", r + "pz.png",
r + "pyr.png", r + "ny.png",
r + "px.png", r + "nx.png"
];

//nacitaj textury
textureCube = new THREE.CubeTextureLoader().load( urls );
textureCube.format = THREE.RGBFormat;

//vytvor novy shader pre skybox
var shader = THREE.ShaderLib[ "cube" ];
shader.uniforms[ "tCube" ].value = textureCube;
var material = new THREE.ShaderMaterial( {
  fragmentShader: shader.fragmentShader,
  vertexShader: shader.vertexShader,
  uniforms: shader.uniforms,
  depthWrite: false,
  side: THREE.BackSide
});

//nova geometria kocky s nastavenym materialom
mesh = new THREE.Mesh( new THREE.BoxGeometry( 200, 200, 200 ),
    material );
mesh.position.y = 20;
scene.add( mesh );
```

4.3.2 Pridanie modelov

Keď je scéna vytvorená, je možné začať s ňou pracovať a umiestňovať do nej objekty. Pri prvom spustení sa zavolá funkcia `build()`, ktorá zavolá a vytvorí všetky objekty, ktoré majú byť zobrazované ako základ, teda napríklad najlepšie predávaný alebo najreprezentatívnejší výrobok. Funkcia `build` sa bude volať aj pri zmene textúry objektov, pretože treba objekty celé prechádzať a aplikovať zmenenú textúru každému bodu. Práca tejto funkcie spočíva v tom, že príjme hodnoty zo všetkých vysúvateľných ponúk pre voľbu objektov a pre každú ponuku spustí funkciu `loadOBJ()`, ktorá je uvedená v kóde 7.

Táto funkcia teda príjme parametre z vysúvateľnej ponuky na stránke. Ako hodnota týchto ponúk je nastavená cesta ku geometrii pridávaného objektu. Pod-

ľa toho, ktorá ponuka odoslala túto hodnotu sa nastaví aj meno objektu v scéne. Práve toto meno bude slúžiť ako identifikátor pre prácu s objektom, jeho presun alebo vymazanie.

Kód 7: Funkcia pre načítanie a pridanie objektu do scény

```
//nacitaj objekt, cesta k objektu je parameter geometry, funkcia
   traverse prejde vsetky vrcholy objektu a priradi im texturu
function loadOBJ( geometry, name ) {
  loader.load( geometry, function( object ){
    object.traverse( function (child) {
      if ( child instanceof THREE.Mesh ) {
        child.material.map = map;
        child.material.envMap = textureCube; //Environment mapa
        child.castShadow = true;
        child.receiveShadow = true;
        child.material.needsUpdate = true;
      }
    });

    //specialne textury su aplikovane pre objekt s nazvom "zem"
    if (object.children[0].name == "zem") {
      var newMaterial = new THREE.MeshPhongMaterial();
      var mapZem = new THREE.TextureLoader().load( "img/ground.jpg" );
      newMaterial.map = mapZem;
      newMaterial.bumpMap = mapZem; //Bump mapa
      newMaterial.bumpScale = 1.2;
      newMaterial.envMap = null;
      newMaterial.shininess = 0;
      object.children[ 0 ].material = newMaterial;
      newMaterial.color = new THREE.Color( 0x999999 );
    }

    //prirad objektu nazov a pridaj ho do sceny
    object.name = name;
    scene.add( object );
  });


}
```

Súčasťou `loadOBJ()` je aj funkcia `traverse`, ktorá vykonáva spomínané prechádzanie objektu a priraďovanie textúr, priradenie Environment mapy alebo nastavenie tieňov. Zvláštna vetva sa vykonáva pre objekt s názvom `"zem"`, ktorý zobrazuje hlinu a teda má inú textúru, nemá Environment mapu ale Bump mapu, ktorá upravuje vzhľad jeho geometrie.

4.3.3 Výber modelu

V prípade potreby práce s modelom, ktorý je už pridaný do scény, je potrebné vybrať želaný model. Najjednoduchšie pre používateľa je prosté kliknutie na objekt. Avšak tento úkon vôbec nie je jednoduchý z hľadiska 3D grafiky. Vyberanie objektu v 3D priestore kliknutím na obrázok scény, ktorý je vykreslený dvojrozmerne vyžaduje určité spracovanie. Toto spracovanie je uvedené v kóde 8.

Ako prvé je potrebné zistiť, na ktoré konkrétne miesto užívateľ myšou klikol. To sa zistí podľa polohy kurzora v momente stlačenia tlačidla myši. Od hodnôt x a y je ale nutné odpočítať prípadný odstup, ktorý má canvas pre vykresľovanie scény od krajov stránky.

Ďalej je nutné vyslať lúč prechádzajúci scénou, ktorý vychádza z pozície kamery a prechádza bodom zisteným v predošlom kroku. Následne sa uložia všetky objekty, ktorými tento lúč prešiel. Jeho trajektória je priama, neláme sa, neodráža, nie je žiadnym spôsobom ovplyvnený materiálom ani svetlom. Ak je takýto uložený objekt aspoň jeden, vyberie sa prvý z radu a to je ten objekt, na ktorý užívateľ klikol a môže sa vykonať požadovaná akcia

Kód 8: Výber objektu zo scény

```
renderer.domElement.addEventListener( 'mousedown', function ( event ) {
        //zisti aktualnu poziciu x,y kurzora mysi na obrazovke
        var mouseX = event.clientX - canvasPosition.left;
        var mouseY = event.clientY - canvasPosition.top;

        event.preventDefault();

        //preved poziciu zo suradnicoveho systemu s nulovou poziciou v
            lavom hornom rohu do systemu s nulovou poziciou v strede
            obrazovky
        var mouse = new THREE.Vector2( (( mouseX) / WIDTH )* 2 - 1,-((
            mouseY) / HEIGHT) * 2 + 1);

        //vysli luc z pozicie kamery do pozicie kurzora mysi
        var raycaster = new THREE.Raycaster();
        raycaster.setFromCamera( mouse, camera);

        //uloz objekty pretate lucom
        var intersects = raycaster.intersectObjects( objects, true );
        if ( intersects.length > 0 ) {
          //akcia ak sa podari vybrat objekt
        }
}
```

4.3.4 Pridanie písma

Menu pre pridanie písma bolo už uvedené v časti Užívateľské rozhranie, avšak je ešte potrebné, aby sa podľa navolených atribútov toto písmo aj pridalo do scény na správne miesto.

Písmo je potrebné pridať do scény v troch po sebe idúcich krokoch.

1. Vytvoriť canvas, ktorý obsahuje písmo podľa jednotlivých atribútov zvolených užívateľom, možnosť vidieť v kóde 9.
2. Nastaviť vytvorený canvas ako textúru nového materiálu.
3. Vytvoriť nový objekt, dvojrozmernú plochu s materiálom vytvoreným v druhom kroku a pridať do scény.

Kód 9: Vytvorenie prvku canvas pre písmo

```
function fillCanvas() {
    //vytvor canvas ak neexistuje
    if (document.getElementById('canvas')==null) {
        var canvas = document.createElement('canvas');
  } else {
    var canvas = document.getElementById('canvas');
  }

  //prirad vytvaranemu textu uzivatelom zvolene parametre
  var vyska = document.getElementById("size").value;
  var font = document.getElementById("font").value;
  var size = 256;
  canvas.width = size;
    canvas.height = size/8;
  var context = canvas.getContext('2d');
  var txt = document.getElementById("myText").value;
  context.font = vyska+"pt␣"+font;

  //vykresli text do stredu prvku canvas
  context.textAlign = "center";
  context.textBaseline = "middle";
  context.fillStyle = document.getElementById("colour").value;
  context.fillText(txt, size/2, size/16);
}
```

S takýmto objektom je už teraz možné pracovať, umiestniť na správne miesto, presúvať alebo vymazať.

Náhľad na pomník v menu písma je riešený vytvorením nového renderera a kamery. Renderer má rozmery malého náhľadu a kamera pozíciu nastavenú priamo pred pomníkom pre správny záber. Pri vyvolaní menu písma sa premennej `getImageData` nastaví hodnota `true` a funkcia `animate()`, uvedená v kóde 10, zodpovedná za vykresľovanie a tvorbu jednotlivých framov začne vykresľovať aj druhý obraz.

Kód 10: Vykresľovanie dvoch rôznych obrazov

```
function animate() {
    //rekurzivna funkcia pre neprestajne vykreslovanie
  requestAnimationFrame(animate);

  camera.lookAt(new THREE.Vector3 (0.0, 5.0,0.0));
  renderer.render(scene, camera);

  //getImageData je pravda ked sa vykresluje druhy canvas v menu pisma
  if(getImageData == true){
        textureCamera.lookAt(new THREE.Vector3 (0.0, 7.0,-10.0));
        rendererTexture.render(scene, textureCamera);
        imgData = rendererTexture.domElement.toDataURL(); //prevedenie
           obsahu vykreslovaneho obrazu do premennej
        getImageData = false;
  }
  controls.update();
}
```

4.3.5 Ovládanie scény

Pre umožnenie ovládania scény pomocou myši je potrebné pridať rozšírenie OrbitControls príkazom . Potom je možné voľne prezerať scénu, približovať, oddaľovať a otáčať. Volné prezeranie však môže viesť ku kolíziám s objektami, prechádzanie cez steny a podobne. Preto je vhodné obmedziť rozsah pohybu pre užívateľa. To sa zabezpečí príkazmi uvedenými v kóde 11. Konkrétne sa obmedzí, že kamera sa nemôže dostať pod úroveň zeme a rozsah priblíženia.

Kód 11: Obmedzenie pohybu kamery

```
var controls = new THREE.OrbitControls(camera, renderer.domElement);
controls.maxPolarAngle = (Math.PI/2)-0.1; //maximalny uhol otocenia
    kamery
controls.minDistance = 20; //minimalne priblizenie
controls.maxDistance = 100; //maximalne priblizenie
```

Ovládanie scény pomocou spodných tlačidiel je však už zložitejšie. Pre každé tlačidlo sa musí vytvoriť `eventListener` , ktorý čaká na akciu s tlačidlom. Priblíženie a oddialenie scény využíva OrbitControls. Vrátenie pohľadu na pôvodné miesto nastaví kamere pôvodnú pozíciu.

Náročnejšími na realizáciu sú tlačidlá otáčania scény. Tie majú za úlohu viesť kameru po kružnici zo stredom zhodným zo stredom scény a s polomerom aktuálnej vzdialenosti ku stredu. V kóde 12 sú využité funkcie sínus a kosínus a ich schopnosť opisovať kružnicu. Číslo `step` určuje veľkosť kroku pri otočení.

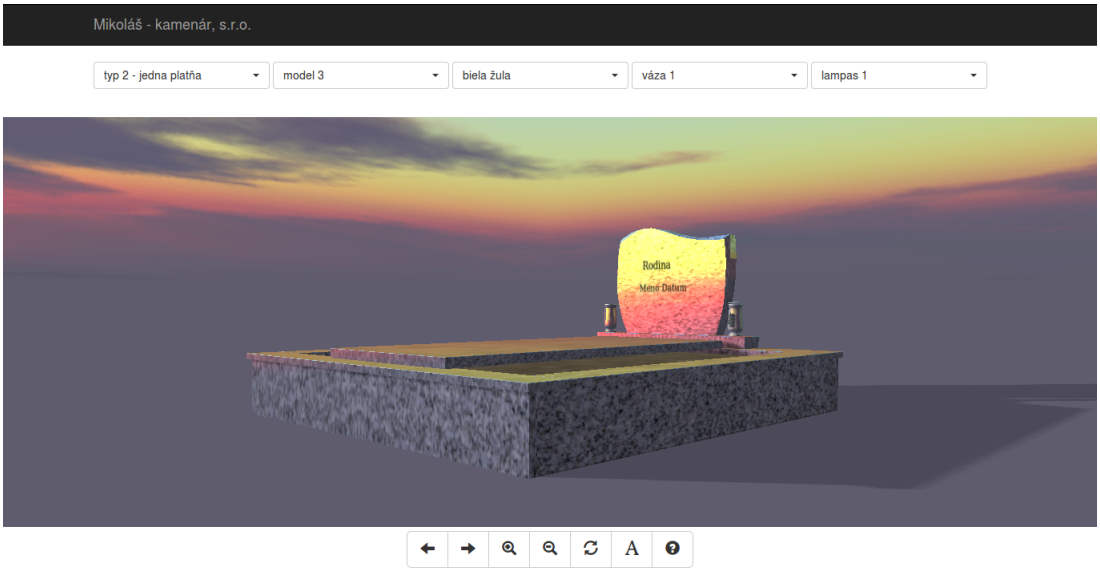
Kód 12: Otáčanie scény tlačidlom

```
document.getElementById("right").addEventListener("click", function(){
  var step = 0.1;
  //actualna pozicia je ulozena
  var x = camera.position.x;
  var y = camera.position.y;
  var z = camera.position.z;

  //pohyb po kruznici
  var moveX = x * Math.cos(step) + z * Math.sin(step);
  var moveZ = z * Math.cos(step) - x * Math.sin(step);

    //kamere je priradena nova pozicia
  camera.position.set(moveX, y, moveZ);
  camera.updateProjectionMatrix();
  camera.lookAt(scene.position);
});
```

Týmito úpravami a nastaveniami je scéna hotová a pripravená na použitie. Spolupracuje so stránkou, zobrazuje modely, umožňuje ich prezeranie a kombinovanie. Je možné pridať písmo a v prípade potreby je možné vybrať objekt scény, presúvať ho a vymazať. Výsledný vzhľad aplikácie je na obrázku 18.



Obr. 18: Výsledná aplikácia

4.4 Testovanie

Výkon hotovej aplikácie je potrebné otestovať na rôznych počítačových zostavách pre zistenie schopností a prípadných užívateľských obmedzení.

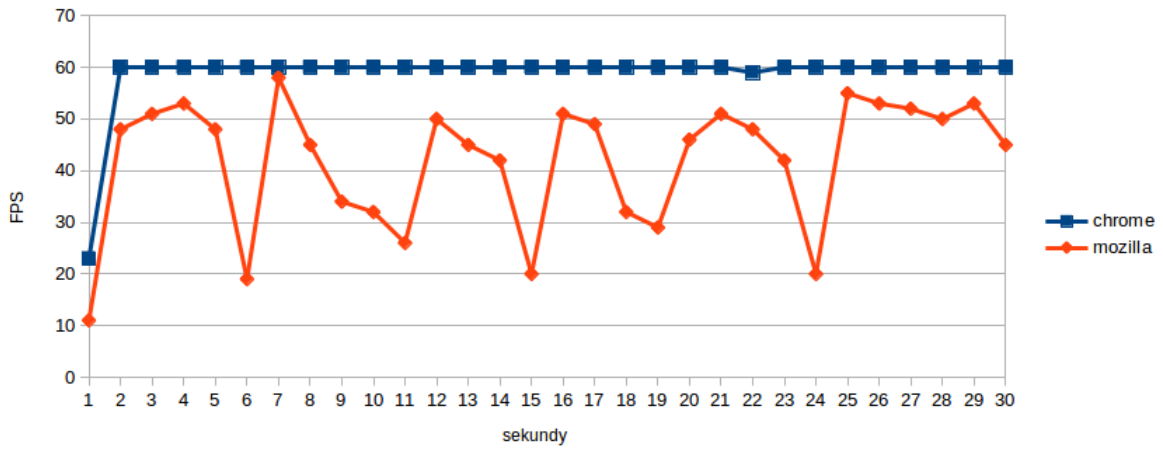
Testovanie prebiehalo na základe merania FPS (obrázkov za sekundu) po dobu tridsiatich sekúnd, vrátane načítavania stránky.

4.4.1 Testovacia zostava 1

Prvým zariadením pre testovanie bol notebook s dvojicou operačných systémov - Windows 7 Ultimate 64-bit a Linux Ubuntu 16.04 LTS 64-bit. Parametre notebooku sú:

- procesor Intel i7-3632QM 2.20GHz
- RAM 8GB
- GPU Intel HD Graphics 4000

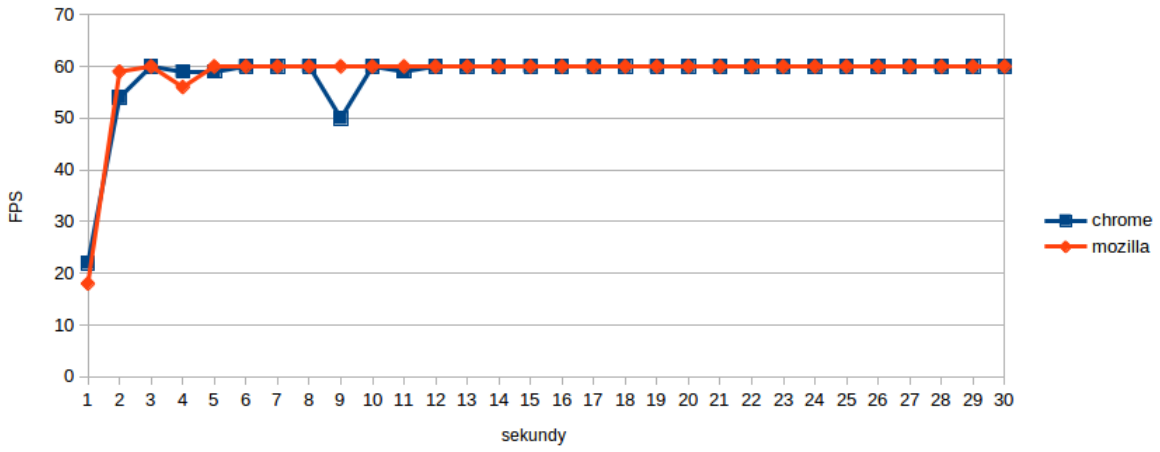
Ide o relatívne výkonný notebook, starý 3 roky, s grafickým čipom integrovaným v procesore.



Obr. 19: Výkon na testovacej zostave 1, OS Ubuntu

Na operačnom systéme Ubuntu boli dva webové prehliadače, Firefox verzia 46.0.1 a Chrome verzia 50.0.2661.102. Ich dosiahnutý výkon je zobrazený na obrázku 19. Je vidieť že Firefox dosiahol značne nižší počet FPS a nestabilný priebeh, ktorý sa už na inom operačnom systéme neopakoval.

Výkon na systéme Windows 7 sa meral opäť pomocou dvoch prehliadačov, Firefox verzia 46.0.1 a Chrome verzia 50.0.2661.102. Z obrázku 20 je vidieť že už oba prehliadače dosahovali porovnateľný výkon. Rozdielny výsledok rovnakých prehliadačov na Ubuntu a Windowse 7 môže mať za následok odlišný ovládač grafického čipu zariadenia.



Obr. 20: Výkon na testovacej zostave 1, OS Windows 7

4.4.2 Testovacia zostava 2

Stolný počítač s nainštalovaným systémom Windows 7 Ultimate 32-bit a nasledujúcimi parametrami:

- procesor Intel Pentium Dual-Core 2.5GHz
- RAM 2GB
- GPU NVIDIA GeForce 8600GT

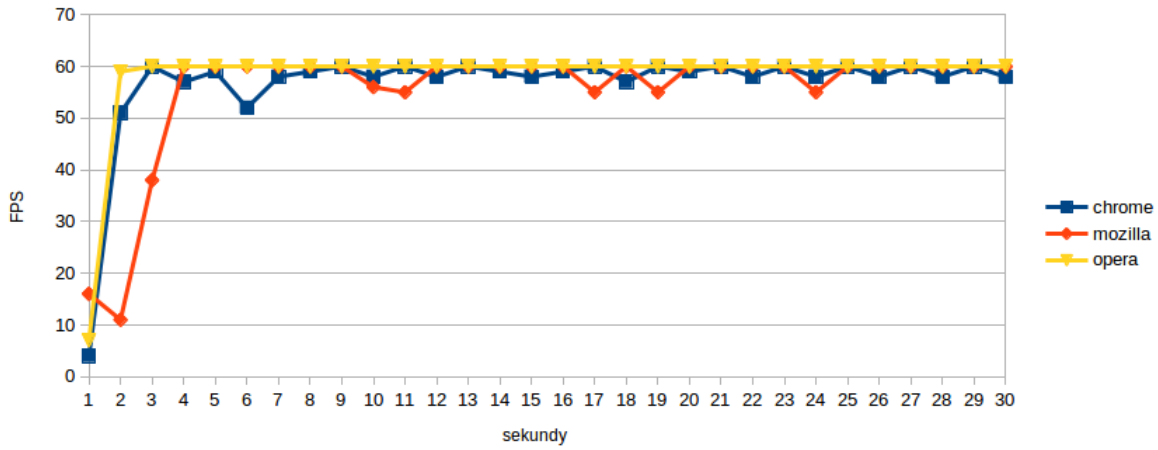
Počítač je starý približne 8 rokov s plnohodnotnou grafickou kartou.

Na zariadení bola trojica webových prehliadačov: Firefox verzia 46.0.1, Chrome verzia 50.0.2661.102 a Opera verzia 37.0.2178.43. Pri dostupnosti bola do porovnania zahrnutá aj Opera, ale išlo skôr o test, či sa bude správať rovnako ako Chrome, pretože zdieľajú rovnaký engine. Výsledok meraní je možné vidieť na obrázku 21 Zaujímavosťou je, že Opera je stabilnejšia ako Chrome.

Počas testovania sa pracovalo so scénou a pridávali sa modely. Je vidieť, že na priebeh výkonu to malo minimálny vplyv.

Ďalšie testy prebehli na dvoch starších notebookoch s operačným systémom Windows Vista. Pri prehliadači Google Chrome, ktorý už pre Vistu neposkytuje najnovšie verzie bolo dosiahnutých 10 FPS. Na aktuálnej Mozille Firefox sa aplikácia nespustila vôbec.

Pre ucelenosť testovania test prebehol aj na mobilnom zariadení Huawei G700 s operačným systémom Android 4.2.1. Na prehliadači Google Chrome aplikácia nefungovala správne a na Mozille Firefox bolo dosiahnutých stabilných 11 FPS.



Obr. 21: Výkon na testovacej zostave 2

4.4.3 Zhodnotenie testovania

Pre bezproblémové použitie pri najnovších ovládačoch grafického čipu zariadenia a najnovšej verzii webového prehliadača je odporúčaný operačný systém Windows 7 a vyšší. To predstavuje 76,5 % všetkých počítačov a 96,96 % počítačov s OS Windows. (W3Schools, 2016)

5 Záver

Testovanie aplikácie ukázalo použiteľnosť bez obmedzení na väčšine počítačov a možnosť spustenia aplikácie aj na výkonnejšom smartfóne. Je teda možné využitie v praxi. Majorita malej testovacej vzorky užívateľov aplikácie uviedla po použití väčšiu spokojnosť s výberom a jednoduchšie rozhodovanie.

Najväčším prínosom práce je možnosť názornej ukážky produktov zákazníkom. Pre človeka je náročné si predstavovať niečo čomu nerozumie a nepozná výrazy a termíny použité napríklad pri vytváraní objednávky. S aplikáciou je možnosť ukázať všetky prvky, riešenia, ponúkané modely, typy kameňa a popísať ich, prípadne hneď aj zodpovedať otázky zákazníka. Po dohodnutí a výbere je možné vytvoriť vizualizáciu už s konkrétnymi nápismi a teda verne simulovať výsledný produkt. Môže sa predísť problémom, že neladí farba písma s typom kameňa alebo iným, ktoré by sa objavili až keď by bolo neskoro.

Webová dostupnosť aplikácie môže mať pozitívny vplyv na počet zákazníkov, keďže sa rozširuje zhromažďovanie informácií pomocou internetu a teda sa osloví nová skupina potenciálnych záujemcov. Takisto sa rozšíri aj plošný rozsah oslovenia práve vďaka širokej dostupnosti a ľudia z iného kraja alebo mesta nenavštevujúci miesto pôsobenia firmy si budú môcť prezrieť ponúkané výrobky.

Budúci rozvoj aplikácie bude hlavne v rozširovaní databázy zobrazovaných modelov. Bolo by vhodné pridať produkty, ktoré nie sú v ponuke firmy, ale nie je problém ich vyrobiť na prianie zákazníka. To by mohlo zvýšiť všeobecný dopyt a obzvlášť pre jedinečné a originálne riešenia, keď by ich zákazníci mali možnosť vidieť a boli by o nich informovaní. Firma by sa teda mohla orientovať viac na zákazkovú výrobu na mieru a ponuku jedinečných výrobkov, lebo nie je problém z ich prezentáciou. Takéto modely sa dajú vytvoriť jednoducho a rýchlo.

Ďalší postup by sa odvíjal od názorov a referencií na aplikáciu po uvedení do prevádzky a podľa toho by sa pridávali alebo upravovali potrebné prvky a riešenia.

6 Zoznam zdrojov

ANYURU, A. *Professional WebGL Programming: Developing 3D Graphics for the Web*. New Jersey: John Wiley & Sons, 2012. 361 s. ISBN 978-1-119-96886-3.

AUTODESK *Overview* [online]. 2016 [cit. 2016-03-09]. Dostupné z: http://www.autodesk.com/products/3ds-max/overview.

AWAY3D *Using a Skybox* [online]. 2012 [cit. 2016-04-27]. Dostupné z: http://away3d.com/tutorials/Using_A_Skybox.

BABYLONJS *WebGL. Simple. Powerful.* [online]. 2013 [cit. 2016-04-6]. Dostupné z: http://www.bablonjs.com.

BLENDER *About* [online]. 2016 [cit. 2016-03-10]. Dostupné z: https://www.blender.org/about/.

BLENDERGURU *Basics of Realistic Texturing* [online]. 2015 [cit. 2016-04-27]. Dostupné z: https://www.blenderguru.com/tutorials/basics-realistic-texturing/.

BOOTSTRAP *About* [online]. 2012 [cit. 2016-05-04]. Dostupné z: http://getbootstrap.com/about/.

BOOTSTRAPCSS *Bootstrap CSS* [online]. 2012 [cit. 2016-05-04]. Dostupné z: http://www.bootstrapcss.com/.

BRUNNER, A. *Implementace 3D interaktivní animace do webového portálu pro prezentaci produktů ve vinařství.*Brno, 2015. Bakalářská práce. Mendelova univerzita v Brně, Provozně ekonomická fakulta. Vedoucí práce Ing. Jaromír Landa, Ph.D..

CANIUSE *WebGL - 3D Canvas graphics* [online]. 2016 [cit. 2016-03-08]. Dostupné z: http://caniuse.com/#feat=webgl.

CANIUSE *CSS3 3D Transforms* [online]. 2016 [cit. 2016-03-08]. Dostupné z: http://caniuse.com/#feat=transforms3d.

CANIUSE *Canvas (basic support)* [online]. 2016 [cit. 2016-03-08]. Dostupné z: http://caniuse.com/#feat=canvas.

DANCHILLA, B. *Beginning WebGL for HTML5*. New York: Apress, 2012. 356 s. ISBN 978-1-4302-3996-3.

DIRKSEN, J. *Learning Three.js - the JavaScript 3D Library for WebGL - Second Edition*. Birmingham: Packt Publishing, 2015. 422 s. ISBN 978-1-78-439221-5.

EBERT *Texturing & Modeling: A Procedural Approach* Morgan Kaufmann, 2003. 687 s. ISBN 9781558608481.

Chopine, A. *3D Art Essentials*. Boston: Focal Press, 2011. 288 s. ISBN 978-0-240-81471-1.

Intel *3D Lighting in Softimage* [online]. 2011 [cit. 2016-05-03]. Dostupné z: https://software.intel.com/en-us/articles/3d-lighting-in-softimage.

Jelínek, P. *Portlet pro 3D vizualizaci časových řad*. Brno, 2014. Diplomová práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce RNDr. Radek Ošlejšek, Ph.D..

Khronos *WebGL* [online]. 2014a [cit. 2015-12-18]. Dostupné z: https://www.khronos.org/webgl/.

Khronos *Getting Started* [online]. 2014b [cit. 2015-12-18]. Dostupné z: https://www.khronos.org/webgl/wiki/Getting_Started.

Khronos *The Standard for Embedded Accelerated 3D Graphics* [online]. 2016a [cit. 2016-04-05]. Dostupné z: https://www.khronos.org/opengles/.

Khronos *Vulkan* [online]. 2016b [cit. 2016-05-02]. Dostupné z: https://www.khronos.org/vulkan/.

Kříž, J. *Mistrovství v 3ds Max*. Brno: Computer Press, 2010. 1152 s. ISBN 9788025124642.

Mazoch, B. *Interaktivní vizualizace 3D objektů pro webové prostředí a její využití v oblasti e-learningu*Brno, 2014. Diplomová práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce doc. Ing. Michal Brandejs, CSc..

Microsoft *Shader Stages* [online]. 2014 [cit. 2016-04-30]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/bb205146%28v=vs.85%29.aspx.

Moya, V. a kol. *A single (unified) shader GPU microarchitecture for embedded systems.* In High Performance Embedded Architectures and Compilers. Springer Berlin Heidelberg, 2005. 286-301..

Open.gl *The graphics pipeline* [online]. 2012 [cit. 2016-05-03]. Dostupné z: https://open.gl/drawing.

OpenGL *Vertex Specification* [online]. 2015a [cit. 2016-04-04]. Dostupné z: https://www.opengl.org/wiki/Vertex_Specification#Vertex_Buffer_Object.

OpenGL *History of OpenGL* [online]. 2015b [cit. 2016-04-05]. Dostupné z: https://www.opengl.org/wiki/History_of_OpenGL.

OpenGL *Rendering Pipeline Overview* [online]. 2015c [cit. 2016-04-30]. Dostupné z: https://www.opengl.org/wiki/Rendering_Pipeline_Overview.

PARISI, T. *Programming 3D applications with HTML5 and WebGL*. Beijing: O'Reilly, 2014. 384 s. ISBN 978-1-44-936296-6.

PLAYCANVAS *Beautiful interactive experiences for every platform* [online]. 2015 [cit. 2016-04-06]. Dostupné z: https://playcanvas.com/.

RHINO3D *Features* [online]. 2016 [cit. 2016-03-10]. Dostupné z: https://www.rhino3d.com/features.

THREEJS *Creating a scene* [online]. 2013 [cit. 2015-12-18]. Dostupné z: http://threejs.org/docs/#Manual/Introduction/Creating_a_scene.

WIKIPEDIA *Lightmap* [online]. 2011 [cit. 2016-05-03]. Dostupné z: https://en.wikipedia.org/wiki/Lightmap.

W3SCHOOLS *OS Platform Statistics and Trends* [online]. 2016 [cit. 2016-05-08]. http://www.w3schools.com/browsers/browsers_os.asp.

ŽARA, J. A KOL. *Moderní počítačová grafika. 2. vyd.*. Brno: Computer Press, 2004. 612 s. ISBN 80-251-0454-0.

Prílohy

A Obsah CD

CD priložené k bakalárskej práci obsahuje:

- súbory aplikácie (zdrojové kódy, modely, textúry)
- prácu v elektronickej podobe