VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# 3D ROZHRANÍ PRO WEBOVÉ STRÁNKY

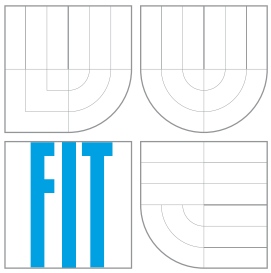
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR KLEPÁRNÍK

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# 3D ROZHRANÍ PRO WEBOVÉ STRÁNKY

3D INTERFACE FOR WEB PAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR KLEPÁRNÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ISTVÁN SZENTANDRÁSI

BRNO 2012

# Abstrakt

Tato práce diskutuje možnosti vytváření 3D webových stránek. Zaměřuje se především na akcelerované vykreslování 3D scény v reálném čase ve webovém prohlížeči s využitím webové grafické knihovny WebGL. Dále se zabývá návrhem 3D webového uživatelského rozhraní a implementací webové stránky prostřednictvím JavaScriptu(WebGL). V závěru práce jsou vyhodnoceny uživatelské testy rozhraní a celkově dosažené výsledky.



# Abstract

This thesis discusses the possibility of creating 3D Web sites. It primarily focuses on accelerated real time rendering of 3D scenes in a web browser. It uses a web-based graphics library WebGL. This thesis also deals with design of 3D user interface, and implementation of Web pages via JavaScript (WebGL). Finally, user assessments on the proposed implementation are evaluated and the overall results are discussed.



# Klíčová slova

WebGL, 3D uživatelské rozhraní, GUI, webové rozhraní, HTML5, JavaScript

# Keywords

WebGL, 3D user interface, GUI, web interface, HTML5, JavaScript



# Citace

Petr Klepárník: 3D rozhraní pro webové stránky, bakalářská práce, Brno, FIT VUT v Brně, 2012

# 3D rozhraní pro webové stránky

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Istvána Szentandrásiho. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Klepárník
14. května 2012

## Poděkování

Tímto bych chtěl poděkovat vedoucímu své bakalářské práce, Ing. Istvánu Szentandrásimu, za pomoc, připomínky, nápady a bezproblémovou komunikaci.

# Obsah

# Seznam obrázků

# Kapitola 1

# Úvod

Webové technologie jsou aktuálně jednou z nejvíce se rozvíjejících oblastí informačních technologií. V dnešní době, kdy je internet pro mnohé již nepostradatelnou součástí běžného života, vyžadují uživatelé stále více aplikací dostupných na internetu, tedy prostřednictvím webového prohlížeče.

Grafické zpracování webové prezentace je velmi důležité. Prostřednictvím obrazu dokážeme sdělovat daleko více informací než pouhým textem. Pokud navíc umožníme vykreslování grafiky v reálném čase (video, animace), množství předaných informací nespočetněkrát zvýšíme.

Hlavní rozdíl mezi klasickými desktopovými aplikacemi a webovými, je právě ve formě uživatelského rozhraní. Vývoj má tendenci vytvářet pro oba typy aplikací stejně vyspělé rozhraní. Týká se to především her využívajících 3D akcelerovanou grafiku. Vytvořit 3D uživatelské rozhraní na webu již dnes není, díky stále se zdokonalujícím technologiím, takový problém jako dříve.

V rámci této práce budu pojednávat o aktuálních možnostech vykreslování 3D grafiky ve webovém prohlížeči (Kapitola 2). Zmíním se o v současnosti dostupných technologiích a hlavním tématem bude akcelerované vykreslování s využitím knihovny WebGL. Od Kapitoly 3 se zaměřím převážně na návrh a implementaci 3D webové prezentace. Tématem návrhu bude interaktivní 3D web rockové kapely, netradiční rozhraní, experimentování. Rozeberu koncept a 3D prvky navrhovaného uživatelského rozhraní. Popíši klíčové části implementace vlastního WebGL frameworku (4) (načítání 3D modelů, animace, mapování textur, začlenění HTML obsahu...), a také problémy, se kterými jsem se setkal. Závěr práce věnuji testování aplikace běžnými uživateli a vyhodnotím celkově dosažené výsledky.

# Kapitola 2

# Možnosti vykreslování 3D grafiky ve webovém prohlížeči

Významnou součástí informačních technologií je počítačová 3D grafika, která modeluje reálné i nereálné objekty na dvojrozměrné zobrazovací zařízení tím způsobem, že se nám jeví jako trojrozměrné. Náročná 3D grafika vyžaduje pro vykreslování hardwarovou akceleraci, kterou zajišťují moderní grafické karty.

## 2.1 Dostupné technologie

Společnosti poskytující internetové prohlížeče reší problém vykreslování 3D ve webovém prohlížeči. Bylo vytvořeno již mnoho pluginů, které dokáží reprezentovat 3D grafiku, ale většina z nich nevyužívala hardwarovou akceleraci. Tyto způsoby řešení mají také podstatnou nevýhodu v tom, že si uživatel musí konkrétní plugin do prohlížeče explicitně doinstalovat. Nejrozšířenějšími platformami jsou Adobe Flash, Silverlight a Java [10].

**Adobe Flash**
Technologie Flash[1] byla jedna z prvních, která umožňovala prezentovat video a audio na internetu. Postupem času byl pro tuto platformu vyvinut objektový programovací jazyk ActionScript, který otevřel možnosti k tvorbě online aplikací, her, bannerů a různých animací. Flash je komplexní systém, obsahující profesionální nástroje, umožňující rychlé navrhování a vývoj. Používání Flashe v prohlížeči vyžaduje instalaci pluginu (zásuvného modulu). Flash nabízí hardwarovou akceleraci videa a do budoucna plánuje také akcelerovanou 3D grafiku.

**Silverlight**
Silverlight[2] od společnosti Microsoft je největším konkurentem technologie Flash, není ale tolik rozšířená. Tato platforma je určená pro tvorbu dynamického online obsahu a interaktivní práce s ním. Silverlight kombinuje text, vektorovou i bitmapovou grafiku, animace a video. S akcelerací je na tom Silverlight podobně jako Flash. Do internetového prohlížeče se instaluje prostřednictvím pluginu.

[1] http://www.adobe.com/cz/flashplatform/.
[2] http://www.microsoft.com/cze/web/silverlight/.

#### Java

Hardwarově akcelerovanou 3D scénu poskytují Java applety[3], softwarové komponenty běžící v kontextu prohlížeče, využívající knihovny např. `Java3D` nebo `JOGL` (Java OpenGL). V případě Javy si uživatel musí nainstalovat prostředí JVM (Java Virtual Machine), jehož součástí je plugin prohlížeče. Java applety mohou být nebezpečné [13].

#### CSS3

Zkratka CSS3[4] označuje kaskádové styly verze 3. Je to jazyk pro popis způsobu zobrazení stránek. Vývoj kaskádových stylů verze 3 sice stále není dokončen, ale v novějších prohlížečích již mají nemalou podporu. Většina nových prohlížečů již alespoň částečně podporuje CSS3 transformace prvků dokumentu. Webkit[5](vykreslovací jádro prohlížeče vyvíjené firmou Apple nebo Google) dokonce umí nastavit perspektivní zobrazení a aplikovat 3D transformační matice. Nejedná se o akcelerované vykreslování, ale na rozdíl od jiných technologií CSS3 pracují přímo s HTML prvky webu.

## 2.2 WebGL

Nový standard WebGL obsažený HTML5 (2.6) řeší problém pluginů a chybějící hardwarové akcelerace. WebGL (Web-based Graphics Library) je multiplatformní volně dostupná nízkoúrovňová 3D grafická knihovna, postavená na OpenGL ES 2.0 [1], rozšiřující možnosti jazyka JavaScript (2.7). Technologie dokáže za přítomnosti HTML5 v elementu objektového modelu dokumentu (DOM), zvaném plátno (canvas), vykreslovat 3D grafiku. Tato knihovna používá přímo grafickou kartu (GPU) počítače, čímž zaručuje vysokou rychlost výpočtů, tedy akcelerované vykreslování.

V březnu roku 2011 byla vydána první specifikace[6] WebGL verze 1.0 společností Khronos Group, která spravuje mimo jiné také známé technologie OpenGL či nový OpenCL. Nutná je podpora grafické karty, která umožňuje vykreslování pomocí shaderů[7]. K programování shaderů slouží stejně jako u OpenGL jazyk GLSL (OpenGL Shading Language). Nezbytný je také moderní prohlížeč obsahující implementaci WebGL. Důležitá je také aktualizace ovladačů GPU. Technologie nepotřebuje žádný plugin a měla by být v budoucnu dostupná také na mobilních zařízeních. Některé typy grafických karet v kombinaci s určitým operačním systémem WebGL nepodporují.

Vykreslování grafiky je založené na specifikaci OpenGL ES 2.0 (odlišnosti jsou popsány v Kapitole 2.4) a GLSL. Na Obrázku 2.1 je znázorněno schéma vykreslování grafického řetězce. K dipozici jsou programovatelné tyto shadery: *Vertex shader* realizuje univerzální programovatelné operace s vrcholy. Běžně se používá pro maticové transformace, výpočet osvětlení a transformaci souřadnic textur. Na *Vertex shader* navazuje modul *Primitive assembly*. Tento modul přeloží vstupující vrcholy na základní vykreslované primitiva. Primitivem může být trojúhelník, bod, čára. V další fázi – *Rasterization* jsou primitiva vykreslena do množin dvojrozměrných fragmentů, které jsou zpracovávány fragment shaderem. *Fragment shader* je program prováděný postupně nad každým zpracovávaným pixelem. Nejčastějšími operacemi jsou modifikace barvy a aplikace textur. Nakonec je výsledek za-

[3] http://java.sun.com/applets/.

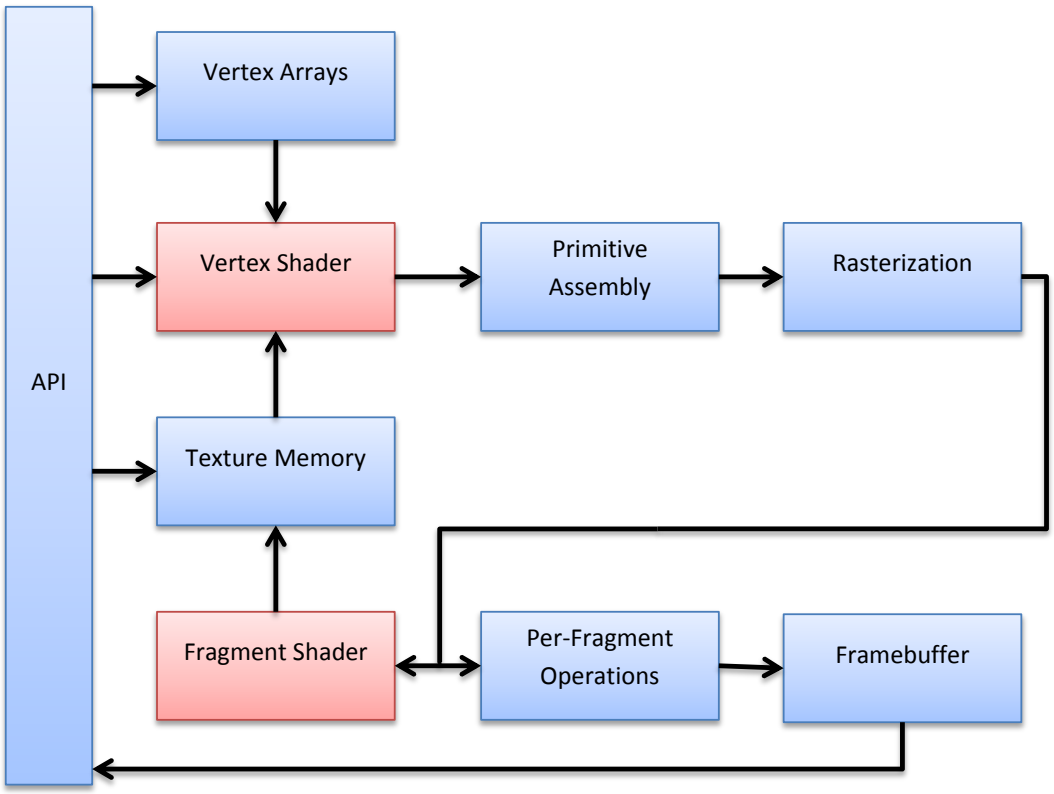
[4] http://www.w3schools.com/css3/.

[5] http://www.webkit.org/.

[6] http://www.khronos.org/registry/webgl/specs/latest/.

[7] Shader je počítačový program sloužící k řízení jednotlivých částí programovatelného grafického řetězce grafické karty (přesněji GPU) [12].

psán do *Frame bufferu*, z kterého je možné zpětně číst. Pro programování shaderů slouží jazyk GLSL, který je syntaxí podobný jazyku C.




Obrázek 2.1: Schéma zřetězeného zpracování WebGL API [1].

## 2.3 Podpora webových prohlížečů

V současné době je WebGL implementována ve většině používaných moderních prohlížečů. V některých případech se zatím jedná pouze o betaverze. Společnosti Apple, Google, Mozilla a Opera tvoří spolupracující skupinu s neziskovou organizací Khronos Group[8].

**Google Chrome**
Google Chrome podporuje WebGL stabilně od verze 9, experimentálně již od verze 8. Společnost Google se aktivně podílí na rozvoji WebGL. Dokládají to především její demonstrační experimenty[9]. Co se týče rychlosti WebGL a JavaScriptu, je na tom Chrome ze všech prohlížečů nejlépe.

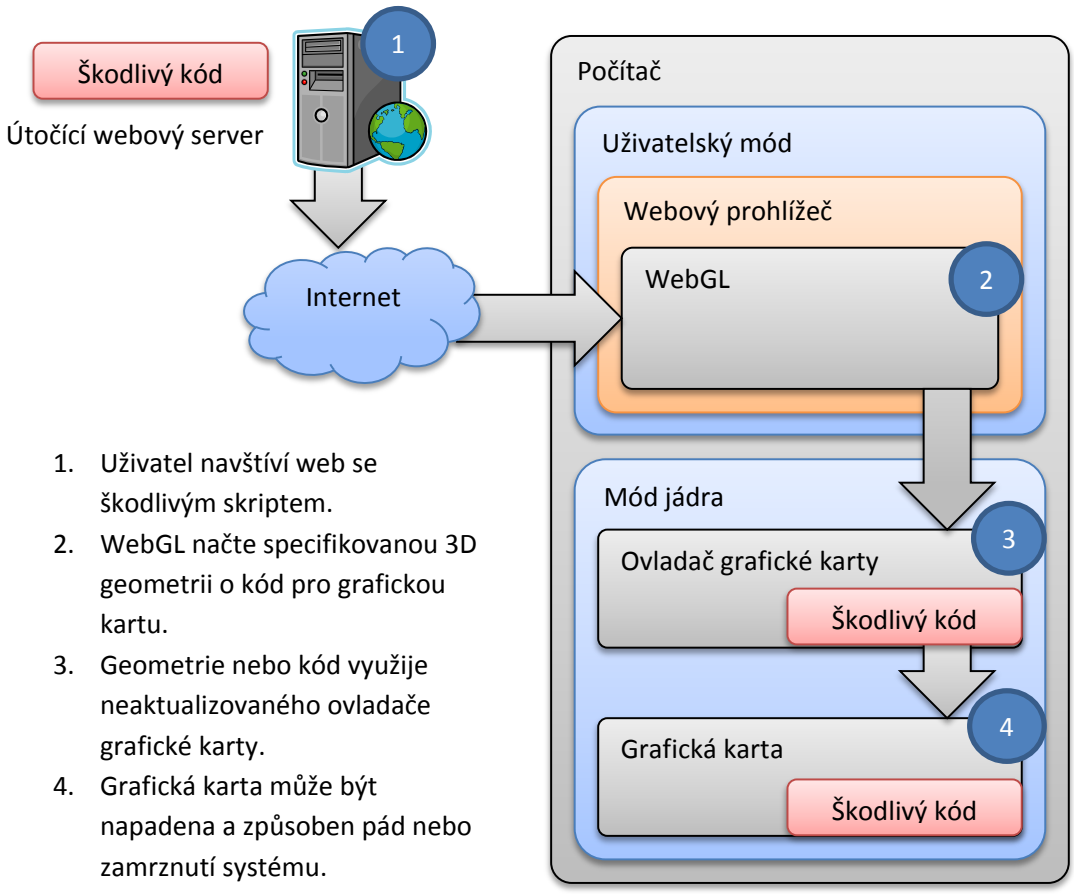
**Mozilla Firefox**
Ve Firefoxu je WebGL dostupná od verze 4. Někdy je v prohlížeči nutné tuto technologii explicitně povolit nebo aktualizovat verzi ovladače grafické karty.

[8] http://www.khronos.org/.
[9] http://www.chromeexperiments.com/webgl.

**Safari**
Uživatelé Mac OS využívající Safari verze vyšší jak 5.1 si mohou akceleraci povolit ručně [5].

**Opera**
Opera implementovala WebGL v připravované nové verzi 12. A slibuje do budoucna funkčnost napříč všemi platformami, tedy ve Windows, Linuxu i Mac OS X, a také v mobilních zařízeních – chytré telefony a televize.

**Internet Explorer**
Internet Explorer od firmy Microsoft WebGL nepodporuje a ani to nemá v budoucnu v plánu [9]. Microsoft upozorňuje na bezpečnostní díry této technologie. WebGL v sobě skrývá spoustu rizik, například to, že umožňuje přistupovat až na úroveň grafických ovladačů. JavaScript by tak mohl posloužit třeba k útoku na grafickou paměť systému. Obrázek 2.2 znázorňuje příklad. Existují již ale pluginy, které do prohlížeče Internet Explorer přidávají plnou podporu WebGL.




Obrázek 2.2: Mechanismus možného útoku pomocí WebGL.

V oblasti mobilních zařízení je WebGL dostupná například v Nokii N900, BlackBerry PlayBook nebo na jiných zařízeních se systémem Android. Pro uživatele iOS Mobile Safari je plně podporována na platformě iAd[10].

[10] https://developer.apple.com/iad/.

## 2.4 Odlišnosti WebGL a OpenGL

WebGL je založena na specifikaci OpenGL ES 2.0, které se, na rozdíl od klasické OpenGL, orientuje především na maximální přenositelnost do mobilních zařízení. Hlavním rozdílem je tedy výkon, který je u klasického OpenGL vyšší. Existuje několik významných rozdílů v chování velmi podobných knihoven OpenGL ES 2.0 a WebGL oproti klasické OpenGL API[11]. OpenGL ES 2.0 a WebGL mají omezenou podporu textur, jejichž rozměry nejsou mocninami dvou. Pokud chceme v aplikaci použít mód, který texturu opakuje (REPEAT wrap mode), tak nelze použít obrázek libovolných rozměrů. Tento problém může řešit možné rozšíření WebGL. Dalšími podstatnými rozdíly je absence podpory objemových textur, nepodpora datového typu `GL_DOUBLE` (v JavaScriptu `Float64Array`) pro pole souřadnic vrcholů a textur, z polygonů umí WebGL vykreslovat pouze trojúhelníky, rozdíl je také ve správě paměti, která je v případě WebGL JavaScriptem alokována a uvolňována automaticky.

## 2.5 Srovnání technologií WebGL, Silverlight a Flash

Hlavními výhodami WebGL je vysoký výkon (akcelerované vykreslování), multiplaformnost a to, že nevyžaduje žádnou instalaci pluginů. Tyto skutečnosti v budoucnosti s největší pravděpodobností způsobí velké rozšíření této technologie. Flash i Silverlight mají již nyní nezanedbatelnou konkurenci v podobě WebGL. Nevýhoda WebGL je její nízkoúrovňové API. Tento problém již řeší v současnosti spousta frameworků[12]. Volně dostupné jsou například O3D[13], GLGE[14] nebo Three.js[15].

V následujících tabulkách 2.1 a 2.2 je srovnání technologií WebGL, Silverlight a Flash podle různých kritérií, ze zdrojů [2] a [4]. Ve výsledku do budoucnosti vychází nejlépe právě WebGL. Samozřejmě by se měla řešit ještě otázka bezpečnosti, kvůli které se Microsoft aktuálně staví bokem. WebGL je ale stále v počátcích svého vývoje a podle mého názoru je zatím předčasné ji označovat jako vytrvale nebezpečnou.

[11] Application Programming Interface, označuje v informatice rozhraní pro programování aplikací.

[12] Framework je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů.

[13] http://code.google.com/p/o3d/.

[14] http://www.glge.org/.

[15] http://threejs.org.

Tabulka 2.1: Srovnání technologií WebGL, Silverlight 5 a Flash 11 pro herní účely[2]

| | **WebGL** | **Flash 11** | **Silverlight 5 (XNA)** |
|---|---|---|---|
| Předchozí vydání | První generace | Flash 10 | Silverlight 4 |
| Podpora prohlížečů | Všechny mimo Internet Explorer (dále jen IE) | Všechny mimo Metro-style IE | IE 10 v desktopovém režimu, Firefox a Chrome |
| podpora iOS | Od verze iOS 5 | Ano, prostřednictvím Adobe AIR | Ne |
| Podpora konzole | Ne | Ne | Xbox 360 |
| Hlavní podporující společnosti | Google, Apple, Mozilla | Adobe | Microsoft |
| Programovací jazyk | JavaScript + GLSL | ActionScript + AGAL | C#, VB .NET, F#, Jscript .NET + HLSL |
| Komerční 3D Game engine | Není | Flare3D | Není |
| Volně dostupné API | O3D, GLGE, Three.js | Proscenium, Alternativa3D, Away3D, Minko | XNA |

## 2.6 HTML5

Technologie WebGL by se neobešla bez HTML verze 5, protože právě ona zprostředkovává nový element `canvas`, určený pro dynamické vykreslování 2D a 3D grafiky prostřednictvím JavaScriptu. HTML5[16] je rozšiřující specifikace jazyka HTML. V současnosti je ve stádiu návrhu organizací W3C[17]. Zavádí například nové značky, perzistentní úložiště formou asociativního pole, relační databáze s podporou transakcí a podporu offline aplikací. Mění zápis specifikace typu dokumentu a přináší celkově kratší a jednodušší zápisy. Významnou novinkou jsou nové značky pro vkládání multimédií. Mimo jiné také rozšiřuje značku input, do které přidává typy pro datum, čas, číslo apod.

## 2.7 JavaScript

Jak jsem již zmínil, WebGL aplikace se programují v jazyce **JavaScript**. JavaScript[18] je multiplatformní objektově orientovaný skriptovací jazyk. Běží na straně klienta, tedy v internetovém prohlížeči. Skript se obvykle vkládá přímo do HTML kódu stránky, případně je v samostatném souboru. JavaScript patří do skupiny technologií, které tvoří dynamické webové stránky. Je také součástí obecného označení technologií AJAX (Kapitola 4.1), umožňující změnu obsahu stránky bez jejího znovunačtení. Pracuje s objektovým modelem dokumentu (DOM) a mimo jiné také zprostředkovává funkcionalitu WebGL.

[16] http://www.w3schools.com/html5/.
[17] http://www.w3.org/.
[18] http://www.w3schools.com/js/

Tabulka 2.2: Srovnání technlologií WebGL (HTML5), Silverlight 5 a Flash [4]

| | **WebGL (HTML5)** | **Silverlight 5** | **Flash** |
|---|---|---|---|
| Podporované prohlížeče | Firefox 4+,Chrome 9+, Safari 5.1, Opera 12 | IE6+, Firefox3+, Chrome, Safari | Všechny prohlížeče |
| Podporovaný operační systém | multiplatformní | Windows a Mac OS | Mutliplatformní kromě iOS |
| Rozšíření v budoucnosti | Všechny současné prohlížeče doplněny IE, rozšíření na platformy iOS, Android a pravděpodobně WP7 (Windows Phone 7) | Rozšíření na WP7 a pravděpodobně XBOX | Většina prohlížečů kromě iOS |
| Výhody | Jedná se pouze o webový standard | Dobrý výkon 2D/3D, přátelské API (založeno na XNA) | Dominantní platforma medií a reklam, podpora Unity3D |
| Nevýhody | Zatím není standardizováno řetezené zpracování, složité vytváření aplikací (bez frameworku), velké výkonové rozdíly mezi desktopovým a mobilním hardwarem | Produkt Microsoftu, není standardizován | Nekompatibilita s iOS, s příchodem HTML5 ztrácí dominantní postavení |
| Závěr | Vítěz, dobrá reklama díky multiplatformnosti a podpora iOS | Dobré řešení desktopové mutlimediální reklamy, ale neřeší mobilní platformy | Nejlepší řešení v blízké budoucnosti |


Obrázek 2.3: Loga technologií WebGL a HTML5.

# Kapitola 3

# Návrh 3D webového rozhraní

Klasická webová stránka nabízí uživateli 2D rozhraní s běžnými prvky HTML jako jsou odkazy, obrázky, formuláře. Může být obohacena o další nestandardní prvky, mediální a dynamický obsah, které jsou implementovány pomocí JavaScriptu nebo zásuvných modulů. Interakce s uživatelem je zajišťována obvykle událostmi klávesnice a myši.

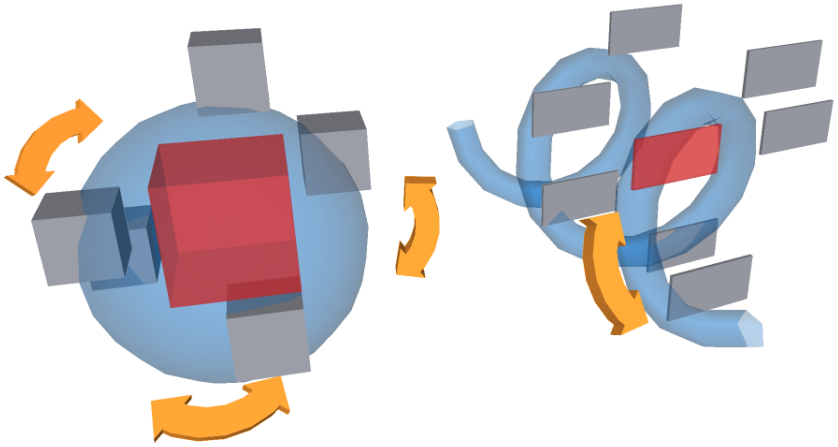
3D grafika vykreslovaná v reálném čase nabízí webové stránce spoustu nových možností. Rozšíříme-li dvoudimenzionální prostor o další úroveň, nabízí se nám například možnost vyhnout se opakovanému načítání různých podstránek pomocí odkazů. Uživatel tak může mít lepší přehled o struktuře webu a celkový dojem z prezentace. Je možné vymodelovat abstraktní 3D model téměř čehokoliv, a poté ho vykreslovat ve webovém prohlížeči, aplikovat na něj různé realistické efekty, animovat ho. Ve virtuálním prostoru můžeme také zobrazovat věci, které jsou ve skutečnosti nerealizovatelné, například modelování chemických sloučenin, 3D grafy nebo kompletní postavu lidského těla včetně jeho vnitřností, nervů a cév. Zajímavým projektem, který je realizován pomocí technologie WebGL je Google Body Browser[1]. V oblasti internetového obchodnictví je tu možnost prezentovat své produkty v podobě 3D modelu.

Typické 2D prvky webu můžeme nahradit jejich 3D alternativou. Nabízí se nám vytvoření animovaného menu webové stránky zobrazeného v prostoru, 3D fotogalerie, videogalerie nebo modelování podstránky webu dalšími objekty. Také zobrazování panoramatických fotografií v prostoru je způsob, který uživatele zaujme. Na následujících Obrázcích 3.1, 3.2 a 3.3 jsou znázorněny některé navrhované 3D prvky. Červeně zvýrazněný objekt označuje aktuálně vybranou položku ze skupiny objektů, případné směry animace jsou znázorněny žlutými šipkami.

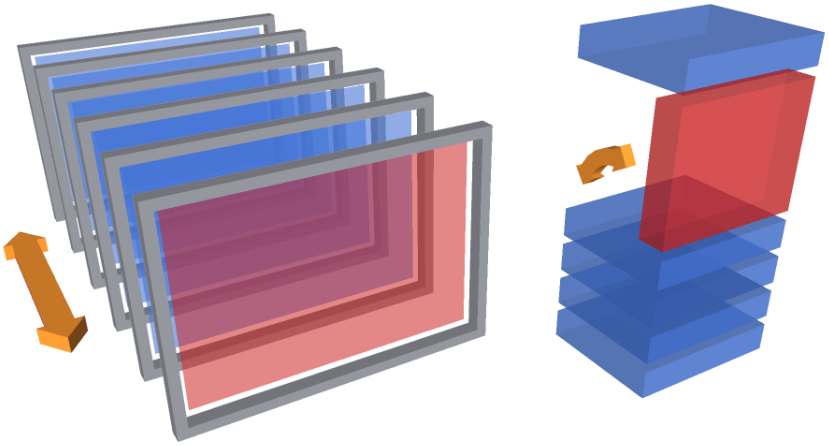
Na Obrázku 3.1 vlevo se nachází 3D prvek, jenž by mohl reprezentovat množinu dalších 3D elementů, které by sjednocoval. Jednotlivé krychle by se pohybovaly po orbitální dráze s tím, že aktivní by byla k uživateli nejblíže a mohla by dále expandovat. Podle mého názoru je také zajímavým nápadem zobrazovat v prostoru prvky, které modelují nějaký časový průběh, například zobrazení kalendáře nějakých událostí rotující spirálou (Obrázek 3.1 vpravo).

Jak by bylo možné reprezentovat například přepínání jednotlivých podstránek webu, záložky apod., je znázorněno na Obrázku 3.2. Vpravo je návrh přepínání stránek, které by mělo především vizuálně zaujmout uživatele. Přepínání stránek na Obrázku 3.2 vpravo by mohlo být praktičtější. Ve výchozím stavu by se uživateli zobrazovala pouze čelní strana panelu se stručnou informací (název stránky), a po výběru by se panel natočil směrem

[1] http://www.zygotebody.com/.



Obrázek 3.1: Návrh možných 3D prvků. Vpravo rotující spirála.

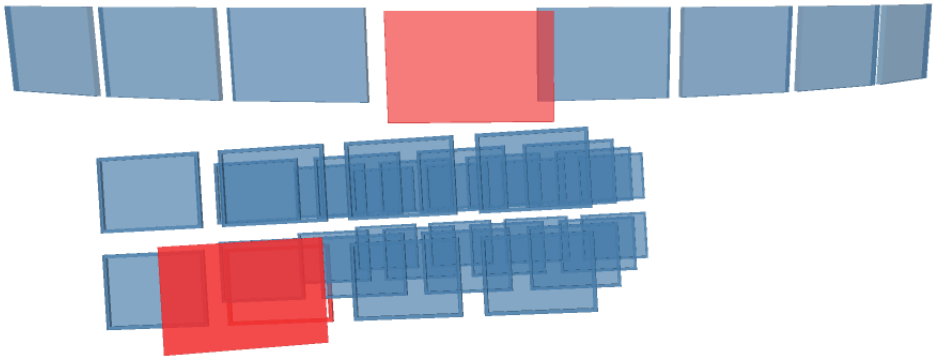


Obrázek 3.2: Návrh možných 3D prvků – přepínání stránek

k uživateli větší plochou, kde by mohl být další obsah. Tyto prvky by mohly mít také i jinou funkcionalitu – návštěvní kniha, zprávy v emailové schránce, reklamní bannery.

Fotografií je na internetových stránkách spousta a fotogalerie je jednou z věcí, která se své 3D podobě přímo nabízí. Máme spoustu fotek a alb, o kterých chceme mít přehled a jednoduše si je prohlížet. Z toho důvodu již existuje spoustu (2D) JavaScriptových knihoven, které dodají webovým fotogaleriím efektivnost a animace. Za zmínku stojí projekt **Lightbox**[2], který poskytuje řadu efektů a výhod pro implementaci webových fotogalerií. Pokud máme k dispozici prostor, můžeme vytvořit fotogalerii daleko efektivnější. Jak je znázorněno na Obrázku 3.3, šlo by fotografie (nebo celá alba) zobrazovat i za sebe nebo je umísťovat na kruhovou dráhu. Uživatel by mohl jednotlivé fotky přemísťovat mezi alby, měnit jejich uspořádání.

[2] http://lokeshdhakar.com/projects/lightbox2/



Obrázek 3.3: Návrh možných 3D prvků – příklady fotogalerie.

Při návrhu nastává jeden podstatný problém, a to pokud chceme do scény začlenit HTML obsah jako je text, a vůbec celé formátování klasické stránky. V současné době neexistuje ideální způsob jak toho docílit. Moje řešení se nachází v rámci popisu implementace v Kapitole 4.9.

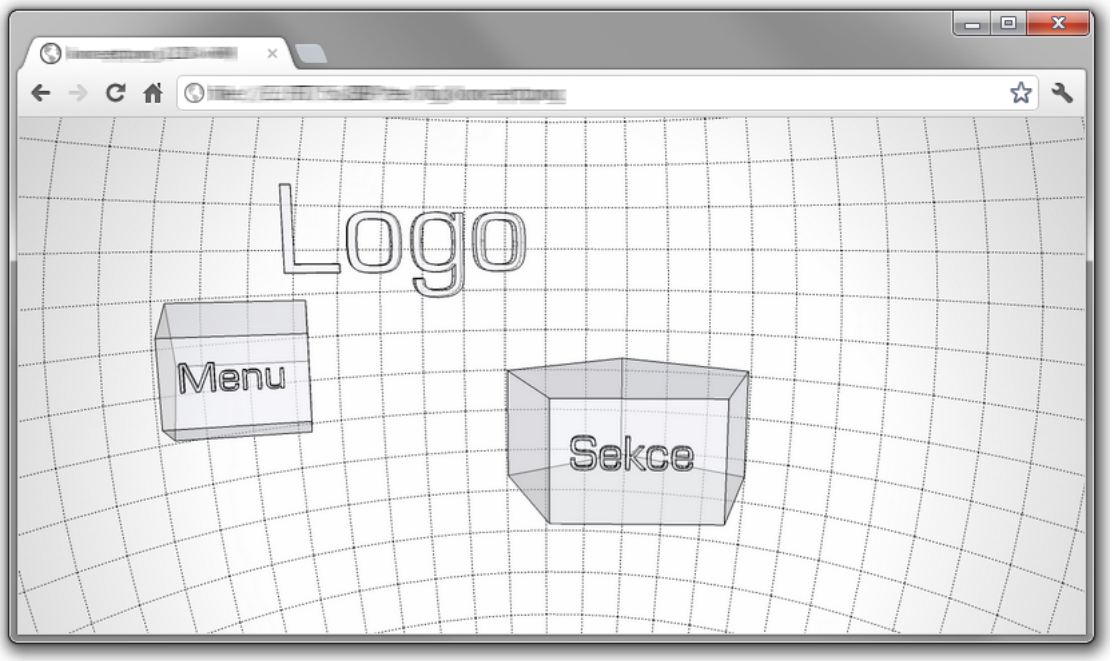
## 3.1 Koncept navrhovaného rozhraní

Jako téma 3D webové prezentace, jsem si zvolil web rockové kapely. Webová stránka by měla uživateli poskytovat úvodní aktuální informace o dané kapele, dále sekci s kalendářem akcí kapely, fotogalerii, prezentaci členů kapely a stránku s kontakty. Navigace mezi jednotlivými stránkami by měla být intuitivní, rychlá, jednoduchá, přehledná. 3D rozhraní by mělo obsahovat přiměřený počet prvků, mezi kterými se uživatel neztratí. V nemalé míře bych chtěl implementovat plynulé animace a různé efekty (např. osvětlení scény), které umožňuje aplikace shaderů. Nicméně, navrhované rozhraní by mělo být univerzální a obecné.

Základní myšlenkou bylo virtuálně umístit uživatele do 3D struktury texturované koule, a tím vytvořit kolem uživatele tématický vizuální prostor. Jedná se o klasický způsob projekce textury využívaný například k tvorbě map v geografii, s tím rozdílem, že pozorovatel nesleduje kouli z vnějšku, ale zevnitř. Do tohoto prostoru chci umístit 3D prvky webového rozhraní. Uživatel by se v tomto prostoru mohl orientovat, tedy alespoň pohybovat s kamerou po orbitální dráze.

Hlavními 3D prvky bude menu, skupiny objektů znázorňující jednotlivé sekce, logo kapely. Menu a logo bude na rozdíl od ostatních objektů statické, aby je uživatel v prostoru neztratil. Koncept rozhraní a rozmístění hlavních prvků je znázorněno na Obrázku 3.4.

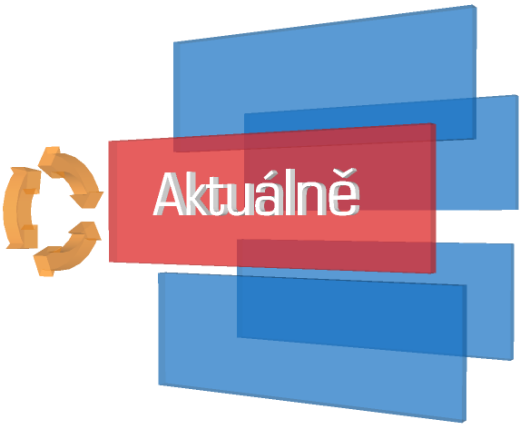
Ovládání rozhraní bude zprostředkováno pomocí klávesnice a myši. Pomocí kurzorových kláves nebo stisknutím pravého tlačítka a pohybem myši bude mít uživatel možnost se v prostoru pohybovat. Pohled kamery bude vždy směřovat do středu scény, tedy do počátku souřadnicového systému. Pohybem ve scéně je myšlen posun po trajektorii ve tvaru kružnice, a to svislé nebo vodorovné. Celé webové rozhraní bude své rozměry přizpůsobovat oknu prohlížeče s tím, že nebudou překročeny určité maximální rozměry.




Obrázek 3.4: Abstraktní koncept rozhraní.

**Menu**

Klíčovým prvkem je menu, které slouží k navigaci mezi jednotlivými sekcemi. Menu by mělo uživatele informovat o struktuře webu. Na Obrázku 3.5 je návrh tohoto elementu. Jedná se o 3D položky, znázorňující jednotlivé sekce webu, rotující kolem jednoho bodu. Aktuální vybranou sekcí je odpovídající položka menu, která je implicitně nejblíže k uživateli, je zvýrazněna a orientuje se směrem k poloze kurzoru myši. Menu je možné otáčet kolečkem myši nebo kliknutím na vybranou položku a způsobí změnu sekce. Položka menu se také lehce zvýrazní po najetí kurzorem myši. Jednotlivé položky budou poloprůhledné, aby se výrazně nepřekrývaly.




Obrázek 3.5: Návrh rotačního 3D menu.

**Přepínání sekcí**

Každá podstránka (sekce) je znázorněna 3D ikonou ohraničenou v rámu, která svým charakterem vystihuje její obsah. Tyto objekty jsou v prostoru uspořádány do mnohoúhelníku a jsou částečně průhledné. Animace přepínání sekcí je synchronizována s rotací menu. V okamžiku, kdy se ukončí animace přepínání stránek, tak se celá skupina ikon natočí aktuální sekcí směrem k uživateli. Aktuální ikona je neprůhledná. Popis tohoto elementu by měl být daleko pochopitelnější z Obrázku 3.6. Po animaci přepínání podstránek ikona nacházející se před uživatelem expanduje v závislosti na charakteru obsahu sekce. V případě textového obsahu stránky, se rámec s ikonou přesune před pozici kamery uživatele a zviditelní se vrstva s HTML obsahem. Pokud jsou obsahem stránky další 3D prvky, ikona zůstane na místě, natáčí se směrem k uživateli a objeví se další 3D objekty. Přepínání sekcí je možné také kliknutím na vybranou sekci.

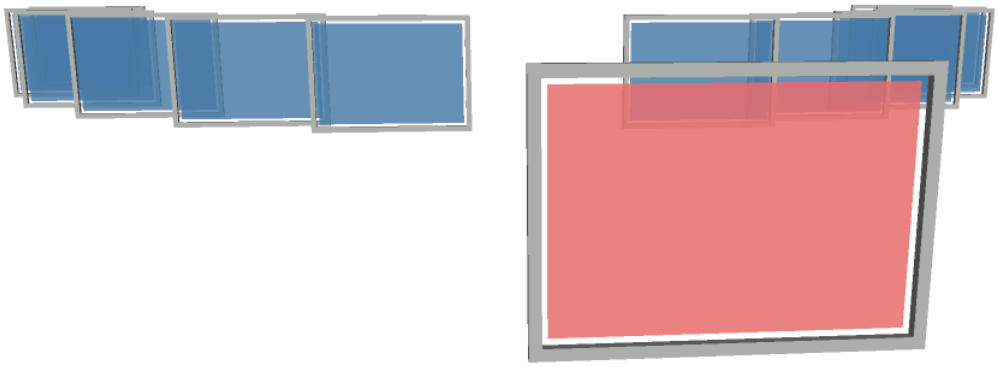

Obrázek 3.6: Uspořádané rámce s ikonami reprezentující jednotlivé sekce.

**Sekce „Fotogalerie“**

Jednou z podstránek je fotogalerie. Po přepnutí do této sekce se před uživatelem objeví řada fotografií a postupně se umístí na vodorovnou trajektorii kružnice se středem v počátku scény. Jednotlivé fotky jsou umístěny v rámečku, aby nesplývaly s tmavým pozadím webu. Po celou dobu, kdy se uživatel nachází v této sekci, jsou fotografie orientovány směrem k pozici kamery. Prohlížení fotografií bude zprostředkováno najetím kurzoru myši na vybranou fotku, tato fotografie se po uplynutí časové prodlevy přiblíží blíže k uživateli a zároveň se před ni nastaví pozice kamery. Pokud zůstane kurzor myši na stejném místě, budou se jednotlivé fotografie automaticky přepínat. Přiblížení fotografie může být inicializováno také kliknutím na vybranou fotku. Po přemístění kurzoru na jinou fotografii nebo kliknutím pravého tlačítka myši do scény mimo fotografii se vybraná fotka vrátí na původní pozici. Během prohlížení je samozřejmě možné se ve scéně pohybovat pomocí kláves nebo myši, jak je popsáno v Kapitole 3.1. Pokud je tedy kurzor myši umístěn cca uprostřed stránky, je pak také možné se mezi fotografiemi přepínat pomocí šipek na klávesnici. V plánu je implementace pouze jedné fotografie s několika demonstračními obrázky, v budoucnu by

bylo ovšem vhodné nějakým způsobem zakomponovat do této podstránky více fotogalerií a přepínání mezi nimi. Schéma fotogalerie je znázorněno na Obrázku 3.7.

Obecně by se dala stejným způsobem implementovat také videogalerie, u které by navíc přibyly prvky pro ovládání videa. Tento nápad pravděpodobně zatím nemá nikde obdoby. Prakticky by bylo možné vykreslovat jakoukoliv další 3D scénu do 2D plochy objektu hlavní scény.



Obrázek 3.7: Navrhovaná fotogalerie.

**Sekce „Členové“**

Sekce „Členové“ by měla prezentovat jednotlivé členy skupiny. O každém členovi obsahuje základní informace, podané v 3D podobě. Princip animace a procházení jednotlivých členů vychází z fotogalerie. Na Obrázku 3.8 je ukázka vybraného člena.

Základ tedy obsahuje fotografie členů kapely. Ovšem po přiblížení vybraného člena se do viditelné scény přemístí další objekty. Prvním je 3D přezdívka člena kapely. A dalším podstatným je 3D model nástroje, na který vybraný člen hraje. Nástroj se otáčí kolem svislé osy. Podrobnější informace o členovi by se mohly objevovat v rámečku poblíž skupiny objektů v HTML textové podobě.

Tato skupina 3D prvků a principy animace a zobrazení by se daly aplikovat na mnoha dalších místech. Nemusí se jednat o prezentaci členů kapely, ale také třeba o nabídku nějakých obchodních produktů nebo předpověď počasí.


Obrázek 3.8: Prezentace člena kapely.

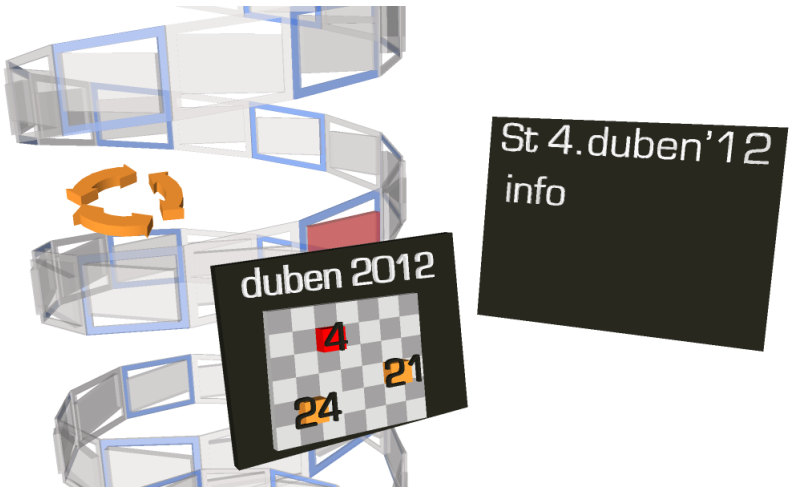
**Sekce „Akce“**
Informace o vystoupení kapely bude obsahovat sekce „Akce“. Měl by se zde tedy vyskytovat kalendář akcí. Pro využití 3D možností bylo v plánu vytvořit spirálu reprezentující časový průběh a na ní umístěné prvky znázorňující měsíce. Spirálou by bylo možné rotovat. Po výběru měsíce by prvek expandoval, objevila by se jednotlivá data akcí, a poté podrobnější informace o akci.

Tento princip zobrazení by šel aplikovat na jakýkoliv obsah, jehož dílčí prvky mezi sebou mají časové rozpoložení. Nabízí se jiný pohled na časovou osu než klasický 2D kalendář. Do tvaru spirály lze uspořádat více prvků v menším prostoru, než kdyby byly rozloženy lineárně na přímce a uživatel by tak neztratil přehled. To by se ovšem muselo experimentálně ověřit.

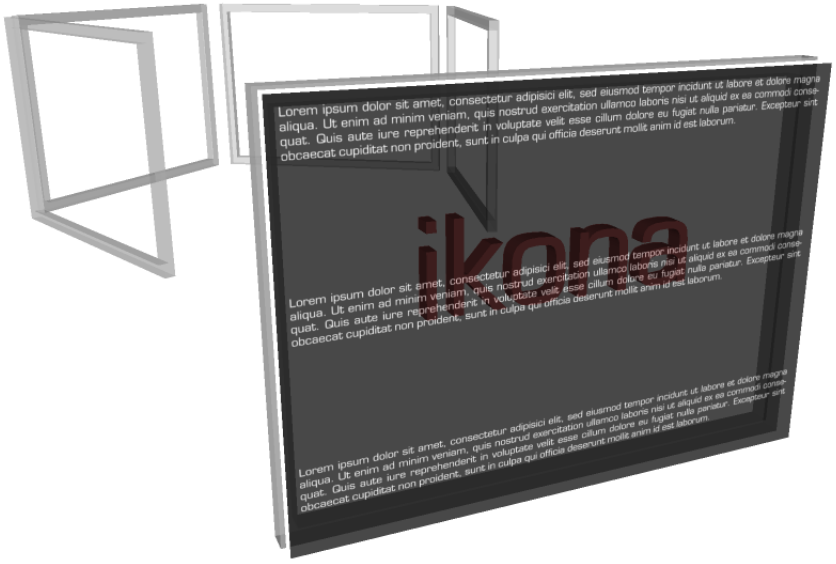
## 3.2 HTML obsah ostatních sekcí

Ostatní sekce budou mít textový charakter. Jedná se o stránky „Aktuálně“ a „Kontakt“. Při přepnutí do jedné z těchto sekcí dojde k animaci odpovídající ikony s rámem blíže k uživateli. Na tento 3D objekt je mapována vrstva s HTML obsahem. Tato vrstva tedy bude měnit svoje měřítko a polohu stejně jako 3D objekt. Při jakémkoliv pohybu ve scéně se bude vrstva s objektem přemísťovat stále do pohledu uživatele. Návrh znázorňuje Obrázek 3.10.

Textovým charakterem nemám na mysli pouze text, ale také další prvky klasické HTML stránky – formulářové prvky, 2D design, zkrátka vše, co nelze jednoduše udělat v alternativní 3D podobě.




Obrázek 3.9: Návrh kalendáře akcí.




Obrázek 3.10: Začlenění textového obsahu.

# Kapitola 4

# Implementace rozhraní

Jak jsem již zmínil v teorii o WebGL (Kapitola 2.2), akcelerované vykreslování je implementováno v jazyce JavaScript (2.7). JavaScript WebGL API je velice nízkoúrovňové, je tedy vhodné, alespoň pro větší projekty, používat nebo si vytvořit nějaký framework, který nám ulehčí a zefektivní práci. Stanovil jsem si cíl vytvořit vlastní framework, a poté s jeho využitím implementovat 3D uživatelské rozhraní.

## 4.1 Volba technologií a cílové prostředí

Kromě nezbytného JavaScriptu jsem použil také skriptovací jazyk PHP[1] pro asynchronní konverzi modelů do formátu JSON[2], tento převod je pouze jednorázový. Web vyžaduje prostředí serveru kvůli asynchronnímu načítání modelů, shaderů a materiálů. Byly využity nové transformační vlastnosti CSS3 (Kapitola 2.1) pro mapování HTML obsahu nad element `canvas`. Rozhraní bylo vyvíjeno v prohlížeči Chrome, se snahou zachovat kompatibilitu i s dalšími prohlížeči, které podporují nové vlastnosti CSS3 a WebGL, konkrétně třeba Firefox. Pro vytváření, editaci a stahování 3D modelů jsem používal volně dostupný program SketchUp[3] od Google.

**AJAX**

AJAX[4] (Asynchronous JavaScript and XML) je technika pro vytváření rychlých a dynamických stránek. Dovoluje asynchronní aktualizaci dat webové stránky bez nutnosti načíst celou stránku znovu. Technologie je postavena na již existujících standardech: `XMLHttpRequest object` pro asynchronní výměnu dat se serverem, JavaScript/DOM pro zobrazování informací, CSS pro stylování dat a XML formát, který se obvykle používá při přenosu dat.

## 4.2 Návrh frameworku

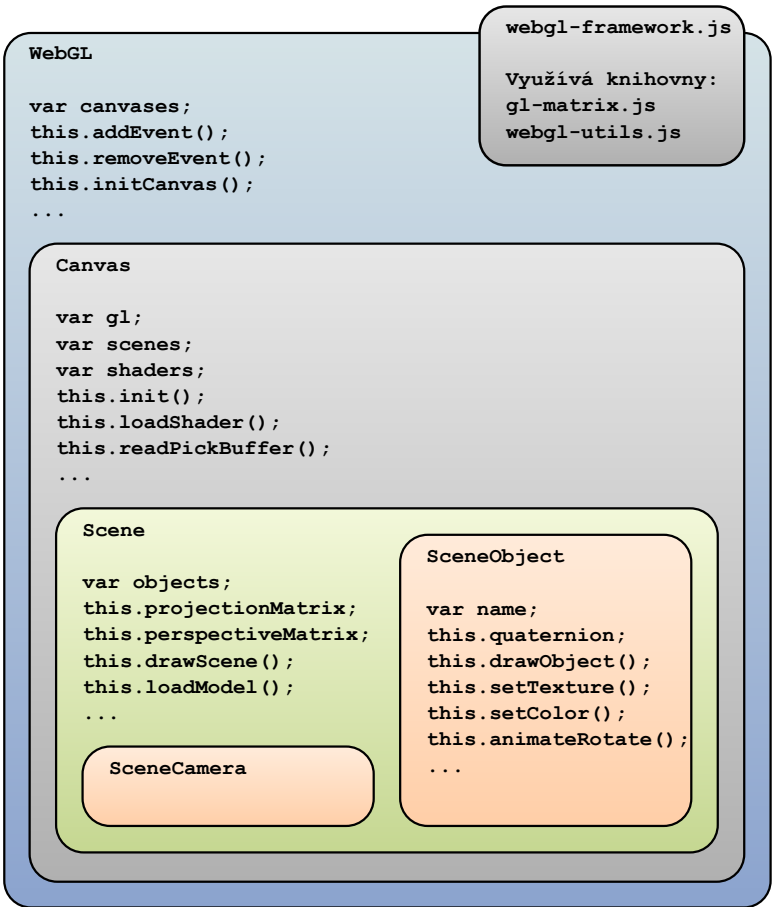
Při návrhu bylo nutné rozmyslet si účel frameworku. Požadovanými vlastnostmi bylo načtení a vykreslení libovolného 3D modelu ve zvoleném formátu, aplikace materiálů objektu (barvy, textury), aplikace shaderů v jazyce GLSL z externích souborů. Dále také zapouzdření transformačních operací a animací, bez kterých se rozhraní rozhodně neobejde. Velice

[1] http://www.php.net/.
[2] http://www.json.org/.
[3] http://sketchup.google.com/.
[4] http://www.w3schools.com/ajax/default.asp.

důležitou součástí bylo integrování funkcionality pro výběr objektů ve scéně kurzorem myši. Na Obrázku 4.1 je schéma navrhovaného frameworku.

**WebGL**

Základem je hlavní objekt třídy `WebGL`, který zapouzdřuje metody pro instanci objektů typu `Canvas` následovanou inicializací kontextu WebGL, obsluhu událostí a další pomocné funkce.




Obrázek 4.1: Stručné schéma frameworku.

**Canvas**

Objekt typu `Canvas` obsahuje již spoustu metod a atributů, které jsou vázány právě k jednomu HTML elementu `canvas` a jednomu kontextu WebGL. Mezi atributy je také struktura objektů, definující jednotlivé scény vykreslované na dané plátno a množina shaderů. Mezi nejdůležitější metody patří vytvoření scény, načtení shaderů, čtení bufferu pro výběr prvků scény a také funkce pro zpracování událostí myši.

**Scene**

Každá scéna (třída `Scene`) má vlastní projekční a perspektivní transformační matici a metody pro operaci s nimi. Obsahuje funkce pro práci s kamerou, vytváření a vykreslování objektů a další operace s ní spojené. Za zmínku stojí také buffery pro řazení průhledných a neprůhledných objektů (více v Kapitole 4.8)

**SceneObject**

Podobjekty scény jsou 3D prvky (objekty typu `SceneObject`). Prvek scény (dále objekt scény) je vytvářen ze struktury, která může obsahovat souřadnice vrcholů, textur, normál, pole indexů, a také popis použitých materiálů. Každý objekt scény má unikátní jméno, modelovou a normálovou matici a kvaternion (Kapitola 4.7) reprezentující rotaci. Klíčovými metodami objektu je samotné jeho vykreslení, animované rotace a posuvy, nastavování materiálů.

## 4.3 Použité knihovny

**gl-matrix**

Bylo nutné použít efektivní a rychlé funkce pro operace s maticemi a vektory (násobení, rotace, posuvy, interpolace, ...). Rozhodl jsem se tedy využít volně dostupné knihovny `gl-matrix`[5]. Knihovnu jsem mírně upravoval a rozšířil o nějaké funkce pro operace s kvaterniony (více o kvaternionech v Kapitole 4.7).

**WebGLUtils**

Správnou inicializaci kontextu WebGL zajišťuji knihovnou `WebGLUtils`[6] od společnosti Google, která v případě nepodpory WebGL prohlížečem uživatele na tuto skutečnost upozorní. Tato knihovna zároveň poskytuje korektní cestou napříč všemi prohlížeči funkci `requestAnimationFrame`, která je základem animací v JavaScriptu [7]. Pokud vytváříme animaci, používáme časovou smyčku, která opakovaně vykresluje scénu po nějakém časovém intervalu. Funkce `requestAnimationFrame` narozdíl od klasického `setTimeout` (`setTimeout` je funkce v JavaScriptu, která slouží k časovanému volání funkcí), který volá časovanou funkci vždy ve stejném zadaném intervalu, nabízí optimalizace. Způsobí například to, že nebudou vykonávány animace, které zrovna nejsou vidět na obrazovce. Výsledkem je snížení nároků na výpočetní prostředky (CPU a GPU).

## 4.4 Načítání 3D modelů

Cílem bylo, aby framework dokázal načítat v podstatě libovolné 3D modely ve zvoleném formátu, včetně jejich jednoduchých materiálů – barev a textur. Každý model jsem vytvořil nebo stáhnul z galerie volně stažitelných nekomerčních modelů 3D Warehouse[7], a poté z programu SketchUp exportoval do formátu `OBJ` a `MTL` (definice materiálů).

Soubor ve formátu `OBJ` může podle specifikace[8] obsahovat mnoho údajů, jako skupiny objektů nebo názvy objektů. Pro mě byly podstatné pouze souřadnice vrcholů, normál, textur a indexy vykreslovaných trojúhelníků definující 3D model, a také informace o použitých materiálech. V souboru formátu `MTL`[9] jsou definované materiály. Tento soubor může obecně obsahovat také spoustu různých informací, jako třeba ostrost nebo různé filtry. V rámci implementace jsem využíval pouze několika údajů: materiál je složený ze 3 barev – ambientní,

[5] https://github.com/toji/gl-matrix.

[6] https://cvs.khronos.org/svn/repos/registry/trunk/public/webgl/sdk/demos/common/webgl-utils.js.

[7] http://sketchup.google.com/3dwarehouse/nn

[8] http://www.martinreddy.net/gfx/3d/OBJ.spec.

[9] http://local.wasp.uwa.edu.au/~pbourke/dataformats/mtl/.

spekulární, difúzní, může mít definovanou hodnotu průhlednosti, a také cestu k souboru s texturou.

Pro zpracování a použití modelových souřadnic frameworkem je třeba data ze souborů načíst a vytvořit z nich použitelné datové struktury. Funkci pro parsování souboru typu `OBJ` jsem implementoval přímo v JavaScriptu, ale také jsem vytvořil skript v jazyce PHP, díky jemuž je možné konverzi spouštět asynchronně. PHP skript zároveň vytvoří soubor ve formátu JSON. Ve výsledku jsem tedy schopen načíst model ve formátu `OBJ`, ale mnohem efektivnější je možnost načíst model popsaný JSON strukturou, která je do objektů rozparsována daleko rychleji. Na rozdíl od souborů se souřadnicemi nejsou soubory s definicí materiálů moc objemné a postačuje přímé parsování do cílových objektů.

Načítání všech souborů je zprostředkováno asynchronním získáváním dat – AJAXem (Kapitola 4.1) a není nutné před vykreslením scény čekat na načtení všech modelů. WebGL API je aktuálně schopné voláním funkce `drawElements` vykreslit najedou maximálně 65535 trojúhelníků. Modely obsahující více než 65535 indexů je potřeba rozdělit a zpracovávat vícekrát. Vzhledem k indexaci použitých materiálů je na tuto skutečnost brán ohled již při parsování souboru.

```
# OBJ Model File

mtllib file.mtl

usemtl Color_I02
v -64.4397 -4.9467 -8
vt 3.93701 9.07707
vn 0.963518 -0.267643 1.04547e-016
v -64.4245 -4.87279 -3.255
vt 3.65834e-016 9.13969
vn 0.990838 -0.135059 5.2757e-017
v -64.4397 -4.9467 -3.255
vt 3.64817e-016 9.07707
f 1/1/1 2/2/2 3/3/1
...
```

```
# OBJ Material File (MTL)

newmtl Color_I02
Ka 0.000000 0.000000 0.000000
Kd 0.117647 0.266667 0.458824
Ks 0.330000 0.330000 0.330000
map_Kd textures/image.png

newmtl Color_000
Ka 0.000000 0.000000 0.000000
Kd 1.000000 1.000000 1.000000
Ks 0.330000 0.330000 0.330000
d 0.530000
...
```

Obrázek 4.2: Ukázka souborů definujících objekt (`OBJ`) a materiály (`MTL`).

## 4.5 Texturování

Mapováním textur – texturováním se označuje princip obarvení povrchu 3D objektů, který jim dodává realistický vzhled. Jak jsem již zmínil v Kapitole 2.4, s WebGL narážíme na problém při mapování bitmapových textur. Pokud chceme aplikovat jako texturu obrázek o rozměrech, které nejsou mocninami dvou a nevyžadujeme opakování textury, postačí nám nastavení parametrů `TEXTURE_WRAP`. Pokud ale opakování vyžadujeme, nelze tuto texturu s danými rozměry použít, toto WebGL v současné době podporuje pouze prostřednictvím rozšíření. Problém vyřešíme pokud upravíme rozměry bitmapy do rozměrů vyhovujících, tedy do následujících nejvyšších mocnin dvou. Vhodné je neupravovat zdrojový obrázek, ale upravit ho v rámci inicializace textury pomocí DOM API a 2D kontextu elementu `canvas`. Pomocí DOM API dynamicky vytvoříme plátno o zadaných rozměrech a vykreslíme do něj obrázek. Toto plátno je potom možné použít jako bitmapu. Postup by měl být zřejmý z následujícího úseku kódu:

```
var newcanvas = document.createElement("canvas"); // Vytvoření plátna
newcanvas.width = nextHighestPowerOfTwo(texture.image.width);
newcanvas.height = nextHighestPowerOfTwo(texture.image.height);
if (newcanvas.width != texture.image.width ||
    newcanvas.height != texture.image.height)
{
    var ctx = newcanvas.getContext("2d"); // Vytvoření 2D kontextu
    // Vykreslení obrázku do plátna
    ctx.drawImage(texture.image, 0, 0, new_width, new_height);
    texture.image = newcanvas;
}
// Aplikace bitmapy
gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, texture.image);
```

V rámci implementace používám Canvas 2D API také pro úpravu textur pro fotogalerii. Zajišťuji zobrazení texturovaných fotografií se správným poměrem stran bez ohledu na původní mapovací souřadnice.

## 4.6 Výběr objektů ve scéně myší

Pro ovládání rozhraní je nezbytné používání myši. Bylo nutné zajistit tzv. **picking** objektů ve scéně, zjištění, zda se kurzor myši nachází nad nějakým 3D objektem a pokud ano, tak nad kterým. Máme k dispozici události o pohybu myši, a také souřadnice pozice kurzoru vzhledem k oknu prohlížeče. Souřadnice je třeba přepočítat do oblasti viewportu – kreslícího plátna.

Pro výběr objektů ve scéně existuje metoda, které se obecně říká **color picking** [8]. Princip této metody je následující: Každému objektu náhodně vygeneruji jednoznačnou RGB barvu. Tuto hodnotu používám také jako identifikátor objektu. Při vykreslování objektů je používán speciální buffer, jehož obsah není zobrazen ve viewportu. Do tohoto bufferu jsou všechny objekty vykresleny vygenerovanou jednoznačnou barvou. Průhledné části objektů vykreslovány nejsou. Aplikován je jednoduchý shader, žádné texturování ani další efekty, jako třeba osvětlení, aplikovány nejsou. Poté funkce `readPixels` zjistí barvu pixelu v souřadnicích pozice kurzoru myši. Následně jsou podle získané barvy a identifikátorů detekovány objekty. Selekce objektů je potřebná například pro klikací menu, fotogalerii, atd.

## 4.7 Animace

Animace jsou jednou z nejdůležitějších součástí rozhraní. Dávají webové stránce další rozměr – časový průběh. Základem animací jsou rychle po sobě vykreslované snímky, v případě mého rozhraní se pohybuji někde mezi 20 až 30 snímky za sekundu, ale záleží na mnoha faktorech – velikost okna, výkon grafické karty, velikost operační paměti, složitost scény a výpočtů, atd. Mým záměrem bylo vytvořit plynulé, dobře vypadající animované transformace objektů.

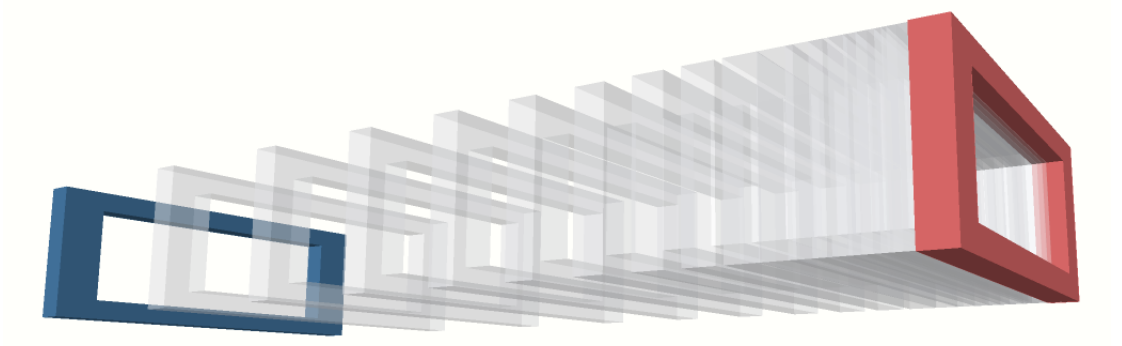
Pro animaci posuvů z jednoho bodu do druhého se obvykle používá lineární interpolace. Jedná se o proložení počátečního a koncového bodu úsečkou, body ležící na této úsečce použijeme pro animaci. Matematický vzorec pro lineární interpolaci je

$$linear \quad = \quad A * (1 - t) + B * t \tag{4.1}$$

kde $A$ je počáteční bod, $B$ je koncový bod a parametr $t$ může nabývat reálných hodnot od 0 do 1 [6].

Pro lépe vypadající animace jsem vztah pro lineární interpolaci aplikoval s konstantním parametr $t$ s tím, že je v každém kroku měněn počáteční bod na hodnotu vypočítanou z předchozího kroku. Výsledná animace má tedy **exponenciální průběh**.

V projektu jsem implementoval třídě `SceneObject` metodu `animateTranslate(dst, t)`, která interpoluje aktuální pozici objektu s parametrem `t` do pozice zadané vektorem `dst`. V případě, že vzdálenost počátečního a cílového bodu dosáhne určité minimální meze, je aktuální pozice objektu nastavena na cílovou a funkce vrací hodnotu `true`, v ostatních případech vrací `false` a objekt je posunut do nové spočítané pozice. Výsledkem použití této metody ve vykreslovací smyčce programu je animace plynulého zpomaleného pohybu objektu, Obrázek 4.7.



Obrázek 4.3: Ukázka interpolované animace s parametrem `t` = 0.2.

Pouze animace posunutí ovšem nestačí. Vzhledem k tomu, že ve scéně pohybuji kamerou, rotuji objekty a natáčím je směrem k uživateli, byla minimálně nutná ještě implementace animovaných rotací. Stejně jako posunutí a změna měřítka, tak i rotace se běžně provádí pomocí transformačních matic. Ty ale v případě rotací mají spoustu nevýhod, a proto jsem se rozhodl reprezentovat rotace pomocí kvaternionů, které nabízí sférickou lineární interpolaci pro požadované animace.

**Kvaterniony**

Kvaterniony jsou zobecněním komplexních čísel ve 3D [3]. Obsahují jednu reálnou a tři imaginární složky, je to tedy uspořádaná čtveřice $(w, x, y, z)$. V počítačové grafice se používají pro reprezentaci rotací (jednotkový kvaternion) a ve srovnání s maticemi mají několik výhod:

- Sférická lineární interpolace dvou kvaternionů poskytuje jejich nejkratší spojnici na povrchu hyperkoule.
- Na rozdíl od transformačních matic, které obsahují 9 nebo 16 hodnot, kvaterniony obsahují hodntoy pouze 4.

- Jednodušší skládání rotací než s maticemi.
- Kvaterniony odstraňují potenciální problém, tzv. gimbal lock, vznikající při reprezentaci rotace pomocí eulerových úhlů. K tomu může dojít při skládání rotací.

Obvykle postup použití kvaternionů je následující: transformační matice se převedou na kvaterniony, provede se sférická interpolace a interpolovaný kvaternion se převede zpět na matici.Vzorec pro sférickou lineární interpolaci mezi dvěma kvaterniony je

$$Slerp(q_1, q_2, t) \quad = \quad q_1 \frac{sin((1-t)\Omega)}{sin\Omega} + q_2 \frac{sin(t * \Omega)}{sin\Omega} \tag{4.2}$$

kde $q1$ a $q2$ jsou interpolované kvaterniony a parametr $t$ je reálné číslo od 0 do 1 .

Podobně jako u lineární interpolace i sférickou aplikuji krokově s konstantním parametrem $t$ tak, že má výsledná animace exponenciální průběh. Byly vytvořeny potřebné funkce pro převod matic na kvaterniony, pro vytvoření kvaternionu z úhlu a osy rotace. Analogicky jako u posuvů jsem pro animovanou rotaci objektu implementoval metody `animateRotate(angle, vec, t)`, kde `angle` je úhel rotace a `vec` osa rotace (vektor) a `animateQuatRotate(newQuaternion, t)`, kde `newQuaternion` je cílový kvaternion. Součástí každého objektu je vedle transformační matice, která je využívána zásadně pro translaci a změnu měřítka, také kvaternion, reprezentující jeho rotaci. Díky kvaternionům jsem také jednoduchým způsobem implementoval tzv. billboarding – natáčení objektů ke kameře, který se jinak řeší daleko složitější cestou.

Další interpolace implementovány nebyly, ale v úvahu připadají například interpolace kubická, spline, Kosinova, které prochází více body a zajistí jemnější průchod definovanou trajektorií.

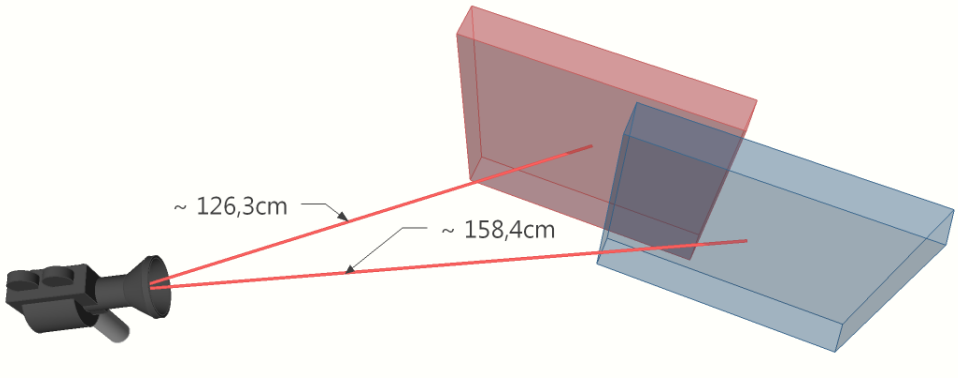
Obrázek 4.4: Ukázka animované rotace.

## 4.8 Řazení poloprůhledných objektů

Pro efektivnost rozhraní jsem se rozhodl používat poloprůhledné objekty. Poloprůhlednost má význam například v menu, kde neaktuální položky, nacházející se za těmi v popředí, jsou alespoň částečně vidět a uživatel tak má o nich přehled. Postupné zviditelňování a zprůhledňování objektů během animací je také nepostradatelné.

Ovšem vykreslování současně několika poloprůhledných objektů, navíc v kombinaci s průhlednými objekty, je problematická záležitost. Největší problém nastává v okamžiku,

kdy se objekty protínají, nebo překrývají například způsobem znázorněném na Obrázku 4.5. Podle zdroje [11] zatím neexistuje žádný standardní ideální algoritmus na řazení protínajících se ploch.




Obrázek 4.5: Demonstrace problému řazení poloprůhledných objektů.

Problém se obvykle řeší a obchází tak, že se nejprve vykreslí všechny neprůhledné objekty, a poté objekty průhledné od nejvzdálenějšího k nejbližšímu. Pokud nedochází k protínání objektů, je výsledek poměrně dobrý. Objekt, ale většinou není pouze jeden trojúhelník, tedy jedna vykreslená plocha, ale skládá se z mnoha malých elementů a pokud chceme korektně vykreslit celý poloprůhledný objekt, musíme správě seřadit i všechny jeho součásti. Korektně bychom museli kontrolovat každý vykreslovaný fragment.

Způsob řešení, který je sice přesný, ale výpočetně náročný a pomalý se nazývá Dual Depth Peeling. Dalším možným řešením je algoritmus Weighted Average, který je aproximací, je rychlý, ale pro hodnoty průhlednosti větší než 0,25 spočítá výsledné barvy tmavší [11].

Moje řešení spočívá v již zmíněném způsobu. Nejprve vykreslím všechny neprůhledné objekty, poté nastavím filtrování zadních stran (`gl.enable(gl.CULL_FACE)`), které zajistí lépe vypadající následně vykreslené poloprůhledné objekty, jejichž polygony nejsou seřazené. Následuje vykreslení poloprůhledných objektů, které jsou seřazeny podle svých středů. Scéna je vytvořena tak, aby co nejméně docházelo k protínání a blízkému překrývání objektů. Řešení je tedy pro uživatelské rozhraní dostačující.

## 4.9 Začlenění HTML obsahu

Skoro každá internetová stránka obsahuje nějaký textový obsah. Velmi zajímavou problematikou bylo propojení 3D rozhraní s HTML obsahem, například formátovaným textem. Rozhraní jako takové je vykreslováno do elementu `canvas`, plátna, které můžeme chápat jako bitmapový obrázek – rastrovou grafiku. Text a jiné HTML elementy stránky jsou ovšem vektorové.

V úvahu připadalo řešení veškerý textový obsah zobrazovat formou textur. To by bylo možné díky Canvas 2D API, které umí vykreslovat částečně formátovaný text jako 2D grafiku. V programu by se tedy vytvořila bitmapa s textem, která by byla texturována na nějaký objekt. Existují již JavaScriptové nástroje, které renderují HTML do elementu `canvas`. Například projekt **html2canvas**[10] vyvíjí knihovnu pro tento převod. Zatím je ale

[10] http://html2canvas.hertzen.com/

stále v počátečních fázích vývoje a konverze html není stoprocentní.

Další řešení problému nabízí CSS3. Původně jsem chtěl použít 3D transformační vlastnosti, které mají metody pro aplikování transformačních matic (`matrix3d`) a nastavení perspektivy. Nepodařilo se mi ale přijít na to, jak nastavit v CSS3 stejné perspektivní zobrazení jako ve WebGL. V aplikaci vytvářím vlastní perspektivní matici různými parametry, v CSS3 ale nelze nastavit perspektivní projekci maticí, nastavuje se pouze pomocí dvou parametrů (hloubka a střed perspektivy) a možnosti jsou velmi omezené. Navíc tyto vlastnosti aktuálně podporuje pouze jádro Webkit. CSS3 používají také jiné jednotky než WebGL.

Nakonec jsem se rozhodl využít nových vlastností CSS3, ale omezil jsem se pouze na vlastnosti podporované ve většině nových prohlížečů. Používám CSS3 transformační metody `scaleX` a `scaleY`, které mění měřítko HTML elementu a dále klasické absolutní pozicování. Je vytvořena vrstva HTML a dále 3D objekt, nad který se tato vrstva bude mapovat. Z maximálních a minimálních souřadnic 3D objektu

```
bodA = maximální pozice 3D objektu, na který se bude mapovat (největší
hodnoty souřadnic x, y, z)
bodB = minimální pozice 3D objektu, na který se bude mapovat
```

nastavím bodu A a B stejnou hodnotu z-ové souřanice, a přepočítám body na odpovídající pozice na plátně přes transformační matice a rozměry viewportu.

```
bodA = get2DPoint(bodA)
bodB = get2DPoint(bodB)
```

Z těchto hodnot poté spočítám změnu měřítka a pozici mapované HTML vrstvy

```
style["-webkit-transform"] = "scaleX("+(bodA[0]-bodB[0])/width+")
                              scaleY("+(bodB[1]-bodA[1])/height+")";
style["top"] = Math.round(bodA[1]-(height-(bodB[1]-bodA[1]))/2)+"px";
style["left"] = Math.round(bodB[0]-(width-(bodA[0]-bodB[0]))/2)+"px";
```

kde hodnota `width` je šířka a `height` výška 3D objektu a HTML vrstvy v maximálním měřítku.

Výsledné zobrazení HTML obsahu sice není skutečně ve 3D, ale špatně nevypadá.

# Kapitola 5

# Výsledky a testování rozhraní

V rámci této práce bylo implementováno uživatelské rozhraní podle návrhu, výjma sekce „Akce“. U sekce „Členové“ nejsou implementovány podrobnější informace o vybraném členovi. Koncept rozhraní by se dal aplikovat i na jiná témata než web kapely. Pro praktické využití by bylo dobré rozhraní více optimalizovat, jelikož přenáší velké množství dat a s pomalým internetovým připojením se stránka načítá poměrně dlouho. Nabízí se optimalizace kompresí přenášených dat (souřadnic, obrázků), a také by bylo výhodnější použití jednodušších 3D modelů. Například v sekci „Členové“ má některý nástroj i 20 tisíc primitiv a podle následujících testů se ukázalo, že je to v kombinaci s dalšími podobnými objekty opravdu mnoho a aplikace se zasekává. Během vývoje a experimetování s WebGL jsem narazil i na takové problémy, že se mi na kratší dobu zmrazil počítač. Způsobeno to bylo pravděpodobně vyčerpáním operační paměti počítače. Ukázalo se, že WebGL je opravdu mocný nástroj a při nesprávném použití je zatím v této brzké fázi vývoje nebezpečný. Aplikace je funkční v nových verzích Google Chrome, a také v nejnovějších prohlížečích Firefox. Ostatní prohlížeče jsem netestoval, jelikož zatím nemají ve výchozím nastavení podporu WebGL aktivní. Na Obrázku 5.1 je snímek výsledné aplikace.

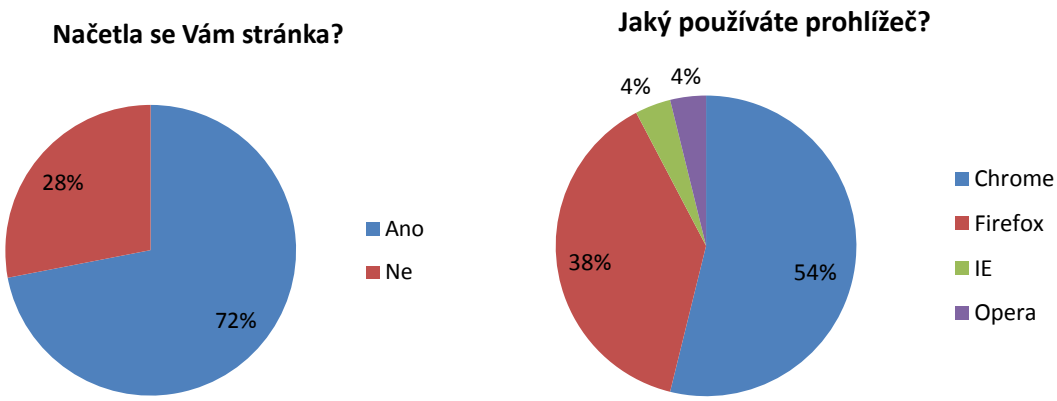
## 5.1 Testování uživateli a vyhodnocení

Uživatelské rozhraní jsem se pokusil otestovat pomocí uživatelských testů. Aplikaci jsem vystavil na internet a vytvořil dotazník. Aplikace je v experimentální fázi, proto jsem nečekal moc povzbudivé výsledky. Vyhodnocení jsem provedl v době, kdy dotazník vyplnilo 25 lidí. Někteří neodpovídali na všechny dotazy. V dotazníku jsem popsal jak ovládat scénu, jak používat myš a klávesnici. Také jsem uvedl informace o podpoře prohlížečů.

Základní otázkou bylo, zda se webová stránka uživateli vůbec načetla. Výsledky mě celkem překvapily. Stránka se na první pokus načetla cca 72% dotázaných (Obrázek 5.2), což bylo více než jsem očekával. Velká většina dotázaných používala prohlížeč Google Chrome 18 a Firefox 12 - převážně nejnovější verze, a jen velmi malé procento Operu a Internet Explorer (Obrázek 5.2 vpravo). Na dotaz „Jak starý máte počítač?“ odpověděla většina v průměru dva roky. Menšině, kteří měli počítače starší (4, 5 a více let) podle očekávání WebGL převážně nefungovala.

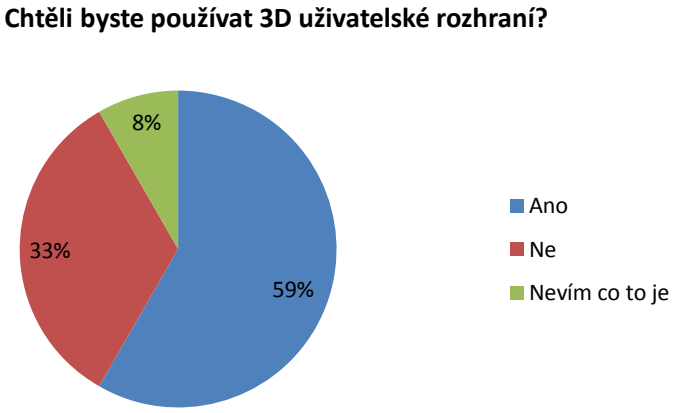
Další dotaz zjišťoval, zda by uživatelé vůbec měli zájem používat 3D uživatelské rozhraní. 59% tázaných by 3D uvítalo, 33% ne a zbytek nevěděl, co to vlastně znamená. Obrázek 5.3.


Obrázek 5.1: Náhled výsledné aplikace (sekce „Členové“).



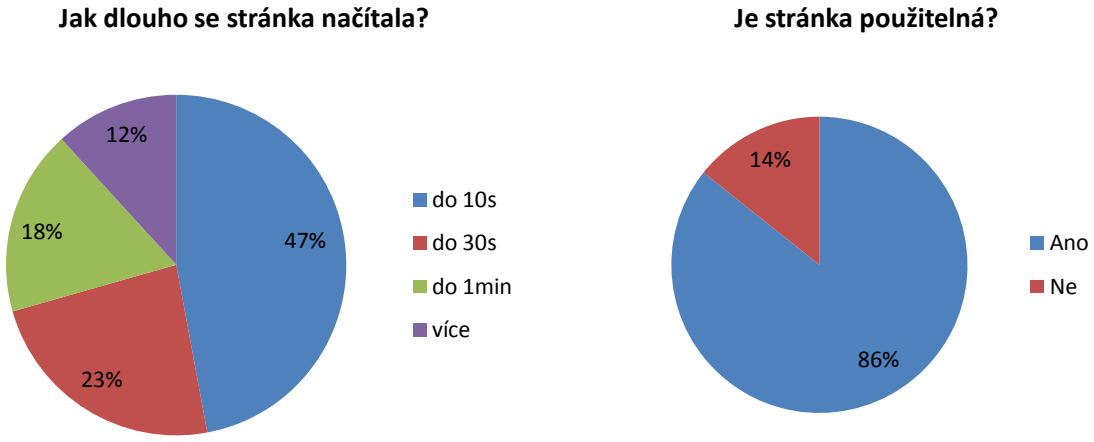

Obrázek 5.2: Grafy výsledků z dotazníku.

Velice mě zajímalo také jak dlouho se lidem webová stránka bude načítat. Během načítání dochází k přenosu poměrně velkého obejmu dat a doba načítání závisí na rychlosti připojení. Ptal jsem tedy na dobu, po kterou se stránka načítala. Odpovědi mě překvapily. Do 10 sekund se stránka načetla 47% dotázaných, což se blíží době při načítání z lokálního serveru. Některým se stránka načetla do 30 sekund, což také není špatné. Uživatelů s pomalejším připojením, kteří čekali déle nebo dokonce více jak minutu, bylo kolem 20%. Tento graf je na Obrázku 5.4 vlevo.

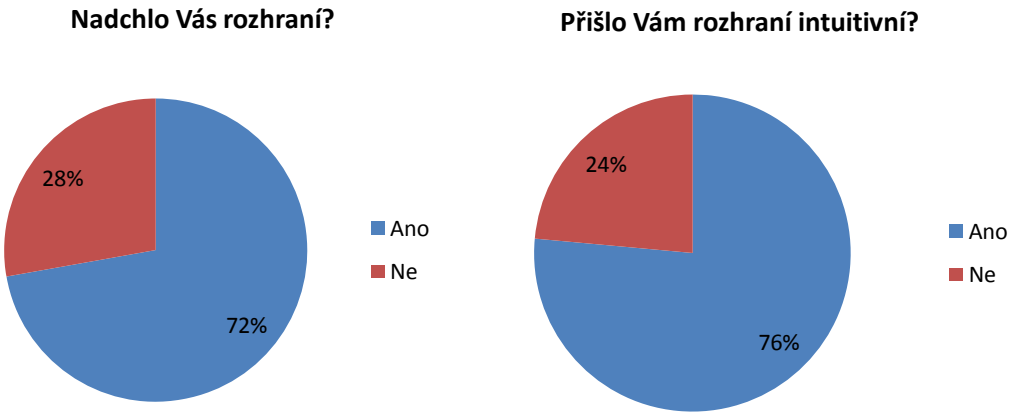



Obrázek 5.3: Graf výsledků z dotazníku.

Dále jsem směřoval otázky přímo k rozhraní. Těm, kterým se stránka podařila načíst, jsem položil otázku na intuitivnost rozhraní, použitelnost, a také jsem se zeptal na to, jestli je rozhraní celkově nadchlo. Výsledky jsou poměrně pozitivní. Nadšení zaznamenalo asi dvě třetiny uživatelů (Obrázek 5.5 vlevo). Lidem přišlo rozhraní převážně intuitivní (Obrázek 5.5 vpravo) a 86% si dokonce myslí, že by byla stránka použitelná (Obrázek 5.4 vpravo). Celkový vzhled přišel dotazovaným nadprůměrný. Ptal jsem se také na konkrétní části prezentace. Menu bylo podle většiny průměrné. Fotogalerie se spíše zalíbila a sekce o členech kapely působila dobrým dojmem. Zde jsem ale zaznamenal několik připomínek na zasekávání a dlouhé načítání sekce. Animace se jevily podle statistik jako zajímavé a působivé. Celkové hodnocení tedy není špatné.

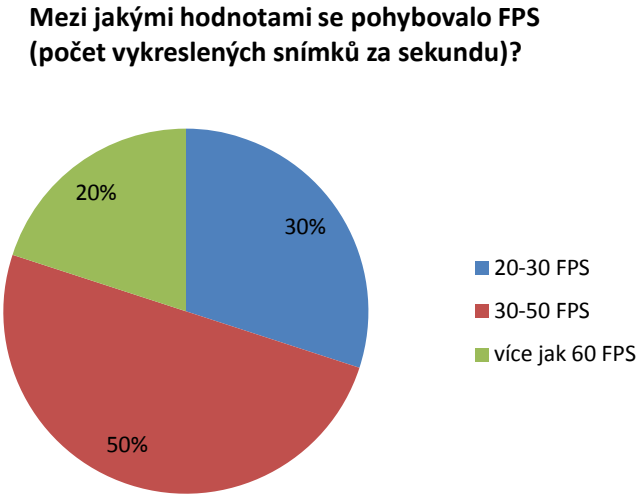



Obrázek 5.4: Grafy výsledků z dotazníku.




Obrázek 5.5: Grafy výsledků z dotazníku.

Abych získal přehled o výkonosti aplikace v různých prostředích, jedna otázka směřovala na hodnotu FPS – počet vykreslených snímků za sekundu. Tato hodnota je zobrazena v levém dolním rohu webové stránky. Získané informace mě překvapily asi nejvíce, jelikož jsem čekal hodnoty podobné těm, které se zobrazují mně. V maximalizovaném okně na 19' monitoru dostávám hodnoty kolem 20 až 30 FPS a jedná se o 3 roky starý notebook. Testování ovšem ukázalo, že se průměrné hodnoty pohybují mezi 30 až 50 FPS – Obrázek 5.6, a to je pozitivní zjištění.




Obrázek 5.6: Graf výsledků z dotazníku – hodnoty FPS.

Díky dotazníku jsem také obdržel spoustu postřehů, nápadů a připomínek, které byly jak negativní, tak pozitivní, a převážně hodně subjektivní. Ukázala se pestrost uvažování různých lidí. Spousta uživatelů vidí 3D jako budoucnost. Data získaná z dotazníků jsou v příloze této práce.

# Kapitola 6

# Závěr

Cílem práce bylo nastudování možností zobrazování 3D grafiky ve webovém prohlížeči se zaměřením na akcelerované vykreslování, použitelnost, přenositelnost. Prozkoumal jsem současné technologie, a dále jsem se zaměřil na WebGL. Tuto technologii jsem prostudoval podrobněji. Dále byl vytvořen návrh možného 3D webového rozhraní, který se konkretizoval na web reprezentující hudební skupinu. Velká část práce byla věnována implementaci WebGL frameworku, který ve výsledku umožnil vytvoření interaktivní 3D webové prezentace. Podstatnou část práce zahrnovalo experimentování s WebGL a pokusy implementovat různé 3D prvky rozhraní. V závěrečných fázích práce bylo implementované rozhraní otestováno uživateli s přínosem zajímavých výsledků. Během vypracovávání projektu a bakalářské práce jsem získal veliké množství informací a zkušeností, které bych mohl v budoucnu uplatnit.

WebGL má podle mého názoru velkou budoucnost. Postupem času se vývojem technologií a prohlížečů pravděpodobně stane běžnou záležitostí, která bude dostupná všude. Výkonost WebGL bude také záviset na schopnostech JavaScriptu. 3D ve vebovém prohlížeči bude pravděpodobně aplikováno v největší míře do her. Ale můžeme čekat i přechod běžných webových stránek do jejich 3D podoby, prezentace 3D předmětů, a různých věcí, které mohou v prostoru předávat informace efektivněji.

Implementovaný projekt by v budoucnu bylo možné optimalizovat, dodělat více 3D prvků a obsahu. Implementovat URL odkazy v rámci stránky. Navržený framework je možné aplikovat na více projektů a dále ho rozšiřovat, zefektivňovat, a na rozdíl od jiných obecných WebGL frameworků se může zaměřovat vyloženě na uživatelské rozhraní a nahradit tak klasické 2D html elementy.

# Literatura


[1] Anyuru, A. *Professional WebGL Programming: Developing 3D Graphics for the Web* [online]. 2012 [cit. 2012-05-01]. Dostupné na: <http://books.google.cz/books?id=F75bsvlmqQsC>. ISBN 978-1-1199-4059-3.

[2] *AREA: The Status of WebGL in Relation to 3D Game Development* [online]. 2011-10-19 [cit. 2012-01-29]. Dostupné na: <http://area.autodesk.com/blogs/chris/the_status_of_webgl_in_relation_to_3d_game_development/>.

[3] Dam, E. B. et al. *Quaternions, Interpolation and Animation* [online]. 1998 [cit. 2012-05-01]. Dostupné na: <http://www.itu.dk/people/erikdam/DOWNLOAD/98-5.pdf>.

[4] Dong, W. *Web goes 3D – does advertising too? The WebGL, Silverlight, and Molehill wars* [online]. 2011-04-28 [cit. 2012-01-29]. Dostupné na: <http://www.domusinc.com/blog/2011/04/web-goes-3d-does-advertising-too-the-webgl-silverlight-and-molehill-wars/>.

[5] FairerPlatform. *New in OS X Lion: Safari 5.1 brings WebGL, Do Not Track and more* [online]. 2011-05-03 [cit. 2012-01-28]. Dostupné na: <http://fairerplatform.com/2011/05/new-in-os-x-lion-safari-5-1-brings-webgl-do-not-track-and-more/>.

[6] Kučera, O. *Knihovna pro výpočet šumů používaných v procedurálním texturování.* Brno: VUT – FIT, 2007. Diplomová práce.

[7] Malý, M. *Zlepšení výkonu animací v JavaScriptu* [online]. 2011-03-15 [cit. 2012-04-07]. Dostupné na: <http://www.zdrojak.cz/clanky/zlepseni-vykonu-animaci-v-javascriptu/>.

[8] Marucchi Foino, R. *Game and Graphics Programming for IOS and Android with OpenGL Es 2.0* [online]. 2012 [cit. 2012-05-01]. Dostupné na: <http://books.google.cz/books?id=9QiKLOZ82roC>. ISBN 978-1-1199-7626-4.

[9] O'Brien, T. *Microsoft decides to pass on WebGL over security concerns* [online]. 2011-06-17 [cit. 2012-01-29]. Dostupné na: <http://www.engadget.com/2011/06/17/microsoft-decides-to-pass-on-webgl-over-security-concerns/>.

[10] *Rich Internet Application Market Share* [online]. [cit. 2012-01-28]. Dostupné na: <http://www.statowl.com/custom_ria_market_penetration.php>.

[11] Rigazzi, A. *Order Independent Transparency* [online]. 2008-07-01 [cit. 2012-04-07]. Dostupné na: <http://www.slideshare.net/acbess/order-independent-transparency-presentation>.

[12] Rost, J. R. et al. *OpenGL shading language*. Třetí vydání. Upper Saddle River: Addison-Wesley, 2010. ISBN 978-0-321-63763-5.

[13] *Securing JAVA: What Is a Malicious Applet?* [online]. [cit. 2012-04-07]. Dostupné na: <http://www.securingjava.com/chapter-four/chapter-four-1.html>.

# Příloha A

# Obsah CD

Na přiloženém disku se nachází následující soubory a složky:

- **obrazky** – adresář s obrázky návrhu rozhraní.
- **projekt** – adresář s projektem.
- **projekt/fotogalery** – adresář s fotkami použitými ve fotogalerii.
- **projekt/lib** – adresář s JavaScritpovými knihovnami, včetně frameworku (webgl-framework.js).
- **projekt/members** – adresář s fotkami použitými v sekci „Akce“.
- **projekt/models** – zdrojové soubory modelů a materiálů.
- **projekt/shaders** – zdrojové soubory shaderů.
- **projekt/index.html** – výsledná webová stránka s 3D rozhraním.
- **projekt/style.css** – kaskádový styl webu.
- **dotaznik.pdf** – tabulky výsledků z dotazníku v pdf.
- **projekt.pdf** – tato práce v pdf.
- **plakat.pdf** – demonstrační plakát.
- **videa** – adresář s videi demonstrujícími rozhraní.

# Příloha B

# Manuál

Pro zprovoznění aplikace je vyžadováno prostředí serveru, z důvodu používání AJAXu.

**Zprovoznění aplikace na serveru**:
1. Nahrát zdrojové soubory (adresář projekt) na server.
2. Otevřít na serveru **projekt/index.html**.

**Spuštění webového rozhraní bez instalace**:
Webová stránka s implementovaným 3D rozhraním je v současnosti dostupná na adresách:
http://www.stud.fit.vutbr.cz/~xklepa01/ISP
http://cleposh.kapela.marshal.cz/ISP

**Ovládání rozhraní**

- Stisknutím a držením pravého tlačítka myši a jejím pohybem se můžete orientovat ve scéně.
- Pro orientaci můžete využít také šipky na klávesnici (vhodné pro prohlížení fotografií s kurzorem myši cca uprostřed obrazovky).
- Vyzkoušejte kolečko myši (otáčí menu).
- Pro výběr sekce je možné klikat na menu nebo přímo na sekce.
- Při prohlížení členů nebo fotografií nemusíte klikat, stačí najet kurzorem na prvek a chvíli počkat.