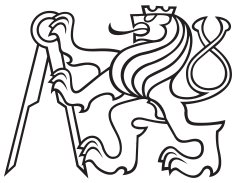
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Aplikace pro výuku programování 3D grafiky

Patrik Schiller

Školitel: Ing. Petr Felkel, Ph.D.
Květen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Schiller** Jméno: **Patrik** Osobní číslo: **474758**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Aplikace pro výuku programování 3D grafiky

Název bakalářské práce anglicky:

Application for teaching 3D graphics programming

Pokyny pro vypracování:

Bakalářská práce se zabývá vývojem aplikace pro podporu výuky programování grafiky. Jejím úkolem je co nejvíce přiblížit problematiku studentům a zájemcům. Zároveň umožňuje jednoduchý přístup ke všem příkladům pomocí webového rozhraní a tím odbourává nutnost instalace vývojového prostředí a kompilace programů. To může být složité, časově náročné a ve výsledku také pro studenty odrazující.
Seznamte se s demonstračními příklady v předmětu (rastrová) počítačová grafika. Proveďte rešerši demonstračních programů používaných na vybraných univerzitách a dem v internetových tutoriálech.
Navrhněte sadu dem pro podporu výuky rastrové grafiky a naprogramujte je ve WebGL a Javascriptu. Jako zdroj témat použijte přednášky předmětu Programování Grafiky, jehož studenty by měla aplikace ve studiu podporovat.
Vybalancujte složitost dem tak, aby na jedné straně příliš nezjednodušovala a na straně druhé příliš nekomplikovala vysvětlovanou látku jinými problémy/tématy.
Zkuste vymyslet jinou strukturu dem, která může usnadnit pochopení problematiky oproti stávajícím demům.
Pokuste se navrhnout způsob testování přínosu aplikace pro uživatele, její přehlednosti a použitelnosti. Jako výchozí bod můžete použít knihu [3].

Seznam doporučené literatury:

[1] Žára, Ondřej: JavaScript – Programátorské techniky a webové technologie,ComputerPress, Brno, 2015.
[2] Žára, J. a kol.: Moderní počítačová grafika, Computer Press, Brno, 2004.
[3] Krug, Steve: Nenuťte uživatele přemýšlet, Computer Press, Brno, 2006.
[4] Bailey, M. and Cunningham, S.: Computer Graphics Shaders: Theory and Practice, Second Edition, CRC Press, 2011.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Felkel, Ph.D., Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **12.02.2020** Termín odevzdání bakalářské práce: **22.05.2020**

Platnost zadání bakalářské práce: **30.09.2021**

Ing. Petr Felkel, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Děkuji Ing. Petru Felkelovi, Ph.D. za vedení mé bakalářské práce, za cenné rady a připomínky, které mi poskytl. Dále bych chtěl poděkovat Ing. Martinu Komárkovi a Ing. Miroslavu Buršovi, Ph.D. za odbornou pomoc. Poděkování patří také všem, kteří se podíleli na testování výukové aplikace. V neposlední řadě děkuji své rodině a přítelkyni za podporu během psaní této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 22. května, 2020

Podpis: .....

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, May 22, 2020

Abstrakt

Práce se zaměřuje na implementaci webové aplikace pro podporu výuky programování počítačové grafiky, jejímž cílem je usnadnit studentům pochopení vývoje grafických aplikací ve WebGL. Základem aplikace jsou demonstrační příklady, které prezentují probíranou látku odpovídající osnovám předmětu *PGR (Programování Grafiky)*. Příklady jsou doplněny výukovým textem, obrázky a ukázkami zdrojového kódu.

Výuková aplikace je založena na principu *SPA (Single-Page Application)* umožňující rychlé procházení obsahu. K implementaci uživatelského rozhraní byl využit framework `React.js` a `Bootstrap`. Obsah kapitol je uložen na serveru a asynchronně stahován pomocí technologie *AJAX*. Výukové příklady jsou vyvíjeny v jazyce JavaScript dle standardu ES6+ v kombinaci s rozhraním `WebGL2`. Díky tomu je výukový materiál včetně příkladů volně dostupný bez potřeby instalace vývojového prostředí. Uživatelé tak mohou obsah číst kdykoliv a kdekoliv nezávisle na použitém zařízení.

Výsledkem práce je série výukových kapitol popisujících problematiku počítačové grafiky od úplných základů. Vzhledem k náročnosti strukturování obsahu je prozatím aplikace určena pokročilejším zájemcům, především studentům předmětu PGR. Většina příkladů je interaktivních, což zjednodušuje pochopení vyučované problematiky.

Klíčová slova: počítačová grafika, výuka, výuková aplikace, WebGL, OpenGL, Javascript, 3D, DCGI

Školitel: Ing. Petr Felkel, Ph.D.

Abstract

The work focuses on the implementation of a web application to support the teaching of computer graphics programming, which aims to make it easier for students to understand the development of graphic applications in WebGL. The application is based on demonstration examples, which present the subject matter corresponding to the syllabus of the subject *PGR (Computer graphics programming)*. The examples are supplemented by tutorials, pictures and source code examples.

The educational application is based on the principle of *SPA (Single-Page Application)* enabling fast browsing of the content. The `React.js` and `Bootstrap` frameworks were used to implement the user interface (GUI). The content of the chapters is stored on the server and fetched asynchronously using *AJAX* technology. Tutorials are developed in JavaScript according to the ES6+ standard in combination with the WebGL2 interface. Thanks to this, the teaching material, including examples, is freely available without the need to install a development environment. Users can read the content anytime and anywhere, regardless of the device used.

The result of the work is a series of teaching chapters describing the matters of computer graphics from the ground up. Due to the complexity of structuring the content, the application is currently intended for more advanced candidates, especially students of the subject PGR. Most of the examples are interactive, which simplifies the understanding of the taught matters.

Keywords: computer graphics, education, educational application, WebGL, OpenGL, Javacript, 3D, DCGI

Title translation: Application for teaching 3D graphics programming

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Počítačová grafika v dnešní době zasahuje do širokého spektra průmyslových odvětví, počínaje herním průmyslem přes kinematografii a multimédia až po různé typy vizualizací. Počítačové vizualizace jsou velmi důležitou součástí pro podporu profesních oborů, jako jsou architektura, věda či lékařství. Díky počítačové grafice je možné zobrazit velmi přesná data, která si bez této pomoci lze jen těžko představit. Příkladem může být vizualizace projektovaných staveb, které lze díky počítačové grafice (programům CAD) vnímat ze všech úhlů pohledu ještě před jejich fyzickým postavením. To vede k razantnímu zjednodušení práce a také může předcházet chybám, které jsou bez vizualizací mnohem hůře rozpoznatelné.

S postupem doby se stále vyvíjí technologie používané v počítačové grafice a s tím přichází také potřeba implementace změn do existujících grafických programů. Většina těchto programů staví svá vykreslovací jádra na low-level grafických rozhraních, mezi která se řadí `DirectX`, `OpenGL` a její nástupce `Vulkan`. Studium těchto rozhraní (a tedy programování počítačové grafiky) je velmi důležité a potřebné pro další vývoj.

Výuka programování počítačové grafiky je pro studenty náročná. Obsahuje velké množství látky z několika oborů, mezi které patří lineární algebra, algoritmizace či programování v jazycích jako jsou C++ či Java. Student se musí naučit poměrně velké množství teorie, než se vůbec pustí do implementace prvního programu.

Pro snazší pochopení vyučované látky jsou vhodné demonstrační příklady (zkráceně demo příklady), které slouží jako praktické ukázky. Realizují pouze probíranou část problematiky a student tak není zatěžován ostatním kódem, který s aktuálně probíranou látkou tolik nesouvisí.

Programování počítačové grafiky je velmi rozsáhlé téma a existuje několik způsobů, jak k němu přistupovat. Pro začátečníky je nejlepší volbou jazyk a programovací rozhraní `Processing`, který je mimo jiné využíván velkým množstvím umělců v oboru multimédií. Processing staví základy vykreslování na OpenGL, která je sama o sobě poměrně složitým rozhraním.

OpenGL se nejčastěji vyskytuje v kombinaci s programovacím jazykem C++, díky čemuž je možné psát vysoce optimalizované grafické programy. Častá je také kombinace s jazykem Java, což umožňuje vývoj multiplatformních aplikací. Ty jsou na tom ale výkonově hůř a to díky jejich kompilaci

za běhu programu. Jazyků, ve kterých je možné psát OpenGL aplikace, je samozřejmě více a je čistě na programátorovi, který si vybere. Mezi další jazyky patří například `Python`, `C`, `Pascal`, `Visual Basic` a jiné ... [26].

Pro vývojáře webových aplikací existuje alternativa v podobě `WebGL`. Ta reprezentuje podmnožinu OpenGL funkcionality, pomocí které lze vykreslovat 3D grafiku do webové stránky s pomocí programovacího jazyka `JavaScript`.

Výuce základů programování počítačové grafiky se věnuje předmět *PGR*, který spadá pod katedru počítačové grafiky a interakce *DCGI, ČVUT FEL*. Výuka se zabývá vývojem grafických desktopových aplikací a to především s pomocí jazyka C++ a OpenGL. Studenti mají ovšem možnost použít i jiné programovací jazyky a grafická rozhraní, pokud splňují podmínky předmětu. Mezi ty patří především podmínka, že vybrané rozhraní nijak nezjednodušuje implementaci probírané látky.

Cílem práce je provedení rešerše volně dostupných výukových materiálů a následná implementace výukové aplikace. Aplikace se má skládat z demo příkladů, jejichž obsah je založen na osnovách předmětu PGR. Na základě provedené rešerše bude vybrán nejlepší způsob rozložení výukových kapitol a prezentace vyučovaného obsahu. Inspirací pro budoucí aplikaci budou především ostatní výukové weby a volně dostupný obsah jiných univerzit.

Smyslem práce je zpřístupnit výukový materiál co nejvíce zájemcům a to především studentům předmětu PGR. Díky využití webových technologií `JavaScript` a `WebGL` bude obsah s příklady stále dostupný bez potřeby instalace vývojového prostředí. Výukové kapitoly budou sestávat z interaktivních demo příkladů zjednodušujících pochopení problematiky.

Kapitola 2
Rešerše

Studium programování počítačové grafiky je čím dál jednodušší a to především díky rostoucímu počtu publikací a internetových zdrojů. Přesto se ale stále jedná o odvětví, které je méně známé, a kvalitních tutoriálů není mnoho. Kdokoliv se odhodlá ke studiu této problematiky, musí být schopen číst články v angličtině a umět si alespoň zjednodušeně představit problém, na kterém pracuje. Počítačová grafika je plná matematiky a vzorců, a různé webové tutoriály řeší některé problémy rozdílnými způsoby.

Kapitola se zabývá rešerší existujících výukových materiálů (tutoriálů), které se týkají programování počítačové grafiky. Hlavním tématem jsou výukové portály zaměřující se na knihovny `OpenGL` a `WebGL`. Důraz byl kladen především na přehlednost, způsob výuky a užitečnost tutoriálu jako takového.

2.1 Výuka na ostatních univerzitách

V případě, kdy si student nevystačí s materiály poskytovanými jeho fakultou, může se nejprve poohlédnout po volně dostupných materiálech ostatních univerzit. U většiny českých vysokých škol se bohužel volně dostupný materiál na téma programování grafiky nenachází (až na předmět PGR, DCGI). Například **VUT** v Brně vyučuje podobný předmět (*Počítačová Grafika - PGR*), ovšem veškeré materiály jsou dostupné až po přihlášení studenta do systému. Je proto nutné se poohlédnout v zahraničí...

MIT

Snad nejznámější univerzitou poskytující výukové materiály v oblasti informačních technologií je `MIT` (*Massachusetts Institute of Technology*). Univerzita zdarma nabízí komplexní kurz zabývající se základy počítačové grafiky v OpenGL (`6.837`), který svým obsahem připomíná náplň předmětu PGR (ČVUT, FEL). Tento kurz spadá pod stránky `MIT Open Courseware`, jež publikuje většinu výukových materiálů univerzity online a zadarmo [29]. Přednášky jsou volně ke stažení a jsou doplněny podpůrnými úkoly. K dispozici jsou také ukázky testů a závěrečných zkoušek.

Samotné přednášky pro studenta předmětu PGR nemusí mít velkou hodnotu. Ovšem série přiložených výukových příkladů (záložka Assignments) je velmi dobrou formou, jak pochopit problematiku, která nemusí být v případě předmětu PGR dostatečně vysvětlena. Student tak má k dispozici více úhlů pohledu a přístupů k dané problematice, a pochopení tak může být o něco snazší.

Vhodným doplňkem výuky mohou také být video nahrávky přednášek kurzu z MIT, které jsou dostupné na YouTube kanále `Justin Solomon` [1]. O tento kanál se stará sám přednášející kurzu počítačové grafiky - pan Justin Solomon.

Na stránkách MIT lze také nalézt sadu demo příkladů pro demonstraci základů OpenGL [30]. Jejich autorem je pan Gordon Wetzstein, který v dnešní době vyučuje na *Standford University*.

University of California

Dalším zdrojem volně dostupných výukových materiálů na téma programování počítačové grafiky je *University of California, Berkeley*. Přednášky kurzu počítačové grafiky (`cs184/284a`) [31] popisují základy vykreslování, transformace, křivky, ale také pokročilé metody, mezi které patří Ray Tracing nebo VR. Doplňující úkoly jsou bohužel dostupné pouze studentům univerzity.

Kampus *Davis*, spadající pod *University of California*, poskytuje video záznamy z přednášek o počítačové grafice na YouTube kanále `UC Davis Academics` [2], ve kterých je detailně a srozumitelně rozebrána teorie počítačové grafiky.

Především video forma přednášek může být některým studentům (PGR) užitečná.

Standford University

Pro potřeby doplňujících materiálů (nebo čistě studijních) je možné také navštívit web Standfordské univerzity, konkrétně stránku kurzu počítačové grafiky (`CS 148`) [32]. K dispozici je několik přednášek zabývajících se problematikou OpenGL, které se z velké části kryjí s náplní předmětu PGR. Bohužel v případě příkladů (úkolů) je k dispozici pouze jejich zadání, ovšem zdrojové kódy k dispozici nejsou. Vzhledem k tomu, že k vypracování úkolů je potřeba základní kostra programu obsahující již předpřipravené funkce, tak jsou tyto příklady spíš bezcenné (slouží pouze studentům univerzity).

[1] `https://www.youtube.com/user/justinmsolomon/`
[2] `https://www.youtube.com/user/ucdavisedu`

2.2 OpenGL tutoriály

V případě předmětu PGR (a tedy OpenGL) jsou nejrelevantnějšími stránkami právě ty, které se zabývají OpenGL. Obsahují fragmenty kódu, které lze jednoduše zkopírovat a vložit, a s trochu štěstí aplikaci hned spustit.

LearnOpenGL.com

Asi nejznámější stránkou zabývající se výukou OpenGL je `learnopengl.com` [1]. Obsah stránky pracuje s OpenGL 3.3 s využitím programovacího jazyka C++. Problematika je velmi důkladně rozepsána v tematicky oddělených kategoriích, každá čítající několik kapitol. Stránka se zabývá popisem od úplných základů, ovšem obsahuje i kapitoly s velmi pokročilými tématy. Autor navíc vydal e-knihu (`Learn OpenGL` - *An offline transcript of learnopengl.com*) [18], která slouží jako offline náhrada (přepis) samotného webu.

Výukové kapitoly jsou velmi dobře zpracované, obsahují mnoho tematických obrázků a nákresů, které umožňují látku lépe pochopit. Každá kapitola je doplněna ukázkami kódu a odkazem na výsledný zdrojový kód vycházející z probírané látky. V kapitolách se také nachází menší úkoly, které čtenář může zkusit vyřešit. Jejich řešení je samozřejmě také přiloženo.

Všechny kapitoly navíc obsahují diskuzi, ve které se čtenáři mohou ptát na různé problémy spojené s konkrétní kapitolou. Často je vidět, že odpovídá samotný autor stránky, ovšem většinou si uživatelé radí sami navzájem.

Web také spolupracuje se zahraničními autory, kteří obsah stránky průběžně překládají do svého jazyka. Díky tomu vzniká mezinárodní komunita vývojářů v OpenGL. K aktuálnímu datu (05.05.2020) existuje pět oficiálních překladů.[3]

opengl-tutorial.org

Dalším poskytovatelem velmi kvalitního výukového materiálu je web `opengl-tutorial`[9]. Výuka se zabývá OpenGL ve verzi 3.3 a vyšší s využitím jazyka C++. Styl výkladu je méně formální, než je tomu u ostatních stránek. Ovšem to určitě není špatná vlastnost a mnoha čtenářům to může vyhovovat. Kapitoly se zabývají OpenGL od základů, včetně nastavení vývojářského prostředí. Text je doplněn názornými ukázkami a na konci kapitol se nachází doplňující úkoly.

Tento web také obsahuje překlady do cizích jazyků. Na výběr je konkrétně ze sedmi, ovšem množství přeloženého materiálu se liší. Pro některé jazyky je přeloženo pouze pár kapitol, pro jiné je zase přeložena většina obsahu.

Pro potřeby uživatelů existuje offline verze kapitol ve formátu PDF [22]. Zdrojové kódy k jednotlivým kapitolám jsou volně ke stažení.

[3] `https://learnopengl.com/Translations`

NeHe

Na české scéně se nachází pár výukových webů, které se zabývají OpenGL. Prvním z nich je web `NeonHelium (NeHe)`[4], za kterým stojí pan Michal Turek a několik dalších autorů.

Jedná se o výukový web s dlouhou historií a obsahuje 48 kapitol. Velkou nevýhodou je ovšem zaměření tutoriálů na OpenGL 2.0 bez využití GLSL shaderů, což je v dnešní době zastaralý přístup. Na druhou stranu se kapitoly zaměřují i na velmi pokročilé problémy, jejichž popis řešení může být užitečný i v dnešní době. I tato stránka obsahuje offline zpracování ve formě PDF knihy [19]. Její vydání je aktuální k datu 29.02.2004 a obsahuje všech 48 vydaných kapitol.

Díky svému obsahu byla tato stránka doporučována začátečníkům i později po vydání OpenGL 3.0 [20]. Uživatelé fóra argumentují tím, že je OpenGL2.0 jednodušší na pochopení, než její novější verze.

Root.cz

Další podobný tutoriál se nachází na portále `root.cz` [21]. Obsahuje 34 kapitol, jejichž autorem je pan Pavel Tišnovský. I zde je bohužel probírána stará verze OpenGL, která se v dnešní době téměř nepoužívá. Kapitoly navíc nejsou tak dobře přehledné, jako tomu je u webu NeHe. Poslední článek zde vyšel 24.02.2004.

2.3 WebGL tutoriály

Pro výuku programování počítačové grafiky, v kontextu webových technologií, také existuje několik výukových stránek. Jejich výhodou je, že prezentovanou problematiku je možné demonstrovat přímo na stránce. Čtenáři tak stačí kompatibilní webový prohlížeč a nic dodatečného nemusí stahovat. Tato varianta je vhodná i pro studium v místech, kde čtenář nemá k dispozici počítač a musí se spokojit například s mobilním telefonem či tabletem. Ukázky v kapitolách mohou být interaktivní a čtenář tak má možnost daný problém lépe pochopit.

WebGLFundamentals.org

Jednou z nejpropracovanějších stránek pro výuku WebGL
je `WebGLFundamentals.org`. Web obsahuje mnoho kapitol v několika kategoriích. Čtenáři je vše vysvětleno od úplných základů a to včetně návaznosti na programovací jazyk JavaScript a HTML.

Text kapitol je založen na rozhraní WebGL 1.0, které je sice starší, ale v dnešní době stále využívané. Na titulní straně se navíc nachází odkaz na projekt `WebGL2Fundamentals.org`[5],

[4] `http://nehe.ceske-hry.cz/subdom/nehe/autori.php`
[5] `https://webgl2fundamentals.org/`

který rozvádí zdejší problematiku o přístupy pomocí novějšího rozhraní WebGL 2.0. Obsah obou stránek je vesměs podobný, ovšem na stránce WebGL2Fundamentals je aktuálnější.

Obě stránky obsahují několik překladů, opět s různým počtem přeložených kapitol. Na přiložené GitHub stránce [6] je navíc návod, jak vytvořit vlastní lokalizaci (překlad) a tím přispět k multijazyčnosti výukového materiálu.

Pro potřeby výuky také slouží pomocná knihovna `WebGLFundamentals API`. Ta se snaží zredukovat zbytečné množství kódu ve výukových příkladech (především tzv. boilerplate kódu, viz kapitola 3.2). Díky tomu není čtenář rušen nadbytečným kódem a celek se stává přehlednějším (na podobném principu funguje také knihovna `CTUGL` (kapitola 5.3), která vznikla pro potřeby této výukové aplikace). Knihovna WebGLFundamentals API obsahuje především funkce, které zjednodušují práci s 3D matematikou, a výrazně redukují počet WebGL příkazů potřebných pro základní používání WebGL. Čtenář zdrojového kódu se díky tomu může soustředit čistě na problematiku dané kapitoly a zbytek je odstíněn zmíněnou knihovnou [23].

2.4 YouTube

Vzhledem k tomu, že někteří studenti preferují mluvené slovo oproti čtení textu, je velmi dobrým zdrojem výukového materiálu portál `YouTube.com`. Videí zabývajících se OpenGL i WebGL je na YouTube mnoho, ovšem za většinou z nich stojí amatérští programátoři, kteří často pokryjí pouze úplné základy. I přesto lze najít obsah, který je velmi kvalitní a k výuce OpenGL užitečný.

The Cherno

Jedním z nejslavnějších kanálů zabývajících se OpenGL (ale i jinými problematikami) je `The Cherno`. Autorem kanálu je Yan Chernikov, který dle svých informací pracuje jako vývojář ve společnosti `Electronic Arts`. Jeho seznam videí s tématikou OpenGL obsahuje 31 videí s celkovou délkou přesahující 10 hodin. Seznam má přes milion shlédnutí a všechna videa jsou velmi kladně hodnocena.

Obsah epizod se zabývá OpenGL ve verzi 3.0 a novější. Problematika je popsána od základů, včetně inicializace projektu a napojení knihoven. Nejnovější kapitoly se zabývají pokročilejšími záležitostmi, mezi které patří například dynamické vykreslování obsahu. Videa začala vznikat až v roce 2017 a vycházejí dodnes (05.05.2020). Díky tomu je obsah velmi aktuální a vhodný pro studium.

[6] `https://github.com/gfxfundamentals/webgl-fundamentals`

2.5 E-learning

Pro uživatele, kterým nevadí za online tutoriály platit, je vhodnou alternativou e-learning. Jedná se o velmi dobrý způsob výuky, který lze v dnešní době korona-virové krize přirovnat k aktuálně probíhající distanční výuce. Autory tutoriálů bývají většinou lidé z oboru, kteří problematice velmi dobře rozumí a formou videí jí předávají dál. Velká část videí je navíc doplněna úkoly nebo projektem, díky čemuž si uživatel nově nabyté znalosti rovnou vyzkouší.

Udemy

Jedním z nejznámějších e-learning portálů je Udemy.com[7]. V den psaní této kapitoly (03.05.2020) se na webu nachází pět tutoriálů zabývajících se především moderní OpenGL a základy tvorby herních enginů[8]. V případě WebGL je tutoriálů dokonce šest[9] s tím, že některé z nich se zabývají knihovnami THREE.js a Babylon.js, jiné zase integrací Unity s WebGL, a ostatní se pak zabývají klasickou WebGL. Většina tutoriálů se snaží případné studenty nalákat na možnost vytvořit si svojí vlastní hru.

2.6 Zdroje obsahu kapitol

Obsah výukových kapitol vyvíjené aplikace se zakládá na látce probírané v předmětu PGR[10]. Velká část obrázků a obecné problematiky je přebrána z přednášek předmětu. Mezi to se řadí základní principy počítačové grafiky, popis základů lineární algebry, práce se shadery a postupy zpracování scény a obrazu. Některé doprovodné obrázky jsou převzaty volně z internetu, a to především z dále uvedených stránek. K takovému obsahu jsou dodatečně uvedené zdroje.

Zdrojové kódy a teorie k jednotlivým kapitolám mají základ v mých předešlých zkušenostech s OpenGL, které vychází ze cvičení předmětu PGR a především z uvedeného výukového portálu `LearnOpenGL.com`[11] (Kapitola 2.2).

Dodatečná teorie potřebná ke znalosti WebGL je čerpána především z webu `WebGL2Fundamentals.org`[12], kde je problematika detailně rozespána. Jako zdroj přesných definic funkcí JavaScriptu a WebGL slouží webový portál `MDN.com`[13], který spadá pod společnost Mozilla vyvíjející webový prohlížeč Mozilla Firefox. Díky tomu se jedná o velmi relevantní zdroj informací.

[7] `https://www.udemy.com/`
[8] `https://www.udemy.com/courses/search/?src=ukw&q=opengl`
[9] `https://www.udemy.com/courses/search/?q=webgl`
[10] `https://dcgi.fel.cvut.cz/courses/pgr`
[11] `https://learnopengl.com/`
[12] `https://webgl2fundamentals.org/`
[13] `https://developer.mozilla.org/en-US/`

2.7 Výsledky rešerše

Z rešerše vyplynulo, že nejpoužívanějším způsobem výuky je členění obsahu na kapitoly, které jsou doplněny úkoly (nebo příklady)na procvičení probrané látky. Většina příkladů počítá pouze s úpravou předpřipraveného kódu a student se tak nemusí zabývat programováním celého příkladu.

Velké množství tutoriálů prokládá výukový text ukázkami kódu. V lepších případech je kód také zvýrazněn a díky tomu je mnohem přehlednější. Ukázky kódu jsou velmi dobrým nástrojem, pomocí kterého lze doplňovat probíranou látku. Čtenář si tak dokáže lépe představit, jak daný problém později implementovat. Ukázky jsou také v mnoha případech založeny na obsahu podpůrných příkladů a jejich následná úprava je díky tomu pro studenty jednodušší. Velmi důležitým doplňkem obsahu jsou také názorné obrázky a animace. V mnoha případech názorná kresba vysvětlí problematiku mnohem lépe, než několik odstavců textu.

V případě WebGL tutoriálů je velmi dobrou výhodou možnost prezentovat problematiku pomocí (interaktivních) demo příkladů. Jejich vypovídající hodnota je ještě větší než u klasických obrázků. A to vzhledem k tomu, že uživatel vidí probranou látku přímo v akci, ještě než se pustí do samotného programování.

U stránky `WebGL2Fundamentals.org` je velmi užitečné propojení jednotlivých kapitol s testovacím prostředím `CodePen.io`[14] a `JSFiddle`[15]. To umožňuje ihned interagovat nejen s výsledným programem, nýbrž také s jeho zdrojovým kódem. Student může kód různě upravovat či zkoumat jeho části, a to bez potřeby vlastního editoru a stahování zdrojového kódu.

Velká část rešeršovaných zdrojů poskytuje pouze statický obsah. Student si může přečíst text, podívat se na ukázky kódu a prohlédnout si přiložené obrázky. Ovšem to, jak se chová výsledek, se dozví až poté, co si příklad naprogramuje (nebo stáhne a spustí). To může být v některých případech zbytečná komplikace a případné zájemce to může od studia odradit. Tento problém se týká především tutoriálů zabývajících se OpenGL. Vzhledem ke kombinaci s programovacími jazyky, jako jsou C++ a Java, nelze výsledný kód v prohlížeči interpretovat.

V následující práci se budu zabývat implementací výukové aplikace, která bude sloužit k podpoře výuky počítačové grafiky. Výsledkem by měl být komplexní tutoriál pokrývající většinu látky probírané v předmětu PGR (viz kapitoly 4.4.2 a 4.4.3). Vzhledem k tomu, že samotné demo příklady jsou určeny pouze k demonstraci probíraného obsahu, budou doplněny výukovým textem. Výsledek tak nebude sloužit pouze jako soubor ukázek, ale jako použitelná množina výukových kapitol.

Demo příklady, které jsou součástí práce, budou implementovány pomocí jazyka JavaScript a rozhraní WebGL. Z výsledků rešerše vychází nejlépe kombinace demo příkladů přímo s výukovým textem a díky tomu se zvolené technologie jeví jako nejlepší.

[14] `https://codepen.io/`
[15] `https://jsfiddle.net/`

Kapitola 3
Analýza

Na základě rešerše vyplynulo, že nejlepším způsobem prezentace obsahu je provázání výukových kapitol dohromady s demo příklady. Aby bylo možné funkční příklady zahrnout do obsahu webové stránky, je nutné využít kombinaci jazyka JavaScript a rozhraní WebGL. Vzhledem k tomu, že syntax OpenGL a WebGL je lehce odlišná, je potřeba analyzovat jejich hlavní rozdíly. Díky znalosti rozdílů je pak možné jednoduše přecházet mezi stylem zápisu obou dvou knihoven.

3.1 Rozdíly mezi OpenGL a WebGL

Zmiňovaný předmět PGR (Programování Grafiky) se zabývá především knihovnou OpenGL spolu s programovacím jazykem C++. V kontextu OpenGL jsou popsány způsoby komunikace s grafickou kartou, funkce programů zvaných shadery, principy 3D matematiky a další postupy pro zpracování dat před jejich vykreslením na obrazovku. Jazyk C++ je pouze využit k přípravě dat a k práci s OpenGL.

Ovšem většinu těchto informací (zkušeností) lze využít i v případě jiných programovacích jazyků a jiných rozhraní pro komunikaci s grafickou kartu. OpenGL je možné použít v kombinaci s jinými programovacími jazyky (viz kapitola 1) a místo OpenGL lze využít rozhraní `Microsoft DirectX` nebo `Vulkan`. Ve všech těchto případech se staví na stejných principech, proto pro programování počítačové grafiky je možné použít jakoukoliv dostupnou kombinaci. V případě předmětu PGR a semestrálních prací je tento výběr omezen na standardní programovací jazyky a grafická rozhraní, která nejsou nijak zjednodušena využitím externích knihoven.

Mezi jednu z možných kombinací patří také rozhraní WebGL spolu s jazykem JavaScript. V praxi se jedná stále o stejnou teorii, liší se pouze přístup daného programovacího jazyka a použitého rozhraní.

Pokud se student rozhodne použít jiný programovací jazyk, nebo dokonce i jiné rozhraní, pak přichází na řadu analýza rozdílů těchto přístupů. V případě záměny programovacích jazyků (například využití Javy místo C++) se nejedná o zásadní změnu a studenta pouze čeká dohledání alternativních knihoven pro daný jazyk. V případě využití WebGL místo OpenGL přichází více změn...

3.1.1 Správa oken, GLUT

Jedním z hlavních rozdílů mezi OpenGL a WebGL je absence knihovny obstarávající vykreslovací okna. V případě práce s OpenGL je nejprve nutné vytvořit okno, do kterého se má obsah vykreslovat. Následně musí být zřízena tzv. `vykreslovací smyčka`, která se stará o zpracovávání událostí a řízení vykreslování jednotlivých snímků. K tomu všemu slouží sady nástrojů (*toolkit*), mezi které patří `GLUT (OpenGL Utility Toolkit)`, `GLFW` nebo `FreeGLUT` (Novější verze GLUT) [24].

V případě WebGL nic takového v základu není. Výjimkou může být knihovna `glfw.js` [25], která se snaží něco podobného implementovat v JavaScriptu, ovšem vzhledem k datu vydání (03.04.2020) by se nemělo s jejím použitím spěchat. Navíc podstata této knihovny je z většiny nahrazena dále popsanou funkcionalitou HTML a jazyka JavaScript.

Díky tomu, že je WebGL zpracovávána v prostředí prohlížeče, které samo o sobě implementuje výše zmíněnou problematiku správy oken, tak není třeba žádnou knihovnu dodávat. Vykreslovací okno (okna) zde reprezentuje HTML element `<canvas>`, jehož úlohou je zobrazování rastrové grafiky.

Zpracovávání událostí má na starosti samotný JavaScript s pomocí tzv. *HTML DOM (Document Object Model)*. DOM je stromová struktura, která uchovává vazby mezi jednotlivými prvky (elementy) HTML stránky. Každý element je reprezentován jako objekt s vlastními atributy, funkcemi a událostmi [27]. Navíc obsahuje podstrom svých potomků, kteří dědí většinu vlastností svého rodičovského elementu (objektu). Události, které vznikají na těchto HTML elementech, putují celým stromem DOM v závislosti na uzlu, kde událost vznikla. Díky tomu je možné WebGL aplikaci ovládat velkým množstvím událostí, které nejsou vázané jen na vykreslovací okno (canvas), nýbrž na celý HTML dokument.

3.1.2 Syntax

Studenti, kteří se rozhodnou psát ve WebGL, si musí zvyknout i na lehce rozdílnou syntaxi od OpenGL. Funkcionalita OpenGL je do programu dodána importováním patřičného knihovního souboru (s příponou .dll). Ten je závislý na implementaci daného výrobce grafických karet a odvíjí se od verze použitých grafických ovladačů. Po importování knihovny jsou funkce a ostatní vlastnosti OpenGL globálně dostupné. Většina názvů takových funkcí začíná prefixem `gl` a u enumerátorů a konstant prefixem `GL_`.

V případě WebGL a JavaScriptu se žádná knihovna nevkládá. WebGL je již samo o sobě součástí JavaScriptu jakožto takzvaný kontext a váže se na již zmíněný HTML element canvas. *Canvas* je definovaná plocha v HTML stránce, do které lze pomocí různých kontextů vykreslovat rastrovou grafiku.

Kontext reprezentuje vykreslovací rozhraní s množinou funkcí, které jsou určené k určitému typu vykreslování. Například existuje rozhraní pro vykreslování 2D grafiky, nebo právě zmíněné WebGL rozhraní, které je primárně určené pro vykreslování 3D grafiky (a také díky tomu obsahuje mnohem více funkcí).

```
glEnable(GL_CULL_FACE /*hodnota enumerátoru*/); // OpenGL
gl.enable(gl.CULL_FACE); // WebGL
```

Listing 1: Ukázka rozdílů syntaxe OpenGL a WebGL

```
#version 300 es          // Definice verze GLSL na prvním řádku
precision highp float;   // Definice zaokrouhlování (2. řádek!)
out vec4 theColor;       // Definice vektoru výstupní barvy
```

Listing 2: Ukázka novinek v GLSL 3.00

Vzhledem k tomu, že WebGL kontext se ukládá do proměnné jakožto reference na WebGL rozhraní, tak záleží na programátorovi, jakým názvem budou WebGL funkce uvozeny. Konvencí je mít referenci uloženou v proměnné pod názvem `gl`. Pak se zápis WebGL příkazů od OpenGL příkazů liší pouze záměnou `gl[příkaz]` a `GL_[hodnota enumerátoru]` za `gl.[příkaz / hodnota enumerátoru]`. Zmíněnou záměnu demonstruje ukázka kódu 1.

3.1.3 Shadery

Shader je program, který je vykonáván na grafické kartě. V základu existují dva typy shaderů - `Vertex Shader` a `Fragment Shader`. Vertex shader pro každý vrchol ve scéně počítá novou pozici na základě dodaných dat. Pro výpočet jednoho snímku je paralelně spuštěno tolik vertex shaderů, kolik je ve scéně vrcholů. Fragment shader počítá barvu konkrétního fragmentu (pixelu) na základě vlastností scény (osvětlení, materiály objektů, vzdálenost mlhy apod.). Při výpočtu každého snímku je paralelně spuštěno tolik fragment shaderů, kolik fragmentů je výstupem rasterizace.

Pro OpenGL i pro WebGL se shadery píší v jazyce *GLSL (OpenGL Shading Language)*. Díky tomu jsou až na menší záležitosti shadery mezi OpenGL a WebGL přenosné. Nutné je si dát pozor na verze GLSL, jelikož pro OpenGL a WebGL se liší. V případě WebGL 1.0 je potřeba použít verzi `GLSL 100` a pro WebGL 2.0 verzi `GLSL 300`. V případě OpenGL je verze GLSL závislá na použité verzi OpenGL [28].

Další změny se týkají fragment shaderu, kde je nutné na začátku programu definovat přesnost zaokrouhlování desetinných čísel [10, str. 158]. GLSL v této verzi také pracuje jinak s alfa kanálem. Díky tomu je výchozí hodnota pro čtvrtou souřadnici barvy rovna nule a výsledná barva je tak průhledná. Programátor musí explicitně výstupní barvu definovat jako čtyř-složkový vektor s poslední souřadnicí obsahující míru viditelnosti.

Poslední změnou je odstranění některých systémových proměnných, mezi které patří výstupní barva fragmentu `gl_FragColor`. Programátor si musí tuto proměnou definovat sám jakožto výstupní proměnnou `out`. Uvedené změny demonstruje ukázka kódu 2.

3.2 Podpůrné knihovny

Na základě rešerše došlo k zjištění, že některé výukové aplikace (stránky) používají při výuce vlastní podpůrnou knihovnu. Hlavním účelem takové knihovny je separace nadbytečného a opakujícího se kódu od toho, který je předmětem konkrétní kapitoly. V některých případech se tento kód značí pojmem *Boilerplate* a reprezentuje větší množství kódu, kterému nemusí programátor věnovat příliš pozornosti. Jedná se tak hlavně o kód, který je ve stejné podobě obsažen ve většině aplikací podobného typu. V případě výuky WebGL se stává boilerplate kódem i ten, který je dostatečně probrán v předchozích kapitolách a jehož délka negativně ovlivňuje přehlednost studovaného kódu.

V případě výukového webu `WebGL2Fundamentals.org` je součástí výuky volně dostupná knihovna `webgl-utils.js`[1], která obsahuje základní funkce reprezentující dokola využívaný boilerplate kód.

Pro účely předmětu PGR slouží knihovna `PGR-framework`, která obsahuje funkce se základním nastavením a funkcionalitou OpenGL. Implementovány jsou především ty části, které jsou nedílnou součástí každé větší grafické aplikace a jejich opakování v každém příkladu je zbytečné. Knihovna dále sjednocuje potřebné balíčky a ostatní knihovny, aby je student nemusel hledat a zvlášť stahovat.

Na základě analýzy zmíněných knihoven jsem se rozhodl k implementaci vlastní knihovny. Pro potřeby výuky WebGL bude sloužit nová knihovna `CTUGL`, která bude část funkcionality přebírat od zmíněné knihovny PGR-framework. Její obsah se také bude vyvíjet na základě potřeb výukové aplikace - viz kapitola 5.3.

3.3 Aplikace pro zobrazování 3D modelů

Jako úvodní krok ke shromáždění vyučované problematiky jsem se rozhodl k implementaci komplexnější grafické aplikace, která bude využívat většinu problematiky probírané v předmětu PGR. Aby byla pro mě implementace zajímavější, vytyčil jsem si za cíl, že v aplikaci bude možné zobrazit jakýkoliv 3D model vytvořený v některém z modelovacích nástrojů (`Autodesk Maya`[2] či `Blender`[3]).

Model by mělo být možné vykreslit do scény včetně jeho materiálů a textur předem definovaných při jeho vytváření. Dále by mělo být možné se ve scéně volně pohybovat a rozhlížet se, čímž bude umožněno prohlížení modelu ze všech stran a úhlů. Součástí scény by mělo být implementované osvětlení založené na *Phongově* osvětlovacím modelu. Výhodou bude implementace pokročilých funkcionalit, mezi které se řadí *Skybox*, *Mlha* či *Picking* (interakce se scénou pomocí klikání myší).

[1] https://webgl2fundamentals.org/webgl/resources/webgl-utils.js
[2] `https://www.autodesk.com/products/maya`
[3] `https://www.blender.org/`

3.4 Funkční požadavky výukové aplikace

Před implementací výukové aplikace je nutné stanovit množinu funkcí, které má aplikace implementovat a realizovat. Funkce jsou založené na potřebách, které souvisí s procházením výukového obsahu.

- **FR1**: Zobrazení strukturovaného seznamu kapitol v postranním menu.
- **FR2**: Zobrazení obsahu zvolené kapitoly doprovázené funkčními ukázkovými příklady (demo příklady).
- **FR3**: Doplnění výukového textu fragmenty zdrojového kódu včetně barevného zvýraznění jeho syntaxe.
- **FR4**: Zobrazení demo příkladu na samostatné stránce.
- **FR5**: Možnost interagovat s demo příklady (s těmi, u kterých to má smysl) přímo ve výukovém textu.

Kapitola 4

Návrh aplikace

Z rešerše vyplývá, že by aplikace měla kombinovat výukový materiál dohromady s funkčními demo příklady. Díky tomu se očekává, že hlavním prvkem stránky bude statický text doplněný ukázkami kódu, obrázky a demo příklady. Jako jeden způsob řešení připadá v úvahu série HTML stránek, které budou provázány mezi sebou jednoduchými odkazy. Řešení samo o sobě naplňuje potřeby pro výukovou aplikaci, ale v dnešní době je těžkopádné a zastaralé. Při přecházení mezi kapitolami je vždy nutné načíst novou stránku včetně stále stejných elementů, mezi které patří hlavička či menu. Jako mnohem lepší řešení se jeví načítání pouze samotného obsahu.

Z toho důvodu je výuková aplikace navržena jako tzv. *Single-Page Application (SPA)* [33]. Jedná se o speciální design webové aplikace, kdy obsah není aktualizován na základě přecházení mezi stránkami, ale je dynamicky vkládán pomocí JavaScriptu. Díky tomu není potřeba uživateli posílat celou novou stránku, ale pouze aktualizovaný obsah. Pro typ stránky, jako je tato, je řešení na základě SPA ideální.

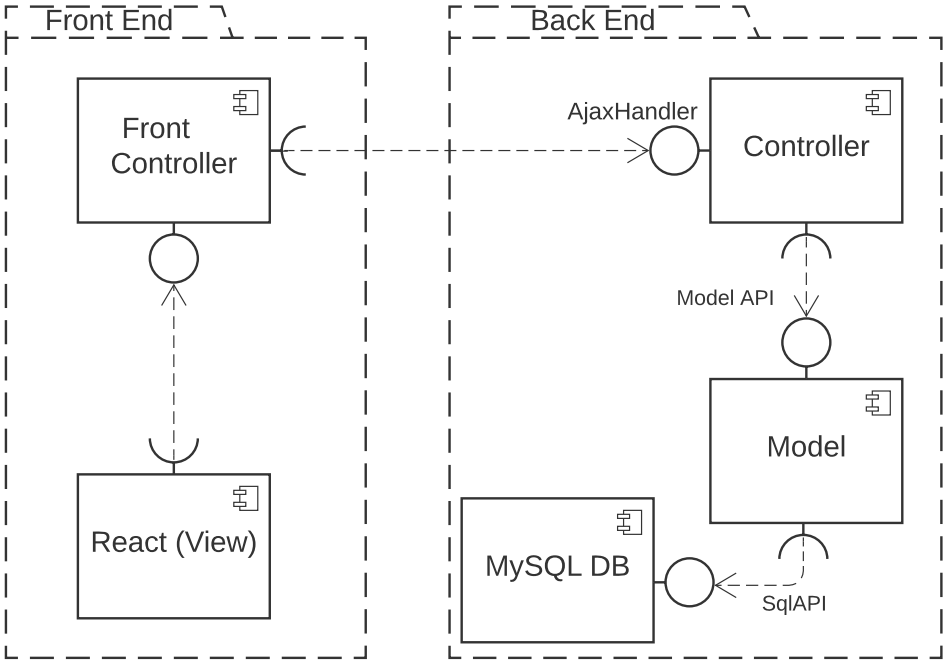
Aplikace je v tomto případě mnohem rychlejší a příjemnější na používání. Obsah se stahuje ze serveru během toho, co si uživatel stránku prohlíží (nebo s ní interaguje), a nemusí tak čekat na stažení celé nové stránky. Stahování obsahu probíhá asynchronně a nijak nenarušuje či nezdržuje chování stránky jako takové. Asynchronní komunikace s webovým serverem je většinou realizována pomocí technologie *AJAX (Asynchronous JavaScript And XML)* [34], která se zakládá především na asynchronních http požadavcích typu GET a POST.

V dnešní době je princip SPA používán na velkém množství webů, které potřebují častěji komunikovat s web-serverem a nahrazovat obsah stránky. Může se jednat o firemní systémy, objednávkové systémy nebo e-shopy.

Snahou SPA je také co nejvíce přiblížit vzhled a chování webové aplikace ke klasické desktopové nebo mobilní aplikaci. Uživatel by pak neměl mít možnost poznat (téměř), jestli je na webové stránce, nebo jestli používá aplikaci či program. K realizaci takové aplikace slouží mnoho frameworků, mezi které patří například `React.js`, kterým se zabývá kapitola 4.2.3.

4.1 Architektura

Jako typ síťové architektury celé aplikace byla zvolena architektura *Client-Server*. Ta odděluje komunikaci s databází a zpracování dat od klientské části, která se stará především o vykreslování obsahu. Klientská část aplikace se značí *Front-End* a v případě webové aplikace je zpracovávána prohlížečem. Serverová část se značí *Back-End* a slouží k obsluhování jednotlivých uživatelů pomocí rozhraní, na kterém přijímá a odpovídá na požadavky z klientské části.



Obrázek 4.1: High-Level návrh aplikace (diagram komponent)

Struktura aplikace se řídí architektonickým vzorem `MVC` (*Model-View-Controller*), který výše popsanou problematiku ještě více rozděluje. Front-End je zastoupen především komponentou *View*, která reprezentuje `GUI` (Uživatelské rozhraní) aplikace.

Komponenta Controller slouží jako prostředník mezi komponentami View a Model. Běžně bývá součástí back-endu aplikace a zpracovává požadavky přicházející z front-endu. Na základě příchozích požadavků komunikuje s back-end komponentou Model, která reprezentuje obecné rozhraní pro získávání dat (například z databáze). Získaná data jsou následně odeslána zpět na front-end v odpovědi na požadavek.

V případě webových aplikací bývá front-end doplněn komponentou *Front Controller*, která slouží k dodatečnému zpracování dat a událostí.

4.2 Front End

Front-End je část aplikace (nebo systému), která je zpracovávána na straně uživatele. Jedná se hlavně o přípravu Uživatelského Rozhraní (GUI), zpracování událostí vzniklých za běhu aplikace (například vstup uživatele - kliknutí na tlačítko), zaslání požadavků na server a zpracování dat, která server vrátí v odpovědi. V případě této aplikace se jedná o celou webovou stránku. Komunikaci se serverem obstarává JavaScript s pomocí zmíněné technologie AJAX.

Front-End výukové aplikace se skládá především z komponenty View, která je reprezentována balíčkem `React`, jež obsahuje hierarchii React komponent. Komponenta View spolupracuje s druhou front-end komponentou `Front Controller`(viz obrázek 4.1). Ta poskytuje aplikační logiku potřebnou především pro získávání a zpracovávání potřebných dat ze serveru. Pro komunikaci se serverem slouží rozhraní `AjaxAPI`, které odesílá Http požadavky na serverové rozhraní `AjaxHandler.php`. Detailnější schéma front-endu zprostředkovává diagram tříd 4.3, na kterém je znázorněn obsah zmíněných komponent. Zbylý neprobraný obsah komponenty Front Controller je popsán v implementační kapitole 5.4.2.

4.2.1 Prototyp

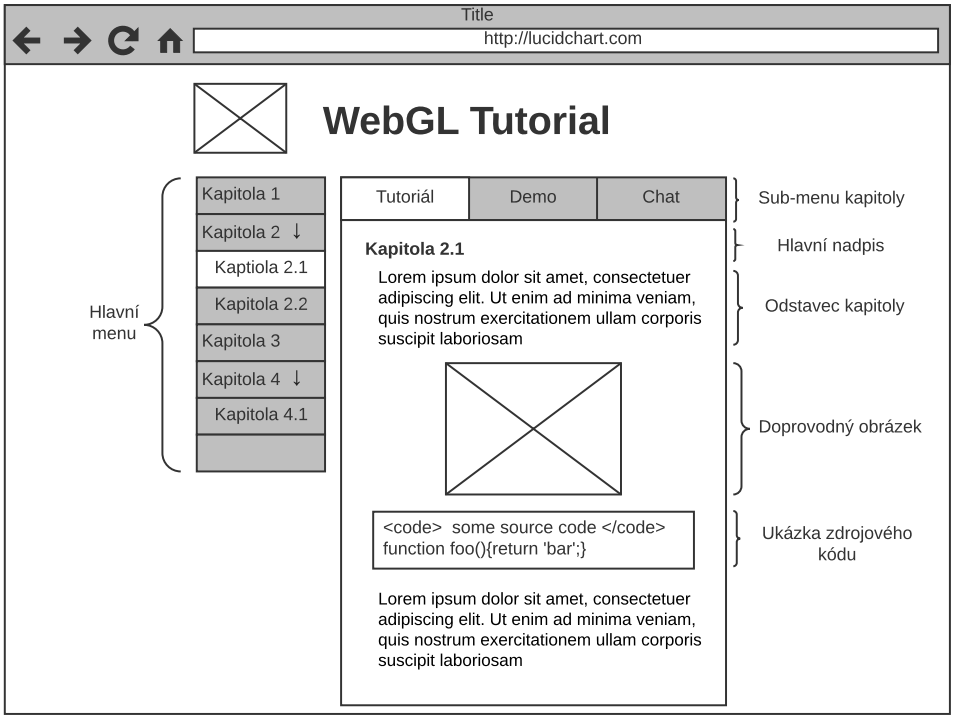
Pro účely návrhu GUI (vzhledu aplikace a rozložení jejích jednotlivých částí) je nejlepší vytvořit tzv. *WireFrames*. Ve své podstatě se jedná o schématické ukázky obrazovek, které zachycují základní podobu budoucí aplikace. Může se jednat o jednoduché černobílé „diagramy“, ale také o funkční prototypy včetně designu a různých funkcionalit. Tvorba wireframes přispívá k jednoznačnosti budoucí implementace a pomáhá také k odhalení prvotních chyb či nedorozumění.

Pro tvorbu prototypu této stránky byla využita aplikace `Lucidchart` [1] se studentskou licencí, která umožňuje vytváření jednoduchých prototypů celých webových aplikací. Oproti ostatním aplikacím (Enterprise Architect apod.) může být výstupem interaktivní série stránek. Při tvorbě wireframes došlo k vytvoření hlavní obrazovky zahrnující většinu obsahu výukové aplikace (viz obrázek 4.2).

4.2.2 Bootstrap

K definici rozložení stránky (kde se co bude nacházet) slouží značkovací jazyk `HTML`. Pomocí něj je možné definovat, kde se bude nacházet hlavička stránky, kde menu a kde samotný obsah webu. HTML ovšem umožňuje jen velmi omezeně ovlivňovat vzhled stránky. K tomu slouží jazyk `CSS` - *Cascading Style Sheets (Kaskádové styly)*, který pomocí pravidel určuje vizuální vlastnosti jednotlivých částí webové stránky (HTML elementů). Procesu aplikace stylovacích pravidel se říká "stylování".

[1] `https://www.lucidchart.com/pages/usecase/education`



Obrázek 4.2: Wireframe stránky - Návrh GUI

Stylování ovšem může být často velmi zdlouhavý proces, který může zabrat mnohem více času, než programátor hodlá obětovat.

V případě tvorby tzv. responzivního designu je časová náročnost ještě větší. *Responzivní design* je způsob návrhu aplikace, kdy autor počítá se zobrazováním stránky na různých typech zařízení. Mezi takové patří stolní počítače, notebooky, tablety, mobily apod. Každé ze zmíněných zařízení má ovšem jiný poměr stran a jiné rozlišení. V případě ignorování těchto zařízení při tvorbě webu jsou jejich uživatelé vystaveni nepříjemně zdeformovaným stránkám, které se mnohem hůř čtou. Právě díky tomu jsou při tvorbě responzivního designu definovány třídy pravidel pro jednotlivé typy zařízení (především s rozdílnými rozměry prvků stránky).

Kvůli potřebě snížit časovou náročnost tvorby designu (responzivního) vznikl framework `Bootstrap` [11]. Ten umožňuje velmi jednoduše designovat vzhled webové stránky pomocí předdefinovaných CSS tříd. Bootstrap celou stránku virtuálně rozděluje na 12 sloupců, kterým programátor nastavuje různou šířku dle své potřeby. Ze stránky se částečně stane tabulka, ve které mají jednotlivé objekty určená pravidla, která nesmí porušit. V případě různých velikostí zařízení, na kterých se webová stránka zobrazuje, se pak automaticky mění šířka a počet použitých sloupců (z celkových dvanácti).

Z důvodu zmíněných výhod bude Bootstrap využit i při vývoji této aplikace. Samozřejmostí je dodefinování vlastních stylů, které Bootstrap neobsahuje. Využito bude také javascriptové rozšíření Bootstrapu (`bootstrap.js`), které umožňuje použití pokročilejších komponent, jako je vyjíždějící menu či přepínání klasického menu do jeho mobilní podoby (v případě zobrazení webové stránky na mobilním zařízení).

4.2.3 React.js

Vzhledem k tomu, že aplikace je koncipována jako *Single Page Application* (SPA), je nutné vyřešit způsob načítání jejího obsahu (především kapitol). Jednou z možností je použití čistého JavaScriptu (Vanilla JavaScript), který se v poslední době stává čím dál mocnějším nástrojem. Jazyk JavaScript byl dlouhou dobu opomíjen díky svým omezeným možnostem a byl používán jen pro nejdůležitější potřeby, které nešlo vyřešit jinak - myšleno v kontextu JavaScriptu ve verzi ES5 (a níž), respektive ECMAScript 2009 (synonymum). Verze ES5 sice již obsahovala některé zajímavější funkce [10, str. 12-13], včetně podpory asynchronního zpracovávání dat (AJAX), ovšem stále chyběla spousta věcí, mezi které patří například koncept tříd. Jeho napodobenina ve formě prototypů sice mohla nahradit třídy, ale mnoha programátorům tento způsob nevyhovoval. Před mnoha lety také vznikla knihovna jQuery, která měla práci s JavaScriptem usnadnit.

Ovšem JavaScript se od verze `ECMAScript 2015` (ES 6) stal moderním jazykem, pomocí kterého lze vyřešit spoustu složitějších situací, včetně systému na dynamické načítání obsahu do stránky.

Další možností je použití zmíněné knihovny `jQuery`. Tato knihovna byla vytvořena právě pro usnadnění práce s vytvářením uživatelského rozhraní (GUI) webových aplikací. Slouží k provázání HTML a JavaScriptu s cílem vytvořit příjemné uživatelské rozhraní, které plynule reaguje na uživatelovi akce. Zároveň se ale snaží oddělit logiku webové stránky od samotné definice její struktury a vzhledu (HTML + CSS). To je v případě implementace větších dynamických webů důležité.

V dnešní době se ovšem jQuery považuje spíš za zastaralou záležitost a používají se jiné frameworky. Mezi nejznámější patří `React.js` a `Angular`. Jejich podstata je podobná a záleží na preferencích programátora, který framework využije.

V případě této aplikace bude využit framework React.js. Základním prvkem tohoto frameworku je struktura zvaná *Komponenta*. Ta reprezentuje jak vizuální stránku (HTML), tak i funkcionalitu (JavaScript), některé části webu. Částí webu může být myšleno tlačítko, ale třeba i formulář, nebo celé menu. Webová aplikace je pak složena z hierarchie takových komponent, která se podobá struktuře HTML DOM (*Document Object Model*). DOM představuje stromovou strukturu, která reprezentuje hierarchické uspořádání elementů ve stránce. Právě díky tomu se struktura React komponent nazývá *ReactDOM*. Každá komponenta obsahuje funkci `render()`, která je zavolána hned po vykreslení rodičovské komponenty (vzhledem ke stromu ReactDOM).

Metoda render() vykreslí obsah komponenty do stránky a zároveň donutí vykreslit obsah celého podstromu svých potomků. Vedle vykreslovací metody (která je povinná) může programátor v komponentě definovat nespočet svých vlastních funkcí, které slouží ke zpracovávání událostí či komunikaci se serverem.

Hlavní podstatou React frameworku je načítání obsahu závislé na komponentách, které vyžadují aktualizaci svého obsahu. Jinak řečeno, ze serveru se načítá pouze obsah, který se má ve stránce změnit, zbytek zůstane zachován. Nahrazení komponenty a všech jejích potomků umožňuje právě zmíněná hierarchie ReactDOM. Aktualizace obsahu komponenty proběhne pouze v případě, že se změní její stav a nebo na její rodičovské komponentě dojde k zavolání metody render().

Díky reprezentaci částí webu jako komponent je také možné rozdělit webovou aplikaci do modulů nebo tříd a pracovat s ní jako s aplikací psanou v Javě nebo jiných objektově orientovaných jazycích. Tím je zjednodušena přehlednost v kódu aplikace a přibývá možnost jednoduché správy či refaktorizace jednotlivých částí.

4.3 Back End

Back-End je část systému (e-shop, internetové bankovnictví apod.), která je provozována na serveru. Slouží k zpracování dotazů z klientské části systému (například webových stránek), získání a zpracování požadovaných dat a jejich případné zaslání zpět klientské části (Front-End).

Vzhledem k tomu, že funkce back-endu výukové aplikace bude sloužit pouze k manipulaci s daty uloženými v databázi (popřípadě zápis dat), tak se počítá pouze s jednoduchou aplikací, která bude zpracovávat Http požadavky. Pro mé účely se jako nejlepší volba jeví jazyk `PHP`, který je i v dnešní době stále využíván právě k tvorbě serverových aplikací. Jazyk PHP jsem vybral také z důvodu, protože s ním mám zkušenosti a nemá smysl se zabývat jinou technologií, vzhledem k menšímu rozsahu serverové části aplikace.

Při implementaci bude vytvořena jednoduchá struktura Controller-Model, která bude ovládána ze vstupního modulu zpracovávajícího Http požadavky (`AjaxHandler.php`), viz obrázky 4.1 a 4.4. Požadavky budou typu GET a POST v závislosti na typu prováděné CRUD operace. Zkratka CRUD zahrnuje všechny typy operací, které mohou nastat při manipulaci s daty - *Create* (vytvoření), *Read* (čtení), *Update* (nahrazení) a *Delete* (smazání). Většina požadavků však bude typu GET, vzhledem k tomu, že budou využívány k získávání obsahu kapitol z databáze (operace Read). Požadavky budou ze vstupního modulu předávány komponentě Controller, která je dále zpracuje s pomocí komponenty Model (obrázek 4.4).

4.3.1 Kontroler

Jako vrstva mezi vstupním modulem (`AjaxHandler.php`) a databází (komponenta `Model`) bude umístěna komponenta Kontroler (`Controller`). Ta bude zpracovávat již konkrétní požadavky přijaté pomocí vstupního rozhraní AjaxHandler.php. Bude se starat o připravení případných vstupních dat před voláním Model vrstvy (a databáze), volání metod Modelové vrstvy, zpracovávání přijatých dat z databáze a nakonec jejich navrácení do vstupního modulu, který je odešle zpět na front-end (obrázek 4.1).

4.3.2 Modelová vrstva

Modelová vrstva bude sloužit pouze jako rozhraní ke komunikaci s databází (viz obrázek 4.4). Bude obsahovat předpřipravené funkce, jejichž názvy budou odpovídat typu prováděné akce v databázi (například funkce `getUser(name)` získá z databáze informace o daném uživateli). Hlavním obsahem těchto funkcí budou SQL dotazy upravené na míru konkrétní databáze. Případné přejmenování sloupců v databázi bude kritické právě pro tyto funkce. Díky jejich separaci od zbytku kódu by ovšem mělo být jednoduché chyby opravit.

4.3.3 Databáze

Vzhledem k tomu, že ukládání kapitol nebude realizováno ukládáním celých HTML dokumentů, byla jako způsob úložiště vybrána SQL databáze. Databáze se jeví jako nejlepší řešení také s výhledem do budoucna, kdy by aplikace mohla být rozšířena o komentáře a další funkcionality. Dalším důvodem výběru databáze je fakt, že v budoucnu se počítá s multijazyčností.

Mezi ukládanými daty kapitol se bude nacházet pouze čistý text a ukázky kódu. Obsah jako obrázky či zdrojový kód samotné demo aplikace budou uloženy ve speciálních složkách, na které bude v databázi uložena reference. Pro reprezentaci kapitol budou vytvořeny dvě tabulky (viz obrázek 4.5).

První tabulka `Chapters` bude nést základní informace potřebné pro výpis kapitol z databáze. Mezi těmito daty bude identifikátor kapitoly, reference na složku dat této kapitoly, či informace o zanoření kapitoly v kategoriích. Na základě výsledků testování byl návrh doplněn atributem `chaining`, který se stará o provázaní kapitol v návaznosti na tlačítka "předchozí"a "následující". Druhá tabulka `ChaptersTranslate` bude obsahovat již lokalizovaná data konkrétní kapitoly. Vzhledem k tomu, že aplikace by se měla v pozdější fázi stát multijazyčnou, tak provázání tabulky se základními informacemi (Chapters) a této tabulky bude řešeno vazbou *1:N*. Každý záznam v tabulce bude obsahovat referenci na kapitolu, ke které patří. Dále bude obsahovat zkratku jazyka, ve kterém je daná lokalizace napsána (například "cz", "en"). Následovat bude nadpis kapitoly a obsah kapitoly v konkrétním jazyce.

Obsah kapitoly bude uložen v serializovaném formátu JSON a to především z důvodu, že bude tvořen různými typy prvků. Mezi ty se řadí *Nadpis*, *Odstavec*, *Obrázek*, *Ukázka kódu* a *Demo příklad*. Prvky "Obrázek"a "Demo příklad"budou obsahovat referenci na konkrétní soubor v příslušné složce.

Návrh počítá s tím, že každá kapitola bude mít svojí vlastní složku, ve které se budou nacházet strukturovaná data (například složka s obrázky či složka s demo příklady). Odkazování na obsah ve složkách pak bude řešeno pomocí relativních odkazů, které se budou odvozovat od kořenové složky konkrétní kapitoly.

Do budoucna se počítá s tabulkami potřebnými pro vedení uživatelů a jejich komentářů. Uživatelé budou vázáni s komentáři vazbou *1:N* a komentáře s kapitolami vazbou *N:1*.

4.4 Obsah výukových kapitol

Velmi důležitou součástí stránky je samotný obsah kapitol. Předpokladem je, že uživatel navštíví stránku buď za účelem studia některé z kapitol a nebo s cílem vyhledat řešení konkrétního problému.

V případě návštěvy kvůli konkrétní kapitole se dá předpokládat, že má uživatel zájem se problematice dlouhodoběji věnovat. Je možné, že již některé kapitoly dokonce absolvoval. Pro tento typ uživatelů je velmi důležité obsah výuky správně strukturovat. I samotná výuka předmětu PGR dělí výuku na cvičeních dle tématiky. Čtenář tak musí mít představu o tom, co se v kapitole dozví, ještě před tím, než si ji vůbec zobrazí. Pro zvýšení přehlednosti by měly být kapitoly navíc rozděleny do skupin, čímž by se oddělily jednotlivé tematické okruhy.

V případě, že návštěvník stránky hledá řešení konkrétního problému, tak se předpokládá, že již má nějaké znalosti. Jinak řečeno, má alespoň tušení, u které problematiky by se mohlo nacházet řešení jeho problému. Pro tyto uživatele je nejzásadnější, aby řešení problému našli co nejdříve, a nemuseli se prodírat zbytečně velkým množstvím obsahu. I právě z tohoto důvodu musí být obsah dělen logicky do tematických okruhů a samotné kapitoly by se měly zabývat konkrétní problematikou.

4.4.1 Cíle výuky

Hlavním cílem aplikace je předat uživatelům základy programování grafiky. Vzhledem ke spolupráci s `DCGI`[2] je kladen důraz především na studenty, kterým má aplikace sloužit jako doplňující výukový materiál. Rozdělení kapitol a jejich obsahu bude částečně kopírovat osnovu předmětu PGR. Obsah bude samozřejmě zaměřen na WebGL a proto se výklad bude v některých částech lišit od látky probírané na cvičeních.

4.4.2 Seznam probíraných témat

- Základy WebGL (relevantní i v případě OpenGL)
- Vykreslovací smyčka a zpracování událostí

[2] `https://dcgi.fel.cvut.cz/`

- 3D transforamce v prostoru
- Kamera, volný pohyb ve scéně
- Osvětlení scény (směrové, bodové a reflektor), Phongův osvětlovací model
- Materiály
- Aplikace textur
- Pokročilé techniky - Mlha, Skybox, Picking

4.4.3 Vynechaná témata

Vzhledem k obsáhlosti látky a složitosti přípravy výukových kapitol došlo k vypuštění některých témat (ta se ovšem v budoucí práci mohou objevit - viz kapitola 8.1).

Křivky a plochy

Křivky využívané v předmětu PGR jsou probírané na dvou přednáškách a je možné si je vyzkoušet na jednom cvičení. I přes to, že jsou interpolační křivky nedílnou součástí předmětu, nejedná se přímo o problematiku vykreslování počítačové grafiky. Příprava výukových demo příkladů i samotného textu k této problematice je časově náročnější, než příprava obsahu ostatních kapitol, a to především díky obsáhlosti tématu.

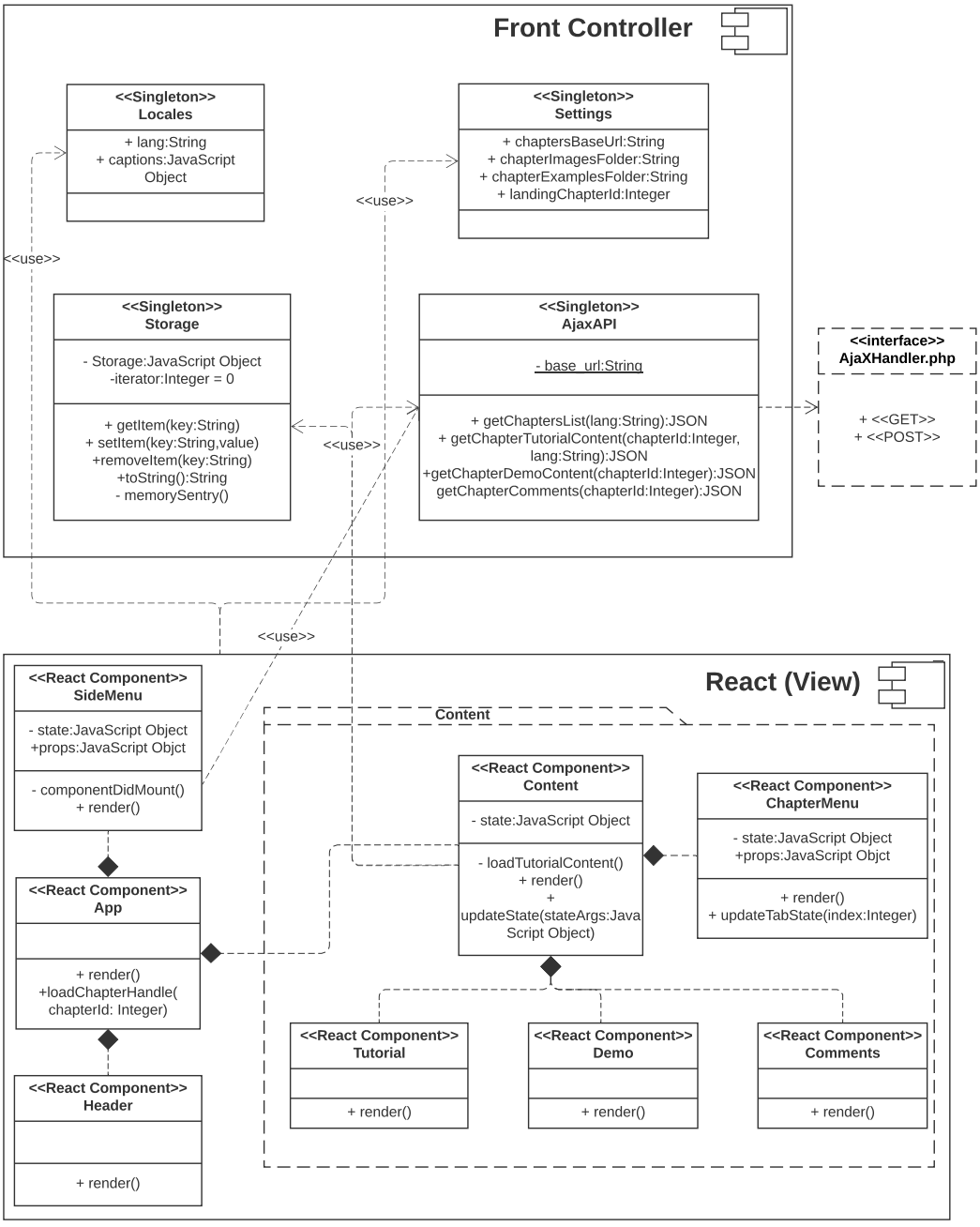
Vzhledem k tomu, že již existuje aplikace pro výuku křivek jako takových, kterou vytvořil pan Ing. Michal Vomastek [13], tak řešení této stejné problematiky by bylo zbytečné.

Načítání modelů

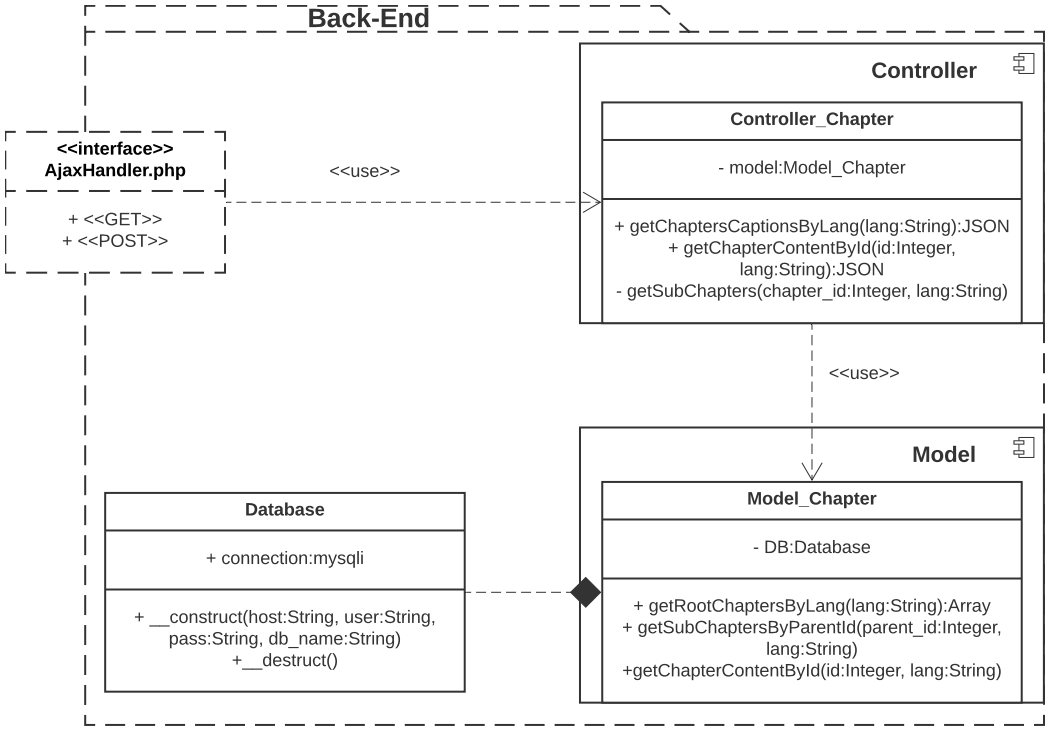
Okrajovým tématem výuky v předmětu PGR je načítání 3D modelů do scény. Načítání modelů je využíváno na několika cvičeních, ovšem samotná problematika je přenechána na knihovně `Assimp` (*Open Asset Import Library*)[3]. Základy použití této knihovny jsou demonstrovány na jednom ze cvičení, ovšem zbytek je ponechán na studentech.

Výuková aplikace jako taková s načítáním modelů nepracuje. Jedná se o pokročilejší tématiku, která není pro základy programování grafiky podstatná. Ovšem mimo výuku je pro zájemce dostupné rozšíření knihovny CTUGL o hierarchii tříd, ve které je také obsažen skript pro načítání 3D modelů dle standardu Assimp. Ten pracuje se soubory typu .OBJ (a .MTL) převedenými do serializovaného souboru JSON. Toto je detailně rozebráno v kapitolách 5.2 a 5.3.

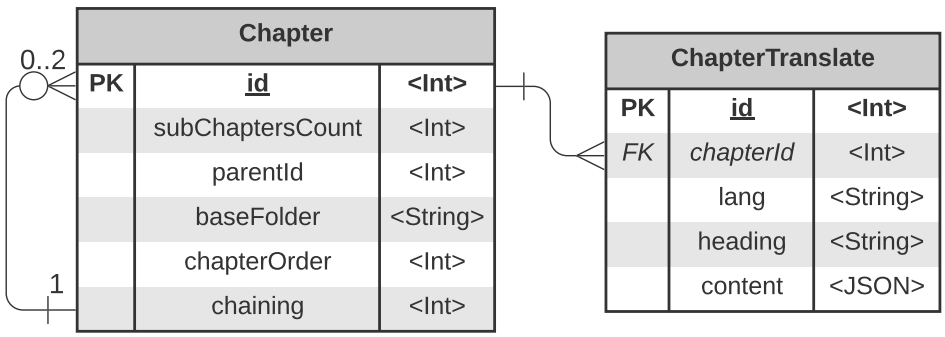
[3] https://github.com/assimp/assimp



Obrázek 4.3: Class Diagram - Front End



Obrázek 4.4: Class Diagram - Back End



Obrázek 4.5: ER diagram databáze

Kapitola 5
Implementace

Kapitola se zabývá postupem implementace aplikace. Popisuje provedené kroky, problémy a jejich řešení.

5.1 Použité technologie

Jak již bylo uvedeno na začátku, vyvíjená aplikace se zabývá výukou programování počítačové grafiky. Vzhledem k tomu, že aplikace je určena k podpoře výuky předmětu Programování Grafiky (PGR) , tak její náplň musí být podobná. Základem je programovací rozhraní OpenGL, které slouží ke komunikaci s grafickou kartou. Pro práci s tímto rozhraním se v předmětu PGR používá programovací jazyk C++, ovšem lze použít i jiné jazyky, jako je například Java.

Pro potřeby výukové aplikace bylo použito rozhraní `WebGL`[3], které je odvozené od OpenGL ES. Jedná se o javascriptové rozhraní (API), které slouží ke komunikaci s grafickou kartou podle standardu OpenGL ES a umožňuje velmi rychlé grafické výpočty. Díky tomu je možné efektivně vykreslovat počítačovou grafiku do webové stránky s pomocí grafické karty namísto pomalého procesoru.

`OpenGL ES` je grafické rozhraní spadající pod OpenGL, které je určeno pro zařízení, jako jsou mobilní telefony, tablety či vestavěné systémy [6]. Použití WebGL rozhraní navíc koresponduje s pravidly předmětu PGR, která připouští použití WebGL jakožto jedné z možných technologií vhodných pro psaní semestrální práce.

Pro aktuálnost byla použita verze WebGL2 (2.0) vycházející z OpenGL ES 3.0, která obsahuje více funkcionalit [4] a odpovídá požadavkům dnešní doby. Aktuálně již větší část prohlížečů WebGL2 podporuje [5] a proto nemá smysl se zbytečně zaobírat WebGL1, která by svými rozdíly mohla studenty mást. Výjimku tvoří uživatelé prohlížečů iOS Safari a Opera Mini, které WebGL2 nepodporují. Takových uživatelů je dle webu `Can I Use` zhruba 14 procent ze všech uživatelů internetových prohlížečů.

S použitím WebGL se mění použitý jazyk pro psaní grafické aplikace a tedy i samotných výukových příkladů. V případě webové aplikace je nejlepší (a snad i jediná možnost) JavaScript. Ten je od verze ES6 (ECMAScript 2015) spolu s HTML5 velmi mocný nástroj. JavaScript verze ES6 přináší mnoho novinek [7], díky kterým se dá považovat za plnohodnotný programovací jazyk (už se nejedná o starý skriptovací jazyk s omezenými možnostmi).

Místo podpůrné knihovny (Free)GLUT (*OpenGL Utily Toolkit*) obstarávající vykreslovací okna (a jejich události) pro OpenGL [8], je použita kombinace HTML DOM (Document Object Model) a JavaScriptu. Většinu rozdílů mezi použitím OpenGL a WebGL popisuje kapitola 3.1.

5.2 Aplikace pro vizualizaci modelů

Před vznikem samotné výukové aplikace došlo k vytvoření obsáhlejší grafické aplikace pro vizualizaci modelů dle požadavků z kapitoly 3.3. Její implementace měla sloužit k osvojení si programování v JavaScriptu a WebGL a k vyřešení většiny problémů na jeden zátah. Mezi problémy bych především zařadil implementaci pro mě méně známých funkcionalit, kterými jsem se dříve (alespoň ve WebGL) nezabýval. Výsledná aplikace mi pak měla zjednodušit přípravu a strukturování výukových kapitol.

První velký problém, který jsem musel při implementaci vyřešit, bylo nalezení optimálního způsobu, jak do aplikace nahrávat modely.

5.2.1 Nahrání a zpracování modelu

Vzhledem k tomu, že na vstupu aplikace by měla být data, která mají jednotný formát, bylo nutné vymyslet způsob jejich reprezentace. Volba padla na jazyk JSON, který se běžně používá k serializaci dat. Největší problém bylo přijít na to, jak dostat data modelu z modelovacího nástroje do souboru JSON. Po delším hledání a zkoušení jsem se spokojil s konvertorem `assimp2json` [12], který staví na OpenGL knihovně `Assimp` (*Open Asset Import Library*). Ta slouží k nahrávání dat modelů ze souboru do grafické aplikace.

Program Assimp2json nedělá nic jiného, než že pomocí knihovny Assimp zpracuje přiložený .OBJ (a .MTL) soubor a data následně serializuje do JSON formátu.

Zde bylo nutné uživatelům striktně přikázat používání tohoto programu, protože ostatní konvertory mezi soubory .OBJ a .JSON vytváří jinou vnitřní strukturu (reprezentaci dat) ve výstupním JSON souboru.

5.2.2 Výsledek

Výsledkem je aplikace, která umí z dodaných vstupních dat zpracovat jednotlivé části modelu a ten vykreslit na obrazovku (ve webové stránce). Na vstupu se počítá se souborem ve formátu JSON, který obsahuje veškerá data modelu (mimo textury). Uživatel je instruován, aby svůj model nejprve vyexportoval ve formátu `.OBJ (+ .MTL)` a následně použil doporučený program

`assimp2json` pro převedení .OBJ modelu do formátu JSON. Obrázky textur jsou nahrány do určené složky, ze které si je aplikace vezme při generování textur.

Po nahrání modelu je možné se volně pohybovat ve scéně, ve které je model vložen. Dále je možné pomocí kláves upravovat některé vlastnosti vykreslování a pomocí klikání myší interagovat s určitými částmi scény. Scéna je doplněna o možnost generování mlhy, vykreslení skyboxu a tzv. environment mapping (odrazy okolí na povrchu objektu). Chování aplikace lze navíc jednoduše upravit v přiloženém konfiguračním souboru (RenderSettings.js). Součástí této aplikace je také rozšíření knihovny `CTUGL`, která byla během vzniku bakalářské práce vytvořena. Knihovně se věnuje následující podkapitola 5.3.

5.3 Knihovna CTUGL

Během práce na výukové aplikaci (a především na aplikaci pro vizualizaci modelů) vznikla pomocná knihovna `CTUGL` (*Czech Technnical University Graphics Library*). Prvotní motivace ke vzniku této knihovny byla potřeba ekvivalentu knihovny `PGR-framework` ve WebGL. Pomocná knihovna PGR-framework je používaná především v předmětu PGR a slouží k zjednodušení některých záležitostí při programování v OpenGL. Hlavní náplní této knihovny je implementace funkcí, které dokáží nahradit nadbytečný kód ve výukových příkladech a semestrálních pracích. Mezi takový kód patří například rutinní vytváření a kompilace Shaderů, generování textur apod. - více v kapitole 3.2.

Dále slouží knihovna PGR-framework k integraci všech potřebných nástrojů a knihoven, se kterými se v PGR pracuje, do jednoho souboru (knihovny). Při práci na výukových příkladech nebo implementaci semestrální práce pak student nemusí složitě dohledávat jednotlivé soubory na internetu a stačí mu pouze stáhnout zmíněnou knihovnu PGR-framework.

Základní modul knihovny CTUGL realizuje potřebnou podmnožinu knihovny PGR-framework, která je navíc doplněna o nové funkcionality.

Vedle toho vznikla také rozšiřující skupina tříd, s jejichž pomocí lze celkem jednoduše naprogramovat složitější 3D webovou aplikaci. Zmíněné třídy reprezentují základní stavební kameny takové aplikace. Díky tomuto rozšíření je možné importovat modely z modelovacích programů (Autodesk Maya, Blender) či používat pokročilejší funkce vykreslování a zpracování uživatelského vstupu.

Knihovna vznikala při počátečním vývoji aplikace pro vizualizaci 3D modelů a umožňuje znovupoužitelnost nově implementovaných funkcionalit. Jinými slovy řečeno, pokud dojde k rozšíření aplikace pro vizualizaci 3D modelů, změny by se měly také objevit v knihovně, aby bylo možné nové funkce použít kdekoliv jinde.

Využití tohoto nadstandardního rozšíření (většiny tříd) je však omezeno v případě implementace semestrální práce v předmětu PGR. Hlavním důvodem je to, že uživatel je odstíněn od velkého množství problematiky, která je v předmětu PGR probírána. Pro případ semestrálních prací a úkolů je

tak možné použít pouze základní verzi knihovny, tedy jen hlavní soubor `ctugl.js`.

5.3.1 Obsah knihovny

Knihovna se skládá ze dvou logických částí. První část reprezentuje množinu funkcionalit, která může být využita při implementaci semestrální práce (nejsou nijak ohroženy požadavky na znalosti studenta). Většina funkcí kopíruje chování originální knihovny PGR-framework. Některé funkce navíc umožňují asynchronní nahrávání textur či zdrojových kódu shaderů. Tato část knihovny se nachází v souboru `ctugl.js` a je využívána ve všech demo příkladech výukové aplikace (výukové kapitoly se mimo jiné o pomocném balíčku tříd nezmiňují).

Druhá část knihovny se skládá z balíčku užitečných tříd, které lze různě kombinovat a používat při implementaci grafických aplikací. Tyto třídy umožňují jednoduše strukturovat kód celé aplikace, a to tak, aby nebyla tvořena tisíci-řádkovými soubory, ve kterých se dá jen těžko orientovat. Každá třída implementuje většinu záležitostí, které souvisí s "objektem", jež je třídou reprezentován. Například třída `Cursor`, reprezentující obecný kurzor na obrazovce, obsahuje funkci na jeho inicializaci, funkci na zpracování prokliku myší do scény či funkci na vykreslení kurzoru na obrazovku. Uživatel knihovny pak složitou problematiku řeší pár řádky ve svém kódu a o zbytek se de facto nemusí starat.

Je sice pravda, že tato část knihovny by neměla být používána při implementaci závěrečných prací předmětu PGR, ovšem i povaha výukových kapitol se snaží studenty nabádat k tomu, aby svůj kód strukturovali. Výsledkem tutoriálu je pak skupina tříd, které částečně odpovídají struktuře zmíněného balíčku. Uživatel se může později rozhodnout, zdali si funkcionalitu napíše sám, anebo využije již hotovou knihovnu. Samozřejmě pouze v případě, že se nejedná o práci spojenou s předmětem PGR.

Struktura souborů knihovny CTUGL

- `ctugl.js` - Základní modul knihovny volně použitelný při práci spojené s předmětem PGR.
- `utils/` - Složka obsahující rozšiřující balíček tříd. Jeho použití je možné pouze pro potřeby nesouvisející s předmětem PGR.
- `Camera.js` - Třída reprezentující kameru. Obsahuje aktuální pozici ve scéně včetně natočení a směru pohledu. Na základě těchto dat poskytuje aktualizované matice *View* a *Projection*.
- `CubeMap.js` - Třída umožňující jednoduché přidání skyboxu do scény. Také podporuje metodu environment mapping.
- `Cursor.js` - Třída reprezentující obecný kurzor a jeho metody. Proklik do scény je realizován pomocí přídavného FrameBufferu. Součástí je i metoda a geometrie pro zobrazení kurzoru na obrazovce.

- `Light.js` - Třída reprezentující obecný zdroj světla. Dle parametrů je možné určit, zdali se má jednat o světlo směrové, bodové nebo o reflektor.
- `Mesh.js` - Třída reprezentující síť vrcholů daného modelu. Jeden model se může skládat z více sítí (meshů). V této třídě dochází k napojení dat modelu do paměti WebGL.
- `Model.js` - Třída reprezentující jeden model ve scéně. Obsahuje aktuální geometrickou transformaci modelu a seznam meshů, jež tvoří geometrii modelu.
- `RenderSettings.js` - Třída, která slouží jako soubor s aktuálním nastavením aplikace. Některé nastavení se dynamicky mění za běhu aplikace v závislosti na akcích uživatele. Uživatel tak může ovlivňovat některé kroky při renderování (vypínání mlhy, vypínání skyboxu, skrytí kurzoru).
- `Shader.js` - Třída pro reprezentaci shader programu. Obsahuje funkce na kompilaci shaderů a sestavení programu.

5.4 Front End

Front-End aplikace se dělí na tři hlavní části. První se stará o vykreslování obsahu do stránky a zpracovávání uživatelských vstupů. Tato část je reprezentována hierarchií React komponent. Druhá část slouží především ke komunikaci s back-endem (serverem) aplikace a je reprezentována balíčkem tříd `Front Controller` - viz kapitola 4.2 . Tím je umožněno dynamické načítání obsahu do stránky. Poslední část reprezentuje obsah jednotlivých kapitol. Každá kapitola má svojí vlastní složku, čímž je obsah demo příkladů jednotlivých kapitol od sebe izolován.

5.4.1 Vykreslování obsahu

Vykreslování (renderování) obsahu do stránky je řízeno hierarchií ReactDOM. Jak již bylo řečeno v kapitole 4.2.3, aktualizace obsahu je závislá na změně stavu konkrétní komponenty a vykreslení je provedeno pouze na této komponentě a na podstromu jejích potomků. Všechny React komponenty této aplikace jsou umístěny ve složce `React`. Každá React komponenta reprezentuje logickou část webu. V samotné složce se tak nachází komponenty, které web rozdělují na hlavní bloky: *Hlavička* (`Header.js`), *Boční menu* (`SideMenu.js`) a *Obsah stránky* (složka `Content`). Tyto komponenty jsou vkládány do kmenové komponenty `App.js`, která reprezentuje vstupní bod dynamického zpracovávání stránky. Veškeré HTML a obsah, které jsou ve struktuře DOM nad tímto prvkem (komponentou), již chováním Reactu ovlivněny nejsou. Výsledný vzhled GUI aplikace je vidět na obrázku 5.1.

Složka Content obsahuje komponentu `Content.js`, která reprezentuje hlavní kontejner, do kterého se vykresluje obsah zvolené kapitoly.

Ve vrchní části je vykreslena komponenta `ChapterMenu.js`, která slouží jako menu k přepínání mezi obsahem kapitoly ("Tutoriál"), mezi záložkou "Výsledné Demo", která slouží k prezentaci využití probrané látky v komplexnější aplikaci, a mezi záložkou "Komentáře", ve které je možné číst a přidávat komentáře k dané problematice. Komentáře se vážou na konkrétní kapitolu a zobrazují se pouze u ní.

Pro každou možnost v menu existuje vlastní komponenta. Pro obsah kapitoly (výukový text s obrázky) slouží komponenta `Tutorial.js`, jež je výchozí komponentou, která se k dané kapitole vykreslí. To znamená, že pokud uživatel klikne v bočním menu na nějakou kapitolu, tak první co uvidí, je text kapitoly. Její komentáře a nebo položku "výsledné demo"si pak může zobrazit kliknutím do horního menu.

Komponenta `Demo.js` se zobrazí po kliknutí do horního menu, v českém jazyce na popisek "Výsledné demo". V této komponentě se nachází ovladatelný demo příklad, popřípadě jeho stručný popisek. Vzhledem k tomu, že jsem se na začátku špatně rozhodl, jak chci výuku strukturovat, tak k využití této záložky nakonec nedošlo. Z testování aplikace vyplynulo, že její využití bude muset být upraveno na základě poznámek uživatelů (kapitola 6.5.1). Tato úprava ovšem znamená větší časovou investici, která už nespadá pod zadání bakalářské práce, proto je uvažována jako výhled do budoucna - viz kapitola 8.1.

Komponenta `Comments.js` má na starost zobrazování a přidávání komentářů. Obsahuje funkci, která umí hierarchicky seřadit pole nesetřízených komentářů, které přijdou v odpovědi ze serveru. Respektive, komentáře setřízené po příchodu ze serveru jsou, ale pouze podle data jejich přidání. V případě této aplikace je možné reagovat na již existující komentáře (odpovídat na ně) a kvůli tomu je potřeba příchozí komentáře ze serveru správně strukturovat, aby na sebe navazovaly.

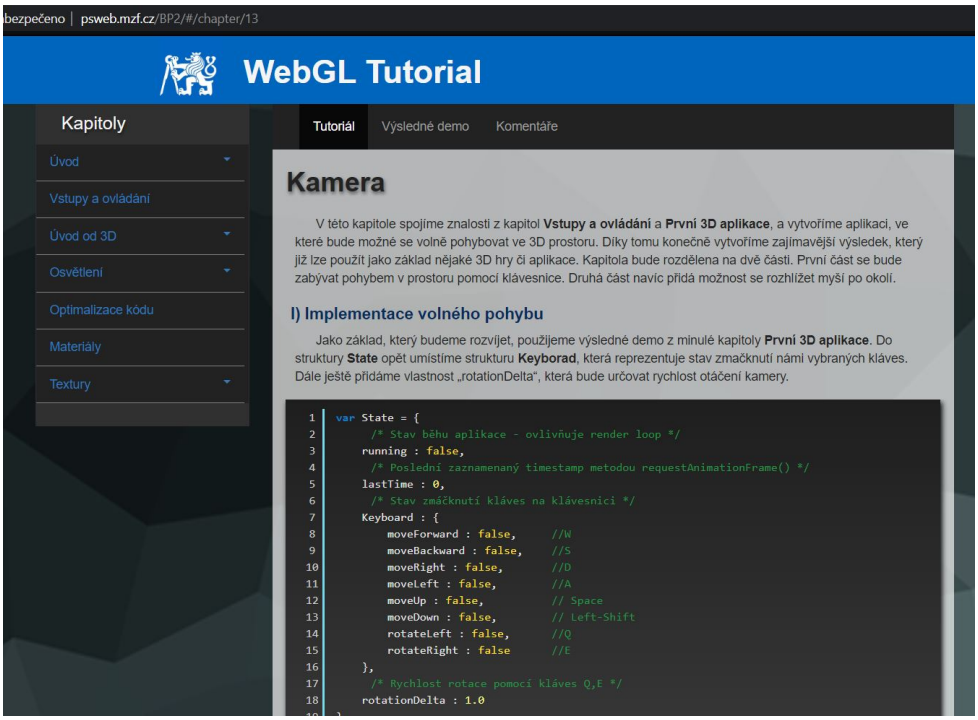
Samotná funkcionalita komentářů je připravena na napojení na databázi, které prozatím realizováno nebylo. Hlavním důvodem je absence accountingu - nelze zakládat uživatelské účty. Toto rozšíření je plánováno jako práce do budoucna a detailněji rozepsáno v kapitole 8.1.

Jednotlivé komponenty využívají API pro komunikaci se serverem a to především k získávání vykreslovaného obsahu. Funkce API je popsána v kapitole 5.4.2.

JSX

JSX je syntaktické rozšíření jazyka JavaScript a ve své podstatě umožňuje zápis HTML a javascriptového kódu dohromady [15]. To je sice možné i v klasickém JavaScriptu, ovšem pouze zápisem HTML do proměnné a její následné nahrání do některého z HTML elementů pomocí JavaScriptu. V případě JSX je možné psát JavaScript a HTML téměř neodděleně, skoro jakoby se jednalo o jeden jazyk.

Použití JSX je velmi doporučováno právě při využití frameworku React.js. Díky tomu je zjednodušen návrh a implementace jednotlivých komponent.



Obrázek 5.1: Ukázka GUI aplikace

JSX je pouze tzv. *syntactic sugar*, tedy slouží pouze ke zpestření / zkrášlení kódu pro programátora. Samotný kód má funkcionalitu úplně stejnou, jako by žádné JSX nebylo využito. Použití tohoto rozšíření pak počítá s tím, že výsledný kód je před použitím na webu překompilován do čistého JavaScriptu. To může být při vývoji aplikace nepříjemné, a proto existuje i skript, který lze do stránky přiložit a kompilace tak probíhá pokaždé až při načítání stránky. Ovšem tento způsob je určen pouze pro případy vývoje aplikace a při jejím nasazení by měl být odstraněn (právě kvůli kompilaci za běhu, která může načítání stránky zpomalit).

Během mé snahy využít JSX jsem narazil na problémy, které mě nakonec od jeho použití odradily. Jedním z problému je fakt, že JSX kód je kompilován do staré verze JavaScriptu (ES5), která nepodporuje některé funkcionality (například postrádá možnost modularizace kódu, se kterou jsem počítal). Použití staré verze JavaScriptu je dobré v případě, že vývojář chce, aby jeho webová aplikace byla zpětně kompatibilní se starými prohlížeči. Ovšem v případě mé výukové aplikace jsem počítal s využitím nových verzí jazyka JavaScript (už jen kvůli WebGL2) a na jeho staré verze jsem se nechtěl vázat. Nemluvě o tom, že zastaralé prohlížeče používá malé procento uživatelů, mezi kterými se určitě nebudou nacházet zájemci o problematiku WebGL2.

Při zpětném pohledu na problematiku JSX musím uznat, že v příští práci bych se vydal tímto směrem. Správa klasických React komponent je bez JSX poměrně složitá a kód není moc přehledný. Následující práci bych řešil pomocí správce balíčků `NPM` (*Node.js Package Manager*)[1], který umožňuje použití široké škály knihoven a rozšíření. Rozšíření jsou instalována a spravována pomocí nmp příkazové řádky. Ta umožňuje i kompilaci JSX skriptů a podporuje interní modularizaci (pouze na straně vývojáře). Výsledný javascriptový kód je transpilován do jednoho velkého souboru, který již dále není upravován. Úpravy jsou prováděny na straně vývojáře, který je nakonec opět zahrne do nové verze transpilovaného kódu a tím nahradí původní javascriptový soubor.

5.4.2 Komunikace se serverem

Komunikace se serverem je řízená centralizovaně pomocí jedné třídy (modulu), který slouží jako rozhraní pro posílání Http požadavků na server a přijímání jejich odpovědí. Modul je reprezentován souborem `AjaxAPI.js`, který poskytuje *singleton* třídu AjaxAPI. Ta obsahuje výčet metod, které reprezentují jednotlivé systémové požadavky. Proto metody zde obsažené obsluhují celé procesy, jako jsou *Získání obsahu kapitol* nebo *Získání obsahu konkrétní kapitoly*. V budoucnu pro případ rozšiřování aplikace o *accounting* (uživatelské účty) by přibyly metody pro registraci a přihlášení uživatelů.

Soubor AjaxAPI.js je umístěn v balíčku s názvem `Front Controller`, který reprezentuje stejnojmennou komponentu (viz kapitola 4.2). Tento balíček dále obsahuje soubory `Locales.js`, `Settings.js` a `Storage.js`.

Locales.js

Tento soubor obsahuje lokalizace (překlady) jednotlivých částí webu. Jedná se o statické překlady prvků jako jsou popisky, tlačítka a další obsah samotné stránky (nikoliv obsah kapitol). Díky tomu je možné jednoduše přepínat mezi jazyky a také nové přidávat. Soubor zatím počítá s jazyky *čeština* a *angličtina*, ovšem obsah kapitol je zatím pouze v češtině.

Settings.js

Tento soubor reprezentuje jednoduchý konfigurační soubor aplikace. Slouží především k nastavení aplikace v případě jejího přesunu na jiný hosting, nebo pro případ restrukturalizace složek uvnitř aplikace. V nastavení je také možné určit, která kapitola se bude zobrazovat na uvítací stránce (při načtení webu).

[1] `https://www.npmjs.com/`

Storage.js - Cache

Z důvodu většího objemu dat, který reprezentuje jednotlivé kapitoly, bylo nutné vytvořit strukturu (cache), která bude již stažená data ukládat na straně klienta. Díky tomu je výrazně snížen nárok na stahování obsahu ze serveru. Cache je reprezentována singleton třídou Storage.js, která poskytuje metody pro zápis nových dat a pro získání uloženého obsahu. Třída navíc jednou za čas kontroluje dobu, po kterou se daný záznam v cache nachází, a po uplynutí stanoveného limitu je konkrétní obsah smazán.

5.4.3 Renderování obsahu kapitol

Obsah kapitol je základem výukové aplikace. Obsahem je myšlen text, obrázky, ukázky kódu a interaktivní demo příklady, které demonstrují probíranou problematiku. O vykreslování samotného obsahu do stránky se stará React komponenta `Tutorial.js`, která je popsána v kapitole 5.4.1. Struktura textu kapitoly je velmi důležitá pro pochopení problematiky. Text je potřebné doprovázet ukázkami kódu, obrázky a případně demo příklady, které reprezentují probranou problematiku v praxi. Uživatel díky tomu hned vidí to (a ve většině případů s tím i interaguje), o čem se v dané kapitole dočetl.

Díky různorodosti obsahu bylo nutné vybrat správný formát reprezentace dat celé kapitoly. Volba padla na kombinaci serializovaného textu v databázi (JSON) a složky obsahující zbytek potřebných dat kapitoly, mezi která patří obrázky a demo příklady. Jedna kapitola je tvořena serializovaným polem JavaScript objektů reprezentujících jednotlivé části kapitoly (viz ukázka kódu 3), které je uloženo v databázi, a příslušnou složkou se zbytkem potřebných dat, která se do databáze neukládají.

Obsah každé kapitoly je členěn do bloků, které mají v serializovaném JSON poli své vlastní pojmenované atributy uvozené daným klíčem. Při parsování dodaného JSON pole v komponentně Tutorial.js je v závislosti na klíči objektu vytvořen příslušný HTML element, jehož obsah je doplněn daty uvozenými tímto klíčem. V případě klíče "caption"je vytvořen element podnadpisu `<h4>`, v případě klíče "paragraph"je vytvořen element odstavce `<p>`. Za klíčem "image"se nachází místo obyčejné hodnoty (textu) další JavaScript objekt, ve kterém je uložena relativní cesta k obrázku (vzhledem ke složce patřící dané kapitole), popisek obrázku a odkaz na zdroj obrázku.

Podobným způsobem funguje objekt demo příkladu, který je uvozen klíčem "example". V objektu se nachází relativní cesta k HTML dokumentu příkladu a popisek příkladu. React komponenta se postará o vytvoření `<iframe>` elementu, který umožňuje vkládat externí HTML dokumenty do webové stránky (jinými slovy HTML stránka v HTML stránce). Díky tomu je kód demo příkladu oddělen od kódu výukové aplikace. Popisek demo příkladu je vykreslen jakožto odkaz, pomocí kterého lze demo příklad samostatně otevřít na nové stránce.

```
[
    {"caption" : "nadpis"},
    {"paragraph" : "Lorem ipsum dolor sit amet, consectetuer adipiscing"},
    {"code" : {
        "data" : [
            "function foo(i){",
            "   return 'bar ' + i;",
            "}"
        ],
        "lang" : "javascript"
    }},
    {"img" : {
        "src" : "image.png",
        "caption" : "popisek obrazku",
        "img_source" : "htttp://a.b.c"
    }},
    {"example" : {
        "caption" : "popisek demo prikladu",
        "src" : "demo.html"
    }}
]
```

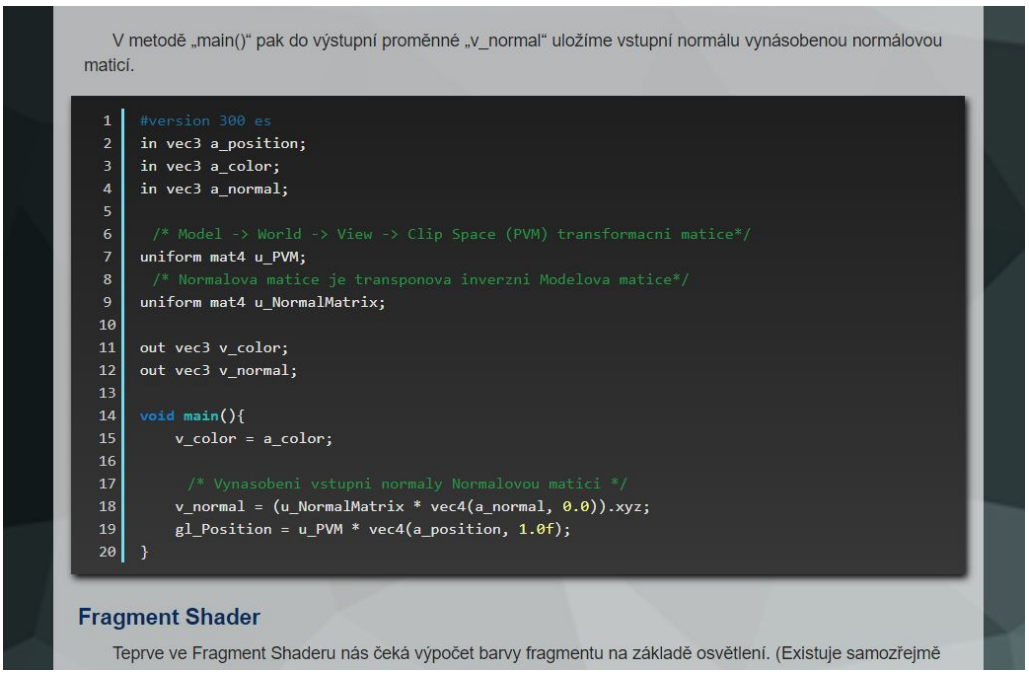
Listing 3: Ukázka struktury JSON

Lehce odlišný přístup potřeboval blok obsahující ukázku zdrojového kódu. Ten za klíčem "code"obsahuje také JavaScript objekt, ve kterém se nachází definice jazyka, ve kterém je ukázka kódu napsána. Definice jazyka je potřeba pro následné obarvení kódu pomocí `HLJS` - viz následující podkapitola. Dále se v objektu nachází zdrojový kód, který je po řádcích vložen v klasickém poli. Tento přístup jsem zvolil poté, co mi JSON parser nechtěl zpracovat text, ve kterém je obsaženo odřádkování (`\r\n`).

Výsledkem parsování dodaného JSON pole je pole HTML elementů (reprezentovaných React komponentami), které je nakonec vykresleno se zbylým obsahem komponenty Tutorial do rodičovské komponenty `Content`.

HLJS (highlight.js)

Z důvodu, že nemalou část kapitol tvoří ukázky zdrojového kódu, bylo nutné zařídit jeho přehlednost a čitelnost. Ta je mimo jiné podpořena "obarvením"klíčových slov syntaxe daného jazyka (neboli *highlighting* - zvýrazňování). Pro inspiraci jsem navštívil ostatní výukové weby, abych zjistil, jakým způsobem kód obarvují. Nejčastějším pomocníkem se jevila knihovna `HLJS (highlight.js)` [17], kterou například využívá portál `WebGLFundamentals.org` [16].



Obrázek 5.2: Ukázka zvýraznění zdrojového kódu

Knihovna HLJS může zdrojový kód zpracovávat ještě v serverové části aplikace v případě, kdy je tato část psána v JavaScriptu (Node.js). Druhá možnost je zpracování již vykresleného obsahu (HTML kódu) ve stránce. Tento způsob jsem využil v mé práci a postačilo mi k tomu vložit do stránky potřebný skript. Vzhledem k tomu, že jsem chtěl uživatelům prezentovat již obarvený kód, rozhodl jsem se, že při načtení stránky bude zdrojový kód skrytý. Teprve až po zpracování všech bloků zdrojového kódu knihovnou HLJS dojde k odkrytí těchto bloků. Uživatel je pak odstíněn od nepříjemného probliknutí nezpracovaného textu.

Vzhledem k tomu, že základní stylování je závislé na šabloně pro světlé editory (tedy bílé pozadí), rozhodl jsem se styly upravit do podoby zvané *Dark Theme*, neboli tmavé prostředí (obrázek 5.2). Tento vzhled je většinou programátorů upřednostňován, protože je pro oči příjemnější. Kolekce mnou upravených stylů se nachází v souboru `hljs-dark-theme.css`.

Později jsem zjistil, že knihovna HLJS podporuje i jiné barvy prostředí pomocí tzv. templatů, ale to už jsem měl své styly upravené.

Číslování řádků není v základní verzi HLJS implementováno. K jeho dodání je potřeba využít rozšíření `highlightjs-line-numbers.js`[2]. Vzhled číslování a jeho pozici lze upravit přiloženým souborem CSS tříd.

[2] `https://github.com/wcoder/highlightjs-line-numbers.js`

5.5 Výukové příklady

Výukové demo příklady jsou realizovány jako mini-projekty vkládané do textu kapitol. Každý příklad je reprezentován vlastní složkou se svými soubory. První část příkladů je realizována jedním HTML souborem, ve kterém se nachází veškerý kód. Čtenář díky tomu vidí, jakým způsobem má propojit HTML stránku se skriptem grafické aplikace.

Pokročilejší příklady jsou rozděleny na více souborů, počínaje rozsáhlejšími shadery, konče strukturou inspirovanou rozšiřujícím balíčkem knihovny CTUGL (Kapitola 5.3).

Demo příklad je do textu kapitoly vložen pomocí značky `<iframe>`, která umožňuje vložení celé HTML stránky do jiné HTML stránky. Díky tomu je také možné demo příklad otevřít v jiné záložce prohlížeče jako samostatnou stránku (odkaz je obsažen jako součást popisku pod demo příkladem).

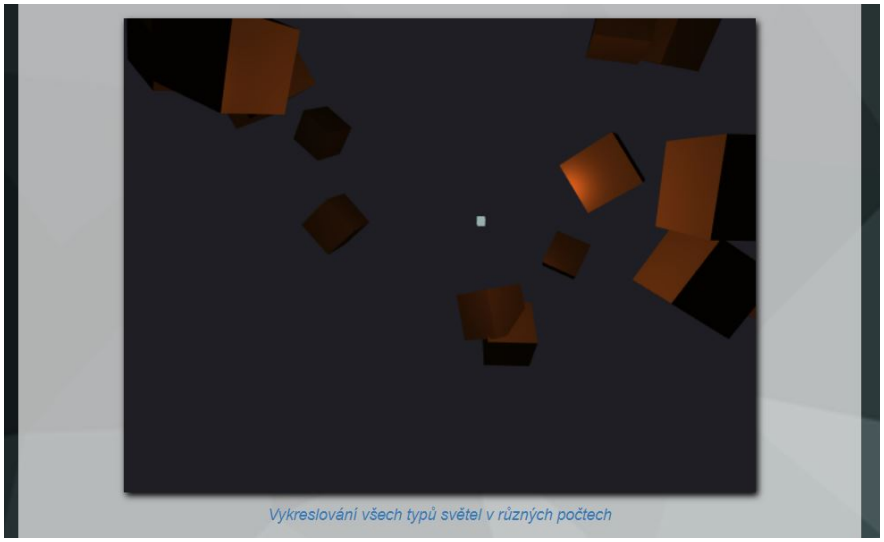
Čtenář si tak může nerušeně prohlédnout příklad v plném rozlišení a zabývat se pouze jeho obsahem - není zbytečně rušen kódem výukové aplikace. Celý zdrojový kód příkladu lze momentálně zobrazit pouze pomocí nástrojů webového prohlížeče. První možností je stisk klávesové zkratky `CTRL + U`, která zobrazí zdrojový kód stránky. Druhou možností je otevření vývojářské konzole pomocí klávesy `F12`. Způsob, jak si zdrojový kód zobrazit, je čtenářům dostatečně vysvětlen.

Při implementaci jsem se setkal s problémem, kdy některé webové prohlížeče nezobrazují obsah demo příkladů správně. Někdy není obsah vykreslen vůbec, což bývá způsobeno absencí podpory WebGL2, někdy je obsah vykreslen s chybami (této problematice se také věnuje kapitola 6.5.2). S chybně vykresleným obsahem jsem se setkal například na svém mobilním telefonu (Android 7.0 `NRD90M`, aplikace Google Chrome ve verzi `81.0.4044.138`). Na obrázku 5.3 je znázorněn rozdíl ve vykreslování v kapitole *Materiály*. V případě počítače je obsah vykreslen správně (osvětlení a materiály), ovšem mobilní verze tento způsob nezvládá. Podobných problémů se vyskytuje více a ve většině případů jsou spojeny s implementací WebGL v konkrétních prohlížečích.

Pro správné vykreslení obsahu je dle mého výzkumu nejlepší použít webové prohlížeče `Google Chrome` nebo `Mozilla Firefox`. V ostatních prohlížečích není obsah vykreslen vůbec nebo jen částečně. Pro referenci použitelnosti webových prohlížečů je dobré sledovat stránku `Can I Use` [5]. Důležitá je také verze prohlížeče a jeho podpora grafické akcelerace. Grafickou akceleraci je možné povolit či zakázat v nastavení webového prohlížeče. Poslední podmínkou jsou aktualizované a kompatibilní ovladače grafické karty.

V případě mobilních telefonů je podpora různorodá a zřejmě záleží na konkretní verzi operačního systému `Android`. Mobilní telefony založené na jiných operačních systémech (iOS) mají podporu pochybnou. Nejlepším mobilním prohlížečem podporujícím výukovou aplikaci je zřejmě mobilní verze Google Chrome. Ta vykreslování až na uvedené nedostatky zvládá.

(a) : Mobilní WebGL



(b) : Desktopová WebGL

Obrázek 5.3: Porovnání vykreslení na mobilním zařízení a na počítači

5.6 Tvorba obsahu kapitol

Pro největší komfort jsem se rozhodl psát kapitoly v textovém editoru, konkrétně `Microsoft Word`. Ten obsahuje většinu potřebných funkcionalit, mezi které patří korektura textu nebo členění obsahu do kapitol. Do budoucna ovšem počítám s implementací administrace, ve které bude příprava článků jednodušší (viz kapitola 8.1).

Jako nejlepší způsob se jeví některý `Wysiwyg` editor (What You See Is What You Get). Ve volném překladu se jedná o editor, který výsledný text bude prezentovat stejným způsobem, jako vypadá při jeho zápisu. To se například o HTML říct nedá, protože součástí formátování textu je velké množství HTML značek. Tento typ editoru se chová například jako zmíněný Microsoft Word a o formátování pomocí HTML se postará sám.

Během psaní kapitol vznikl obsáhlý dokument, který se dělí na kapitoly a podkapitoly stejným způsobem, jak je možné pozorovat na webu. Formátování textu na webové stránce je závislé především na stylování hlavních elementů. Veškerý text tak přebírá vlastnosti svého elementu (odstavec, nadpis). Pro případné úpravy uvnitř textu je nutné použít HTML značky - tedy pro případ, kdy je potřeba změnit barvu či tloušťku některé části odstavce či nadpisu.

5.6.1 Tématika textu

Vzhledem k tomu, že výukové kapitoly mají především podporovat předmět PGR, tak jejich obsah vychází z náplně jeho přednášek a cvičení. Základem pro psaní textu pro mě byly již hotové demo příklady, které jsem implementoval ještě před psaním obsahu kapitol. Psaní pak bylo pro mě mnohem jednodušší. Demo příklady jsem derivoval z komplexní `Aplikace pro vizualizaci 3D modelů` (kapitola 5.2), kde jsem poprvé implementoval danou problematiku.

Přístup k psaní kapitol stylem *Rozděluj a panuj* byl pro mě osobně velmi dobrý a určitě bych ho doporučil i pro tvorbu jiných prací. Především jsem se vyhnul problémům, kdy bych nevěděl o čem psát, nebo bych úplně zapomněl na popsání některých důležitých záležitostí. Implementací složitější aplikace jsem si pojistil, že jsem na nic nezapomněl.

5.6.2 Přenos textu do databáze

Aktuální způsob přenosu textu kapitol na webový server je uživatelsky velmi nepřívětivý. Vzhledem k tomu, že kapitoly píši sám, tak to neberu jako velký problém, ovšem do budoucna počítám se zmíněným wysiwyg editorem, jehož obsah bude zpracován automaticky.

Nyní je obsah přenášen ručním převodem textu do JSON pole a jeho následným nahráním do databáze. Zmíněné pole obsahuje struktury podobné JavaScript objektům. Každá struktura reprezentuje jeden prvek textu kapitoly (nadpis, odstavec, obrázek, ukázka kódu, demo příklad). Typ obsahu takového objektu je určen klíčem, kterým je uvozen obsah uvnitř objektu.

V případě obrázku, ukázky zdrojového kódu, a demo příkladu je obsah reprezentován dalším vnořeným objektem. Ten bylo nutné použít pro dospecifikování potřebných hodnot, mezi které patří cesta k souboru, popisek nebo typ programovacího jazyka.

Toto JSON pole v případě stažení kapitoly na front-end prochází parsováním v React komponentě Tutorial.js, jehož výsledkem je pole HTML elementů.

5.7 Shrnutí implementace

Výsledkem časově náročné implementace jsou tři hlavní výstupy. Nejdůležitějším výstupem je výuková aplikace splňující požadavky vycházející ze zadání bakalářské práce. Druhým výstupem je podpůrná knihovna CTUGL (kapitola 5.3), jejíž základní modul byl využit k implementaci demo příkladů ve výukové aplikaci. Posledním výstupem je aplikace pro vizualizaci 3D modelů (kapitola 5.2), která sama o sobě nijak s BP nesouvisí, ale pro vznik výukové aplikace byla důležitá.

Výuková aplikace

Během implementace výukové aplikace jsem si vyzkoušel pro mě nové technologie, mezi které patří React.js a Bootstrap. Bohužel některé cesty, které jsem zvolil, nebyly nejlepší a díky tomu není aplikace v takovém stavu, který bych si přál. Svůj účel samozřejmě může aplikace bez problému plnit, ovšem s menšími nedostatky, které byly objeveny především v závěrečné fázi testování.

Jako největší problém vidím špatné rozvržení práce s frameworkem React.js. Případné budoucí úpravy jsou složitější na provedení a celková přehlednost kódu není na nejlepší úrovni. Vzhledem k tomu, že jsem pracoval s React.js poprvé, tak jsem také nevěděl, jak správně řešit přístup ke komunikaci mezi komponentami nebo jak provádět jejich aktualizaci. Problém mi také dělalo propojení frameworku React.js s knihovnou HLJS.js, především synchronizace vykreslování komponent s následným zvýrazňováním zdrojového kódu.

Při budoucí práci s frameworkem React.js bych využil správce balíčků NPM, s jehož pomocí je možné jednoduše importovat rozšíření, modularizovat aplikaci do logických celků či kompilovat JSX kód. Tím by se kód aplikace stal více přehledným a jednoduše upravitelným.

Výsledkem implementace výukových kapitol je série demo příkladů, jejichž ukázky se nachází v příloze B. Celkově vzniklo dvacet příkladů obsažených v patnácti implementačních kapitolách. Větší část příkladů je navíc doplněna výukovým textem.

Knihovna CTUGL

Na počátku jsem se zaměřil pouze na implementaci funkcí, které jsou obsaženy v knihovně PGR-framework. Knihovna vznikala při vývoji aplikace pro vizualizaci modelů a proto byla postupně upravována k potřebám WebGL (došlo k úpravám některých funkcí a další funkce přibyly). Hlavní modul byl později rozšířen o balíček tříd, které byly implementovány jakožto součást aplikace pro vizualizaci modelů.
Třídy jsou upravené tak, aby je bylo možné nezávisle na ostatních použít při implementaci jakékoliv jiné 3D WebGL aplikace.

Úvodní nápad na strukturování aplikace (a tedy i knihovny) jsem přebral z výukového portálu LearnOpenGL.com (Kapitola 2.2). Strukturu a její obsah jsem ovšem přizpůsoboval svým potřebám a implementoval dle svých představ.

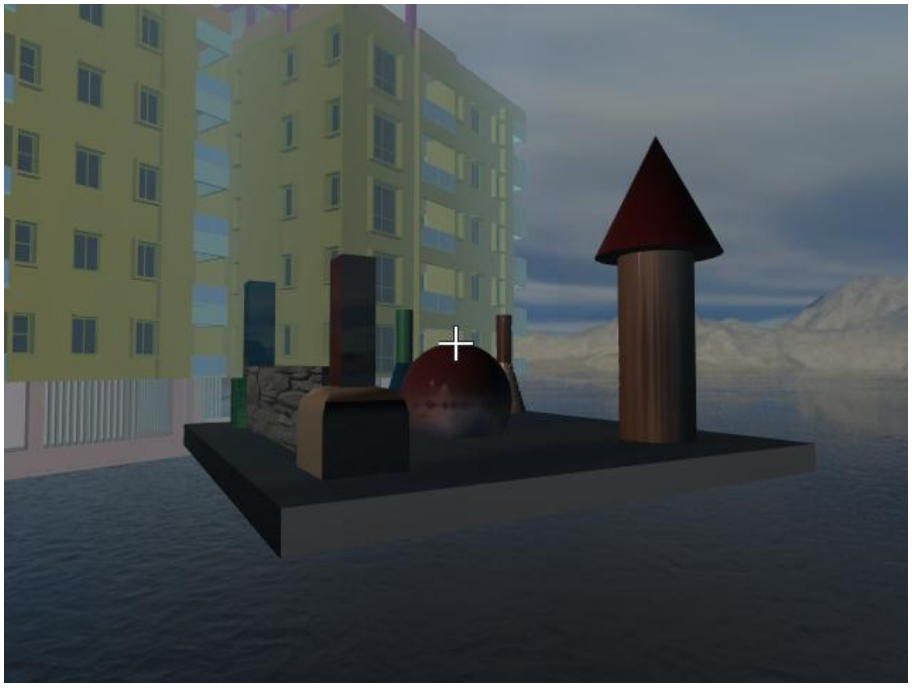
Použití knihovny je možné prozatím pouze pomocí vkládání jednotlivých skriptů s třídami do hlavní stránky (HTML souboru). Později je uvažováno s modularizací a vystavením knihovny na veřejném serveru, což práci s knihovnou usnadní.

Aplikace pro vizualizaci modelů

Implementace této aplikace předcházela vývoji výukové aplikace. Její hlavní podstata je popsána v kapitole 5.2. Během implementace jsem chtěl kód strukturovat dle nových možností jazyka JavaScript ES6 a jeho rozložení přirovnat struktuře podobné objektově orientovaným jazykům, jako jsou C++ nebo Java. Bohužel jsem narazil na problém, kdy nelze jednoduše vkládat JavaScriptové třídy do jiných. To lze vyřešit pouze modularizací skriptů, k čemuž je potřeba webový server.

Druhou možností je modularizace na bázi správce balíčků NPM, který je spuštěn na straně vývojáře. Výsledný kód je ovšem transpilován do velkých JavaScriptových souborů, které tento typ hierarchie neobsahují. To mi v případě vývoje grafické aplikace přijde nevhodné. Proto je aplikace prozatím závislá na vkládání jednotlivých tříd do kořenového HTML souboru. Jedná se o stejný princip, na kterém je závislé používání tříd knihovny CTUGL.

Obrázek 5.4: Ukázka aplikace pro vizualizaci modelů (Skybox + Kurzor). Model představuje moji Minecraft mapu převedenou do .OBJ pomocí programu `Mineways` [39].



Obrázek 5.5: Ukázka aplikace pro vizualizaci modelů (Skybox + Kurzor + Mlha + Environment mapping). Model domu je stažený z `Turbosquid` [38], model uprostřed je můj vlastní vytvořený v Autodesk Maya.

Kapitola 6
Testování

Kapitola pojednává o návrhu a realizaci testování výukové aplikace. Konkrétně se jedná o `Uživatelské testování`, které má za úkol zjistit použitelnost výsledné aplikace. Jednou z podmínek je opakování testování v průběhu implementace pro dřívější odhalení případných nedostatků a chyb.

6.1 Cíle testování

Hlavním cílem testování výukové aplikace je zjištění její přehlednosti a použitelnosti budoucími uživateli. Předmětem testování bude především přehlednost uživatelského rozhraní (GUI), které bude testováno dle praktik popsaných v knize `Nenuťte uživatele přemýšlet`, jejímž autorem je pan Steve Krug [14]. Tím dojde k odhalení případných chyb a nejasností při používání aplikace.

Druhá část testování se bude zabývat srozumitelností a přínosem obsahu výukové aplikace. Testujícím uživatelům, kteří si projdou alespoň větší část výukových kapitol, bude předán vědomostní test, který má zjistit míru porozumění probrané látky. Otázky budou pokrývat většinu probraných témat. Vzhledem k tomu, že většina testovaných uživatelů nemá žádné znalosti v oblasti programování počítačové grafiky, tak nemá smysl zjišťovat jejich znalosti na začátku testování aplikace.

Během procházení kapitol budou navíc testovaní uživatelé zpracovávat přiložené úkoly. Na konci testování budou úkoly vyhodnoceny a zaneseny spolu s časovou náročností do porovnávací tabulky. Následně dojde k zhodnocení užitečnosti aplikace v porovnání s ostatními tutoriály zmíněnými v kapitole 2.

6.2 Úvod do testování aplikace

V úvodu implementace bylo nutné pochopit základní problematiku testování *Uživatelských rozhraní*. K tomu velmi dobře posloužila zmíněná kniha `Nenuťte uživatele přemýšlet`. Vybraná část knihy (`Kapitola 9`: *Testování použitelnosti za 10 centů denně*) popisuje základní chyby při návrhu uživatelských testů, základní pravidla testování a následně je představena ukázka fiktivního testování aplikace.

Výsledkem uživatelských testů jsou data reprezentující přehlednost a použitelnost dané aplikace (respektive jejího uživatelského rozhraní). Díky tomu lze také najít slabá místa, která mohou případné uživatele odrazovat od používání testované aplikace.

6.3 Uživatelské testování

Při začátcích vývoje aplikace je vhodné uspořádat tzv. *Skupinové testování*. Jedná se o nejlepší způsob, jak získat co nejvíce užitečných informací (názorů) za krátký čas. Skupinového testování se účastní skupinka lidí (od tří až po deset) a spočívá v kladení otázek na hlavní rysy vznikající aplikace, ke kterým se jednotlivý účastníci testování vyjadřují. Může se probírat vzhled, rozložení aplikace a vlastně cokoliv, k čemu je dobré znát názor budoucích uživatelů. Díky většímu počtu účastníků testování lze rozvinout diskuzi, ve které je vyřčeno mnoho názorů a následných reakcí. Názor většiny pak lze brát jako důležitou informaci, kterou lze při vývoji aplikace zohlednit [14, str. 115].

Skupinové testování však slouží pouze před a při začátku implementace aplikace (systému). Testování již implementovaných funkcí je nutné provádět jednotlivě pomocí *Testů Použitelnosti*. Jedná se většinou o monitorované používání vznikající aplikace testovaným uživatelem, přičemž je zjišťována přehlednost a použitelnost jejích částí. Vývojář (nebo jejich skupina) pak vidí, které části aplikace jsou pro uživatele problematické a hůř pochopitelné. Tento typ testů by měl probíhat ve více kolech (s více uživateli) a opakovaně (iterovaně). Je totiž dobré provádět testování častěji, odhalené chyby pak opravovat a výsledek opět testovat.

6.3.1 Vybraní uživatelé

Výběr uživatelů je proces, který by neměl být důležitější, než samotné testování. Respektive jejich výběr by neměl ohrozit výsledný počet provedených testů a čas nad nimi strávený. Vždy je tedy lepší vybrat takové lidi, kteří splňují nějaké základní požadavky, než hledat někoho, kdo splňuje několik kritérií [14, str. 120]. Vzhledem k tomu, že výuková aplikace nebude určena jen pro studenty ČVUT FEL (především pak studenty předmětu PGR), rozhodl jsem se k výběru širšího spektra lidí. Podařilo se mi sehnat 4 uživatele, kteří byli ochotni se testování podrobit. Adeptů jsem sehnal více, ale z obav, že nebudou látku zvládat, se testování nakonec nezúčastnili.

	Pohlaví	Věk	Absolvoval PGR	Student FEL
Tester1	Muž	49	Ne	Bývalý (Ing.)
Tester2	Muž	22	Ne	Ano
Tester3	Muž	42	Ne	Bývalý (Ing.)
Tester4	Muž	24	Ano	Ano

Tabulka 6.1: Tabulka vybraných testerů

Mezi testovanými uživateli se objevil bývalý student předmětu PGR, dále student ČVUT FEL, pro kterého je problematika programování počítačové grafiky novinkou, ale také lidé, kteří reprezentují případné zájemce z řad příležitostných programátorů (viz tabulka 6.1). Tímto výběrem tak byla pokryta většina skupin potencionálních návštěvníků výukového webu.

6.3.2 Způsob testování

Testování proběhly celkem dvě iterace, jedna úvodní a druhá v závěru implementace. Jedná se o celkem nízký počet opakování, který dle knihy pana Kruga není úplně ideální, ovšem díky menší velikosti celé aplikace nebylo moc věcí, které testovat. Přišlo mi zbytečné ztrácet čas nad vymýšlením nových testů, proto jsem tento čas raději investoval do vývoje. To stejné se dát říct i o mých testovaných subjektech, které také neměly mnoho volného času.

6.3.3 První kolo - Skupinové testování

První kolo testování proběhlo ještě před začátkem implementace výukové aplikace. Díky špatné organizaci bohužel neproběhlo skupinové testování (ve smyslu více uživatelů) a sezení se tak konalo pouze v přítomnosti mě a mého kolegy ze školy. Ovšem jak uvádí pan Krug, nějaké testování je stále lepší, než žádné.

Kolegovi jsem představil návrh aplikace a zeptal se ho, co si myslí, že má být její náplní (k čemu má sloužit). Jeho odpověď z většiny odpovídala mé představě, což mě potěšilo. Dále mě zajímal názor na rozložení stránky. Po následné návštěvě "konkurenčních"webů jsme dospěli k menším úpravám v oblasti menu.

Výsledkem testování byla necelá stránka náčrtků a poznámek, které mi následně ulehčily implementaci částí aplikace (nemusel jsem nad nimi dlouho přemýšlet).

6.3.4 Druhé kolo - Testování použitelnosti

Díky nečekané pandemii Koronaviru jsem musel přehodnotit způsob uživatelského testování použitelnosti. Testy, které mají probíhat v přítomnosti uživatele a vývojáře tak bylo nutné provézt vzdáleně. Z toho důvodu jsem připravil seznam otázek a úkolů, které jsem testovaným uživatelům předal, a dále jsem jim nechal týden na volné zkoumání aplikace. Během tohoto týdne bylo jejich úkolem přečíst alespoň část kapitol výukové aplikace, zkusit naprogramovat přiložené demo příklady a následně splnit úkoly zkoumající porozumění problematiky.

Dle vzorového testu z knihy `Nenuťte uživatele přemýšlet` [14, str. 124] jsem připravil seznam otázek, které se postupně ptají na předešlé zkušenosti testovaného uživatele a pomalu přechází k porozumění jednotlivých části webové aplikace. Jak již bylo řečeno, mezi testovanými se nacházely osoby různých povolání s různými přístupy k programování. Z jejich odpovědí jsem nakonec složil univerzální odpověď, která popisuje jejich všeobecné mínění k dané otázce.

Otázky k uživatelskému testování použitelnosti dle testovacího scénáře Steva Kruga

Následuje seznam otázek, na základě kterého jsem vyzpovídal účastníky testování.

- **Kdybyste se potřeboval naučit nový programovací jazyk či framework, jaký způsob by Vám nejvíce vyhovoval?**

  Testeři obecně zmiňovali e-learning kurzy nebo portál YouTube a to především díky tomu, že obsah raději poslouchají, nežli čtou. Pokud by se setkali se zajímavým výukovým webem s textový obsahem, tak by ho ve většině případů alespoň vyzkoušeli.

  V některých případech by se při učení nové problematiky řídili oficiální dokumentací konkretní technologie či jazyka. Knihu by si spíš nekoupili.

- **Hledáte na internetu řešení problémů? Jaký styl jejich řešení a vysvětlení preferujete?**

  Všichni dotázaní uvedli, že při řešení problémů při programování často používají internet. Někteří opět zmínili formu videí, ale jako nejdůležitější považují stručnost a konkrétnost. V případě textových tutoriálů očekávají dostatek obrázků a ukázek kódu s možností si ho vlastnoručně vyzkoušet.

- **Máte nějaké oblíbené webové stránky, které využíváte při řešení problémů v programování? Proč?**

  Někteří testovaní uživatelé se spokojí s portálem `StackOverflow`[1] (především díky stručnosti) a stačí jim, že dané řešení funguje. Zbytek ocení detailnější rozbor problematiky na specializovaných stránkách typu `W3Schools` [2] či `Instructables` [3].

- **Co si myslíte, že je hlavní náplní této stránky? K čemu slouží?**

  Respondenti se více méně shodli na tom, že aplikace slouží jako tutoriál k WebGL (soudili tak podle názvu v hlavičce stránky).

- **Co si myslíte, že reprezentují jednotlivé části webu?** Hlavní rozložení stránky pochopila většina testovaných uživatelů, ovšem menší rozpory budilo horizontální menu u kapitol. Někteří uživatelé si zpočátku

[1] `https://stackoverflow.com/`
[2] `https://www.w3schools.com/`
[3] `https://www.instructables.com/`

mysleli, že menu slouží jako globální přepínač a špatně si vyložili jeho podstatu.

Testovaní uživatelé si zprvu nebyli jistí, která stránka je úvodní a jak si takovou stránku zobrazit.

Pár uživatelů již v úvodu ocenilo sekci s komentáři, která by dle jejich názorů mohla částečně nahradit funkci zmíněného webu StackOverflow.

K sekci "demo příklad"vzniklo spoustu nejasností, ale také nápadů, které by zde uživatelé uvítali (viz závěr této kapitoly 6.5).

- **Co byste udělal jako první po příchodu na tuto stránku?**

  Většina testovaných uživatelů by hledala úvod, který by popisoval záměr tutoriálu. Zajímali by se především o shrnutí toho, co se v tutoriálu naučí, ale také o ukázky, které demonstrují, co se dá po absolvování tutoriálu vytvořit. Některým by stačila galerie s obrázky, jiní by naopak ocenili interaktivní ukázky.

  Následně by prozkoumali hlavní strukturu kapitol a zběžně se podívali na jejich obsah. Dle nabytého dojmu by hledali úvodní kapitolu tutoriálu.

- **Proč si myslíte, že jsou některé prvky menu rozjíždějící?**

  Testeři se shodli na tom, že se jedná o hierarchii podkapitol reprezentující nějaký tematický okruh problematiky.

- **Jak byste si otevřel další kapitolu? Přijde Vám způsob přehledný?**

  Otevření nové kapitoly nedělalo testovaným uživatelům problém. Po prvotních nesnázích s určením stránky (úvodní), která se jim zobrazí po úvodním navštívení webu, jim přišla navigace intuitivní.

  Většina testovaných uživatelů kritizovala absenci tlačítek pro navigaci mezi kapitolami. Tím jsou myšlena tlačítka *Předchozí kapitola* a *Následující kapitola*, která by se měla nacházet na konci každé kapitoly.

- **Co si představíte pod pojmem "Demo"či "Demo příklad"?**

  Testeři si pod pojmem představují ukázku k probírané problematice. Ukázka by měla být pokud možno interaktivní a mělo by být možné si zobrazit její zdrojový kód.

- **Jak byste si zobrazil zdrojový kód Demo příkladu?**

  Jako první způsob by testovaní uživatelé vyzkoušeli možnost *Zobrazit zdrojový kód stránky* po kliknutí pravým tlačítkem myši do stránky (nebo klávesovou zkratkou CTRL + U). Někteří by použili vývojářskou konzoli dostupnou pomocí klávesy F12.

  Po delším zkoumání stránky většina testerů přišla na možnost rozkliknutí demo příkladu na nové záložce a zobrazení zdrojového kódu pouze samotného příkladu.

Díky tomu část testovaných uživatelů navrhla přidat kapitolu popisující základní používání výukové aplikace.

Dále byla zmíněna těžkopádnost zobrazení zdrojového kódu pomocí nástrojů prohlížeče. Někteří uživatelé by ocenili zdrojový kód přímo ve stránce, do kterého by se mohly odkazovat jednotlivé části kapitol.

- **Dokázal byste si zdrojový kód zobrazit i bez rad v textu? I za cenu hledání řešení na internetu?**

  Odpovědi se lišily a závisely především na zkušenostech testovaného uživatele s vývojem webových aplikací.

- **Jak byste si uložil knihovnu CTUGL a jak byste ji vložil do projektu?**

  Všichni testeři se dostali ke zdrojovému kódu knihovny a jejich přístup se lišil pouze ve způsobu stažení souboru. Část testerů zdrojový kód zkopírovala do schránky a vložila do nového javascriptového souboru v projektu. Zbytek testerů soubor stáhl kliknutím myší do obrazovky a výběrem *Uložit jako* si ho uložil do adresáře s projektem.

6.4 Závěrečné testování vědomostí

Závěrem týdenního testování aplikace byl také vědomostní test, který měl ověřit, kolik se toho testovaní uživatelé naučili. Otázky testu byly zaměřeny především na úvodní okruh kapitol, především kvůli časové náročnosti procházení tutoriálu jako celku (ne všichni testovaní uživatelé prošli látku až do konce).

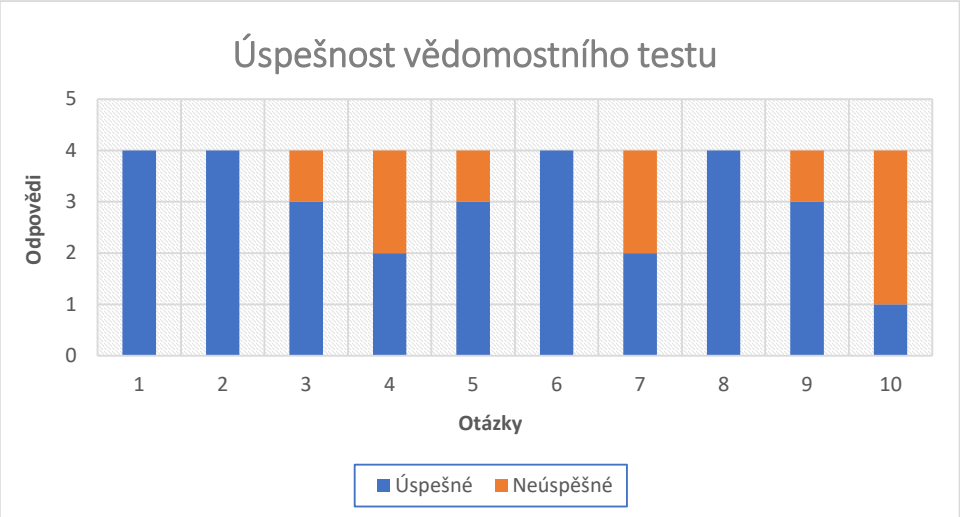
- **Z jakého geometrického prvku (primitiva) se skládají modely v počítačové grafice?**
- **K čemu slouží EBO (Element Buffer Object) ?**
- **Kde a jak se počítá barva pixelu?**
- **Na čem je závislá míra odraženého světla od povrchu objektu?**
- **Jakým způsobem byste předali pozici kamery do shaderu?**
- **Zkuste vymyslet definici atributu (pomocí gl.vertexAttribPointer()), který obsahuje tři složky, jejich datový typ je Float, před tímto atributem se nachází atribut se čtyřmi složkami a za tímto atributem se nachází atribut se dvěma složkami. (zajímá mě především hodnota „offset“ a „stride“)**
- **Jaké složky obsahuje Phongův osvětlovací model a jaký je jejich význam?**
- **Pomocí kterého příkazu řekneme prohlížeči, že chceme vykreslit další snímek (animace / hry) ?**

- **Jakými způsoby lze do shaderu předat data (jaké typy vstupních proměnných existují)?**

- **Bonus: Jakým způsobem byste v GLSL uložili do proměnné datového typu „vec3“ součet proměnné typu „vec2“ a proměnné typu „vec4“? Ukažte na konkrétních datech: vec2(0.4, 1.9) a vec4(1.1, 2.5, 0.8, 1.0).**

6.4.1 Úspěšnost

Úspěšnost vědomostního testu byla poměrně vysoká i přes to, že někteří testovaní uživatelé nedočetli tutoriál až do konce (viz Graf 6.1). To vypovídá o tom, že dohledání potřebných odpovědí v textu není složité. Z toho vyplývá, že výuková aplikace může sloužit i příležitostným návštěvníkům, kteří hledají řešení konkrétního problému.



Obrázek 6.1: Graf úspěšnosti testovaných uživatelů

Kapitola:	#2	#3	#4	#5	#6	Hodin celkem	Úkolů splněno
Tester1	15min	45min	1h 15min	35min	40min	**3h 30min**	**2/4**
Tester2	20min	1h	1h 20min	30min	45min	**3h 50min**	**3/4**
Tester3	15min	40min	1h	25min	35min	**3h 5min**	**3/4**
Tester4	15min	25min	50min	30min	30min	**2h 30min**	**4/4**
Průměr:	**16,25**	**42,5**	**66,25**	**30**	**37,5**	**3h 12 min**	**3/4**

Tabulka 6.2: Tabulka časové náročnosti kapitol

6.5 Závěr testování

Výsledkem uživatelského testování je spousta užitečných dat a postřehů. V případě první kategorie kapitol jsem testované uživatele požádal o měření času, který stráví nad čtením dané kapitoly a řešením jejího obsahu. Díky tomu jsem získal přibližný čas, který testovaní uživatelé strávili nad jednotlivými kapitolami i nad jejich celou sérií (Tabulka 6.2). Z tabulky je také patrné, že zkušený uživatel (`Tester4`) si s obsahem poradil mnohem rychleji, než ostatní.

Z naměřených dat vyplývá, že pro nastudování a vyzkoušení základní problematiky WebGL stačilo testovaným uživatelům průměrně něco málo přes tři hodiny. Pro porovnání s předmětem PGR probraný obsah ve výukové aplikaci odpovídá přibližně dvěma a půl přednáškám a dalším dvěma či třem cvičením. To, co dokáže uživatel výukové aplikace naprogramovat za necelé 4 hodiny (včetně základní teorie), dokáže student předmětu PGR až na konci třetího cvičení (+- 4,5 hodin pouze cvičení).

Tento styl porovnávání samozřejmě není úplně ideální. V předmětu PGR je probráno mnohem více teorie, která je na cvičeních opakována. Ovšem pokud sečteme třeba jen polovinu času dvou přednášek a dvou cvičení a připočteme k tomu čas domácího samostudia (například jednu hodinu) dostaneme časovou náročnost zhruba pět hodin.

Z naměřených dat v tabulce také vyplývá, že i úplný nováček na poli WebGL (i JavaScriptu) je schopný za relativně přijatelnou dobu naprogramovat (nebo alespoň složit z fragmentů kódu a zprovoznit) základní WebGL aplikaci včetně vykreslovací smyčky.

Oproti ostatním (i profesionálním) tutoriálům zmíněným v kapitole 2 je hlavní výhodou český jazyk. Dnešním standardem je studium programování v angličtině, ale mnoho uživatelů ocení vysvětlení složitého problému v jejich rodném jazyce.

Další výhodou oproti většině tutoriálů (až na WebGL tutoriály typu `WebGL2Fundamentals.org` je úzká provázanost textu s fungujícím výsledným demo příkladem, se kterým je možné interagovat přímo v prohlížeči. Pro potřeby výuky se tak jedná o nejideálnější stav.

Oproti tutoriálu WebGL2Fundamentals je čtenář veden k jiné struktuře aplikace. Složitější problematika je rozdělena do více souborů (tříd) a v některých případech je propojena i s jinými funkcionalitami (probranými dříve). Pro některé tak může být kód přehlednější a určitě se jedná o výhodu do budoucna.

6.5.1 Dodatečné poznámky testerů

Dle názorů testovaných uživatelů je aplikace vhodným nástrojem k výuce základů WebGL. Oceňují především stručnost a názornost kapitol a příkladů. Struktura kapitol se zamlouvá jak nováčkům tak i zkušenému uživateli OpenGL (absolvent předmětu PGR). Text popisují jako přehledný a zábavný.

Student, který sám absolvoval předmět PGR, uvedl, že aplikace může velmi dobře doplňovat výuku předmětu. Také poznamenal, že zpracování aplikace může více motivovat uživatele k implementaci svých grafických aplikací (na rozdíl od složitější kombinace jazyka C++ a OpenGL). Připravení a spuštění projektu je mnohem jednodušší a tedy méně odrazující. Jako jedinou nevýhodu vidí potřebu pochopit základy webových technologií (což je ale oproti jazyku C++ jednodušší).

Bohužel v případě dvou testovaných uživatelů docházelo v druhé polovině tutoriálu (po sérii `Úvod do WebGL`) k nejasnostem, které následně způsobovaly problémy s implementací probírané látky. Bylo zjištěno, že na méně znalé programátory v oblasti webových technologií je tempo výkladu příliš rychlé. V textu kapitol se nacházelo více fragmentů kódu, o kterých nevěděli, kam je vložit. Tento problém byl způsoben především jiným strukturováním kódu JavaScriptu, než je zvyklé v běžných programovacích jazycích. Jeden testovaný uživatel přiznal, že se kód snažil psát stejným způsobem, jako tomu je například u jazyka C - což je špatně především v kontextu vkládání jiných skriptů (pořadí apod.).

Dočasné řešení je přimět tyto uživatele prohlížet si celé zdrojové kódy pomocí nástrojů webového prohlížeče a snažit se pochopit problematiku z výsledného kódu.

Řešení do budoucna je nutné ovšem důmyslně promyslet. Zadání bakalářské práce původně nepočítalo s uživateli, kteří nejsou studenty počítačové grafiky. Případným rozšířením obsahu kapitol o ukázku celého zdrojového kódu (což bylo navrženo jedním z testovaných) nebo o interaktivní rozhraní pro editování zdrojového kódu (`JSFiddle`[4] či `Codepen.io`[5]) by došlo k znatelnému zlepšení chápání problematiky a způsobu implementace.

Připomínky, nápady

Většina testovaných uživatelů zmínila těžkopádnost ovládání aplikace jako takové. Mezi to se řadí absence tlačítek *Předchozí kapitola* a *Následující kapitola*, nutnost scrollovat na začátek stránky a hledat v menu následující kapitolu či horší provázanost výukových kapitol s kompletním zdrojovým kódem příkladů.

Dále byla zmíněna absence titulní stránky (respektive prokliku skrze logo stránky) a občasné chyby v textu či komentářích ve zdrojových kódech.

Mezi nápady ze strany uživatelů byla navržena implementace elementu *Video*, pomocí kterého by bylo možné do stránky vkládat video ukázky (prozatím jsou k dispozici jednoduché odkazy). V případě absence podpory WebGL ve webovém prohlížeči bylo navrženo nahradit demo příklad jednoduchým obrázkem, který zachycuje hlavní podstatu příkladu.

[4] `https://jsfiddle.net/`
[5] `https://codepen.io/`

Testovaní uživatelé také uvažovali nad využitím záložky "Výsledné demo". Někteří navrhovali využití záložky k zobrazení celého zdrojového kódu, ostatní by naopak uvítali ukázky začlenění problematiky do větší aplikace.

Navržena byla také možnost si stáhnout zdrojový kód jako ZIP soubor. V případě pokročilejších příkladů jsou projekty složeny z několika souborů a jejich manuální dohledání je náročnější. Většinu takových souborů si uživatel musí prozatím zobrazit pomocí odkazů ve zdrojovém kódu hlavního HTML souboru.

6.5.2 Odhalené technické problémy

Během testování aplikace došlo k identifikaci dalších problémů, které znemožňují (nebo omezují) používaní výukové aplikace. V některých prohlížečích není možné spustit aplikaci vůbec a to kvůli zhoršené podpoře JavaScript modulů. Díky tomu nedojde ani k načtení hlavní stránky. Chyba byla objevena v prohlížečích `Internet Explorer`, `Microsoft Edge`, `iOS Safari` a `Google Chrome pro iOS`.

Z detailnějšího studia problému vyplynulo, že je způsoben samotným příkazem `export default`, jehož implementace ve zmíněných prohlížečích funguje jiným způsobem. V konzoli prohlížeče se neobjeví žádná chyba, pouze není obsah vykreslen. Přitom dle stránky `MDN`[6] jsou javascriptové moduly samy o sobě podporovány.

V některých prohlížečích také chybí podpora WebGL2, ale to jsou ve většině případů ty stejné prohlížeče, které byly již uvedeny.

Do budoucna je proto potřeba aplikaci přepsat tak, aby nebylo nutné používat JavaScript moduly. V případě WebGL2 bohužel dobré řešení neexistuje.

[6] `https://caniuse.com/#feat=es6-module`

Kapitola 7

Diskuze

Kapitola se zabývá hlavními problémy a nedostatky aplikace, a návrhy jejich možného řešení.

7.1 Indexování obsahu Google boty

Vzhledem k tomu, že jsem si při návrhu výukové aplikace neuvědomil potřebu zpřístupnit obsah stránky Google botům, nedostal jsem se ani k implementaci, která by zmíněnou problematiku řešila. Úlohou botů je zmapování obsahu stránky, aby ji bylo možné později zahrnout do relevantních výsledků vyhledávání. Díky tomu je následně stránka dohledatelná při zadání správných klíčových slov do Google vyhledávače.

Bohužel Google boti umí pracovat jen s čistou URL adresou, tedy hodnotami oddělenými lomítky. Výuková aplikace však pracuje s tzv. *Hash hodnotami*, které se nachází za mřížkou v URL adrese [35] [36].

Tento nedostatek naštěstí není v aktuální fázi pro aplikaci zásadní, protože aplikace bude ze začátku sloužit především pro potřeby fakulty. V případě následujících úprav, které by měly za úkol zpřístupnit obsah i širší veřejnosti, bude nutné upravit trasování požadavků.

Vyřešení problému s indexováním si představuji jako součást budoucích úprav aplikace, které souvisí s odhalenými nedostatky a problémy.

7.2 Podpora ostatních prohlížečů

Při testování aplikace došlo k zjištění, že některé prohlížeče mají problém se zobrazením výukové aplikace. Tímto problémem se zabývá kapitola 6.5.2. Detailnějším zkoumáním problému jsem dospěl k závěru, že bude nutné přepsat způsob modularizace aplikace. Novější prohlížeče typu Microsoft Edge sice modularizaci podporují [37], ale z nepochopitelných důvodů nezvládají interpretovat některé způsoby exportování a importování modulů. Jako výsledek se zobrazí skoro prázdná stránka (vykreslí se jen HTML obsah, který není ovlivněn JavaScriptem). Ovšem ve vývojářské konzoli se žádné chyby nezobrazují.

Před úpravou aplikace bude nutné provézt rešerši, na základě které bude odvozen způsob následných oprav. Pro aktuální fázi nejsou zmíněné úpravy zásadní, protože velká část uživatelů používá prohlížeče Mozilla Firefox a Google Chrome [37]. V případě mobilních zařízení je podporovaná verze Google Chrome for Android, ve které se obsah zobrazuje relativně správně.

7.3 Rozhraní pro přidávání nových kapitol

Vzhledem k aktuální náročnosti přidávání nových kapitol je vhodné vymyslet lepší způsob, jak práci s rozšiřováním obsahu usnadnit. Jako nejlepší varianta připadá textový editor na bázi `Wysiwyg` *What You See Is What You Get*[1]. Správcem obsahu jsem prozatím jen já. V případě, kdy by se měl o obsah starat někdo jiný, bylo by nutné se na rozhraní domluvit.

7.4 Úpravy aplikace

Díky nalezeným nedostatkům a návrhům uživatelů vzniklo pár nápadů na úpravy a rozšíření aplikace. Realizace rozšíření a úprav by probíhala zároveň s úpravami souvisejícími z dříve uvedenými problémy.

Úpravy / změny

- Oprava URL odkazů (routing).
- Přepsaní aplikace s pomocí balíčkového systému NPM a využitím technologií Node.js a JSX.
- Zpřístupnění aplikace většině prohlížečů (s tím souvisí předešlý bod).
- Rozšíření kapitol o nové prvky (video, zástupný obrázek v případně absence WebGL2).
- Vytvoření vlastních doprovodných obrázků či animací.
- Využití záložky "Výsledné demo"pro prezentaci způsobu začlenění probrané látky do komplexního projektu. V textu kapitol (záložka "Tutoriál") se budou nacházet pouze jednoduché ukázky zaměřené na konkrétní problematiku. Vedle toho se bude postupně vyvíjet komplexní aplikace, která bude v každé kapitole rozšířena o probranou problematiku. Výsledek bude prezentován ve zmíněné záložce "Výsledné demo", která bude pro svůj účel přejmenována.

[1] `https://www.adaptic.cz/znalosti/slovnicek/wysiwyg/`

Rozšíření

- Implementace uživatelských účtů a komentářů.
- Implementace editoru pro psaní a přidávání nových kapitol.
- Rozšíření aplikace o ukázku kompletního zdrojového kódu konkrétního demo příkladu. To bude řešeno buď novou stránkou, nebo pomocí interaktivního rozhraní `JSFiddle`[2] či `Codepen`[3]. S tím souvisí také zpřístupnění výukového obsahu i zájemcům, pro které může být aktuální styl výkladu příliš rychlý. Veškeré kroky budou lépe popsány a uživatel si je bude moci rovnou vyzkoušet.

7.5 Náročnost výuky

Na základě předem určeného využití aplikace byla náročnost (respektive tempo) výuky odvozeno od předmětu PGR. Vzhledem k tomu, že má aplikace studenty podporovat především v praktické části výuky, došlo k omezení teorie na potřebné základy. Naopak rozbor implementace byl rozdělen na jednotlivé kroky, kterým je věnován dostatek prostoru na vysvětlení.

Vzhledem k mé idee zpřístupnit problematiku i ostatním uživatelům jsem se snažil text psát stylem, kterému by mohl porozumět i laik v oboru počítačové grafiky a webových technologií. Dle výsledků testování a zpětné reakce testovaných uživatelů tomu tak doopravdy bylo, bohužel jen v případě první série kapitol. S příchodem obsáhlejších kapitol popisujících 3D prostor a složitější výpočty se začali někteří testovaní uživatelé v problematice a kódu ztrácet. Tento problém popisuje další podkapitola 7.6.

Pro zjednodušení výuky a práce s WebGL vznikla podpůrná knihovna `CTUGL` (kapitola 5.3). Studentům předmětu PGR je dostupný základní modul, který vychází ze školní knihovny `PGR-framework`. Ten slouží k redukci nadbytečného kódu a usnadňuje tak práci a orientaci ve zdrojovém kódu daného výukového příkladu (demo příkladu). S tímto modulem pracuje většina výukových příkladů vzniklé aplikace.

Pro zájemce vznikl také balíček rozšiřujících tříd knihovny CTUGL, pomocí kterých je celkem jednoduché sestavit základ složitější 3D WebGL aplikace. V balíčku je implementované řešení kamery, shader programu, reprezentace modelu, reprezentace sítě vrcholů (mesh), cube-mapy (skybox, environment mapping) a kurzoru. Pomocí balíčku je možné nahrávat do scény WebGL 3D modely ve formátu JSON dle standardu .OBJ včetně jejich textur a materiálů.

[2] `https://codepen.io/`
[3] `https://jsfiddle.net/`

7.6 Provázání textu a ukázek kódu

Při psaní výukových kapitol jsem se setkal s problémem, kdy od určité kapitoly bylo obtížné určit nejlepší způsob výkladu látky. S postupem v tutoriálu je při programování příkladů potřeba čím dál více funkcionality, což vede k razantnímu zvýšení počtu řádků výsledného kódu. I příklad, který se snaží zabývat jen danou problematikou, musí pracovat s kódem z předešlých kapitol (včetně úprav).

Pokud konkrétní kapitola počítá s přidáním například dvou řádků do již existující funkce, která má třeba třicet řádků, přišlo mi zbytečné zatěžovat text kapitoly ukázkou dlouhou třicet řádků. Do textu jsem zahrnul pouze nové řádky s vysvětlením, kam mají být řádky v kódu umístěny. Bohužel toto řešení vedlo v některých případech ke zmatení testovaného uživatele.

Jako jedno z možných řešení mě napadá rozdělit obrazovku vertikálně na dvě půlky, kdy v jedné by se nacházel výukový text s odkazy do zdrojového kódu a v druhé by se nacházel celý zdrojový kód. Tyto problémy nejvíc postihovaly uživatele, kteří jsou v oboru programování grafiky a webových technologií nováčci. Je možné, že detailnější uvedení pravidel HTML a JavaScriptu by zmíněné problémy mohlo zredukovat. Ovšem to je nyní těžké určit a výsledky by přineslo až měření a testování. V případě úprav GUI (uživatelského rozhraní) by bylo vhodné provézt výzkum mezi potenciálními uživateli a na základě toho nové GUI navrhnout.

Kapitola 8

Závěr

Hlavním cílem bakalářské práce bylo navrhnout sadu demo příkladů ve WebGL, které mají pomáhat studentům s pochopením problematiky programování počítačové grafiky. Obsah příkladů se měl odvíjet od látky probírané v předmětu PGR, navíc měl být ale uzpůsoben dle rešerše stylů výuky na ostatních univerzitách či výukových webech.

Vznikla výuková aplikace čítající 19 kapitol popisujících problematiku programování grafiky od úplných základů. Celkem bylo vytvořeno 20 demonstračních příkladů, které se nacházejí v patnácti implementačních kapitolách. Kapitoly se skládají z výukového textu doplněného obrázky, ukázkami kódu a výslednými demo příklady. Čtenář se může řídit ukázkami zdrojového kódu přímo v textu kapitol, ale také si může jakýkoliv příklad otevřít v nové záložce a zobrazit si jeho kompletní zdrojový kód (prozatím pomocí nástrojů webového prohlížeče).

Vedle výukové aplikace vznikla také podpůrná knihovna CTUGL, která slouží k zjednodušení práce s WebGL. I v případě jednoduchých aplikací je nutné napsat spoustu inicializačního kódu, který lze pomocí této knihovny nahradit jediným řádkem, na kterém je volána příslušná funkce, jež daný kód obsahuje.

Na počátku implementace navíc vznikla grafická aplikace pro vizualizaci 3D modelů. Její předností je možnost vykreslení jakéhokoliv 3D modelu s příponou .OBJ a .MTL včetně jeho textur a materiálů. Ve scéně s modelem (nebo více modely) je možné se volně pohybovat, implementovány jsou pokročilé funkce jako mlha, skybox či environment mapping. Aplikace vznikla především z důvodu shromáždění poznatků o programování WebGL ještě před psaním výukových kapitol.

Vzhledem k časové náročnosti přípravy obsahu výukové aplikace vznikla série kapitol, které jsou určeny především pro studenty PGR. Z uživatelského testováni vyplynulo, že obsah a zpracování kapitol je pro běžné uživatele občas náročný na pochopení, a z toho důvodu je nutné aplikaci upravit. Vzhledem k tomu, že zadání bakalářské práce s ostatními uživateli nepočítalo, připadá předělání aplikace na vizi do budoucna.

8.1 Budoucí práce

Aplikace je ve fázi, kdy je použitelná a svému účelu může sloužit. Bohužel díky velikosti a náročnosti projektu nebylo možné vyřešit všechny problémy. Ty samy o sobě se zadáním bakalářské práce nijak nesouvisí, ale znepříjemňují celkový dojem z používání aplikace. Práce na těchto změnách představuje další stovky odpracovaných hodin, které by mohly reprezentovat nějakou budoucí semestrální či diplomovou práci.

Předmětem budoucí práce jsou především opravy odhalených chyb a nedostatků. Změny se budou týkat použitých technologií či obsahu kapitol. Úpravy by měly být zpočátku zaměřeny na zlepšení uživatelské přívětivosti aplikace. Mezi to se řadí například jednodušší přístupnost ke zdrojovým kódům nebo úpravy uživatelského rozhraní. Až poté by měly následovat technické úpravy zahrnující opravu URL adres, aby jednotlivé stránky byly dohledatelné pomocí Google botů, nebo implementace nových funkcí.

V případě možných rozšíření připadá v úvahu přidání uživatelských účtů, díky kterým bude možné přidávat komentáře k jednotlivým kapitolám. Část této funkcionality již je dokonce připravena.

Díky časové náročnosti projektu také nebyl u některých kapitol dopsán výukový text. Pro zájemce je prozatím dostupný zdrojový kód s doprovodnými komentáři. Pro zpřístupnění obsahu méně zkušeným čtenářům je ovšem potřeba výukový text doplnit.

Samozřejmostí je případné rozšiřování probíraných témat i nad rámec předmětu PGR.

Literatura

[1] LearnOpenGL: *Learn OpenGL, extensive tutorial resource for learning Modern OpenGL*. LearnOpenGL [online]. [cit. 25.4.2020], Dostupné na: `https://learnopengl.com/`

[2] DCGI: *Programování Grafiky*. [online]. [cit. 25.4.2020], Dostupné na: `https://dcgi.fel.cvut.cz/courses/pgr`

[3] WebGL: *2D and 3D graphics for the web*. Web APIs | MDN [online]. Copyright © 2005 [cit. 22.04.2020]. Dostupné na: `https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API`

[4] WebGL2: *What's New*. WebGL2Fundamentals [online]. [cit. 22.04.2020]. Dostupné na: `https://webgl2fundamentals.org/webgl/lessons/webgl2-whats-new.html`

[5] Can I Use: *WebGL2 - Next version of WebGL. Based on OpenGL ES 3.0*. Can I use... Support tables for HTML5, CSS3, etc [online]. [cit. 22.04.2020]. Dostupné na: `https://caniuse.com/#feat=webgl2`

[6] OpenGL ES: *Overview*. The Khronos Group Inc [online]. Copyright © The Khronos [cit. 23.04.2020]. Dostupné na: `https://www.khronos.org/opengles/`

[7] *ECMAScript 6*. W3Schools Online Web Tutorials [online]. [cit. 23.04.2020]. Dostupné na: `https://www.w3schools.com/js/js_es6.asp`

[8] The freeglut Project: *About*. Sourceforge [online]. [cit. 23.04.2020]. Dostupné na: `http://freeglut.sourceforge.net/`

[9] opengl-tutorial: *Home*. opengl-tutorial [online]. [cit. 23.04.2020]. Dostupné na: `http://www.opengl-tutorial.org/`

[10] ŽÁRA, Ondřej. *JavaScript: Programátorské techniky a webové technologie*. Brno: Computer Press, 2015. ISBN 978-80-251-4573-9.

[11] Bootstrap: *Bootstrap 4 Get Started*. W3Schools Online Web Tutorials [online]. [cit. 26.04.2020]. Dostupné na: `https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp`

[12] Alexander Gessler: *JSON exporter for Open Asset Import Library to make 3D models accessible from JS/WebGl*. The world's leading software development platform · GitHub [online]. Copyright © 2020 GitHub, Inc. [cit. 28.04.2020]. Dostupné na: https://github.com/assimp/assimp2json

[13] Bc. Michal Vomastek: *Nástroj pro výuku základních křivek*. ČVUT DSpace [online]. Copyright ©K [cit. 29.04.2020]. Dostupné na: https://dspace.cvut.cz/bitstream/handle/10467/82558/F3-DP-2019-Vomastek-Michal-Nastroj_pro_vyuku_zakladnich_krivek.pdf?sequence=-1&isAllowed=y

[14] KRUG, Steve. *Nenuťte uživatele přemýšlet; 2. aktualizované vydání*. Brno: Computer Press, 2006. ISBN 80-251-1291-8.

[15] ReactJS: *Introducing JSX – React*. React – A JavaScript library for building user interface [online]. [cit. 29.04.2020]. Dostupné na: https://reactjs.org/docs/introducing-jsx.html

[16] *WebGL from the ground up*. WebGL Fundamentals [online]. [cit. 30.04.2020]. Dostupné na: https://webglfundamentals.org/

[17] highlight: *Syntax highlighting for the Web*. highlight.js [online]. [cit. 30.04.2020]. Dostupné na: https://highlightjs.org/

[18] Joe de Vries: *Learn OpenGL*. Learn OpenGL, extensive tutorial resource for learning Modern OpenGL [online]. Copyright © [cit. 03.05.2020]. Dostupné na: https://learnopengl.com/book/learnopengl_book.pdf

[19] Michal Turek: *NeHe OpenGL Tutoriály*. [online]. Copyright ©Nf [cit. 05.05.2020]. Dostupné na: http://nehe.ceske-hry.cz/subdom/nehe/download/download/cz_nehe_opengl.pdf

[20] ABCLinuxu: *Dotaz: Je OpenGL obtizne? (vyřešeno)*. AbcLinuxu.cz - Linux na stříbrném podnose [online]. Copyright © 1999 [cit. 05.05.2020]. Dostupné na: https://www.abclinuxu.cz/poradna/programovani/show/390614

[21] Pavel Tišnovský: *Seriál Grafická knihovna OpenGL*. Root.cz [online]. [cit. 05.05.2020]. Dostupné na: https://www.root.cz/serialy/graficka-knihovna-opengl/

[22] Opengl-Tutorial: *opengl_tutorial_2017_06_07*. opengl-tutorial [online]. Copyright ©Sg [cit. 05.05.2020]. Dostupné na: http://www.opengl-tutorial.org/assets/pdf/opengl_tutorial_2017_06_07.pdf

[23] WebGL Fundamentals API Docs: *Home - Documentation*. WebGL Fundamentals [online]. [cit. 05.05.2020]. Dostupné na: https://webglfundamentals.org/docs/index.html

[24] Khronos: *Related toolkits and APIs - OpenGL Wiki*. The Khronos Group Inc [online]. [cit. 05.05.2020]. Dostupné na: https://www.khronos.org/opengl/wiki/Related_toolkits_and_APIs

[25] GLFW: *Open Source, multi-platform library for OpenGL*. GLFW - An OpenGL library [online]. [cit. 05.05.2020]. Dostupné na: https://www.glfw.org/

[26] Khronos: *Language bindings - OpenGL Wiki*. The Khronos Group Inc [online]. Copyright © The Khronos [cit. 05.05.2020]. Dostupné na: https://www.khronos.org/opengl/wiki/Language_bindings

[27] W3Schools: *What is the HTML DOM?*. W3Schools Online Web Tutorials [online]. [cit. 05.05.2020]. Dostupné na: https://www.w3schools.com/whatis/whatis_htmldom.asp

[28] Matt DesLauriers: *GLSL Versions · mattdesl/lwjgl-basics Wiki · GitHub*. The world's leading software development platform · GitHub [online]. Copyright © 2020 GitHub, Inc. [cit. 06.05.2020]. Dostupné na: https://github.com/mattdesl/lwjgl-basics/wiki/GLSL-Versions

[29] MIT: *Computer Graphics*. Computer Graphics | Electrical Engineering and Computer Science | MIT OpenCourseWare [online]. Copyright © 2001 [cit. 06.05.2020]. Dostupné na: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/index.htm

[30] Gordon Wetzstein: *OpenGL Examples*. mit.edu [online]. [cit. 06.05.2020]. Dostupné na: http://web.media.mit.edu/~gordonw/OpenGL/

[31] cs184/284a: *Computer Graphics and Imaging*. University of California, Berkeley [online]. [cit. 06.05.2020]. https://cs184.eecs.berkeley.edu/sp19

[32] CS 148: *Introduction to Computer Graphics and Imaging*. Standford University [online]. [cit. 06.05.2020]. Dostupné na: https://web.stanford.edu/class/cs148/assignments.html

[33] DZone Web Dev: *What Is a Single-Page Application?* DZone [online]. [cit. 07.05.2020]. Dostupné na: https://dzone.com/articles/what-is-a-single-page-application

[34] MDN: *Ajax*. Developer guides | MDN [online]. Copyright © 2005 [cit. 14.05.2020]. Dostupné na: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

[35] John Mueler on Twitter: *"Correct – we generally don't support "#"URLs. Sometimes there are "weird"exceptions, where we find it's the only way to highlight that content. Definitely not something I'd rely on though. Good job finding an example like this :)!... https://t.co/QHPPqTg2L7"*. Twitter. It's what's happening. [online]. Copyright © 2020 Twitter [cit. 15.05.2020]. Dostupné na: `https://twitter.com/JohnMu/status/1118844020816584707?ref_src=twcamp%5Ecopy%7Ctwsrc%5Eandroid%7Ctwgr%5Ecopy%7Ctwcon%5E7090%7Ctwterm%5E3`

[36] Viacheslav Spiridonov: *How does Google index content for SPA sites? - Search Console Community*. Google Help [online]. Copyright ©2020 Google [cit. 15.05.2020]. Dostupné na: `https://support.google.com/webmasters/thread/14272290?hl=en`

[37] Can I Use: *JavaScript modules via script tag*. Can I use... Support tables for HTML5, CSS3, etc [online]. [cit. 15.05.2020]. Dostupné na: `https://caniuse.com/#feat=es6-module`

[38] arch_3d: *3D model Apartment Building_26*. TurboSquid 1380467 [online]. Copyright © TurboSquid 2020 [cit. 21.05.2020]. Dostupné na: `https://www.turbosquid.com/FullPreview/Index.cfm/ID/1380467`

[39] *Mineways*. Real-Time Rendering Resources [online]. [cit. 21.05.2020]. Dostupné na: `http://www.realtimerendering.com/erich/minecraft/public/mineways/`

Příloha A

Uživatelská příručka

A.1 Uživatel

Aplikaci je možné navštívit na adrese `http://psweb.mzf.cz/BP2/`. Pro správné zobrazení obsahu a funkčnost demo příkladů je vhodné použít webové prohlížeče `Google Chrome` (od verze 56), `Mozilla Firefox` (od verze 51) nebo `Chrome for Android`. Prohlížeče `Safari` a `iOS Safari` prozatím podporované nejsou. Použití ostatních prohlížečů může a nemusí fungovat - viz stránka *Can I Use*[1]. Detailnějším popisem funkčnosti v prohlížečích se zabývají kapitoly 6.5.2 a 7.2.

+ Aplikace pro vizualizaci modelů

Aplikace pro vizualizaci modelů je dostupná na adrese `http://psweb.mzf.cz/BP/`. Vzhledem k načítání a inicializaci geometrie modelů se samotná stránka načítá delší dobu. Nahrávání modelů přímo z GUI aplikace zatím není implementované a díky tomu je závislé na úpravách přímo v kódu aplikace.

A.1.1 Jak číst výukové kapitoly

Po otevření aplikace je zobrazena úvodní kapitola. Text je vhodné číst postupně dle řazení kapitol. Další kapitolu lze otevřít pomocí menu na levé straně obrazovky nebo pomocí tlačítek `Předchozí kapitola` a `Následující kapitola` .

Práce se zdrojovým kódem

K implementaci příkladů je vhodné použít některé z vývojářských prostředí (`IDE`). Doporučují například `Visual Studio Code`[2] nebo `JetBrains: WebStorm`[3]. Ovšem možné je použít jakýkoliv programovací editor.

Před inicializací projektu je vhodné si připravit složku, ve které se má nacházet obsah příkladu. V té stačí vytvořit HTML soubor (například `index.html`),

[1] `https://caniuse.com/#feat=webgl2`
[2] `https://code.visualstudio.com/`
[3] `https://www.jetbrains.com/webstorm/`

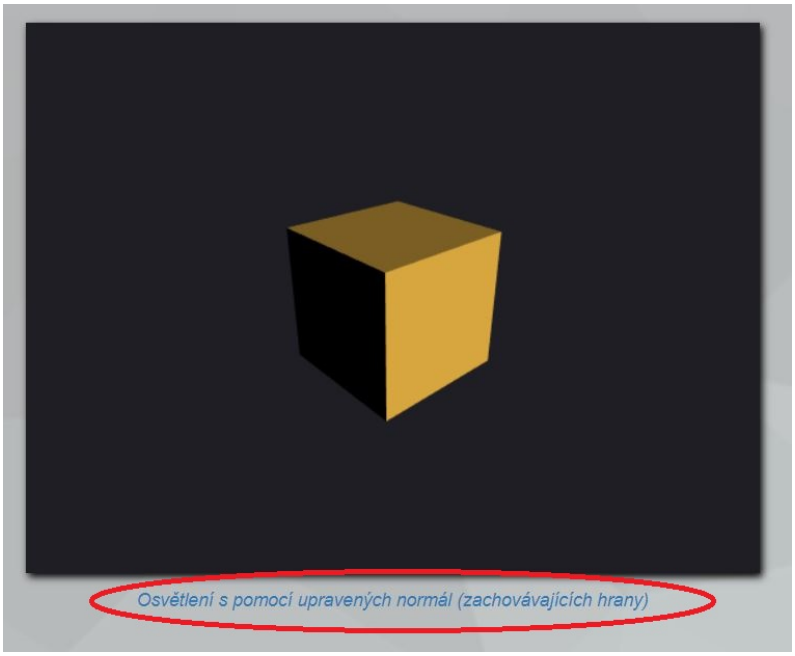
do kterého se bude kód příkladu zapisovat. Většina úvodních příkladů pracuje s jediným HTML souborem.

V úvodní implementační kapitole (*Hello World*[4]) je popsána HTML kostra, která je základem každého demo příkladu. Hlavní skript příkladu se zapisuje mezi značky `<script>` a `</script>` pod ukončovací značku `</body>`. Ostatní skripty (knihovna CTUGL, shadery apod.) je nutné vložit do hlavičky (mezi značky `<head>` a `</head>`), aby byly v době překládání hlavního skriptu dostupné. Vložení knihovny CTUGL je také popsáno v kapitole Hello World.

Výukové kapitoly staví na kódu předešlých kapitol, proto již probraný kód není znovu probírán. U každého fragmentu kódu je popsáno, kam má být vložen. Kdykoliv je možné si zobrazit celý zdrojový kód v případě nejasností - více v následující kapitole.

A.1.2 Přístup ke zdrojovému kódu demo příkladů

Fragmenty zdrojového kódu jsou dostupné v textu kapitol. Samotný demo příklad lze otevřít v nové záložce pomocí odkazu, který je součástí popisku demo příkladu (viz obrázek A.1). Zdrojový kód pak lze zobrazit pomocí klávesové zkratky CTRL + U. Pro uložení zdrojového kódu stačí kliknout pravým tlačítkem myši do zdrojového kódu a zvolit "Uložit jako...".



Obrázek A.1: Odkaz na zobrazení příkladu v nové záložce

[4] `http://psweb.mzf.cz/BP2/#/chapter/4`

A.1.3 Ovládání demo příkladů

Pro možnost interakce s demo příkladem je nejprve nutné kliknout myší do příkladu. Tím dojde k uzamčení kurzoru a registraci stisknutých kláves. Pro zrušení interakce stačí zmáčknout klávesu *Escape*. Počínaje kapitolou *Úvod do 3D/Kamera* je možné se ve scéně příkladu volně pohybovat.
Pomocí myši je možné se volně rozhlížet. Následuje výčet kláves, pomocí kterých lze příklady ovládat:

- **W** - Pohyb dopředu
- **S** - Pohyb dozadu
- **A** - Pohyb doleva
- **D** - Pohyb doprava
- **Space** (mezerník) - Pohyb nahoru
- **Levý Shift** - Pohyb dolu

V případě aplikace pro vizualizaci modelů jsou ještě definovány klávesy:

- **F** - Vypnutí / zapnutí mlhy
- **C** - Skrytí / zobrazení kurzoru
- **X** - Skrytí / zobrazení skyboxu

A.2 Správce

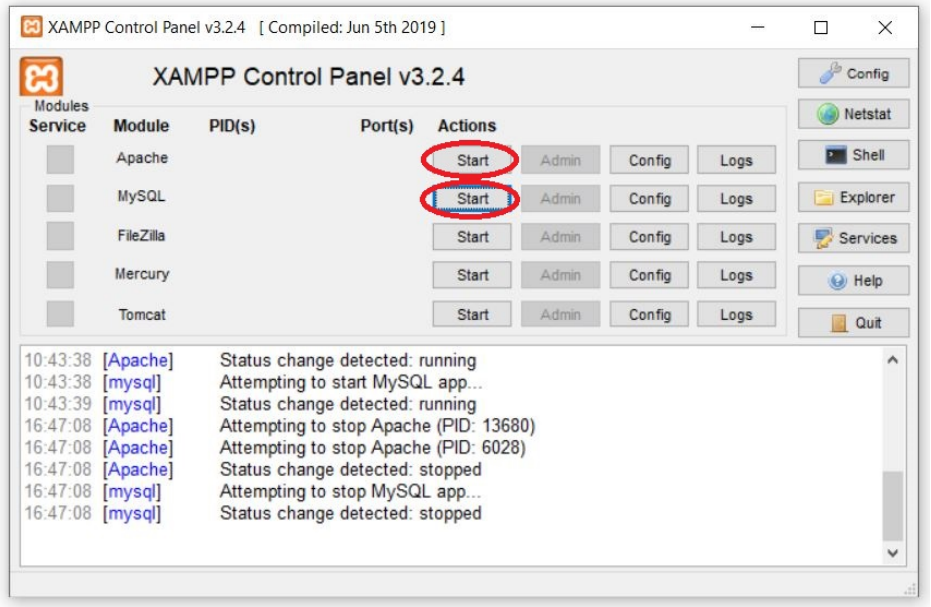
A.2.1 Základní nastavení aplikace

Zdrojové soubory výukové aplikace se nachází ve složce `BP/project/edu_app`. Pro spuštění aplikace je potřeba mít spuštěný webový a databázový server. Jednou z možností je instalace programu `XAMPP`[5], pomocí kterého je možné tyto servery spravovat. Po instalaci stačí složku `BP/project/edu_app` nakopírovat do kořenového XAMPP adresáře `./xampp/htdocs/`.

Pomocí XAMPP ovládací konzole (*XAMPP Control Panel*), která se nainstaluje spolu s XAMPP, je potřeba spustit webový a databázový server pomocí tlačítek "Start"dle obrázku A.2.

Dále je nutné dodat obsah kapitol. To lze provézt importováním databáze pomocí administračního rozhraní `phpMyAdmin`, které se otevře po kliknutí na tlačítko `Admin` v řádku s databázovým MySQL serverem (viz obrázek A.2) vedle tlačítka "Start".

[5] `https://www.apachefriends.org/index.html`



Obrázek A.2: Spuštění webového a databázového serveru pomocí XAMPP Control Panel

V administračním rozhraní phpMyAdmin je nutné vytvořit novou databázi kliknutím na tlačítko `Nová` v levém menu (viz obrázek A.3). Při jejím zakládání stačí zadat jako název nové databáze "bp"[6] (tak se databáze originálně jmenuje). Po jejím vytvoření stačí kliknout na položku *Import* (červeně zvýrazněné) dle obrázku A.3. Zobrazí se formulář pro import databáze.

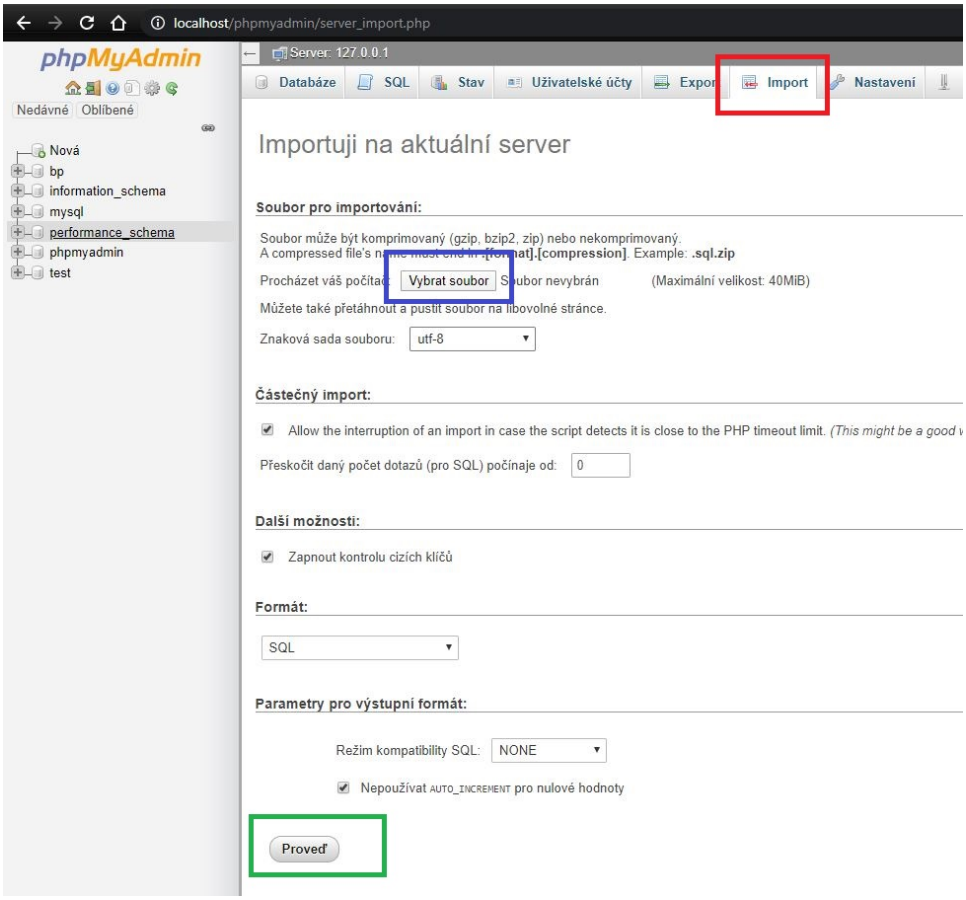
Pomocí tlačítka "Vybrat soubor"vybereme soubor exportované databáze `db.sql`, který se nachází ve složce s výukovou aplikací (`BP/project/edu_app/db_dump/db.sql`). Následně stačí jen potvrdit import tlačítkem "Proveď"(zeleně zvýrazněné).
V souboru `edu_app/Back-End/AjaxHandler.php` je popřípadě nutné nastavit přihlašovací údaje do databáze. Pro localhost bývá výchozí přihlašovací jméno "root"a heslo prázdný string (tedy bez hesla).

Stránku je následně možné navštívit na odkazu `http://localhost/edu_app`, kde *localhost* reprezentuje kořenový adresář `./xampp/htdocs/`.

A.2.2 Přidání nové kapitoly

Text kapitoly je aktuálně nejvhodnější psát v textovém editoru, jako je `Microsoft Word` a podobné. Obsah textu je následně nutné převézt do JSON pole dle ukázky 3 v kapitole 5.4.3.

[6] `https://help.one.com/hc/en-us/articles/115005588189-How-do-I-import%2Da%2Ddatabase%2Dto%2DphpMyAdmin%2D`



Obrázek A.3: Import databáze včetně dat

Pro referenci je dobré použít následující mapování:

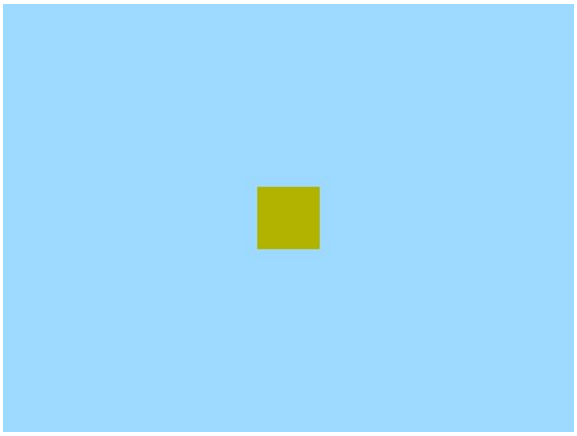
- **Název kapitoly v databázi** - Nadpis se stylem "Nadpis2"
- **caption** - Podnadpisy v kapitolách se stylem "Podnadpis"
- **paragraph** - Jeden odstavec textu
- **code** - Ukázka kódu
- **img** - Obrázek
- **example** - HTML soubor s demo příkladem

V databázi je nutné přidat záznam nové kapitoly. Nejprve je nutné zaregistrovat novou kapitolu záznamem do tabulky `chapters` (strukturování lze odpozorovat již od vložených kapitol). Následně v tabulce `chapters_translate` je potřeba vyplnit obsah kapitoly v daném jazyce. Tento obsah se mapuje s tabulkou `chapters` pomocí atributu `chapter_id`. Detailnější popis lze najít v kapitole 4.3.3.

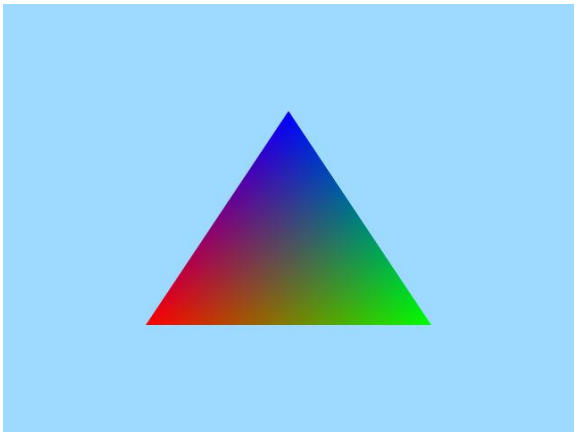
Většinu aktuálně sepsaného textu je možné najít v Microsoft Word souboru `BP/project/Documents/ChaptersContent.docx`.

Příloha B

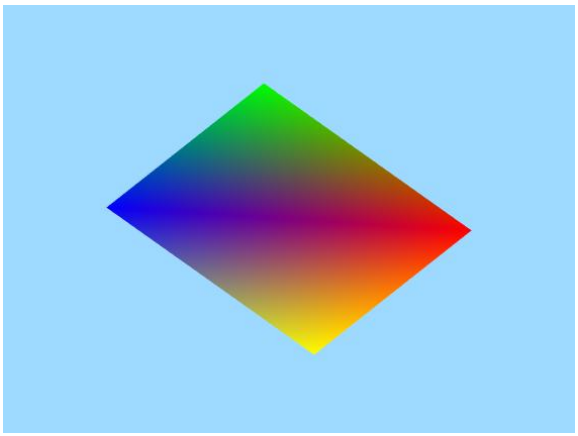
Ukázky demo příkladů



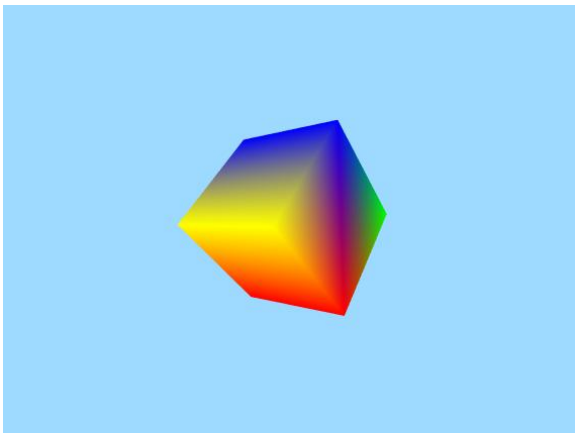
Úvodní Hello World příklad - vykreslení jednoho bodu.



Příklad kombinující VAO a VBO k vykreslení trojúhelníku.

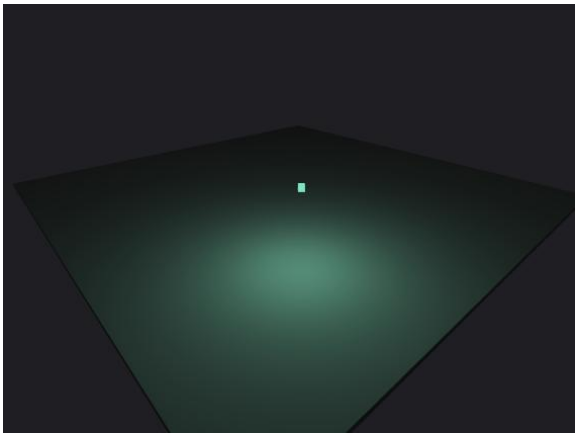


Kombinace příkladů o indexovaném vykreslování, vykreslovací smyčce a zpracovávání vstupů.

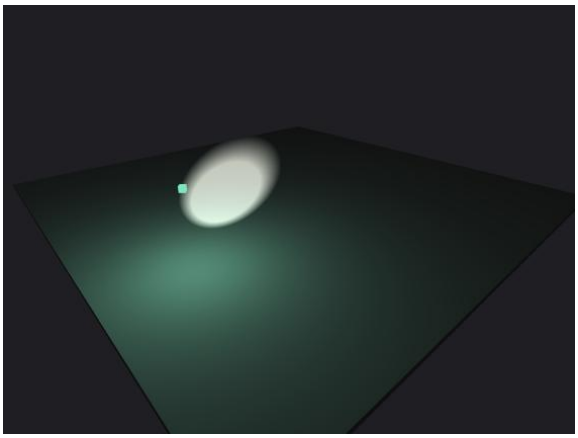


Kombinace příkladu o vykreslení objektu ve 3D prostoru s příkladem implementujícím volný pohyb kamery.

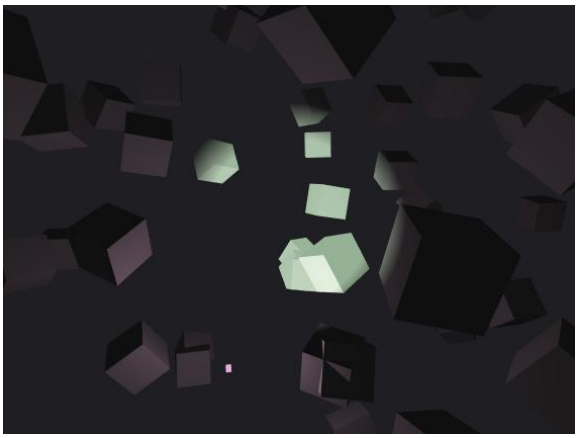
Příklad implementující směrové osvětlení (directional light).



Příklad implementující všesměrové bodové osvětlení (point light).

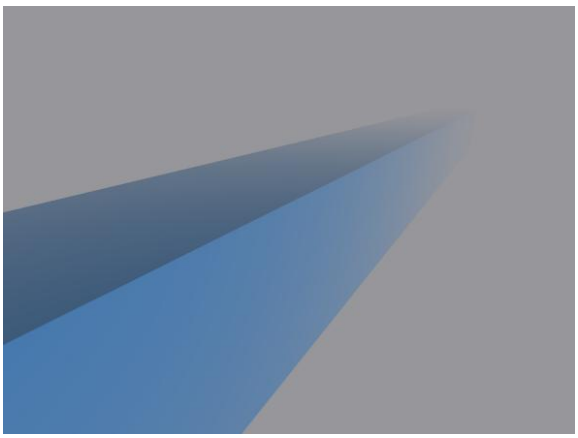


Příklad implementující osvětlení typu reflektor. Pozice reflektoru je závislá na svítící kostičce, která navíc reprezentuje point light.

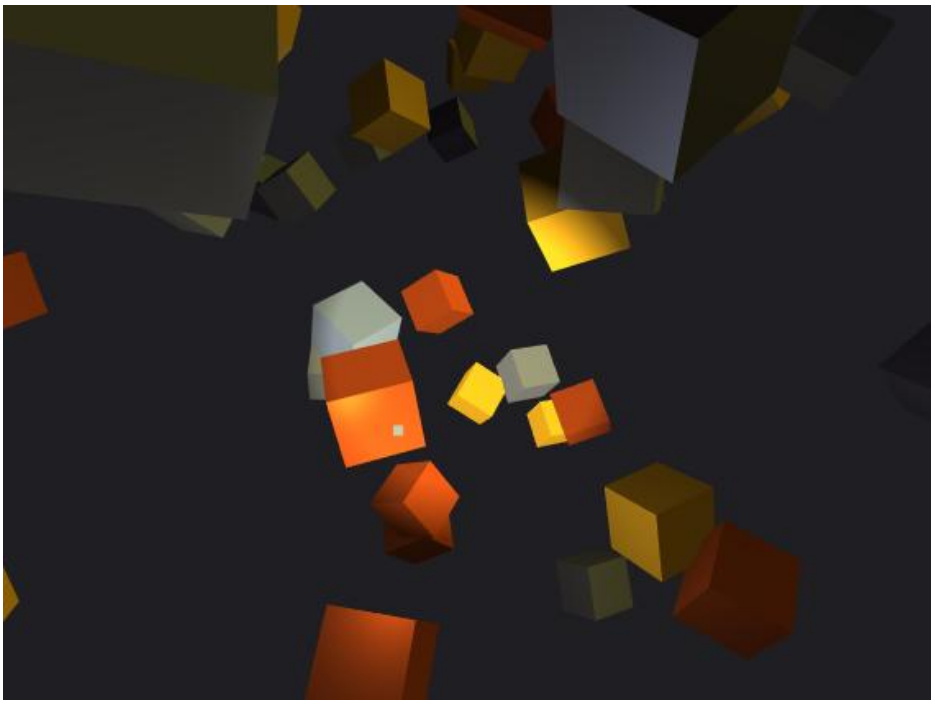


Příklad o optimalizaci kódu a vykreslení několika objektů v kombinaci se všemi typy osvětlení.

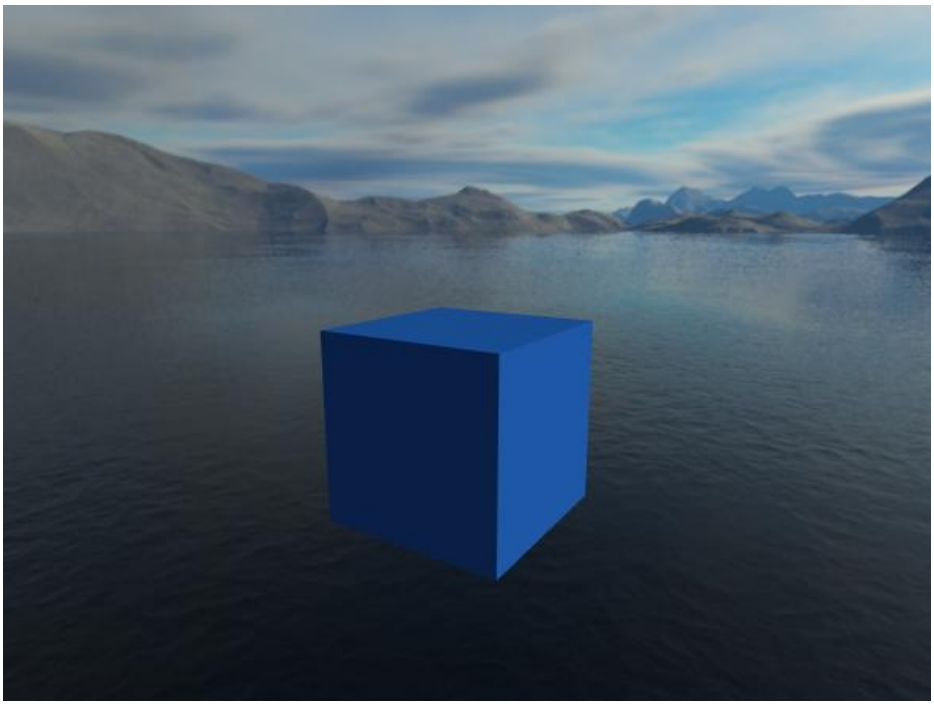
Příklad realizující základní aplikaci textury.



Příklad implementující exponenciální a lineární mlhu.

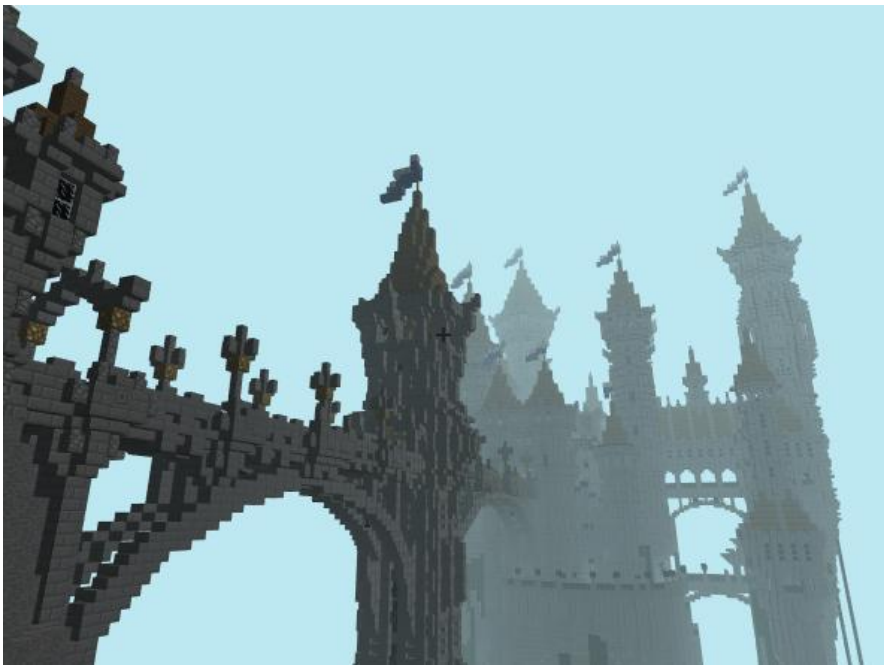


Příklad implementující materiály (ambientní, difúzní a spekulární složky barvy). Opět v kombinaci se všemi typy osvětlení.



Příklad implementující skybox pomocí cubemap textury. Promítání skyboxu je realizováno na obdélník definovaný v NDC souřadnicích.

`Aplikace pro vizualizaci modelů`: Ukázka vizualizace mé další Minecraft mapy[1]. *(Spoluautor mapy: Antonín Kabelka)*



`Aplikace pro vizualizaci modelů`: Vizualizace modelu mé Minecraft mapy[2] dohromady s efektem mlhy. *(Spoluautor mapy: Antonín Kabelka)*

[1] `https://www.planetminecraft.com/project/chinese-temple-lobby/`
[2] `https://www.planetminecraft.com/project/the-lonely-castle-3239371/`

Příloha C

Obsah CD

```
BP/
├── project/
│   ├── Documents/
│   ├── edu_app/
│   │   ├── Back-End/
│   │   │   ├── Controller/
│   │   │   ├── Model/
│   │   │   ├── AjaxHandler.php
│   │   │   └── Database.php
│   │   ├── Front-End/
│   │   │   ├── FrontController/
│   │   │   └── React-View/
│   │   ├── Chapters/
│   │   ├── img/
│   │   ├── index.html
│   │   └── db_dump/
│   │       └── bp.sql
│   ├── CTUGL/
│   │   ├── ctugl.js
│   │   └── utils/
│   └── vis_app/
│       ├── js/
│       │   ├── data/
│       │   └── render/
│       │       ├── include/
│       │       └── model_loader/
│       └── index.html
├── BP_text/
│   ├── BP.pdf
│   └── BP_latex_src/
└── Images/
```