Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Render manager s podporou vyvažování zátěže a webovým rozhraním

Bc. Dominik Jančík

Vedoucí práce: Ing. Ivan Šimeček, Ph.D.

26. června 2015

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 26. června 2015 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2015 Dominik Jančík. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Jančík, Dominik. *Render manager s podporou vyvažování zátěže a webovým rozhraním.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Práce si klade za cíl navrhnout online službu renderovací farmy za využití výkonu školních počítačů, které jsou většinu času nevyužity. Práce prozkoumává požadavky kladené na tuto službu, nabízenou sílu vybraných školních počítačů a způsoby jak nárazově využívané počítače zařadit do render farmy bez omezení jejich uživatelů. Výsledkem je prototypová implementace pod názvem FITRender, která umožňuje zpracování zadaných scén modulárním propojením několika softwarových celků. Do budoucna práce navrhuje četná zlepšení s výhledovým cílem službu nasadit pro využívání studenty ČVUT.

Klíčová slova renderovací clustery, online render farma, webové služby, modulární architektura, FITRender, distribuované výpočty, HTCondor

Abstract

The objective of this thesis was to design an online cloud rendering service able to employ mostly idle school computers. The explored areas of this work include requirements gathering, evaluation of computing potential of selected school computers and insights into methods of sharing these computers with their standard users. The result of this work is a prototype implementation called FITRender which allows rendering of submitted scenes through

a decoupled cooperation of multiple software elements. Finally a number of observations and suggestions is made towards the project to help it reach its goal of using the school's idle computational power to serve its students.

Keywords render clusters, online render farm service, web services, modular architecture, distributed computing, HTCondor, distributed rendering

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Výpočetní clustery pro renderování počítačové grafiky, neboli render farmy, jsou používány všude po světě v různém měřítku. Cílem render farem je pomocí kombinace výkonu propojených počítačů zkrácení doby potřebné pro často časově náročný proces renderování - výpočtu finálního obrazového výstupu [1].

Výhodou síťového renderování, které využívá většího počtu méně výkonných počítačů oproti jednomu silnějšímu je často lepší poměr cena/výkon a teoreticky neomezená škálovatelnost. Takovou infrastrukturu je v malém měřítku možno využít pro osobní účely či ve větším pro potřeby animačních studií [2] a nebo pro poskytování renderovacích služeb [3][4].

Jedním z problémů je však složitější instalace a správa celé infrastruktury. Další problém tkví v potřebné režii pro distribuci jednotlivých úloh. Naštěstí existuje velké množství proprietárních i otevřených systémů [5][6][7], které jsou navržený pro řešení právě tohoto problému; některé z nich jsou zkoumány podrobněji dále v této práci. Díky této nabídce je možné rychle vytvořit efektivní render farmu. Pro sestavení je však nutné mít k dispozici množství volných výpočetních strojů.

Ty je možné nalézt na půdě naší školy - množství počítačů, které jsou využívány většinou pouze v rámci výuky a to pouze pokud se student právě nerozhodl využít vlastní počítač. Dvě z učeben jsou dokonce kompletně vybaveny výkonnými grafickými akcelerátory Nvidia, které samy o sobě nabízí nezanedbatelný renderovací výkon [8][9][10][11].

Nebylo by tedy vhodné nabídnout výkon nevyužitých počítačů jinde, kde by mohl pomoci? Například pro renedurování architektonických vizualizací studentů architektury na fakultě stavební a fakulty architektury v rámci posílení mezifakultních vztahů? To mě inspirovalo k vytvoření této práce, která zkoumá reálné možnosti školních počítačů a způsob, jak na těchto počítačích zprovoznit renderovací farmu s přístupem z uživatelsky přívětivého webového rozhraní.

Cíl práce

Cílem této práce je tedy co nejpřesněji zhodnotit výpočetní potenciál školních počítačů pro potřeby renderování. Navrhnout systém render farmy s podporou nahrávání a správy renderovacích úloh pomocí webového rozhraní, který je možné přizpůsobit na školní síťové prostředí. Identifikovat jednotlivé komponenty této služby, kde je to možné vyhledat vhodné podpůrné technologie a ostatní části navrhnout a vytvořit pro ně prototypovou implementaci.

Na základech, které tato práce vytvoří, by mělo být možné v budoucnu postavit cloudovou renderovací službu, která studentům zpřístupní dosud nevyužitý výpočetní potenciál školních počítačů při zachování jejich stávající použitelnosti.

Jako vedlejší produkt nabídne konstrukce této infrastruktury způsob, jak školní počítače využít pro celou škálu dalších distribuovaných úloh.

Struktura práce

Kromě Úvodu, jehož čtení právě dokončujete je práce strukturována do standardních částí Analýza, Návrh, Implementace, Testování a Závěr.

V kapitole Analýza jsou prozkoumány aktuální možnosti studentů ohledně akcelerace renderovacích úloh včetně stavby vlastní render farmy a využití online výpočetních clusterů 1.1. Dále je zde nabídnuto zhodnocení výpočetního výkonu školních počítačů zaměřené na učebny s grafickými akcelerátory 1.2. Zhodnocení je podloženo četným měřením. Další sekce se zabývá průzkumem provedeným mezi studenty architektury 1.3, který zjišťoval mimo jiné jejich preference, nároky, technickou výbavu a znalost. Posléze jsou krátce představeny možnosti paralelizace a distribuce renderovacích úloh 1.4, které by navrhovaný systém měl podporovat pro efektivní distribuci práce. V sekci Požadavky 1.5 jsou pak zjištěné potřeby a záměry se systémem sepsány do přehledné formy funkčních a nefunkčních požadavků. Dále jsou identifikovány uživatelské role v systému, které jsou pak využity v přehledu hlavních případů užití, tzv. Use Cases 1.7. V následující sekci jsou pak analyzovány jednotlivé moduly systému potřebné pro jeho plnou funkcionalitu 1.8. Závěrečné sekce se pak věnují hledáním již hotových řešení pro tři ze 4 identifikovaných modulů 1.9 1.10 1.11 a výběrem vhodných technologií pro implementaci posledního modulu 1.12.

Kapitola Návrh se věnuje přípravě na implementaci celého systému. Prvně je navrženo několik hardwarových architektur 2.1, které by měl systém podporovat. V těchto architekturách je také modelováno zpracování požadavky o render scény. S touto vědomostí pracuje dále prezentovaný návrh komponent 2.2, který určuje konkrétní propojení zvolených technologií a nabízí způsob umožňující lepší modularitu a obecnost systému. Dále jsou navrženy třídní hierarchie 2.3 pro efektivní implementaci požadovaných vlastností jednotlivých komponent. Závěrečná sekce kapitoly se věnuje návrhu referenčního uživatelského rozhraní 2.4 a představení loga služby.

Následující kapitola, Implementace 3, se věnuje práci provedené při implementaci prototypového řešení. Zmiňuje použité podpůrné technologie a použité postupy. Nejdříve jsou popsány práce provedené pro komponentu WS Adaptér 3.1 a poté postupy pro komponentu Frontend 3.2. V rámci kapitoly jsou také nabídnuty snímky obrazovky z výsledné aplikace.

Předposlední kapitola Testování se věnuje nejprve uživatelskému testování 4.1, které bylo provedeno na mockupové verzi uživatelského rozhraní. Dále pak pokrytí prototypové implementace jednotkovými (unit) testy 4.2 a v závěru zhodnocení funkčnosti prototypu v rámci akceptačního testování 4.3.

Kapitola Závěr nabízí shrnutí výsledků a postupy do budoucna.

Tím končí hlavní část práce, mnohé materiály je však možné nalézt v přílohách.

V rámci textu je navrhovaný systém často odkazován pod pracovním názvem FITRender.

Přeji příjemné čtení.

Kapitola 1

Analýza

1.1 Aktuální stav

V současné době se studentům nabízí několik možností jak zrychlit renderování svých vizualizací.

Kromě možnosti zvýšení výkonu svého počítače se nabízí možnost stavby vlastní malé render farmy, pro což existuje množství materiálů online [12][13]. Provedený výzkum, k dispozici dále, však ukázal, že studentů využívající tuto možnost je velmi málo.

Další možností je využití jedné z velkého množství online renderovacích služeb [4]. Jednotlivé služby se liší zejména podporovanými programy, rychlostí a cenou.

Pro ocenění práce na placených online render farmách se většinou používá jednotka gigahertz na jednu hodinu - GHz/h. Tato jednotka symbolizuje práci jednoho jádra o taktu 1 GHz po dobu jedné hodiny. To znamená že například dvoujádrový procesor o taktu 2 GHz vykoná za jednu hodinu při plném vytížení $2jádra \cdot 2GHz \cdot 1hodina = 4GHz/h$ práce. Samozřejmě vzhledem k různé výkonnosti procesorů se nejedná o zcela objektivní jednotku. Tím pádem není možné brát srovnání cen jednotlivých služeb založené na této jednotce zcela jednoznačně [14]. Pro hrubý odhad, v jaké cenové rovině se jednotlivé služby pohybují však tato míra dostačí.

Abych nesrovnával poměrně abstraktní cenu za jednu GHz/h, která se většinou pohybuje v řádu centů, využiji odhad potřebných GHz/h pro modelový snímek architektonické vizualizace, s dobou renderování 10 hodin na počítači s 500 bodovým výsledkem (průměr studentů z výzkumu dále 1.3.2 je 556) CineBench R15 (o tomto benchmarku více v sekci Výkon školních počítačů). Spotřeba takové scény je dle kalkulátoru služby RANCH [15] 265 GHz/h. Pro odhad celého většího projektu používám cenu deseti takovýchto snímků.

Právě již zmíněný RANCH základní cenu této scény odhaduje na €4–€11 dle zvolené priority [15]. RANCH však nabízí poloviční studentskou slevu [16]. Tím pádem se odhad pohybuje spíše mezi €2–€6 dle priority. Priorita určuje pořadí v celkové frontě všech úloh [17]. Další slevy je možné získat zakoupením balíčku většího množství předplacených GHz/h. Pro zadání, které vyžaduje 10 podobných pohledů by se tedy cena pohybovala okolo €40, tedy přibližně 1000 korun, což se při nabízeném zrychlení (RANCH snímek této scény vyrenderuje po příchodu na řadu za zhruba 2 minuty, celou sadu pak za 20 minut oproti 100 hodinám [15]) nezdá jako až tak obrovská suma. Pro některé studenty však může jednat o vysokou částku, především pokud je třeba rendery opakovat a platit tak opakovaně.

Většina ostatních placených služeb se pohybuje v rozmezí €0.008 (RenderSpell Zombie [18]) přes €0.029 (RebusFarm) po €0.06 (RenderFlow vysoká priorita [19], GarageFarm vysoká priorita [20]). To pro jeden snímek naší modelové scény znamená cenu €2 až €16. Cena renderu celé sady 10 snímků se tedy na placených render farmách pohybuje přibližně mezi 500 až 4300 korunami.

Na trochu jiném principu funguje služba Autodesk A360 Cloud, která nabízí cloudové výpočty pro širší škálu úkolu prováděných z Autodesk aplikací. Kromě renderování z produktů AutoCAD, Revit a dalších [21] nabízí například i strukturální simulaci budov či simulace vstřikování plastových forem. Jako univerzální platidlo využívá Cloud Credits, které se dávají zdarma k zakoupení či prodloužení licenci k vybraným Autodesk produktům [22]. Po vyčerpání těchto kreditů je však nutné si další dokoupit.

Mezi online renderovacími službami jsou však i služby bezplatné.

Například Pea render nabízí možnost bezplatného renderování na jejich serverech za cenu přidaného vodoznaku. Jeho odstranění je pak zpoplatněno v podobných relacích jako ostatní služby na $0.03 za GHz/h [23]. Taková služba se může hodit například pro náhledy před placeným renderem na téže nebo například jiné rychlejší službě. Pea render je však poměrně omezen, co se týče podpory softwaru. Nabízí podporu pouze balíčku 3DS Max s rendererem V-Ray [24].

Dále mezi bezplatné render farmy patří distribuované online render farmy. Tyto farmy nepočítají snímky na vlastních strojích, ale poskytují infrastrukturu pro sdílení práce napříč uživateli. Aby rozdělení výkonu mezi uživatele bylo spravedlivé, využívají tyto služby kreditový systém, který odměňuje uživatele za vykonanou práci. Kredity se pak platí výpočet vlastních projektů. Je tedy možné nechat počítač pracovat pro ostatní ve chvíli, kdy není využit a v případě potřeby využít výkon ostatních uživatelů sítě.

Mezi tyto služby patří například BURP-BOINC. BURP je zkratka pro Big and Ugly Rendering Project. Pro distribuci úloh využívá tento projekt právě BOINC (Berkeley Open Infrastructure Network Computing) [25][26], která možná bude někomu povědomá. Právě tento distribuční systém se používá pro mnohé výzkumné projekty jako je například SETI@home hledající mimozemskou inteligenci [27] či World Community Grid, který zašticuje například výzkumy ohledně léčby AIDS, rakoviny a eboly, zlepšení solárních článků a dalších [28]. Služba podporuje pouze program Blender s jeho vestavěnými renderery.

Další takovou službou, také pouze pro Blender, je Sheep It! Render Farm [29]. Pro distribuci využívá vlastní infrastruktury s vlastní klientskou Java aplikací, která zajišťuje stažení Blenderu, scény, zpracování a odeslání zpět na server [30]. Z vlastní zkušenosti je používání velmi jednoduché a spolehlivé. Služba poskytuje statistiky, ze kterých je jasně vidět aktuální zaplnění a výkon [31].

I přes velké množství nabízených služeb není jejich využití dle provedeného výzkumu příliš rozšířeno. Konkrétním důvodům se věnuji v samostatné sekci o průzkumu.

Pro zkušené počítačové uživatele ještě existuje možnost v použití obecných cloudových výpočetních infrastruktur, jakými je například *MetaCentrum* sdružení CESNET [32] nebo *Amazon EC2* [33]. Toto řešení však počítá se samostatnou konfigurací celého procesu a může tak být časově náročné.

c

Tabulka 1.1: Hardwarová výbava školních učeben

Místnost	Procesor	Grafický akcelerátor
T9:349	Intel Core i5-4570S (Haswell), 2.9GHz	-
T9:350	Intel Core i5-2400 (Sandy Bridge), 3.1 GHz	Nvidia GeForce GTX 480
T9:351	Intel Core i5-3470 (Ivy Bridge), 3.2 GHz	Nvidia GeForce GTX 560 Ti

1.2 Výkon školních počítačů

Pro zjištění dostupného výkonu jsem provedl několik měření na školních počítačích v GPU Laboratořích v místnostech T9:350 a T9:351 a také na počítačích v místnosti T9:349 bez výkonných GPU akcelerátorů, ale s novějšími procesory postavenými na architektuře Haswell.

1.2.1 HW Výbava

Pro přehled zde uvádím konkrétní momentálně instalovanou HW výbavu počítačů v jednotlivých místnostech obsažené v tabulce 1.1

Jedná se o procesory stejné kategorie i uváděcí ceny. Všechny instalované procesory mají 4 jádra a nemají podporu HyperThreadingu, který umožňuje spouštět ke každému jádru dvě vlákna namísto jednoho, což je bohužel zrovna pro renderování vcelku přínosné. Generační rozdíly spojené s pokročilejšími vyrobními procesy o menší litografii mají nejpozorovatelnější vliv na spotřebu. Zároveň mají novější Intel procesory pokročilejší vestavěný grafický akcelerátor, který však pro potřeby této práce nemá užitek [34][35][36]. Je tak možné od všech procesorů očekávat obdobný renderovací výkon.

Mezi GPU jednotkami GTX 480 a GTX 560 Ti je však co se týče specifikací a tudíž i předpokládaného výkonu větší rozdíl. GTX 560 Ti je sice novější kartou, ale nižší kategorie. Toto se projevuje zejména v nižším množství paměti - 1024MB oproti 1536MB na GTX 480 - a nižší počet CUDA jader - 384 oproti 448 [37][38]. GTX 560 Ti má pouze jako novější karta mírně vyšší takt. Vyšší třída karty GTX 480 napovídá lepšímu renderovacímu výkonu, zejména díky vyššímu počtu CUDA jader. Větší množství paměti také umožňuje pracovat s komplexnejšími scénami.

1.2.2 Benchmarky

Na počítačích jsem spustil benchmarky *CineBench R15*, zaměřen na renderování na CPU, a LuxMark v3.0, zaměřen na renderování pomocí OpenCL podporujících zařízení, v tomto případě GPU.

CineBench R15 využívá renderovacího enginu Cinemy 4D a jeho výsledky se používají například pro odhad ceny na renderovacích službách jako RebusFarm [39], RenderStorm [40] a dalších [41].

Tabulka 1.2: Výsledky školních benchmarků

Místnost	CineBench R15	LuxMark Simple
T9:349	407	-
T9:350 (GTX 480)	432	4433
T9:351 (GTX 560 Ti)	454	3107

Oba benchmarky hodnotí výkon počítače pomocí skóre, díky kterému je možné výsledek porovnat s ostatními.

Tabulka 1.2 obsahuje naměřené výsledky jednotlivých benchmarků.

Výsledky CineBench benchmarků jsou pro jednotlivé procesory dle předpokladu řádově podobné; mírně lepších výsledků dosáhly procesory s vyšší základní frekvencí.

Pro představu, jak si tyto procesory vedou v globálním měřítku je možné nahlédnout do databáze výsledků CineBench R15 Benchmarku dostupné na cbscores.com. Například desktopové procesory vyšší řady i7 se stejným počtem jader a podporou technologie HyperThreading dosahují výsledků 606 (i7-2600) až 899 (i7-4770K). Nejvýše se samozřejmě řadí serverové sestavy s několika procesory Xeon či Opteron, ale při jednoprocesorové konfiguraci si vede nejlépe osmijádrový Intel Core i7-5960X s 1707 body [42][43].

Procesory dostupné v učebnách se tak řadí dle očekávání spíše do slabší kategorie. Hlavním cílem těchto benchmarků však bylo získat možnost porovnání s výkonností počítačů studentů architektury, které je k nalezení v následující sekci Průzkum mezi studenty architektury 1.3.2.1.

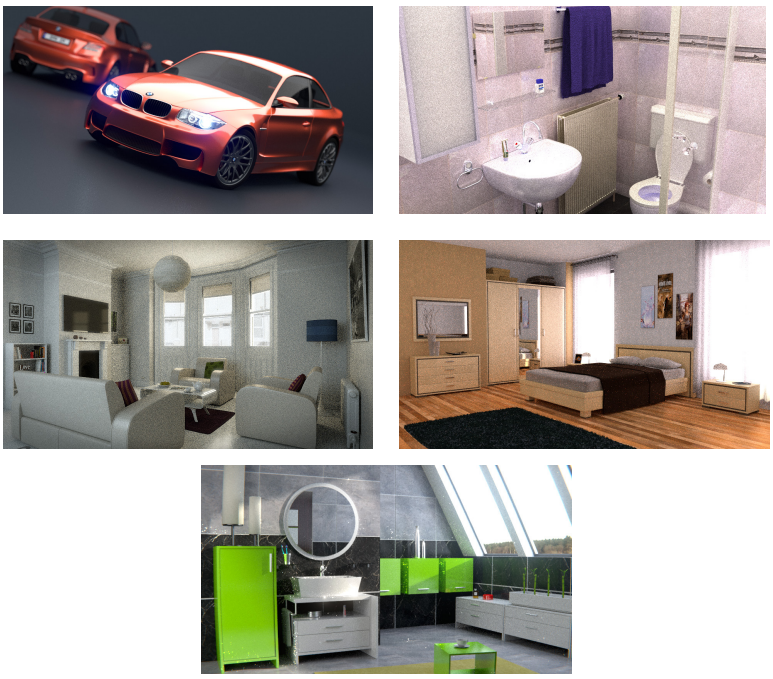
Výsledky LuxMark benchmarku potvrzují předpoklad o nezanedbatelně vyšším renderovacím výkonu karty GTX 480, která dosáhla přibližně 1,4 násobného výsledku. Jaký vliv to má pro reálné scény je možné nalézt v následující sekci.

1.2.3 Zkušební scény

Rozdílné typy provedených benchmarků pro CPU a GPU neumožňují jejich přímé porovnání a zjištění možnéhoo zrychlení renderování na GPU oproti CPU.

Proto jsem na počítačích provedl sérii zkušebních renderů pomocí rendereru Cycles, který je vestavěn v balíčku Blender. Tento renderer umožňuje renderovat stejné scény na CPU i GPU a nabízí tak možnost přímého srovnání.

Zvolil jsem rozšířenou benchmarkovací scénu pro Blender s vozidlem BMW [44], dále pak čtveřici volně dostupných interiérových scén ze serveru Blend Swap. Jednu koupelnovou od uživatele bobal57 [45] (Bathroom), druhou koupelnovou od uživatele cenobi [46] (Lazienka), obývák od uživatele Jay-Artist [47] (The White Room) a ložnici od uživatele SlykDrako [48] (Quarto). Veškeré



Obrázek 1.1: Výsledky měřících renderů. Zleva doprava BMW, Bathroom, Lazienka, The White Room, Quarto.

tyto scény jsou dobře optimalizované co se týče paměťové náročnosti, žádná z nich nevyžaduje více než 200MB. To je umožňuje bez obav renderovat na GPU.

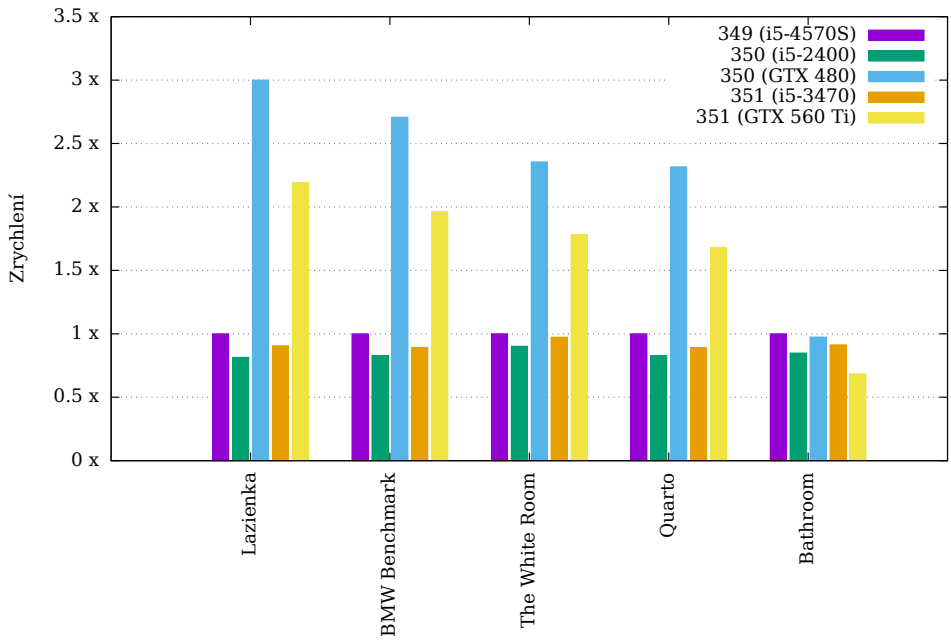
Vzhledem ke zkoumání relativních rozdílů rychlosti mezi jednotlivými konfiguracemi jsem scénám nastavil nižší kvalitu pro rychlejší dokončení. Proto jsou výsledné obrázky znatelně zašuměné.

Výsledky měření uvádím v grafu 1.2. Uvádím pouze násobné zrychlení oproti referenčnímu výsledku a ne absolutní čas, aby bylo možné srovnávat výkon mezi jednotlivými scénami. Jako referenční jsem zvolil výsledky naměřené na CPU i5-4570S v místnosti T9:349, které se překvapivě (oproti benchmarku CineBench) ukázaly být nejrychlejšími z výpočtů na procesoru. Konkrétní naměřené časy spolu s renderovanými scénami včetně použitého nastavení je možné nalézt na přiloženém disku.

Toto měření znovu potvrdilo výkonnostní převahu grafické karty Nvidia GTX 480. Zrychlení od GTX 560 Ti se stabilně pohybovalo mezi 1,3 a 1,4. To mimochodem i vcelku přesně odpovídá rozdílů bodů těchto karet v benchmarku LuxMark jehož výsledky jsou prezentovány v minulé sekci.

Z grafu 1.2 je patrné, že na zkušebních scénách dosáhla karta GTX 480 až třínasobného zrychlení oproti nejrychlejšímu CPU výsledku a to pro scénu Lazienka.

Z ostatních výsledků se však vymyká scéna Bathroom. Proto jsem se ji



Obrázek 1.2: Graf zrychlení renderování na GPU oproti CPU

rozhodl analyzovat, abych zjistil, čím je slabý výkon na GPU způsoben.

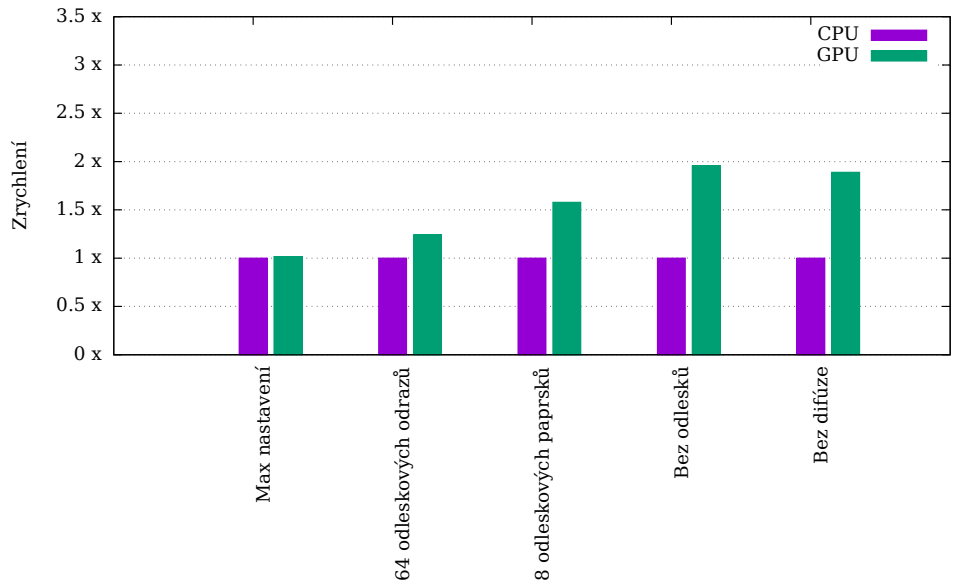
1.2.3.1 Analýza scény Bathroom

Jedná se o poměrně jednoduchou scénu, která neobsahuje složitou geometrii ani velké množství světel, ve skutečnosti je osvětlena pouze jednou emisivní plochou umístěnou nad scénou. Pro své celkové osvětlení však spoléhá na velké množství světelných odrazů v rámci místnosti.

Proto autor volil poměrně neobvyklé brute force nastavení světelných odrazů. V rendereru Cycles je možné specifikovat množství odrazů samostatně pro skupiny difúzních (Diffuse), odleskových (Glossy), transmisivních (Transmission) a volumetrických (Volume) paprsků [49]. Autor scény zvolil pro difúzní, odleskové i transmisivní paprsky 256 odrazů.

Zkusil jsem tedy experimentálně měnit tato nastavení a zaznamenával jsem změny ve zrychlení CPU oproti GPU a rozdíly ve výsledném obrazu. Tuto analýzu jsem prováděl na svém stroji s konfigurací s CPU Intel Core i7-5930K a grafickou kartkou GTX Titan Black. Výsledky tedy není možné přímo aplikovat na školní PC, ale je možné sledovat obdobné trendy.

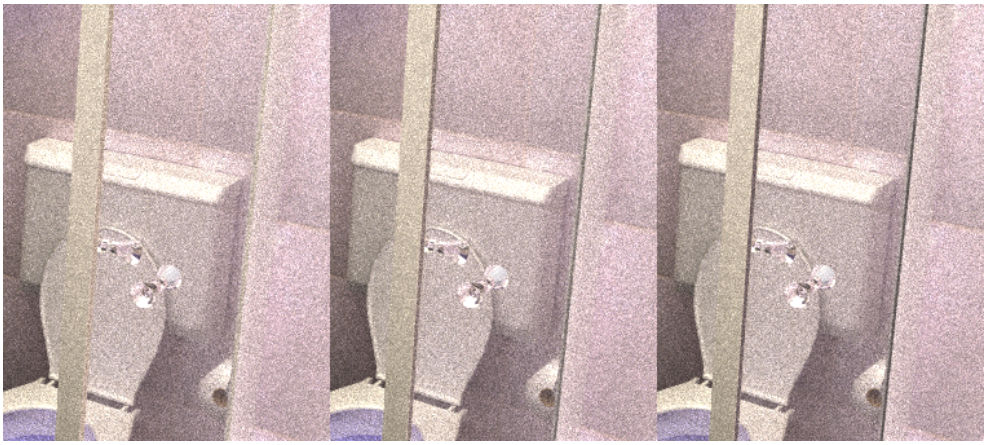
Výsledky v grafu 1.3 ukazují, že na vině špatného výkonu GPU karty oproti CPU je opravdu vysoké množství paprskových odrazů. To naznačuje, že vhodná úprava nastavení může docílit výrazného zrychlení za cenu malého (význam rozdílu je samozřejmě subjektiviní) snížení kvality.



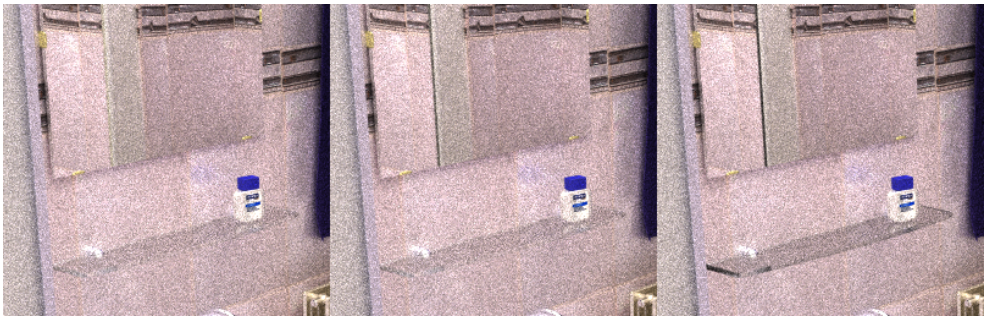
Obrázek 1.3: Graf zrychlení renderování na GPU oproti CPU pro scénu Bathroom v různých konfiguracích

Obrázek 1.4: Náhled renderů scény Bathroom v různých nastaveních

Scény od uživatelé však mohou být konfigurovány libovolně a bylo by tedy samozřejmě vhodné při přidělování práce brát v rámci možností v potaz konkrétní konfiguraci scény.



Obrázek 1.5: Detail rozdílu výsledků scény bathroom. Snížené osvětlení tenkých ploch sprchového kouta se sníženým počtem odrazových paprsků.



Obrázek 1.6: Detail rozdílu výsledků scény bathroom. Tenká plocha sprchového kouta v odrazu zrcadla v jiné části obrazu. Zatmavená skleněná plocha při radikálním snížení odlesků.

1.2.3.2 Závěr

V každém případě je z tohoto měření patrně, že GPU v učebnách nabízejí nezanedbatelný renderovací výkon. GTX 480 ve většině případů alespoň dvojnásobný a GTX 560 Ti jeden a půl násobný. To znamená, že při možné kombinaci GPU a CPU renderování vydá v podstatě každý počítač s GTX 480 za tři a s GTX 560 Ti za dva.

Je však třeba mít stále na paměti, že GPU jsou při renderování omezena zejména dvěma faktory.

Prvním je množství dostupné paměti VRAM, jelikož valná většina GPU rendererů potřebuje celou scénu umístit do grafické paměti. Výjimku tvoří renderery Redshift a FurryBall, které jsou schopny za cenu malého snížení výkonu swapovat geometrii a textury z VRAM do hlavní RAM paměti [50][51].

Druhým je podpora softwaru. Ne všechny renderery v dnešní době podporují renderování na GPU. Na druhou stranu však existují mnohé pouze GPU renderery a omezené GPU verze populárních CPU rendererů, jako je například V-Ray RT [52] pro V-Ray nebo iRay pro Mental Ray [53].

1.3 Průzkum mezi studenty architektury

Pro lepší seznámení s potřebami potenciálních koncových uživatelů této práce jsem provedl výzkum mezi studenty architektury ČVUT formou dotazníku o přibližně 30 otázkách. V rámci tohoto dotázníku jsem zjišťoval schopnosti, metodiky, preference a technické zázemí studentů ohledně architektonických vizualizací, zkušenosti s cloudovými renderovacími službami a zájem o takovou službu poskytovanou přímo školou. Odpovídalo celkem 52 studentů, z toho většina z fakulty architektury.

Také jsem díky tomuto dotazníku sestavil seznam zájemců o předběžné testování této služby. Získáná jména a emailové adresy však v tomto veřejném dokumentu z důvodu ochrany osobních údajů neuveřejňuji.

Uvedu zde klíčové výsledky a závěr z provedeného výzkumu. Kompletní výsledky je pak možné nalézt na přiloženém disku.

1.3.1 Softwarové preference

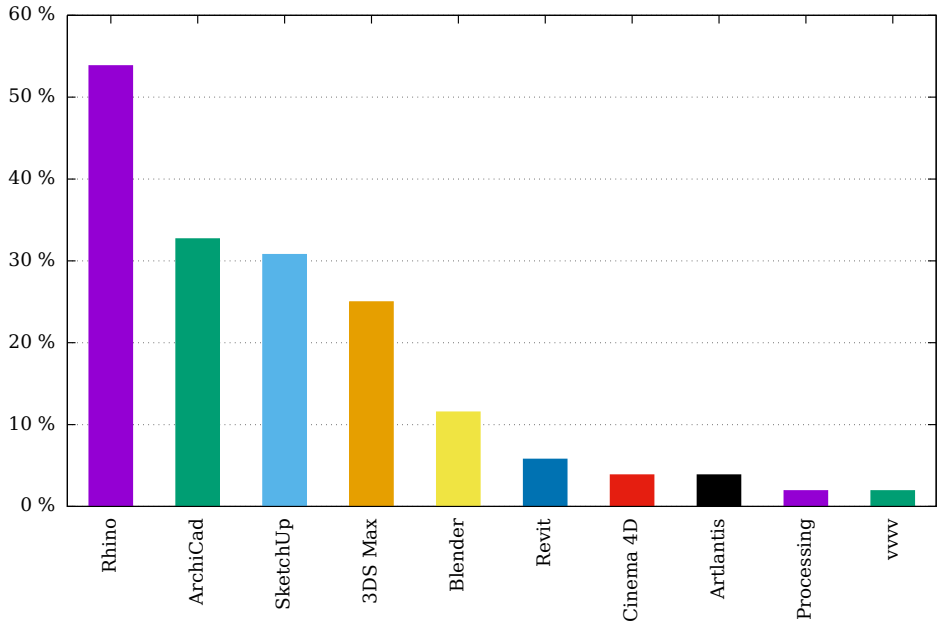
Co se týče používaných softwarů, tak pro tvorbu scén mezi dotazovanými architekty jasně vede Rhino s 28 z 52, tedy 53%. Dále se pak v podobných měrách používá Autodesk ArchiCad (33%), SketchUp (31%) a Autodesk 3DS Max (25%). Ne zcela zanedbatelné zastoupení má jako pátý v pořadí i volně dostupný Blender (12%), který je používán v prototypové verzi této práce.

Jako další se v odpovědích objevily Revit, Cinema 4D, Artlantis a pro mě překvapivě se v odpovědích našly i vizuelně zaměřený programovací jazyk [54] Processing a generativní vizualizační program [55] vvvv.

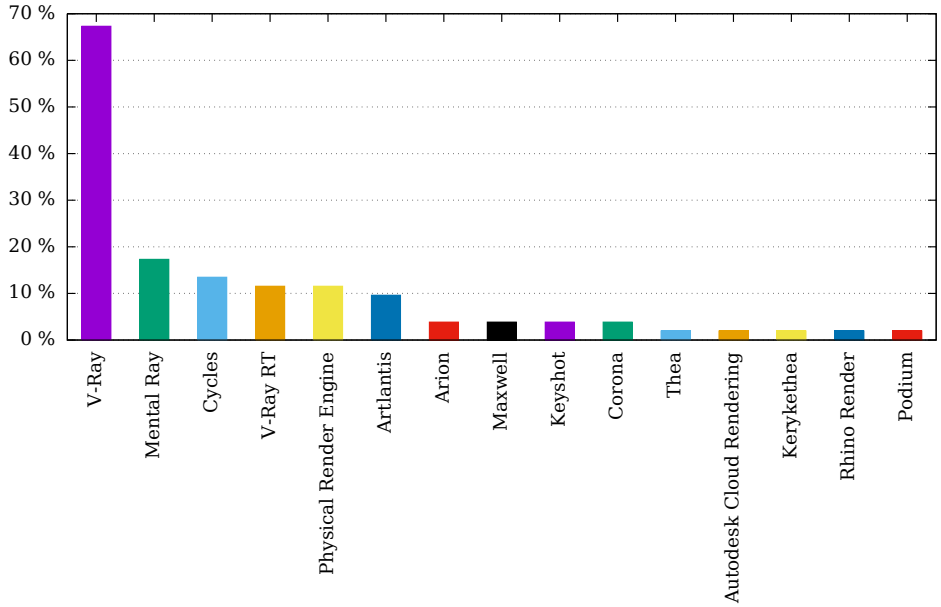
Důležitější pro render farmu jsou však používané renderery. Zde jsou výsledky mnohem jednoznačnější a pro znalé v oboru pravděpodobně předpokládané. S 67% využitím (35 dotázaných) vede V-Ray. To hned z několika důvodů. Vzhledem k tomu že se v oboru jedná o takzvaný industry standard [56], tak pro V-Ray existuje mnoho učebních materiálů zaměřených na architektonické vizualizace. Má širokou nabídku kvalitních přednastavených renderovacích materiálů a kvalitní dodatečné zdroje. Dále má nativní pluginy pro nejpoužívanější 3D balíčky [57] [58] [59] [60] [61] [62] a pro ostatní je možné využít samostatné verze V-Ray standalone [63]. Výsledné obrázky se dají do velké míry označit za fotorealistické. Nejedná se však o ryze fyzikálně korektní raytracer, díky čemuž je V-Ray ve své třídě CPU rendererů je velice rychly.

Další místa kromě čistě raytrace verze V-Raye vytvořené pro renderování na GPU se umístili vestavěné renderery 3DS Maxu Mental Ray, Blenderu Cycles, ArchiCadu a Cinemy 4D Physical Render Engine. Pro kompletní výsledky viz graf 1.8 nebo data na přiloženém disku.

Vzhledem k jasné převaze bych renderer V-Ray označil jako nejvhodnějšího kandidáta pro rozšíření FITRenderu. Použití V-Raye se už bohužel váže na placenou licenci. Ta se při ročním pronájmu pro 15 počítačů pohybuje okolo 30 tisíc Kč bez zahrnutí HW licenčních donglů [64]. Naštěstí není nutné pořizovat



Obrázek 1.7: Používané 3D balíčky mezi studenty architektury



Obrázek 1.8: Používané renderery mezi studenty architektury

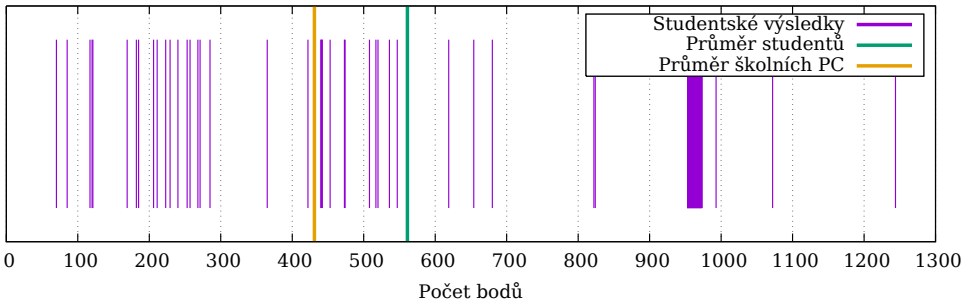
licenci pro každý podporovaný 3D balíček, ale stačí pouze pro verzi V-Ray standalone, která podporuje formát .vrscene, do kterého je možné exportovat z většiny V-Ray pluginů [65] [66] [67] [68].

1.3.2 Technická výbava

1.3.2.1 Výpočetní výkon

Školní render farma by neměla příliš velký význam, pokud by neposkytovala vyšší výpočetní výkon než ten, kteří mají studenti dostupný na svýj strojích. Proto jsem nechal každého studenta, který odpovídal dotazník změřit svůj výkon procesoru pomocí benchmarku CineBench R15. Při porovnání s výsledky školních počítačů to umožňuje zjistit, jakého zrychlení je možné dosáhnout oproti aktuálnímu stavu studentů.

Graf 1.9 znázorňuje jednotlivé výsledky studentů a je doplněn o celkový průměr z odpovědí (zelená linka, hodnota 556), který je podpobný průměru studentů, kteří mají zájem službu využít (565), a průměr měřených školních PC v místnostech T9:349, T9:350 a T9:351 (oranžová linka, hodnota 431). Tlustší čára značí, že konkrétní výsledek zadalo více studentů. Například na hodnotě 963 se sešlo 13 studentů s podobnou konfigurací počítače.



Obrázek 1.9: Výsledky benchmarku CineBench R15 jednotlivých studentů včetně průměrné hodnoty studentů a školních PC

Z výsledků je patrné, že výkon CPU školních počítačů se pohybuje pod průměem dotazovaných studentů. Z grafu 1.9 je ale také patrné, že dostupný výkon jednotlivých studentů je rozprostřen ve velmi širokém spektru od 70 do 1244 bodů.

V každé ze tří měřených učeben je 24 počítačů, dohromady se tedy jedná o *72* potenciálně využitelných strojů. Pokud bychom využili závěry z minulé sekce a počítali s renderováním na GPU, tak je možné 24 počítačů s výkonnějšími GPU v místnosti T9:350 brát jako trojnásobek, tedy 72. Počítače se slabšími GPU z T9:351 pak jako dvojnásobek, tedy 48. Dohromady se při zahrnutí GPU se tak renderovací síla těchto tří místností dá přirovnat ke *144* počítačům pouze s CPU.

Jeden školní počítač tedy bez GPU při tomto přístupu odpovídá průměru, 431 bodů. Všech 72 bez GPU *31032* bodů a se zahrnutím GPU, kdy se počítače počítají vícekrát dle výkonnosti jejich grafické karty se dostáváme na hodnotu *62064*.

Pojďme se tedy s těmito výsledky na paměti podívat, jaké možné zrychlení se nabízí studentům, kteří odpovídali na tento dotazník.

Bez GPU by nejvýkonnější studentský počítač s 1244 body by potřeboval k nahrazení svého jednoho počítače necelé tři školní a potenciál pro zrychlení oproti jeho aktuálnímu stavu by byl při kompletním využití včech 72 počítačů v měřených učebnách pouze 25 násobný. Při zahrnutí GPU pak zhruba 50 násobný.

Výsledek 1244 však odpovídá vysoce výkonnému procesoru typu Intel XEON X5650 nebo Intel Core i7-4930K [42] [43], které je možné najít většinou spíše v desktopových pracovních stanicích nebo serverech. Pro majitele takto výkonných strojů nneí tato služba primárně určena.

Na druhé straně nejnižší výsledek 70 bodů by teoreticky mohl bez GPU získat už při spuštění svého renderu na jednom počítačů přibližně až 6 násobného zrychlení. Při využití všech tří učeben až 443 násobného zrychlení. S GPU pak přibližně 886 násobný. Konkrétně tento student využívá pro renderování V-Ray a renderování jednotlivých hlavních pohledů mu zabere déle než 8 hodin. Při volné render farmě by tento student měl práci dokončenou výrazně rychleji a měl možnost se více soustředit na samotný obsah práce a teoreticky tak zvýšit kvalitu výsledku.

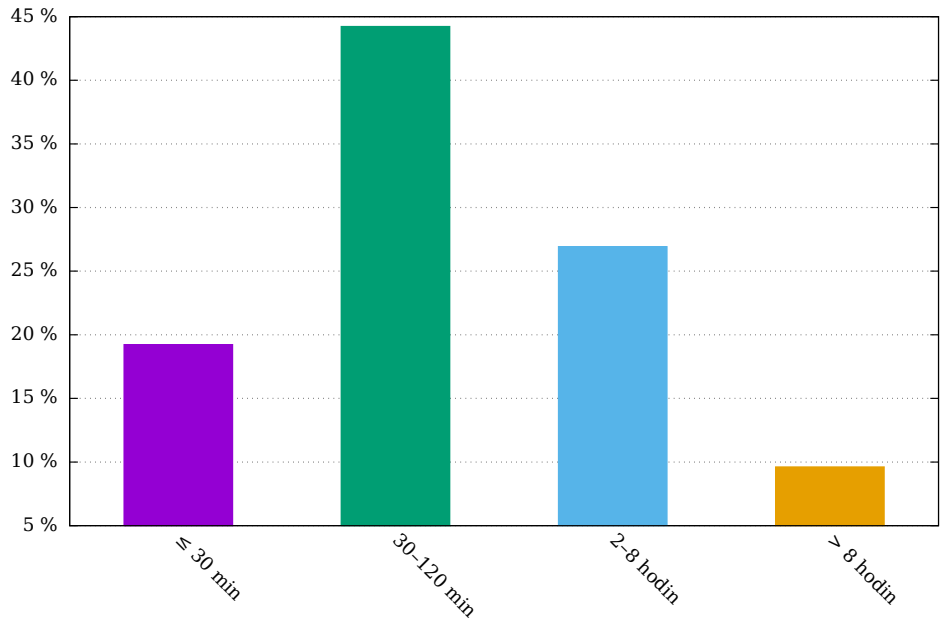
Pokud bychom se zaměřili na průměr studentů, kteří chtějí využít službu, hodnotu 565, tak je patrné, že na jednom počítači není možné zrychlení dosáhnout. Při použití všech měřených počítačů je možné potenciálně dosáhnout zrychlení až 55 násobku bez GPU a 110 s GPU.

Pokud si vytvoříme modelovou scénu a sadu scén na základě zjištěných dat v rámci dotazníku, tak je možné provést orientační odhady vytížení služby. Tento odhad je v další sekci Doba renderování.

1.3.2.2 Doba renderování a odhad zátěže

Dotazník zjistil, že 20% (10) studentům zabere renderování jednoho hlavního pod 30 minut, 44% (23) 30-120 minut, 27% (14) 2-8 hodin a 9,6% (5) více než 8 hodin. Pokud bychom použili prostřední hodnoty těchto intervalů a odhad pro poslední interval například 12 hodin, je možné vytvořit si představu o průměrné době renderu scény studentů. Tato hodnota je rovna $10 \cdot 15 + 23 \cdot 75 + 14 \cdot 300 + 5 \cdot 720)/52$, tedy 186 minut - přibližně 3 hodiny.

Za předpokladu, že službu využívá 100 studentů a každý si chce vyrenderovat 10 snímků o průměrné době 186 minut na průměrném studentském počítačí hodnoceném 565 body CineBench R15 benchmarku, odhadneme zda jsou počítače schopny zvládout takovou zátěž. Jedná se tedy celkem o 1000 snímků.



Obrázek 1.10: Průměrné časové intervaly doby renderování studentů

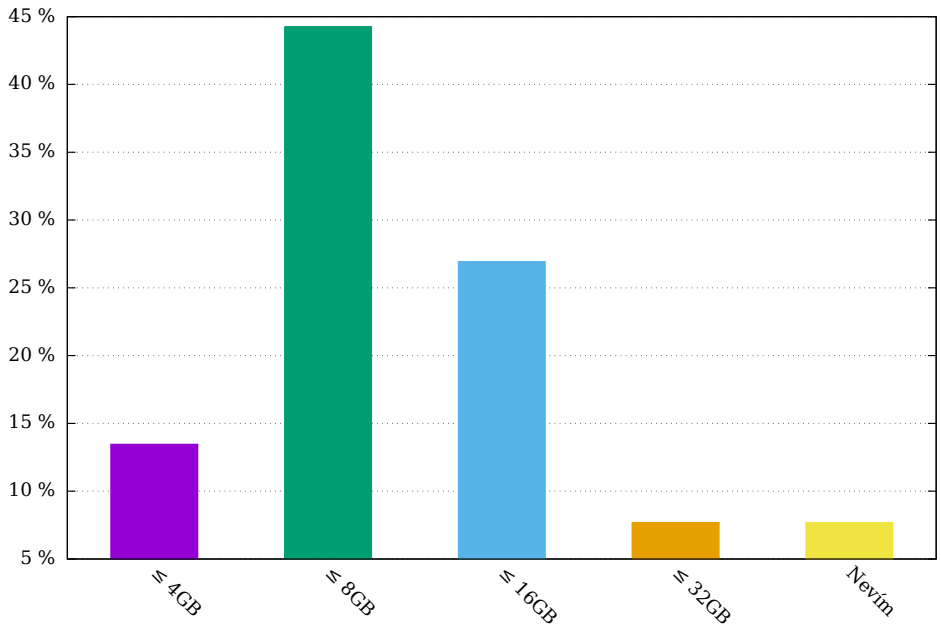
Také předpokládáme, že tyto požadavky jsou zadávány v rámci dvou měsíců při odevzdávání závěrečných prací, rovnoměrně, tedy každý den zhruba 16 snímků, které na studentských počítačích zaberou v součtu 2976 minut, přibližně 50 hodin.

Při odhadech z minulé sekce o síle render farmy je možné docílit 55 násobného zrychlení bez GPU a 110 násobného zrychlení s GPU. Všechny snímky by tak byly zpracovány za 55 minut bez GPU a za 27 minut s GPU. To znamená, že i při zadání najednou by nejdelší doba čekání na snímek, který se renderuje 3 hodiny na studentském počítači, byla jednu hodinu. A to bez jakéhokoli zatížení studentského počítači, který je v tu chvíli možné plně využít pro další práci.

55 minut renderování však nevyužívá farmu plně. Ta je schopna za celý den práce teoreticky zpracovat až 425 bez GPU a 851 s GPU. Pokud bychom vzali v potaz, že počítače jsou přes den využívany, dejme tomu 12 hodin, tak dosáhneme hodnot 212 respektive 425 snímků denně, což stále pokryje modelovou průběžnou zátěž. To znamená teoreticky schopnost obsluhovat celkem 1272 respektive 2550 studentů.

1.3.2.3 Operační paměť RAM

Z odpovědí je patrné, že většina studentů má ve srovnání se školními počítač méně nebo stejně paměti RAM. Konkrétně se jedná o 30 studentů, kteří mají stejně či méně a 18, kteří mají více. To znamená, že pro většinu studentů, kteří



Obrázek 1.11: Dostupná paměť RAM studentů

mohou scény renderovat na svém počítači, nebude nutné kontrolovat a snižovat paměťové nároky jejich scén před odesláním na školní render farmu. U většiny scén je však možné očekávat nároky nižší. Pro možnost využití renderování na GPU je samozřejmě nutné brát ohled na jejich omezenou kapacitu VRAM.

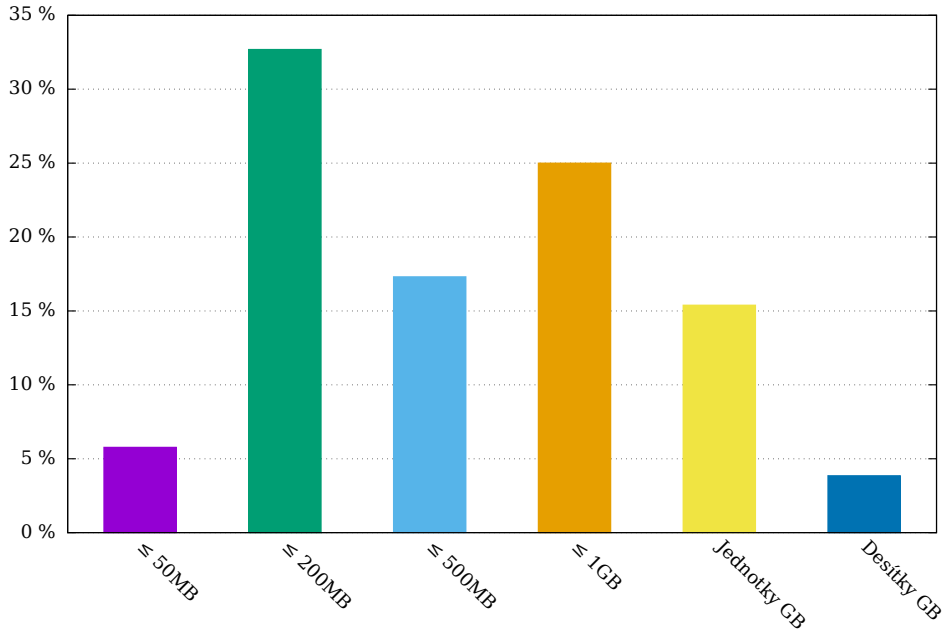
1.3.2.4 Rychlost nahrávání na internet

Nezanedbatelnou částí času stráveného prací s cloudovou renderovací službou je doba potřebná pro nahrání scén. Je teoreticky možné, že tato doba bude tak dlouhá, že ožné zrychlení samotného renderu postrádá smysl.

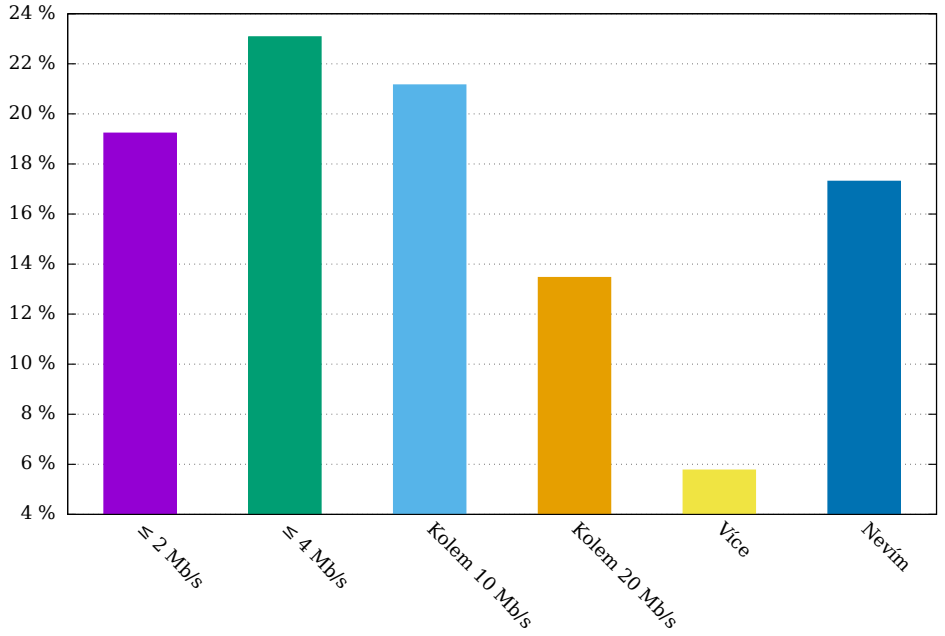
Abych si mohl vytvořit konkrétní představu o tom, kolik času stráví student nahráváním scény na server, tak jsem zjišťoval průměrnou velikost projektů a dostupnou rychlost uploadu. Odhad velikosti projektů je také důležitý pro správný odhad vyžadované kapacity souborového uložiště pro render farmu. Výsledky odpovědí je možné vidět v následujících grafech.

Z grafu velikostí scén 1.12 je jasně vidět, že velké množství projektů, 38% (20), je menších než 200 MB. Spolu se scénami o velikosti do 500MB se jedná o více než polovinu a při zahrnutí projektů do velikosti jednoho GB se jedná o naprostou většinu 81% (42). Velkých projektů, které zabírají jednotky až desítky GB je pak pouze něco okolo 4% (2).

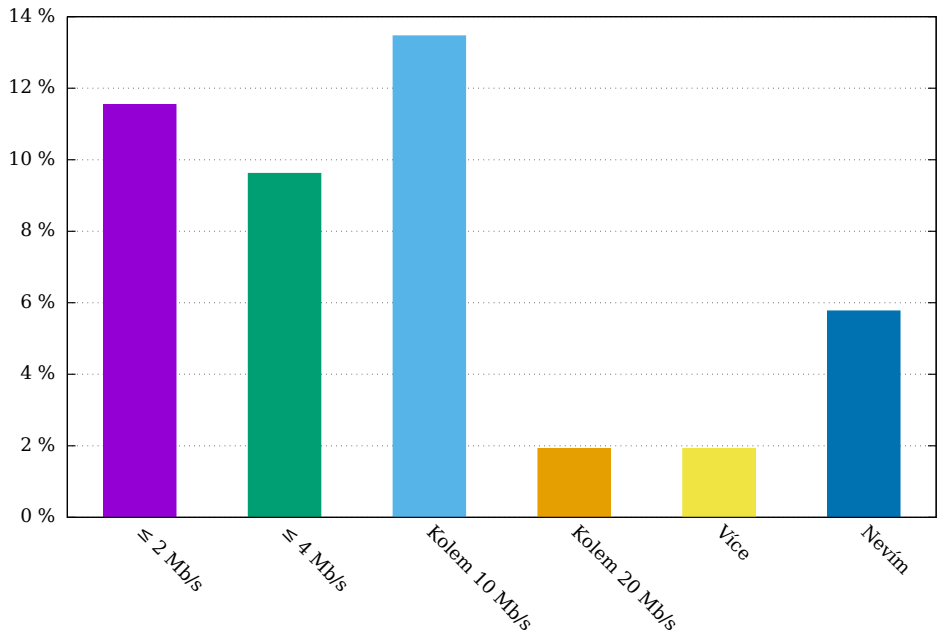
Při poměrně nízkém uploadu 2 Mb/s, jaký nabízí například O2 ADSL [69] zabere upload souboru o velikosti 1GB přibližně jednu hodinu. Při 4 Mb/s pak půl, 10 Mb/s méně než čtvrtinu a tak dále.



Obrázek 1.12: Průměrné velikosti souborů projektů studentů



Obrázek 1.13: Rychlost uploadu studentů



Obrázek 1.14: Rychlost uploadu studentů, kteří mají scény velké 1GB a více.

Vzhledem k tomu, že velikost většiny projektů se pohybuje v rozmezí 200MB až 1GB, tak se i při pomalejším připojení dá předpokládat čtvrthodina až hodina strávená nahráváním scény na server. Pro scény, kde render jednoho snímku vyžaduje několik hodin nebo je dokonce z jedné scénu renderováno více snímku (animace či více pohledů) je patrné, že doba nahrávání je vůči potenciálnímu zrychlení renderu zcela únosná.

Graf 1.14 ještě prezentuje rychlosti uploadu pro uživatele s průměrnou velikostí scén 1GB a výše.

1.3.3 Technické znalosti

Zrychlení renderů je kromě navýšení výpočetního výkonu možné docílit i optimalizací scén a maximalizací využití vlastního výkonu pomocí efektivního naplánování renderování jednotlivých snímků.

Zjišťoval jsem tedy, jaké optimalizační postupy studenti používají (znají) a jak postupují při renderování jednotlivých snímků. Tato znalost se sice na první pohled může zdát jako vcelku zbytečná pro potřeby této práce, ale existují minimálně dva důvody proč mít o technických znalostech uživatelů služby přehled a případně se jí snažit zlepšit. Tím prvním je, že použití dobře optimalizovaných scén může snížit celkovou zátěž render farmy. Druhým je pak fakt, že studenti, kteří jsou schopni samostatně a efektivně jednotlivé rendery připravit do fronty nebo je dokonce automaticky distribuovat na více vlastních počítačů, mohou mnohá zadání vyřešit v únosném času na svých strojích.

Čímž se může dále ušetřit v době vyššího náporu relativně omezených prostředků farmy pro potřebnější.

1.3.3.1 Optimalizace scén

Pro zjištění schopností optimalizace scén jsem studentům dal možnost vybrat z vyčerpávajícího seznamu deseti typů optimalizací s případnou možností připsat vlastní.

Tabulka 1.3 obsahuje nebízené možnosti spolu se svým kódovým označením, které ro ušetření místa použito v grafech. Zároveň jsem si je na základě svého uvážení označit jako esenciální či nikoliv. Esenciálními optimalizacemi mám na mysli ty, které je vhodné použít ve valné většině případů. Zkratky těchto optimalizací u sebe mají hvězdičku.

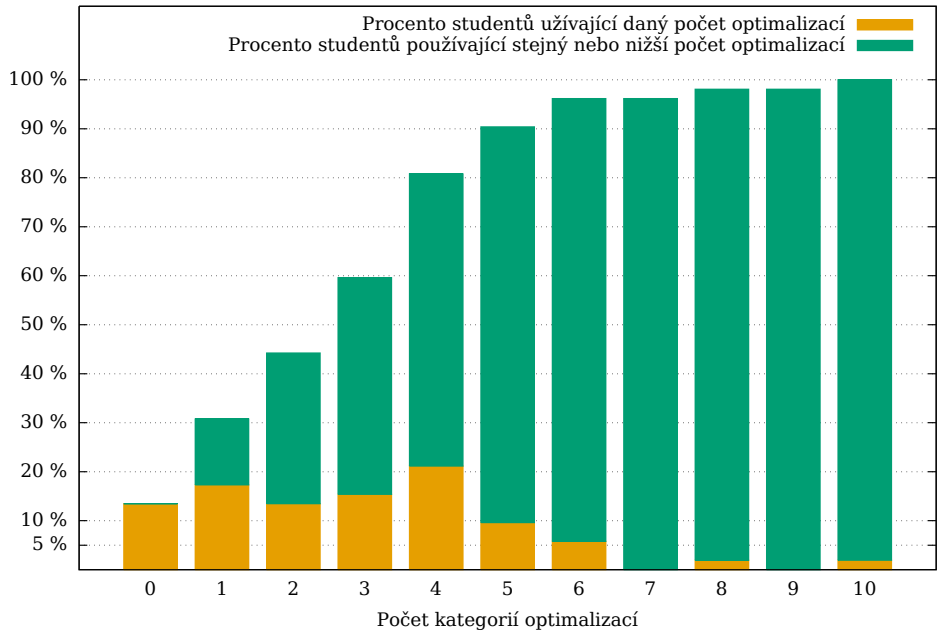
Tabulka 1.3: Přehled možných optimalizací 3D scén

Zkratka	Popis	Esenciální
Deak*	Deaktivace objektů, které jasně nepřispívají do obrazu - nejsou jasně vidět (V podstatě manuální *frustum culling*).	Ano
Maz*	Mazání nepoužívaných objektů, zejména pro snížení velikosti projektového souboru.	Ano
Nast*	Ladění nastavení renderu pro nejlepší poměr čas/kvalita.	Ano
Mat*	Omezení náročnejších materiálu. Vědomý výběr materiálů podle jejich náročnosti.	Ano
Inst*	*Instancování* geometrie, zejména pro snížení paměťových nároků.	Ano
Geom*	Čištění geometrie, minimalizace počtu polygonů.	Ano
SvětMin	Minimalizace počtu světel	Ne
Průch	Rozdělení renderu na několik průchodů	Ne
SvětPap	Nastavení počtu paprsků pro jednotlivá světla	Ne
LOD	Snížení detailu vzdálených objektů (LOD).	Ne

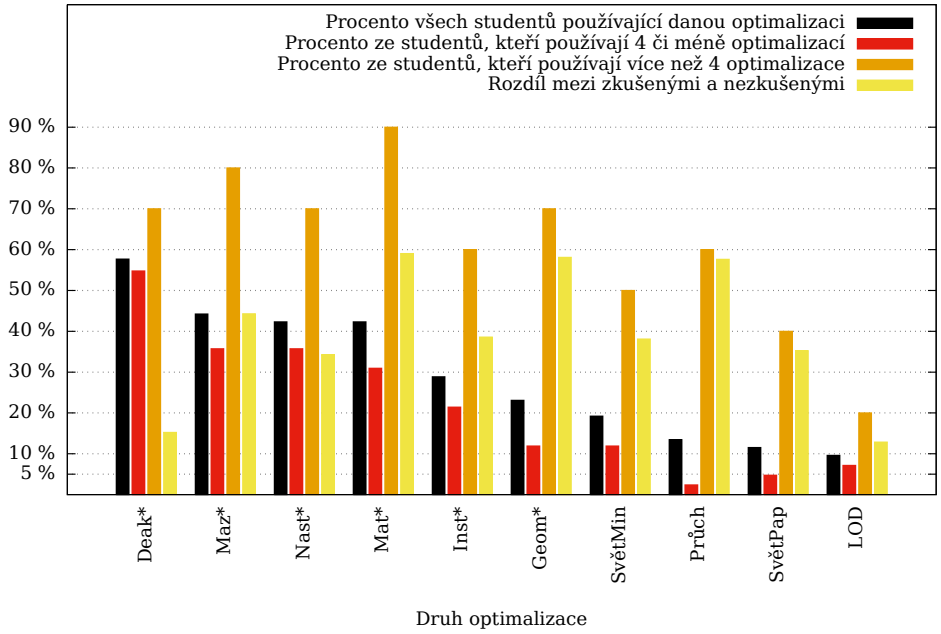
Graf 1.15 rozděluje studenty do jednotlivých skupin podle počtu využívaných optimalizací. Oranžově označené hodnoty ukazují počet studentů využívající daný počet optimalizací. Zeleně je pak graf doplněn o součet studentů využivající nižší počet optimalizací. Všichni studenti využívají 10 nebo méně, proto zde graf dosahuje hodnoty 100% a pouze kolem 10% nevyužívá ani jednu, což je vyznačeno v levé straně.

Ukazuje se, že valná většina - 80% z dotázaných - využívá pouze 4 či méně optimalizací. Desetina neoptimalizuje své scény vůbec.

Graf 1.16 zobrazuje kolik z odpovídajících využívá danou optimalizaci. Graf nabízí samostatný pohled na tři různé skupiny studentů, všechny (černá), méně zkušené používající 4 a méně optimalizací (červená), zkušené používající více než 4 optimalizace (oranžová) a rozdíl mezi posledními dvěmi skupinami (žlutá). Toto rozdělení nabízím, aby bylo možné nalézt typy optimalizací, které jsou mezi méně zkušenými uživateli méně známé.



Obrázek 1.15: Znalostní skupiny studentů vzhledem k optimalizačním metodám



Obrázek 1.16: Přehled používanosti jednotlivých optimalizačních metody podle úrovně zkušenosti

Ukazuje se, že nejrozšířenější optimalizací je dekativace objektů, které příliš neovlivňují render. Jedná se o jednoduchou a jasně viditelnou optimalizaci. Těsně nad 40% všech studentů pak maže nepoužívané objekty, ladí nastavení a dbá na volbu materiálů v ohledu k jejich náročnosti.

Instancování, dle mého názoru velmi důležitý typ optimalizace i pracovního postupu,zejména pro renderování na GPU, používá pouze 29% všech a 21% méně zkušených studentů.

Mezi metody používané výrazně víc zkušenými se dá zařadit vhodný výběr materiálů, čištění geometrie a použití více renderovacích průchodů - tato metoda byla v méně zkušené skupině zvolena pouze jednou. Zrovna u více renderovacích průchodů nejde pouze o zrychlení renderu, ale i o rozšíření možností postprodukce, takže je zajisté vhodné být obeznámen s touto možností.

Méně než 10% rozšíření snižování detailu, LOD, mezi studenty je možné připsat jeho komplikovanější aplikaci a také omezené škále scén, kde je ho možné použít k velkému užitku. Tato optimalizace by byla vhodná například v exteriérové scéně s komplikovanou geometrií rostlin.

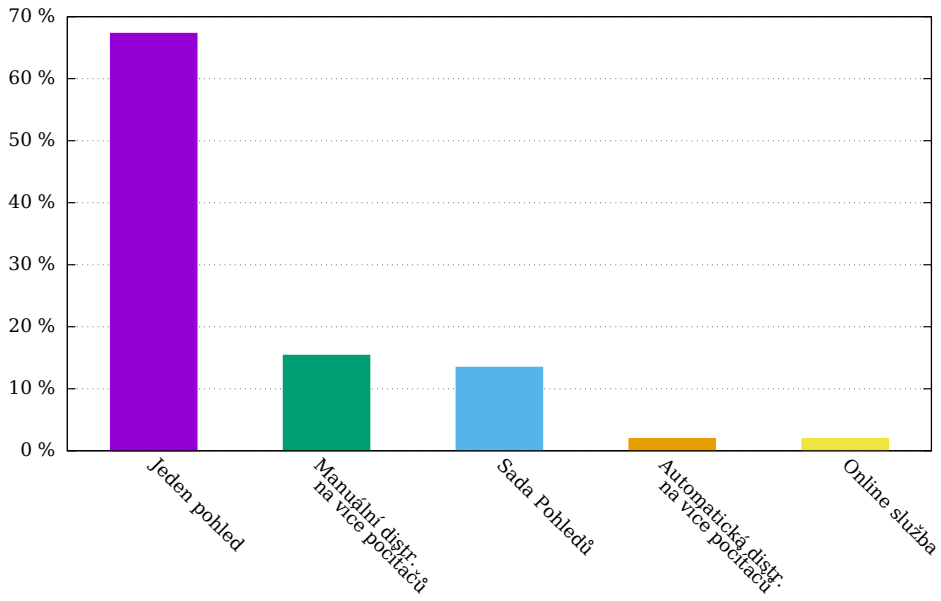
1.3.3.2 Postup při renderování

Architektonické vizualizace se většinou sestávají z více než jednoho pohledu. Při renderování více snímků je možné postupovat několik způsoby. Nejjednodušším a nejméně efektivním je každý pohled připravit, vyrenderovat a přejít na další. To vede ke špatně rozložené práci, kdy je v pracovních hodinách počítač zaneprázdněn renderováním. Lepším způsobem je jednotlivé pohledy ověřit a nastavit a následně zařadit do fronty, která je spuštěna po dokončení uživatelské práce, například přes noc. Další alternativou práci distribuovat na jeden či více jiných počítačů, což je možné provádět manuálně nebo lépe automaticky s pomocí nějakého správce render farmy. Poslední možností je úloha odeslat ke zpracování do nějaké online renderovací služby nebo jiného výpočetního cloudu.

Graf 1.17 zobrazuje způsoby, jakými studenti většinou postupují při renderování jednotlivých pohledů.

Valná většina z dotázaných studentů bohužel postupuje právě tou nejjednodušší a nejméně efektivní cestou - připravit pohled, vyrenderovat, přejít na další. Pouze 7, tedy 13 procent studentů využívá renderovacích front.

Tyto výsledky naznačují, že mnozí studenti by byli při efektivizaci svých pracovních postupů schopni ušetřit čas. Bylo by tedy vhodné studenty architektury v těchto ohledech vzdělávat. FITRender tomu může napomoci umístěním speciální sekce, která by se věnovala postupům a nástrojům k tomu vhodnými. Tato sekce by mohla být ve formě veřejné wiki.



Obrázek 1.17: Přehled postupů při renderování scén studentů

1.3.4 Zkušenosti s cloud renderovacími službami

Většina studentů má pouze okrajové zkušenosti s těmito službami. Aktivně je používají pouze dva. Mezi tyto používané služby patří *Autodesk A360* a *REBUS farm*.

1.3.5 Požadované funkce systému

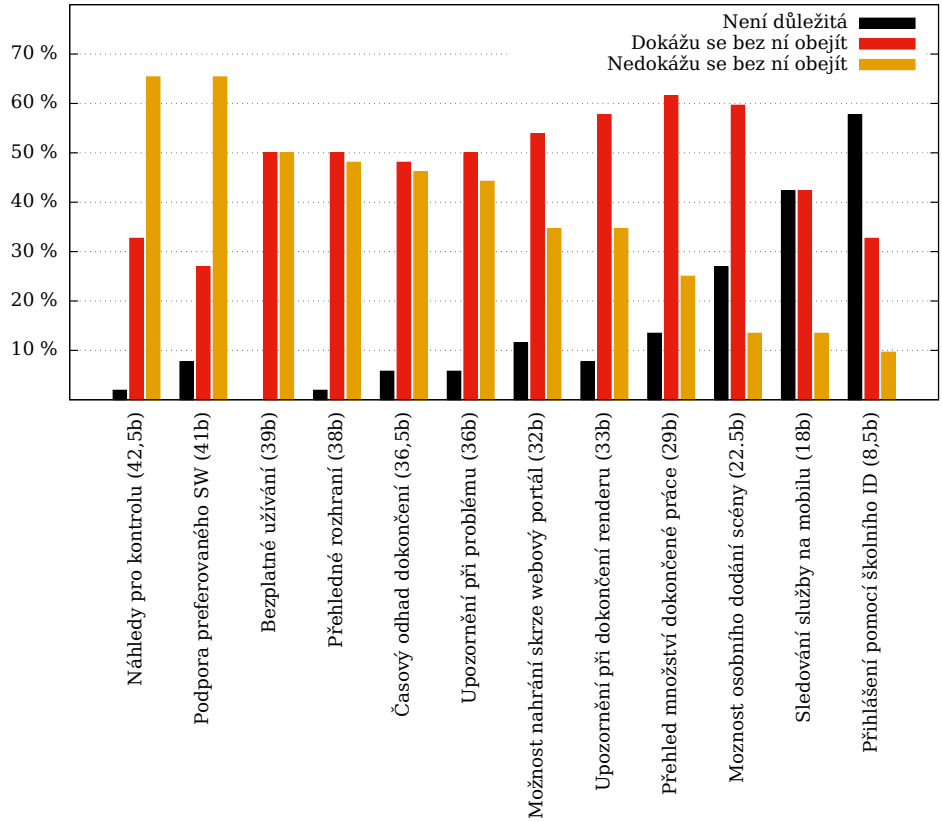
V rámci dotazníku jsem se ptal na důležitost jednotlivých možných dodatečných funkcí pro uživatele. Pro každou vlastnost studenti vybírali ze tří možností důležitosti: Neni pro mě vůbec důležitá (Není důležitá, černá). Je pro mě důležitá, ale dokážu se přizpůsobit a službu přesto využívat (Dokážu se bez ní obejít, červená). Bez této vlastnosti bych službu nemohl/a využívat (Nedokážu se bez ní obejít, oranžová).

Jednotlivé vlastnosti jsem pak ohodnotil body. Za odpověď, že vlastnost není důležitá jsem udělil 0 bodů, za střední důležitost 0,5 a za absolutní důležitost 1. Podle tohoto hodnocení jsem jednotlivé vlastnosti seřadil.

Výsledky prezentuje graf 1.18.

Mezi nejvýše hodnocené se umístila vlastnost poskytnutí náhledů pro kontrolu a poté pochopitelně podpora preferovaného softwaru studenta.

Dále pak vlastnost bezplatného používání, i když je třeba podotknout, že polovina studentů se vyjádřila, že službu dokáže využívat i bez této vlastnosti. Existuje zde tedy možnost případně zpoplatnit například nadužívání



Obrázek 1.18: Přehled potřeby jednotlivých vlastností systému

služby nebo používání služby absolventy za poplatek, což byla prosba jednoho odpovídajícího.

Přehledné rozhraní je alespoň mírně důležité pro valnou většinu studentů.

Mezi méně důležité vlastnosti pak patří přihlášení pomocí školního ID, sledování služby na mobilním telefonu nebo možnost osobního dodání scény (pro větší scény, které by se dlouho nahrávaly).

Žadná ze zmíněných vlastnostní nebyla zcela zavrhnuta.

1.3.6 Závěr

Z provedeného výzkumu plyne několik závěrů.

Prvním je nutná podpora preferovaného softwaru studentů, pokud by měl být systém nasazen v širším měřítku. Jako jasný kandidát se nabízí renderer V-Ray. Je tedy třeba hlouběji analyzovat aktuální stav školních licencí a případně

cenové podmínky licencování tohoto rendereru a jejich výhodnost. Eventuálně do této analýzy zahrnout i možnost mírného zpoplatnění služby.

Jeden z odpovídajících se zmínil, že podobnou myšlenku vystavění školní render farmy zkoumala jeho bývalá škola, Kansas State University v Manhattan, Kansas. Na této škole se však rozhodovalo mezi zakoupení hardwaru pro render farmu a rozdáním kreditů na online renderovací farmy[1].

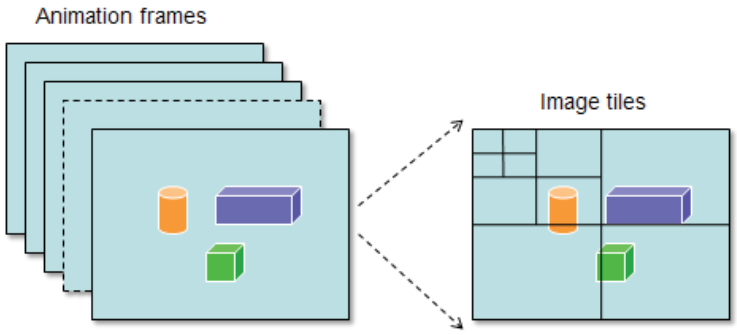
Dále bych zmínil, že by bylo vhodné lépe studenty instruovat ohledně některých postupů vzhledem k tomu, že k renderování často přistupují značně neefektivně.

[1]Tuto informaci jsem získal z e-mailové korespondence v návaznosti na komentář v dotazníku.

1.4 Možnosti paralelizace a distribuce renderovacích úloh

Co se týče paralelizace a možností distribuce, vykazuje úloha CGI renderovaní velmi dobré vlastnosti.

Pokud bychom se zaměřili na možnosti paralelizace na jednom počítači, tak se nabízí hned několik způsobů, jak úlohu efektivně rozdělit a využít plného paraleliního výkonu počítače. V podstatě triviální, ale přesto efektivní a hojně využívanou metodou je rozdělení obrázku na velké množství dlaždic, tzv. tiles, které jsou poté postupně přiřazovány dostupným jadrům ke zpracování [2]. Pro možnost vhodné distribuce a pro maximalizaci pozitivních vlivů této metody se velikosti dlaždic většinou volí poměrně malé, v rozmezí přibližně 16x16 až 64x64 pixelů [8].



Obrázek 1.19: Náznak možné distribuce renderovaných snímků. Animaci je možné rozdělit na jednotlivé snímky a snímky na dlaždice. [2]

Výhodou tohoto postupu je, že je aplikovatelný v podstatě na veškeré úlohy tohoto typu - díky tomu, že výsledný obrázek je vždy možné rozdělit a množství diskrétních oblastí, které se posléze jednoduše spojí.

Pro některé typy hardwaru nebo typy úloh se však nejedná o zcela optimální řešení. Další možnosti, jak úlohu rozdělit mezi více výpočetních jednotek se nabízí při renderování pomocí ray tracingu.

Ray tracing, neboli sledování paprsku, spočívá v simulaci šíření jednotlivých světelných paprsků v rámci scény. Existuje několik způsobů, jak k této simulaci přístupovat - směr simulace, různé optimalizace či heuristiky - ale stále se jedná o simulaci velkého množství nezávislých jevů [70].

To nahrává masivně paralelní architektuře GPU jednotek, které je díky vzestupu CUDA a OpenCL iniciativ v posledních letech možné velmi efektivně využít právě pro tento typ přirozeně paralelních úloh. Při ray tracingu na GPU se proto většinou úloha nedělí na jednotilvé dlaždice, ale na samotné paprsky.

Renderery jako V-Ray RT [71], iRay [71], Cycles [72], Arion [73], FurryBall [51] a mnoho dalších tohoto využívají a jsou často schopné nabídnout

srovnatelné výsledky s CPU renderováním za výrazně kratší časy. Je však třeba mít na paměti, že rozdílná architektura GPU s sebou nese určitá omezení, zejména v množství dostupné paměti VRAM [74].

Při distribuci úloh a více počítačů, jako tomu je například právě v případě render farmy, se dá využít totožných postupů. Tedy například každému počítači přiřadit jiný úsek výsledného obrázku či jiný snímek animace nebo rozdělit sady paprsků. Pokud bychom zanedbali dobu nutnou pro přípravu scény na každém počítači (přenos scény po síti, vybudování akceleračních struktur), tak je vzhledem ke skvělé distribuovatelnosti úlohy očekávat lineární zrychlení.

Obě metody je možné kombinovat a tím docílit potřebné granularity. Například při distribuci na školní render farmu se tak úloha může rozdělit na velké množství menších podúloh. To s sebou sice ponese větší režijní náklady, na druhou stranu se ale výpočty provedou rychleji. To pak nabízí možnost vměstnat výpočty do kratších prostojů v uživatelském používání počítačů, jakými jsou třeba přestávky.

Spojení obrazů je možné zadat jako další úlohy pro výpočetní clusteru. Toto řešení však vyžaduje podporu určení závislostí mezi úkoly[1].

[1]Nejprve se musí dokončit rendery jednolivých částí. Až poté je možné přejít k jejich sloučení.

1.5 Požadavky

Zde nabízím přehledný seznam jednotlivých požadavků na systém plynoucích buď přímo ze zadaní, z dalších požadavků vedoucího práce nebo byly zjištěny v rámci provedené analýzy.

1.5.1 Funkční požadavky

1.5.1.1 F-FR1 Poskytnout webové rozhraní pro zadávání a vyzvedávání renderování 3D scén

Toto je základní součástí samotného zadání. Výsledkem práce má být renderovací služba, která má být schopna renderovat scény zaslané uživateli. Ty tedy musí být možné zadat skrze webové rozhraní a následně možné vyzvednout po dokončení.

1.5.1.2 F-FR2 Upozornit uživatele na dokončené scény

Systém by měl upozornit uživatelu při ukončení renderování scény, například e-mailem či pomocí SMS. Scéna může být ukončena jak při úspěšném dokončení, tak i při selhání.

1.5.1.3 F-FR3 Podporovat správu nahraných scén

Systém by měl umožnit zobrazovat, upravovat a mazat jednotlivé scény.

1.5.1.4 F-FR4 Podporovat přihlášení pomocí celoškolského přihlášení ČVUT

Toto je dodatečnou součástí zadání. Studentům by mělo být umožněno přihlásit se do služby pomocí své celoškolské identity.

1.5.1.5 F-FR5 Podporovat render jednotlivých snímků, ale i animací

Někteří uživatelé nerenderují pouze jednotlivé statické snímky, ale i animace. Proto by měl systém podporovat práci v obou těchto režimech.

1.5.1.6 F-FR6 Podporovat nastavení granularizace úkolů

V závěru analýzy možností distribuce renderovacích úkolů v sekci 1.4 bylo poznamenáno, že větší množství kratších výpočtů je možné lépe vložit do krátkých časových úseků, jakými jsou například přestávky. Proto by bylo vhodné, aby bylo možné upravovat nastavení dělení jednotlivých úloh.

1.5.1.7 F-FR7 Podporovat scény programu Blender

Program Blender byl vybrán jako první program pro naplnění funkce renderovacího modulu. Systém by tedy v každém případě měl podporovat renderování scén pomocí tohoto programu.

1.5.1.8 F-FR8 Možnost správy výpočetního clusteru z webové aplikace

Systém by měl umožňovat upravit vybraná nastavení aplikace, která se mohou vztahovat i na samotný výpočetní cluster. Mezi tyto nastavení může patřit například určení použitých cest pro ukládání souborů či správa jednotlivých renderovacích uzlů.

1.5.1.9 F-FR9 Možnost časového rozvrhování využití farmy a jednotlivých počítačů

Systém by měl umožnit rozvrhnout výpočty tak, aby se nekryly s výukou. Toto zadání pochází od vedoucího práce.

1.5.1.10 F-FR10 Sbírat statistiky ohledně prováděných výpočtů

Systém by měl sbírat alespoň časové statistiky o prováděných úkolech. Díky nim bude možné optimalizovat nastavení pro zrychlení výpočtů či odhadnout dobu nutnou pro výpočet scény. Tyto statistiky by měly být dostupné skrze webové rozhraní.

1.5.1.11 F-FR11 Umožnit správu uživatelů

Webové rozhraní by mělo umožňovat správu uživatelů, zejména přiřazování uživatelských rolí.

1.5.2 Nefunkční požadavky

1.5.2.1 N-FR1 Systém by měl zachovat použitelnost využitých PC pro uživatelskou práci

Počítače, které jsou součástí render farmy by si měly zachovat použitelnost uživateli. To znamená, že počítač může provádět výpočty pouze, pokud je neaktivní. Toto je rozšíření požadavku ohledně časového rozvrhování. Počítače totiž mohou být používany uživateli i mimo učební hodiny.

1.5.2.2 N-FR2 Systém by měl nabídnout přehledné uživatelské rozhraní

Jedná se o těžko měřitelný požadavek. Systém by se však měl v rámci uživatelského rozhraní snažit o prioritizaci hlavních use cases, vhodné použití barev

a grafických prvků a sledování dalších dobrých zvyků návrhu uživatelského rozhraní.

1.5.2.3 N-FR3 Systém by měl být navržen pro více platforem

Systém by se měl snažit o to, aby byl funkční na více operačních systémech, zejména však Linux Gentoo a Microsoft Windows, popřípadě Mac OS X. Tento požadavek je vznesen zejména z eventuální potřeby podporovat více rendererů, z nichž některé fungují pouze na omezeném množství platforem. Příkladem může být Artlantis, ArchiCAD [75] či Cinema 4D [76].

1.5.2.4 N-FR4 Systém by měl umožnit snadnou výměnu jednotlivých modulů

Vzhledem k očekáváné situaci, kdy by bylo vhodné využít jiný distribuční modul 1.9.2, je potřeba, aby bylo relativně snadné jednotlivé moduly vyměnit.

1.5.2.5 N-FR5 Systém by měl být jednoduše rozšířitelný o podporu dalších rendererů

U tohoto typu služby se jedná o kritickou vlastnost.[2] První Blender s rendererem Cycles by neměl být jediným podporovaným softwarem. Vzhledem k průzkumu v podsekci 1.3.1 by jako další na řadu mohl přijít V-Ray a poté mnohé další. Proto by měl být systém rozšířitelný o další renderery.

1.6 Role v systému

1.6.1 Webové rozhraní

Webové rozhraní vyžaduje minimálně tři uživatelské role - hosta, uživatele a správce. Pro lepší možnost rozdělení práv k jednotlivým akcím by však základní verze FITRenderu měla pracovat alespoň se čtyřmi rolemi - *Guest*, *User*, *Elevated User* a *Admin*.

1.6.1.1 Guest

Jedná se o nepřihlášeného uživatele. Tento uživatel může zobrazovat pouze veřejné části stránky (domovská stránka, informace o službě) a má možnost se přihlásit, čimž nabyde jedné z dalších rolí.

1.6.1.2 User

Jedná se o základní rolí systému přiřazenou po registraci každému uživateli. Uživatel s touto rolí může na farmu zasílat scény a volit jejich nastavení, sledovat stav jejich provádění a vyzvednout výsledek. Může také své scény smazat a zrušit tím zpracování. Ke scénám ostatních uživatelů nebo do nastavení výpočetního clusteru User nemá přístup.

1.6.1.3 Elevated User

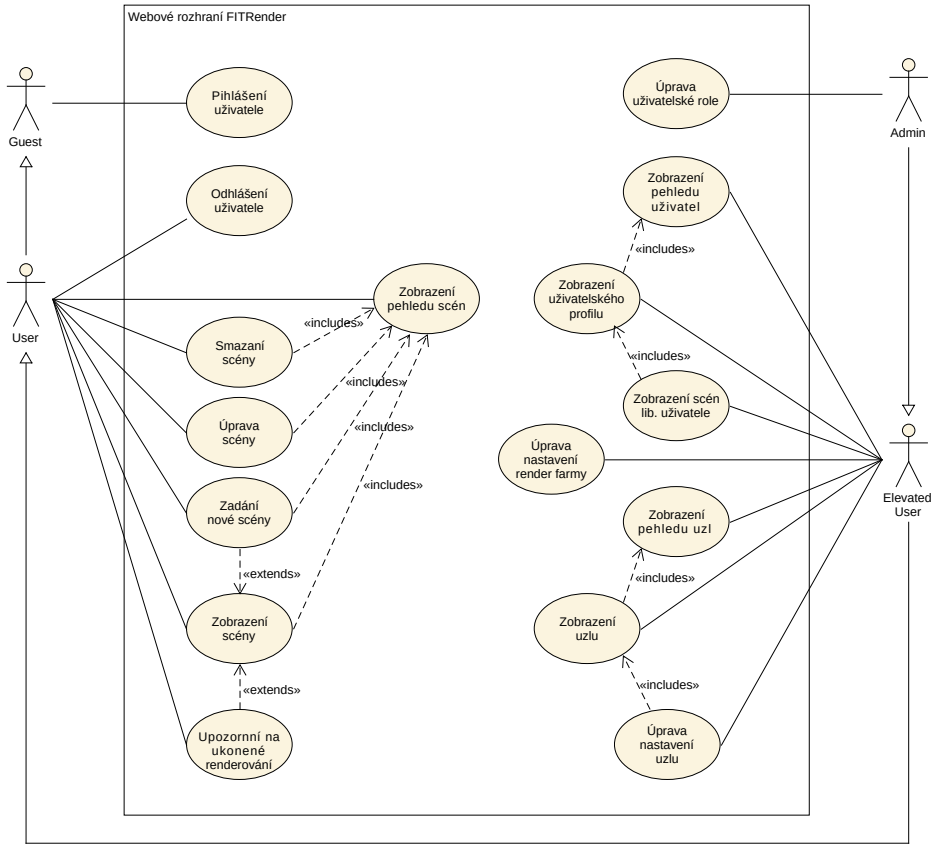
Tato role umožňuje správu systému. Elevated User má přístup k operacím nad všemi scénami a nastavení renderovacího clusteru.

1.6.1.4 Admin

Admin, neboli administrátor má stejná práva jako Elevated User. Kromě toho však může i upravovat nastavení uživatelských rolí.

1.6.2 Správa infrastruktury

Do systému je nutné zahrnout i role fyzických osob starající se o jeho údržbu. Zejména systémové administrátory. Před nasazením do provozu by bylo vhodné tyto role hlouběji identifikovat a rozdělit konkrétní zodpovědnosti.



Obrázek 1.20: Use case diagram nabízející přehled případů užití a jejich propojení

1.7 Use Cases

Abych získal lepší představu o tom, jak se požadavky projeví v interakci systému s uživateli, jsem navrhl sadu případů užití při interakci uživatelů s webovým rozhraním. U těchto požadavků jsem také zpětně ověřil, zda opravdu předepsané požadavky plní.

Navržené případy užití posléze posloužily zejména při návrhu průchodu uživatelským rozhraním.

1.7.1 Seznam zjištěných případů užití

1.7.1.1 UC-FR1 Přihlášení uživatele

Role: Guest

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne přihlásit do systému.
2. Systém zobrazí formulář použitého autentizačního modulu
3. Uživatel vyplní a odešle formulář.
4. Autentizační modul provede kontrolu údajů.
5. Autentizační modul schváli údaje, spraví o tom systém a ten uživatele přihlásí.

Alternativní scénář

1. Scénář je stejný do čtvrtého kroku hlavního.
2. Autentizační modul údaje neschválí a uživatel je vyzván k opakování akce.

1.7.1.2 UC-FR2 Odhlášení uživatele

Role: User, Elevated User, Admin

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne odhlásit ze systému.
2. Systém uživatele odhlásí a tím ho převede do role Guest.

1.7.1.3 UC-FR3 Zadání nové scény

Role: User, Elevated User, Admin

Hlavní scénář

1. Případ užití začíná, jesliže se uživatel rozhodne vložit do systému novou scénu pro vyrenderování.
2. Systém zobrazí výběr dostupných programů.
3. Uživatel zvolí program, pro který nahrává scénu.
4. Systém zobrazí formulář pro přidání scény obsahující tyto atributy:
   - Název
   - Soubor scény
   - Specifické nastavení rendereru, které obsahuje alespoň seznam snímku pro renderování

5. Uživatel formulář vyplní a odešle.

6. Systém formulář ověří. V případě zadání správných údajů akci dokončí. V opačném případě vyzve uživatele k upravení chybných položek a scénář pokračuje od kroku 4.

1.7.1.4 UC-FR4 Zobrazení scény

Role: User, Elevated User, Admin

Omezení: Uživatelům s rolí User je povoleno zobrazovat pouze jejich vlastní scény. Ostatním rolím je povoleno zobrazování všech scén.

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne zobrazit scénu v systému.

2. Systém uživateli zobrazí detailní pohled na scénu, který obsahuje

   - Název scény
   - Nastavení scény
   - Náhled vyrenderovaných snímků s odkazem na verzi v plném rozlišení
   - Statistiky o výpočtu

1.7.1.5 UC-FR5 Úprava scény

Role: User, Elevated User, Admin

Omezení: Uživatelům s rolí User je povoleno upravovat pouze jejich vlastní scény. Ostatním rolím je povolena úprava všech scén.

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne upravit scénu.

2. Systém zobrazí formulář, který nabidne úpravu názvu scény.

3. Uživatel formulář vyplní a odešle.

4. Systém formulář ověří. V případě zadání správných údajů akci dokončí. V opačném případě vyzve uživatele k upravení chybných položek a scénář pokračuje od kroku 2.

1.7.1.6 UC-FR6 Smazání scény

Role: User, Elevated User, Admin

Omezení: Uživatelům s rolí User je povoleno mazat pouze jejich vlastní scény. Ostatním rolím je povoleno mazání všech scén.

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne smazat scénu.
2. Systém uživatele požádá o potvrzení akce.
3. Po potvrzení systém smaže scénu a zruší její výpočet.

1.7.1.7 UC-FR7 Zobrazení přehledu scén

Role: User, Elevated User, Admin

Omezení: Uživatelům s rolí User je povoleno zobrazit pouze jejich vlastní scény. Ostatním rolím je povoleno zobrazování všech scén.

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne zobrazit přehled všech scén.
2. Systém zobrazí seznam scén obsahující pro každou scénu:
   - Název
   - Procento dokončení
   - Odkaz na detail scény

1.7.1.8 UC-FR8 Zobrazení uživatelského profilu

Role: User, Elevated User, Admin

Omezení: Uživatelům s rolí User je povoleno zobrazit pouze jejich vlastní profil. Ostatním rolím je povoleno zobrazování všech profilů.

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne zobrazit uživatelský profil.
2. Systém zobrazí uživatelský profil obsahující tyto položky:

- Jméno
- E-mail
- Odkaz na seznam scén (pro role Elevated User a Admin)
- Odkaz na úpravu uživatelské role (pro roli Admin)

1.7.1.9 UC-FR9 Zobrazení scén libovolného uživatele

Role: Elevated User, Admin

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne zobrazit seznam scén od jednoho konkrétního uživatele.
2. Systém zobrazí seznam scén daného uživatele obsahující pro každou scénu:
   - Název
   - Procento dokončení
   - Odkaz na detail scény

1.7.1.10 UC-FR10 Úprava uživatelské role

Role: Admin

Omezení: Admin nemůže změnit svou vlastní roli.

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne změnit uživatelskou roli nějakého uživatele.
2. Systém zobrazí formulář obsahující možnost výběru nové role pro uživatele.
3. Uživatel vybere novou roli a formulář odešle.
4. Systém požadavek zpracuje.

1.7.1.11 UC-FR11 Zobrazení přehledu uživatelů

Role: Elevated User, Admin

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne zobrazit seznam scén všech uživatelů systému.
2. Systém zobrazí seznam uživatelů, který pro každého uživatele obsahuje:
   - Jméno
   - Odkaz na uživatelský profil

1.7.1.12 UC-FR12 Úprava nastavení render farmy

Role: Elevated User, Admin

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne změnit některá z nastavení render farmy.
2. Systém zobrazí seznam přístupných nastavení s formuláři pro změnu.
3. Uživatel provede a potvrdí změny.

1.7.1.13 UC-FR13 Zobrazení přehledu jednotlivých uzlů render farmy

Role: Elevated User, Admin

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne zobrazit přehled výpočetních uzlů v systému.
2. Systém zobrazí seznam připojených uzlů, který pro každý uzel obsahuje:
   - Název
   - Stav uzlu, který může nabývat jedné z následujících hodnot
     - Neaktivní
     - Zaneprázdněn
     - Jiný

1.7.1.14 UC-FR14 Zobrazení uzlu

Role: Elevated User, Admin

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne zobrazit podrobnosti o výpočetním uzlu.

2. Systém zobrazí detailní pohled na zvolený uzel, který obsahuje:
    - Název
    - Stav
    - Statistiky
    - Seznam dalších atributů
    - Odkaz k úpravě nastavení uzlu

1.7.1.15 UC-FR15 Úprava nastavení uzlu

Role: Elevated User, Admin

Hlavní scénář

1. Případ užití začíná, jestliže se uživatel rozhodne upravit nastavení uzlu.
2. Systém zobrazí formulář obsahující upravitelná nastavení, který obsahuje alespoň:
    - Možnost aktivace a deaktivace uzlu
    - Nastavení časového rozvrhu práce uzlu
3. Uživatel upraví nastavení a akci potvrdí.
4. Systém zpracuje požadavek.

1.7.1.16 UC-FR16 Upozornění na ukončené renderování

Role: User, Elevated User, Admin

Hlavní scénář

1. Případ užití začíná, pokud ve službě dojde k ukončení renderu uživatele (při dokončení nebo selhání).
2. Systém upozorní uživatele o této skutečnosti pomocí e-mailu, SMS, mobilního upozornění nebo jinou cestou. Upozornění obsahuje popis nastalé události a odkaz na scénu, které se událost týká.

Tabulka 1.4: Vazba jednotlivých use cases a požadavků. Čísla v záhlaví tabulky reprezentují jednotlivé funkční požadavky F-FR1 až F-FR11.

	1	2	3	4	5	6	7	8	9	10	11	Jakýkoli
UC-FR1				●								●
UC-FR2				●								●
UC-FR3	●				●							●
UC-FR4	●		●		●					●		●
UC-FR5	●		●									●
UC-FR6	●		●									●
UC-FR7	●		●									●
UC-FR8											●	●
UC-FR9			●									●
UC-FR10											●	●
UC-FR11											●	●
UC-FR12						●		●	●			●
UC-FR13								●				●
UC-FR14								●		●		●
UC-FR15								●	●			●
UC-FR16		●										●
Všechny UC	●	●	●	●	●	●		●	●	●	●	●

1.7.2 Vazba funkčních požadavků případů užití

Z tabulky 1.4 je patrné, že každý případ užití plní nějaký účel s cílem splnit požadavky. Jeden požadavek však nemá případ užití, který by ho řešil. To by teoreticky mohlo naznačit opomenutí. Tímto požadavkem je však *F-FR7* o podpoře scén programu Blender 1.5.1.7. Tento požadavek je spíše na nižší úroveň systému, kterou případy užití nepokrývají.

1.8 Moduly systému

K identifikaci modulů pomůže popis systému obsahující požadovanou funkcionalitu.

FITRender má poskytovat *webovou službu* (Modul Webového rozhraní), která umožňuje *přihlášení školním ID* (Autentizační modul) pro CG *renderování* (Renderovací modul) pomocí *distribuované výpočetní sítě* (Distribuční modul).

Z textu je, jak jsem naznačil, možné identifikovat 4 softwarové celky - moduly, které by ve spolupráci měly být schopny zajistit požadované vlastnosti systému.

Tyto moduly jsou tedy:

- Modul Webového rozhraní
- Autentizační modul
- Renderovací modul
- Distribuční modul

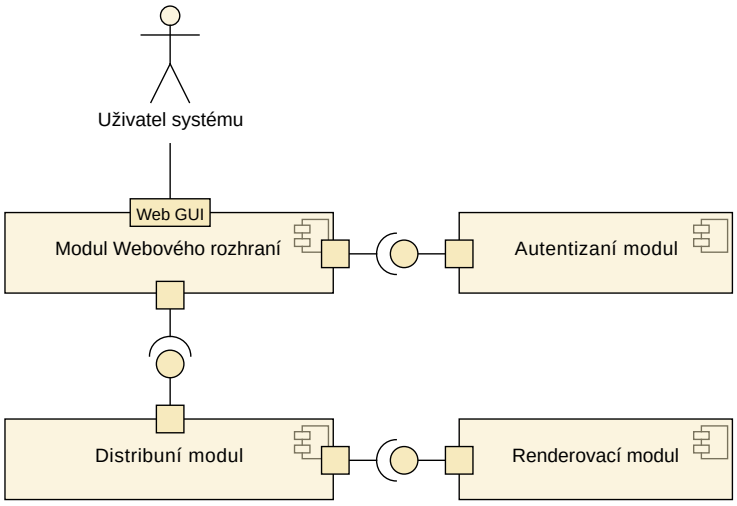
Modul webového rozhraní by měl umožňovat přihlášení uživatele skrze Autentizační modul. Přihlášený uživatel by měl mít možnost odeslat skrze rozhraní úlohu do Distribučního modulu, který ji pomocí Renderovacího modulu zpracuje a pošle zpět k vyzvednutí.

Graficky se tato souvislost dá naznačit pomocí stručného UML diagramu komponent 1.21. Žádné z vazeb by neměly být příliš pevné, aby jednotlivé moduly zůstaly i po kompletaci pouze moduly, které je v případě potřeby možné nahradit. Proto jsem v diagramu zvolil spojení pomocí lízátkové notace. Tuto myšlenku jsem měl na paměti po celou práci na systému.

Modul *Webového rozhraní* má umožnit přihlášení pomocí specifického Autentizačního modulu a komunikaci úloh se specifickým Distribučním modulem. Obé by mělo být v případě potřeby možné vyměnit. Modul Webového rozhraní poskytuje prezentaci služby koncovým uživatelům. Díky tomu se předpokládá potřeba jeho úpravy pro potřeby projektu. Proto se u tohoto modulu neočekává naplnění hotovým řešením, ale samostatná implementace.

Autentizační modul poskytuje standardní webovou službu autentizace Webovému rozhraní, která by dle požadavků měla rozpoznat uživatele pomocí celoškolského přihlášení ČVUT. Vzhledem k potřebě této autentizace nejen školními aplikacemi a faktu, že z principu tento modul potřebuje komunikovat s interní školní databází se předpokládá, že pro tento modul je možné využít již existující řešení. Proto budou tato řešení nalezena, analyzována a jedno vybráno pro prototypovou implementaci systému.

Distribuční modul přijímá zadání of *Webového rozhraní* a rozesílá je na jednotlivé výpočetní uzly, kde se zadání zpracuje pomocí Renderovacího modulu.



Obrázek 1.21: Diagram komponent naznačující jednotlivé moduly systému

Jak již bylo podoktnuto v minulých sekcích 1.1 existuje množství řešení, které nabízí tuto funkcionalitu. Proto budou tato řešení analyzována a nalezeno to, které nejlépe splňuje požadavky systému.

Smyslem *Renderovacího modulu* je přijmout úkol ve formě 3D scény a vrátit jeden či více výsledných snímků. Jedná se tedy buď o kompletní 3D balíček nebo pouze renderer s podporou čtení scén. Do budoucna FITRender počítá dle požadavků s podporou více těchto modulů. Pro prototypovou implementaci je však potřeba zvolit jeden, na kterém se ověří principy fungování celého systému.

1.9 Hledání vhodného distribučního modulu

Kritickou částí služby renderovací farmy je distribuční modul. Tedy součást, která je zodpovědná za distribuci práce na jednotlivé výpočetní uzly.

Je logické, že pro nesporné výhody distribuovaných výpočtů momentálně existuje rozsáhlé množství různorodých systémů pro jeho podporu. To jak obecných, tak i specializovaných na oblast počítačové grafiky. To je samo o sobě dobrou i špatnou zprávou. Je z čeho vybírat a často se jedná o vyzrálé a léty prověřené produkty. Na druhou stranu je potřeba zevrubně zkoumat velké množství nástrojů. Osobně bych však řekl, že pozitivum výběru jasně převažuje.

1.9.1 Zkoumané systémy

Následuje seznam zkoumaných distribučních systémů. Při zkoumání jsem se zaměřoval na schopnost splnění požadavků této části systému. Zaměřil jsem se na *licenci a aktivitu vývoje*, *síťovou architekturu*, *vlastní podporu webového rozhraní*, *kompatibilitu s různými operačními systémy a renderery*, *schopnost reakce na uživatelskou aktivitu*, *schopnosti časového plánování*, *kvalitu dokumentace* a v neposlední řadě snad nejdůležitější schopnost začlenění do většího systému, tzn. *sílu a dostupnost poskytovaného API*.

Pro úplnost jsem zkoumal i několik komerčních řešení, z nichž některá jsou zajímavá pro svoji úplnost - vestavěnou podporu široké škály grafického softwaru, pokročilé statistiky a kompletní cloudové frontendy. Pro svoji licenci jsou však logicky z finálního výběru automaticky diskvalifikována.

Analyzoval jsem komerční CGI specializované produkty *Backburner* a *SquidNet*; Open Source CG specializovaná řešení *DrQueue*, *Puli OpenRenderManagement* a *CGRU Afanasy* a Open Source obecnější systémy pro správu distribuovaných výpočtů *HTCondor* a *Clondike*.

1.9.1.1 Autodesk Backburner

Představení Backburner bude jistě dobře známý pro uživatele softwarových balíčků *Autodesk Maya* a *Autodesk 3DS Max*. Do těchto programů je totiž přímo vestavěná podpora pro odesílání do tohoto systému [77] [78].

Licence a aktivita Jedná se o volně dostupný closed source software vyvíjený firmou Autodesk [79]. To znamená, že v případě potřeby není možné upravit jeho funkcionalitu.

Backburner je dodáván jako základní síťový renderer pro řadu rozšířených produktů firmy Autodesk a tak je možné odhadovat, že jeho vývojová aktivita nezamrzne.

Podpora platforem Existují plně funkční verze pro Windows, Linux i Mac OS [80].

Architektura Systém je klasické architektury klient/server, v jazyce Backburneru Server/Manager.

Vlastnosti Má podporu pro spouštění úloh přímo z Autodesk programů, ale umožňuje i obecnější režim pomocí utility CmdJob [81].

Tato utilita umožňuje odeslání libovolného příkazu na výpočetní cluster. Zároveň vystavuje pokročilé možnosti Backburneru jakými jsou nastavení priority a určení závislostí. Dále je možné při odeslání úkolu specifikovat, na jakém Serveru se má úloha vykonat. Buď volbou konkrétního Serveru nebo zvolením skupiny Serverů H.

API, správa a dokumentace Jak již bylo zmíněno, tak pro odeslání a nastavení nové úlohy je možné využít utility CmdJob. Problémem je však fakt, že spravovat (sledovat stav jednotlivých uzlů, sledovat a modifikovat zadané úlohy, ...) Backburner cluster je možné pouze z výše zmíněných balíčků nebo pomocí GUI utility Backburner Queue Monitor. To v podstatě znemožňuje jeho zařazení do většího systému.

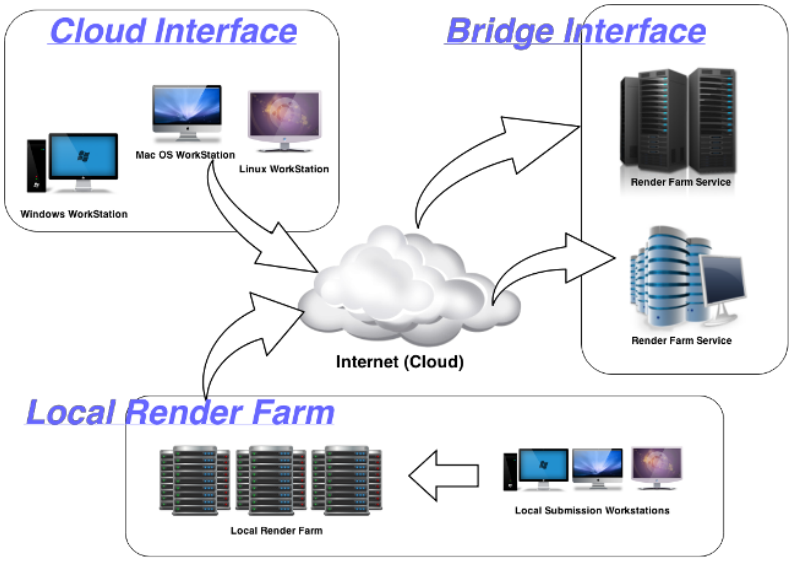
Verdikt Vzhledem k uzavřené licenci a omezeným možnostem napojení do širšího systému je patrné, že Backburner není vhodným kandidátem na výpočetní backend pro systém FITRender.

1.9.1.2 SquidNet

Představení SquidNet je komerčním řešením pokročilého správce render farmy. Možná by se dalo spíše říci, že se jedná o švýcarský nožík renderovacích farem.

Licence a aktivita SquidNet je možné používat bezplatně pro render farmy do velikosti 8 počítačů. Podpora každého dalšího ale stojí 15 amerických dolarů. Případně je možná hromadná licence dostupná za cenu 1200 dolarů za roční a 2400 dolarů za licenci neomezenou. Program má také uzavřený zdrojový kód, takže není možné analyzovat kód z ohledu bezpečnosti a nebo ho upravit v případě potřeby [82].

Podpora platforem Podporovány jsou všchny primární PC platformy a tedy Linux, Windows a Mac OS X [83].



Obrázek 1.22: Možné konfigurace systému SquidNet [85]

Architektura Lokální render farma funguje v režimu klient/server. Je však možné jednotlivé výpočetní uzly nastavit do jednoho ze čtyř režimů. *Peer* - nekonfigurovaný uzel, *Master* - správce aktivity farmy/server, *Slave* - uzel určený pouze pro výpočty a *Client* - počítač určený pro zadávání úkolů, který může dodatečně také renderovat [84]. Již zmíněná možnost připojení k výpočetnímu cloudu dále navyšuje potenciální komplexitu architektury.

Vlastnosti SquidNet samozřejmě poskytuje základní službu správce render farem - automatizaci distribuce renderovacích úkolů na výpočení cluster v lokální síti. Kromě toho však nabízí rozšíření vlastního clusteru o cluster v cloudu transparentně koncovému uživateli. Dále v betaverzi nabízí pokročilý cloudový interface, který lze používat interně, ale je jím i možno proměnit svou lokální render farmu v cloudovou renderovací službu pro veřejnost [85]. V tomto ohledu se jedná o kompletní řešení, které v podstatě působí jako přímá konkurence FITRenderu z komerční sféry.

Celý balíček je hluboboce propracovaný. Nabízí velmi dobrou škálovatelnost. Dynamický load balancing, který v reálném čase reaguje na aktuální zatížení procesorů jednotlivých uzlů. Sběr statistik. Šablony pro úkoly z 24 používaných CG aplikací. Kromě používání skrze CLI nabízí i modulární GUI a již zmíněný cloudový interface. Oznámení pomocí e-mailu, ale i SMS. Správu priorit, závislostí, překlad cest a další. Prostě má všechno a používá ho celá řada firem včetně například cloudové renderovací služby Renderfarm.nu a třeba automobilky Volkswagen [7].

API, správa a dokumentace Pokud by se SquidNet měl použít pouze jako backend a být ovládán z cloudového interfacu vytvořeného přímo pro FITRender, tak by bylo možné k ovládání SquidNetu použít CLI rozhrání [84]. Veškeré používání programu je dobře zdokumentováno v několika uživatelských příručkách [86].

Verdikt V celkovém ohledu se jedná o velmi zajímavé a pokročilé řešení. Vzhledem k proprietární licenci spojené s licenčními poplatky je však tato možnost zavržena.

1.9.1.3 DrQueue

Představení OpenSource projekt DrQueue má poměrně dlouhou historii. Od svého vzniku v roce 2000 [87] prošel jedním kompletním přepisem z C na systém pro interaktivní sdílené výpočty [88] *IPython* v roce 2011 [89]. Používal se na dobře známých projektech, jakými jsou například *Piráti z Karibiku - Truhla mrtvého muže* (2006), či první ze série Open Movie Projects *Elephant's Dream* (2007) [90].

Licence a aktivita Projekt je vyvíjen pod *GPL licencí* a je dostupný buď na GitHub repozitáři [91] a nebo skrze vlastní Redmine. Přepsaná Momentálně je však projekt velice neaktivní, poslední commit do veřejného repozitáře se v době psaní textu datuje více než rok zpátky.

Podpora platforem Díky svému přepisu do čistého jazyku Python 2.7 podporuje širokou škálu platforem, které samozřejěm obsahují Linux, Windows i Mac OS X. Je závislý na modulu *ZMQ*, souborovém systému *SSHFS* a databazovém serveru *MongoDB* [92].

Architektura Aktuální verze využívá architektury Klient/Server, v DrQueue nazvány *Master/Slave* [93]. Pro komunikaci je v Pythonové implementaci použit protokol SSH a souborový systém SSHFS pro sdílení souborů.

Vlastnosti Výhoda CG zaměřených správců výpočetního clusteru tkví hlavně ve vestavěné podpoře mnoha CG softwaru. V případě DrQueue tomu není jinak a aktuálně podporuje generování úloh pro více jak 15 různých programů mezi které patří například *Blender*, *3DS Max*, *Maya*, *V-Ray* a *Cinema 4D* [94].

Kromě toho DrQueue umožňuje dělit clusteru do různých poolů a při zadání úkolu specifikovat pool či hardwarové a platformní požadavky [95]. Nepodporuje však například závislosti mezi úkoly, časové rozvrhování nebo reakce na uživatelskou aktivitu výpočetního stroje.

Zajímavostí je, že v rámci DrQueue vznikal i projekt cloudového rozhrání, které v některých bodech odpovídalo cílům této práce. Proto by se mohlo zdát,

že by bylo vhodné tento projekt využít jako základ implementace této práce. Původně bylo rozhraní implementováno pro C verzi za použití sady vazeb pro Ruby *DrQueueOnRails*. Po převedení projektu do Pythonu byla vytvořena snaha o převod. Ten však není kompletní a poslední aktivita v tomto projektu byla v roce 2012 [96]. Zároveň je DrQueueOnRails postaven na zastaralé verzi 3 frameworku Rails a Ruby 1.8. Mimo jíné silné propojení s jedním konkrétním výpočetním backendem jde proti filozofii této práce.

API, správa a dokumentace Jako API DrQueue je možné použít jeho CLI utilitu drqueue, která umožňuje nejen zadávání a správu jednotlivých úkolů, ale i správu jednotlivých počítačů v clusteru a bezpečnostní nastavení [95].

Dokumentace systému je na poměrně dobré úrovni. Sice není příliš obsáhlá, ale stejně tak není obsáhlý samotný systém. Na Wiki projektu jsou jasně popsány veškeré důležité oblasti - požadavky, instalace, spuštění, práce s programem pomocí CLI a GUI projektu DrQt [97].

Verdikt DrQueue je poměrně sympatickým a přímočarým správcem render farmy s univerzální CLI utilitou, kterou je možné použít jak pro akce ohledně úkolů tak i správu jednotlivých uzlů. Na druhou stranu je možná až příliš základním řešením.

Chybějící podpora závislostí, časového rozvrhování a pokročilejšího plánování by při použití v cloudové renderovací službě pravděpodobně brzo způsobila mnohé komplikace. Dalším negativem je také nízka aktivita celého projektu.

Z těchto důvodů DrQueue vyřazuji z výběru.

1.9.1.4 Puli (OpenRenderManagement)

Představení Puli je render manager vytvořen pro pomoc s prací na vizuálních efektech a post produkcí [98]. Jedná se o minimální sadu nástrojů potřebných pro distribuci výpočetních úloh napříc sítí.

Licence a aktivita Puli je licencován pod takzvanou *Modified BSD* licencí [99]. Vývoj programu je aktivní [100].

Podpora platforem Veškeré součásti Puli jsou napsány v jazyce *Python* s využitím frameworku *Tornado* [101]. Díky tomu má poměrně malé systémové závislosti. S využitím virtuálního prostředí pro Python Virtualenv se dá dokonce Puli nainstalovat bez root přístupu. Bohužel i přes to, že je postaven na multiplatformních technologiích, je aktuálně pouze pro Linuxové operační systémy [102]. To je způsobeno například použitím PID zámků.

Architektura Puli je klasické klient/server architektury - v Puli nazvány *Worker*/*Dispatcher* [102].

Vlastnosti Co se týče pokročilých vlastností, tak Puli podporuje definování závislostí mezi jednotlivými úlohami, sbírání statistik, propojení s cloudovým výpočetním řešením, správu priorit a možnost lokálně ručně deaktivovat výpočet [103]. Tím však seznam končí a například pro určení časového rozvrhu výpočtů by bylo nutné použít externí nástroj jako například *cron* [104]. Reakci na uživatelskou aktivitu by bylo nutné řešit podobným způsobem.

Při testování se také naskytly problémy s detekcí nově připojených výpočetních uzlů. Na Github stránce projektu je však dostupný patch [105], který tento problém odstraní, takže je možné očekávat, že se tato oprava brzy dostane i do hlavního repozitáře.

API, správa a dokumentace V systému jsem neobjevil možnost pokročilých vlastností plánování jako je možnost časového rozvrhování či reakce na uživatelskou aktivitu. Dokumentace k programu je na velmi minimální úrovni.

Verdikt Puli je sice pěkným přímočarým řešením s jednoduchou instalací. Jeho nespolehlivost, nedostatek pokročilých plánovacích schopností a špatná dokumentace ho však z výběru diskvalifikují.

1.9.1.5 CGRU Afanasy

Představení Afanasy je komponenta správy render farmy balíčku CGRU - CG Rules. Rules v názvu má dvojí význam - jako označení sady principů a ve smyslu, že počítačová grafika je skvělá věc. Rules je také název další komponenty pro sledování a kontrolování CG projektů. Dalšími komponentami tohoto balíčku jsou *Movie Maker*, nástroj pro automatizaci konverze obrazových sekvencí do videa integrovaný s Afanasy a *Keeper*, který nabízí přehled o službách CGRU a přístup k jejich funkcím [106]. Afanasy je momentálně bráno jako hlavní součást celého balíčku a ostatní komponenty jsou určeny pro jeho podporu [107].

Podle oficiální stránky CGRU byl balíček použit či se používá alespoň v 15 studiích z Ukrajiny, Ruska, Švédska, Francie a dalších [107]. Zároveň byl použit ve více než 30 celovečerních filmech, z kterých je pro naší kotlinu možná povědomý *Step Up Revolution* [107]. Jedná se tedy o produkčně prověřený software. Několikaletá průběžná aktivita na GitHub repozitáři také napovídá tomu, že se autor snaží udržovat projekt relevantní [108].

Licence a aktivita Projekt CGRU a jeho zdrojový kód jsou distribuovány pod známou *GNU Lesser General Public License* neboli *LGPL licencí*, která umožňuje použití v otevřených i proprietárních aplikacích, volnou distribuci a úpravu při zachování autorství a LGPL či GPL licence [109] [110].

Podpora platforem Kompletní CGRU je možné spustit na platformách Linux, Windows i Mac OS X. Kompilované birnární balíčky jsou k dispozici pro

Windows a Linuxové distribuce AltLinux, CentOS, RedHat Enterprise Linux, Debian, Fedora, Mageia, Open SUSE a Ubuntu [111]. Pro ostatní distribuce a Mac OS X je nutná kompilace ze zdrojového kódu, kterou může zkomplikovat závislost na knihovnách *SIP*, *PySide* a *PyQT 4*.

Architektura Afanasy využívá klasické architektury jednoho spravovacího serveru skrze který jsou zadávány úlohy pro připojené výpočetní uzly. V Afanasy jsou tyto role určeny spuštěním aplikací *Server* [112] a *Render Host* [113].

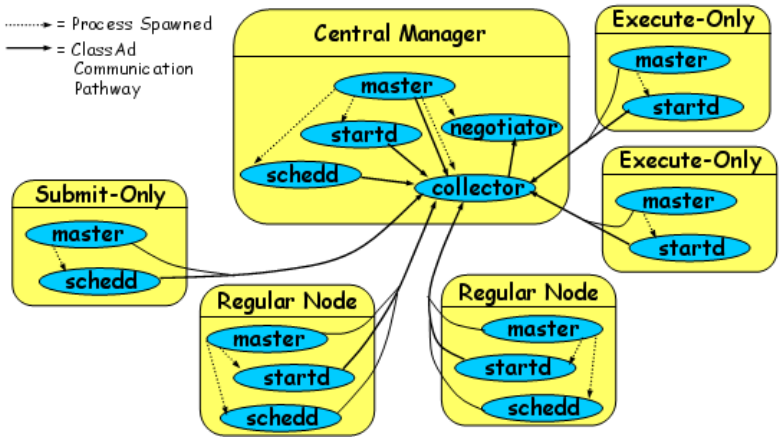
Vlastnosti CGRU nabízí poměrně širokou škálu pokročilých funkcí a nastavení pro distribuované výpočty. Patří mezi ně určení priorit pro jednotlivé úkoly, stanovení kapacit uzlů, závislostí v rámci jednoho úkolu či mezi několika samostatnými úkoly, překlad cest mezi operačními systémy a další. Dále jsou součástí balíčku integrační pluginy do programů *Adobe After Effects*, *Blender*, *Cinema 4D*, *Houdini*, *3DS Max*, *Maya*, *Nuke* a *XSI* [106]. Tyto pluginy umožňují zadávat úkoly do lokální render farmy bez opuštění daného programu. Kromě toho CGRU nabízí i samostatný grafický program a webovou aplikaci pro zadávání úkolů, sledování jejich stav a vyzvedávání výsledků. Afanasy samo o sobě však nenabízí podporu časového rozvrhování úloh či reagování na uživatelskou aktivitu počítače.

API, správa a dokumentace Pro komunikaci s Afanasy serverem je možné využít *JSON API*. Konkrétní JSON struktury je možné vytvářet manuálně a odeslat pomocí utility *afcmd* s parametrem json, využít Afanasy Python modulu nebo Pythonové CLI utility *afjob.py*. Toto JSON rozhraní je zaměřeno na práci s úlohami, tedy zadávání, sledování stavu, úpravu a smazání [114]. Momentálně neumožňuje například sledovat stav jednotlivých výpočetních uzlů či spravovat jejich nastavení, pro to je však možné použít další funkce CLI utility afcmd.

Verdikt CGRU je zajímavý tím, že se z open source možností jedná asi o nejkompletnější balíček. Proto by bylo dobré tento systém nasadit například v grafickém studiu. Vzhledem k omezené možnosti pokročilého plánování, které je vyžadováno ve školním prostředí však tento systém nevybírám.

1.9.1.6 HTCondor

Představení HTCondor, neboli High Throughput Condor je dlouhodobým projektem Center for High Throughput Computing v University of Wisconsin-Madison [115], který započal roku 1988 [116]. To je umožněno kromě propracované architektury a celé škály pokročilých vlastnostní i flexibilním mechanismem určování priorit a požadavků pro jednotlivé úlohy a výpočetní uzly *ClassAds* [117][118]



Obrázek 1.23: Možnosti nastavení uzlů při využití systému HTCondor [121]

Licence a aktivita Projekt HTCondor je licencován pod benevolentní open source *Apache License, Verze 2.0*, která mimo jiné umožňuje volně šířit zdrojový kód při zachování licence, přidávat úpravy s uvedením původce a pro upravené části je dokonce možné si stanovit vlastní licenční podmínky [119].

Podpora platforem Existují verze pro všechny tři hlavní platformy, tedy Linux, Windows a Mac OSX. Binární balíčky jsou z oficiální stránky dostupné pro Windows, Mac OSX a Linuxové distribuce Debian, Ubuntu, RedHat Enterprise Linux. Pro ostatní distribuce Linuxu je třeba vlastní kompilace ze zdrojového kódu, která kromě standardně dostupných závislostí vyžaduje instalaci *Globus Toolkit*. Verze pro různé platformy mají mezi sebou mírné rozdíly [120].

Architektura Architektura systému HTCondor je ve své podstatě také server/klient avšak nabízí větší granularitu rozdělení úloh díky konfigurovatelných daemonům. Různou kombinací spuštěných daemonů *startd* (spouštění úloh), *schedd* (zadávání úloh), *collector* (sbírání systémových informací) a *negotiator* (rozdělování úloh) je možné určit zda úloha daného počítače je výpočet, zadávání nových úloh do systému, správa systému či libovolná kombinace [121]. V rámci clusteru je dokonce možné mít více serverů, které využívají stejné výpočetní prostředky či cluster propojit s nějakým grid řešením [122].

Vlastnosti HTCondor má mnoho pokročilých vlastností. Umožňuje propojení nejen k dalším HTCondor výpočetním clusterům ale i několika jiným typům [122]. Nabízí vysoce roubustní implementaci závislostí pomocí své utility *DAGMan* [123][124]. Umožňuje dočasné pozastavení úloh, například při uživatelské aktivitě na počítači. Pro úlohy využívající HTCondor knihovnu je

možná i hibernace na disk [125]. Je podporován výpočet ve virtuálním prostředí [126] a mnoho dalších.

Unikátní vlastnost systému však tvoří v představení zmíněné ClassAds - *Classified Advertisements*. Tento funkcionální dynamický jazyk je v podstatě univerzálním komunikačním jazykem systému HTCondor [118]. Jeho proměnné mohou být naplněny klasickými literály včetně řetězců, seznamy, specifickým agregačním typem suplujícím objekty classads a funkcemi včetně odkazů na jiné proměnné nebo vestavěné funkce. Kromě toho ClassAds podporují i jmenné prostory a dynamické typování [127][117].

Veškeré atributy výpočetních uzlů jsou sdíleny právě pomocí ClassAds. V základu například aktuální stav, uživatelská aktivita, systémový čas, benchmarkové hodnocení a mnoho dalších. Samozřejmě je možné přidat vlastní - pro FITRender by například mělo smysl přidat informace o instalovaném GPU.

U každé úlohy respektive uzlu je poté možné určit požadavky na uzly respektive úlohy a jejich hodnocení pomocí hodnot *Requirements* a *Rank* [115]. To například umožňuje říci, že úloha se spustí pouze na uzlech bez uživatelské aktivity v době přestávek a přednost budou mít uzly s výkonnějšími GPU kartami. Dále je možné hodnoty ClassAds použít i v argumentech zadání a tím úkol lépe přizpůsobit konkrétnímu výpočetnímu uzlu. HTCondor také podporuje takzvané dynamické sloty, tedy spuštění počtu úloh na počítači podle požadovaných jader [128].

API, správa a dokumentace Jako API HTCondoru lze použít jeho sadu CLI utilit. Mezi ně patří například *condor_submit* - zadání úlohy, *condor_Hstatus* - aktuální stav uzlů, *condor_Hq* - fronta úloh a mnohé další [129][130]. Dále HTCondor umožňuje spustit webserver obsluhující *SOAP* požadavky [131].

Dokumentace systému je na vysoké úrovní. Je verzována a dopodrobna popisuje například různé postupy instalace [132], ale samozřejmě i jednotlivé vlastnosti systému.

Verdikt Konfigurační síla systému ClassAds dělá HTCondor ideální pro heterogenní prostředí s množstvím nededikovaných počítačů. Umožňuje využít veškeré dostupné prostředky a přitom do velké míry neomezovat jejich používání v případě potřeby.

Správa render farmy možná není primárním účelem systému HTCondor, ale i přesto existuje množství dokumentovaných případů, kdy byl pro tento účel použit. Například byl využit jako výpočetní backend pro projekt digitální archivace kulturního dědictví na *University of Sussex* [133]. C.O.R.E Feature Animation použilo HTCondor při práci na filmu *The Wild* pro Disney [134].

1.9.1.7 Clondike

Obrázek 1.24: Snímek z animovaného filmu The Wild, který používal pro distribuci snímků HTCondor. © Buena Vista International

Představení Clondike, neboli *CLuster Of Non-Dedicated Interoperating KErnels*, představuje alternativní přístup k distribuci výpočetních úkolů napříč heterogenním clusterem nededikovaných počítačů tím, že staví na *Peer to Peer principech* a bezpečné izolaci výpočtu do virtuálního stroje [135]. *Je vyvíjen přímo na půdě ČVUT.*

Licence a aktivita Verze Clondike dostupná na serveru GitHub je distribuována pod licencí *GPL*. Aktivita na tomto repozitáři je poměrně malá [136]. Práce na systému však probíhá spíše formou jednotlivých závěrečných prací [137][138][139], které systém posouvají dopředu. Z posledních je možné zmínit práci z roku 2014 tehdy Bc. Pavla Tvrdíka [140] o jejíž přínosech více v sekci architektura.

Podpora platforem Jak již napovídá kompletní název projektu, tak Clondike umisťuje svoji základní funkcionalitu do samotného Linuxového kernelu pomocí patche. To samozřejmě rozšiřuje technické možnosti a zvyšuje efektivitu řešení. Na druhou stranu to však projekt silně váže právě s operačními systémy, které využívají Linuxového jádra. Původní verze pracovala s Gentoo distribucí [137] a aktuální referenční image využívá distribuce Debian [141].

Architektura Zajímavostí Clondike je právě jeho architektura. Zatímco ostatní zkoumané systémy pracují s různými variantami architektury Klient/Server, tak Clondike využívá Peer to Peer architektury. Aktuálně systém využívá implementace založené na *P2P DHT systému Kademlia* [140].

Zadávání úkolů a hledání výpočetních prostředků tak funguje bez centrálního serveru. Tím je systém odolnější proti výpadkům a umožňuje velmi dynamickou škálovatelnost a spolehlivé zadávání úkolů odkudkoli v síti počítačů se spuštěným systémem Clondike.

Pro cloudovou službu, která k výpočetnímu backendu přistupuje skrze jeden bod jsou však výhody tohoto systému pouze omezené.

API, správa a dokumentace V dokumentech, které se věnují systému se zmiňuje implementace submit rozhraní v jazyce Ruby. Nanalezl jsem však podrobnosti o schopnostech ani používání tohoto rozhraní.

Verdikt Clondike je jistě zajímavým projektem s podporou přímo od naší školy. P2P přístupy umožňují dynamickou škálovatelnost a jeho důraz na bezpečnost a vhodné využití momentálně dostupné prostředky je ve školním přostředí silně vítano.

Pravděpodobně pro svoji komplexnost a přerušovaný vývoj je však projekt i přes svůj věk alespoň 10 let[1] stále v uživatelsky poměrně nepřívětivém stavu, zejména pro nedostatek dokumentace. Zároveň silná vazba na platformu Linux je problémem pro render farmu, která by pracovala se softwarem, který Linux nepodporuje.

Z těchto důvodů není Clondike vhodným kandidátem pro výpočetní backend této prototypové implementace služby FITRender. Bude však jistě zajímavé sledovat vývoj projektu nadále a pokusit se snahy propojit.

[1] První zmínky o systému jsou z roku 2005. [142]

1.9.2 Zhodnocení

Tabulka 1.5 prezentuje zkoumané vlastnosti v přehledné a kompaktní formě.

Jednotlivé zkratky znamenají:

- **L** - Licence
- **AV** - Aktivně vyvíjeno
- **A** - Architektura
- **WR** - Webové rozhraní
- **MP** - Multiplatformní
- **RUA** - Reaguje na uživatelskou aktivitu
- **ČP** - Umožňuje časové plánování
- **API** - Možnosti komunikace se systémem
- **SU** - API umožňuje správu jednotlivých uzlů
- **KD** - Kvalitní dokumentace
- **P** - Proprietární
- **K/S** - Klient/Server
- **P2P** - Peer to Peer

Tabulka 1.5: Přehled kandidátů na Distribuční modul pro systém FITRender

Systém	L	AV	A	WR	MP	RUA	ČP	API	SU	KD
SquidNet	GPL	+	K/S[1,2]	++	+	+	?	CLI	+	+
DrQueue	GPL	-	K/S	+	+	-	-	CLI	+	+
Puli	MBSD	+	K/S	-	-	-	-	CLI	+	-
CGRU Afanasy	LGPL	+	K/S	+	+	-	-	CLI JSON	+	+
HTCondor	ALv2.0	+	K/S[2,3]	-	+[4]	+	+	CLI SOAP	+	+
Clondike	GPL	+	P2P	?	-	+	?	?	?	-

Jako výpočetní backend projektu FITRender jsem zvolil systém *HTCondor*. Vzhledem k tomu, že poskytuje vysoce kvalitní dokumentaci a pokročilé vlastnosti, které jsou vhodné pro cílené školní prostředí. Instalace HTCondor na školních počítačích zároveň nabídné výpočetní platformu pro širokou škálu úkolů mimo potřeby render farmy.

Jako případnou alternativu pro kontrolovanější prostředí bych volil *CGRU Afanasy* vzhledem k dobré integraci se širokou škálou CGI programů.

[1] S možným propojením cloudových služeb.
[2] S vyšší granularitou.
[3] S možným propojením do jiných výpočetních systémů.
[4] Podpora platforem není zcela rovnocenná.

Za opakovanou zmínku stojí i systém *Clondike*. Předpokládám, že cílem školy je na systému dále pracovat a v budoucnu ho nasadit jako standardního správce distribuovaných výpočtu na dostupných výpočetních prostředcích. V takovém případě by samozřejmě bylo vhodné Clondike integrovat do systému FITRender.

1.10 Hledání vhodného renderovacího modulu

Renderovací modul pro prototypovou implementaci by měl umožnit odzkoušení principů celého systému, do nichž spadá podpora GPU renderování. Vhodný modul by neměl systém v počátku zatížit placenou licencí. V ideálním případě by tento modul měl také pokrýt alespoň z části pokrýt potřeby koncových uživatelů, tedy studentů architektury. Při jeho výběru je tedy možné se odkázat zpět na provedený průzkum mezi nimi. Zároveň by tento renderer měl podporovat Linux, který bude prvním podporovaných systémem prototypové implementace.

Vzhledem k poměrně specifickým požadavkům je možné k výběru přistoupit poměrně přímočaře.

Vzhledem ke snaze jít naproti koncovým uživatelům do výběru neřadím kvalitní a volně dostupné renderery, jakými je například *LuxRender* [143] či *YafaRay* [144], vzhledem k tomu, že nemají žádné zastoupení mezi studenty architektury dle výsledků prezentovaných v sekci 1.3.1.

Tabulka 1.6 nabízí seznam studenty nejpoužívanějších řešení včetně přehledu splnění jednotlivých požadavků.

Tabulka 1.6: Přehled vlastností rendererů používaných mezi studenty architektury

Renderer	Volná licence	Podpora GPU	Podpora Linuxu	Zastoupení
V-Ray	-[145]	-	+[63]	67%
Mental Ray	-[146]	-[1]	+[147]	17%
Cycles	+[148]	+[149]	+[150]	13%
V-Ray RT	-[145]	+[52]	+[63]	12%
PRE	-	-	-[75][76]	12%
Artlantis	-[151]	-	-	10%

Z tabulky je patrné, že jediným vhodným kandidátem je vestavěný renderer programu Blender - *Cycles*.

Blender je open source konkurencí velkých hráčů mezi 3D renderery, jakými jsou například 3DS Max, Maya, Cinema 4D a další. Byl použit při produkci mnoha animovaných filmů [153] a je stále aktivně vyvíjen [154].

Pro jeho zapojení do celého systému je možné použít CLI rozhraní [155].

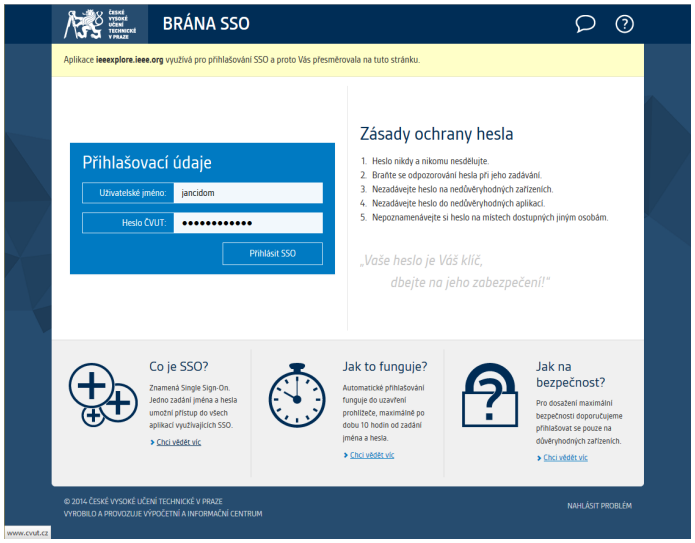
[1] Podpora GPU je ve vývojovém a testovacím stadiu.[152]

1.11 Hledání vhodného autentizačního modulu

ČVUT nabízí dvě formy jednotného přihlášení, *Single Sign-On* neboli SSO, které je možné použít pro autentizaci ve webové aplikaci.

Přihlášení pomocí SSO umožňuje autentizaci uživatele v rámci aplikace skrze poskytovatele autentizace (v našem případě ČVUT) bez sdílení samotných přihlašovacích informací s danou aplikací [156]. Propojením jedné identity s více aplikacemi se však přihlašovací údaje stávají mocnějšími a je proto nutné dbát zvyšené opatrnosti při práci s nimi. Jednotné přihlášení však na druhou stranu snižuje potřebu pamatovat si různá hesla a umožňuje automatizovat registraci v aplikacích a v mnohých případech může být i bezpečnější cestou oproti klasickému přilášení [157][158].

1.11.1 Shibboleth



Obrázek 1.25: Přihlašovací formulář školního Shibboleth systému

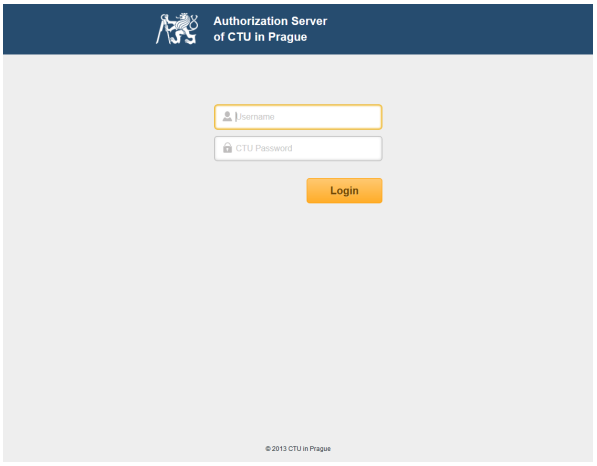
První je systém Shibboleth, ve federačním režimu [159]. Tento režim nabízí propojení institucí, které se nachází ve stejné federaci [160]. ČVUT je součástí dvou federací a to *cvutID* [161] a širší *eduID.cz*, kterou je propojeno se zbytkem vysokoškolských a výzkumných institucí v České republice [162]. Dále je možné Shibboleth SSO použít například pro autentizaci na elektronické knihovny vědeckých prací IEEE Xplore [163].

Vzhledem k propojení s akademickou sférou není možné jednoduše využít toto přihlášení ve vlastní aplikaci.

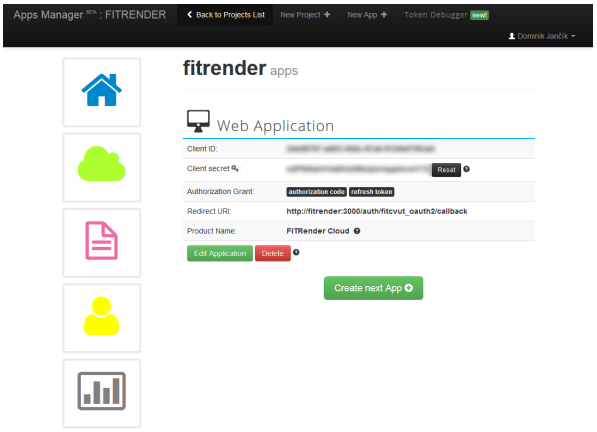
1.11.2 OAuth 2.0

Jednoduchost využití řeší projekt fakultního OAuth 2.0 autorizačního serveru *auth.fit.cvut.cz* [164], který umožňuje samostatně vytvořit novou aplikaci využívající této autentizace pomocí přehledné aplikace *AppsManager* 1.26b. Zde je také možné spravovat OAuth 2.0 klíče [165].

K samotné komunikaci mezi autentizačním serverem a klientskou aplikací se používá rozšířený autentizační framework OAuth 2.0, který využívá například *Google* [166], *SoundCloud* [167] a další.



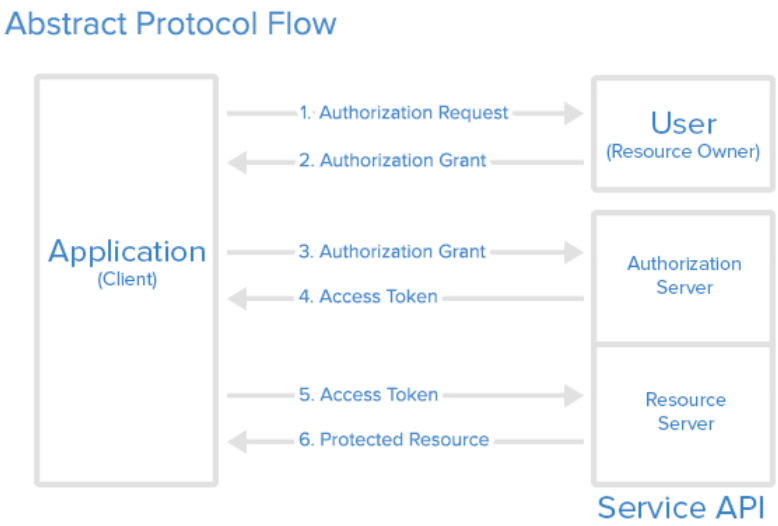
(a) Přihlašovací formulář fakultního OAuth 2.0 serveru



(b) Správce aplikací fakultního OAuth 2.0 serveru, AppsManager

Obrázek 1.26: Screenshoty z fakultního OAuth 2.0 systému

OAuth 2.0 nabízí několik postupů pro autentizaci uživatele [168]. Pro serverou aplikací, jakou je systém FITRender, je však vhodné použít takzvaný



Obrázek 1.27: Průběh autorizace pomocí OAuth 2.0 [170]

Authorization Code Grant, který zpracuje přihlášení pomocí několika přesměrování. V rámci tohoto postupu uživatel nejprve požádá aplikaci o autorizaci, která požadavek zašle na autorizační server a uživatele přesměruje na daný autorizační server. Zde uživatel potvrdí autorizaci pro danou aplikaci pomocí svých přihlašovacích údajů. Na základě toho autorizační server vrátí aplikaci autorizační kód, který aplikace využije pro získání přístupového tokenu, který se využívá pro ověření jednotlivých požadavků vůči dalším API [169]. Přesný průběh je naznačen v diagramu 1.27

Použitím přihlášení pomocí fakultního OAuth 2.0 serveru může navíc aplikace získat přístup k praktickým fakultním službám jako *KOSapi*, *VVVSapi* či *Usermap API* [164], které nabízí přístup k agregovaným datům uživatele z několika služeb [171].

1.12 Technologie pro dodatečnou implementaci

Pro Modul autentizace, Renderovací modul a Distribuční modul jsou v tuto chvíli již vybrána vhodná hotová řešení. Jednotlivé moduly je však nutné propojit a nabídnout jejich funkcionalitu koncovým uživatelům ve formě webové služby.

Tuto službu je nutné navrhnout a implementovat. Pro návrh i implementaci je důležitý vhodný výběr stavebních kamenů, tedy technologií, které implementace využije.

Prvně je třeba rozhodnout, jaké programovací jazyky budou pro stavbu aplikace využity. Pro stavbu webových aplikací se používají zejména *PHP*, *Python*, *Ruby*, *Java*, *.NET jazyky* [172] a *JavaScript* [173].

Je sice možné s těmito jazyky aplikace stavět i na zelené louce, ale pro urychlení vývoje, zajištění bezpečnosti a sdílení kódu a postupů se většinou používají tzv. frameworky [174]. Každý z uvedených jazyků nabízí alespoň jeden takový framework. Pro PHP tím jsou například populární framework *Laravel* [175] [176] či české *Nette* [177]. Pro Python je nejpoužívanější *Django*. Ruby má zejména *Rails* a *Sinatra*. Java má *Spring* či třeba *Play*. Pro C# a jiné .NET jazyky existuje *ASP.NET* [178]. Pro JavaScript patří mezi nejvýznamnější *AngularJS* a *Meteor*. [179]

Po výběru jazyka a frameworku přichází na řadu různé podpůrné technologie a zásuvné moduly...

Jak si z této přehršle možností vybrat? Je třeba vzít v potaz to, kdo bude na aplikaci pracovat. A poté se zaměřit na to, co od technologie chceme.

Odpověď na první otázku pro tuto chvíli je jasná: Já. Ale při širším nasazení projektu je možné počítat s jeho udržováním i jinými stranami. Z toho vychází odpověď na otázku druhou - co od technologie chceme.

Technologie, kterou je třeba vybrat je tedy taková, se kterou mám dobré zkušenosti a zároveň nabízí jednotné konvence psaní kódu. Taková která nabízí možnost jednotkových testů funkčnosti, které poskytují větší jistotu při úpravach. Zároveň je při TDD pravděpodobné, že je výsledný kód méně provázaný, což vede k vyší udržitelnosti projektu. [180]

Mezi webové technologie, se kterými jsem měl tu čest pracovat patří Java EE s frameworkem Spring, PHP s frameworky Symfony a Nette a Ruby s frameworkem Ruby on Rails a Sinatra. Největší zkušenosti mám pak s Nette a Rails v nichž jsem pracoval na několika webových aplikacích, jako je například webové stránky festivalu *Winter Arena* (Nette) [181] či webové stránky organizace *BASS DROP* (Rails) [182].

Z odzkoušených technologií stanovené požadavky splňuje lépe Ruby s frameworkem *Ruby on Rails*.

Vzhledem k tomu, že Ruby on Rails je jasně nejpopulárnější framework v rámci jazyka a druhý nejpopulárnější framework globálně [179], je pravděpodobné, že se v budoucnu naleznou programátoři, kteří budou mít s jazykem a frameworkem zkušenost.

Ruby má vestavěnou podporu pro jednotkové testování ve formě Test::Unit a MiniTest::Unit a s nástroji jako RSpec i velmi příjemnou syntaxi celého testování [183]. Ve frameworku Ruby on Rails je tím pádem taky možné testovat bez větších potíží [184].

Množství volně dostupných rozšíření ve formě takzvaných *Gemů* nabízí podporu široké škály databázových strojů [185], nástroje pro využití různých forem autentizace [186], zpracování nahraných souborů [187] a dalších.

Z těchto důvodů jsem pro prototypovou implementaci zvolil jazyk *Ruby*. Co se týče frameworků, tak se kromě *Ruby on Rails* nabízí i minimalističtejší framework *Sinatra*. Sinatra umožňuje velmi rychle specifikovat zpracování REST požadavků [188], což je vhodné například při implementaci jednoduchých REST služeb.

Kapitola 2

Návrh

2.1 Návrh architektury

Počítačová infrastruktura systému FITRender není stavěna na zelené louce. Proto nenavrhuji jednu konkrétní architekturu potřebnou pro běh systému. Spíše jsem vytvořil seznam modelových konfigurací, které by systém měl být schopen ve své finální podobě podporovat. Pro stručnost návrhy neobsahují síťové bezpečnostní prvky a další nutné součásti infrastruktury, ale samozřejmě se s jejich umístěním v síti počítá.

Dále se věnuji zejména dvěma architekturám *A1* a *A2*, které kladou různé nároky na komunikační strukturu mezi web serverem a výpočetním clusterem. Další možné varianty, které pracují například bez Souborového uložiště jsou k dispozici v příloze E.

2.1.1 A1 - Základní architektura

Tato modelová architektura je v podstatě rozšířením klasické architektury renderovací farmy o Webový server poskytující uživatelům samotnou cloudovou službu.

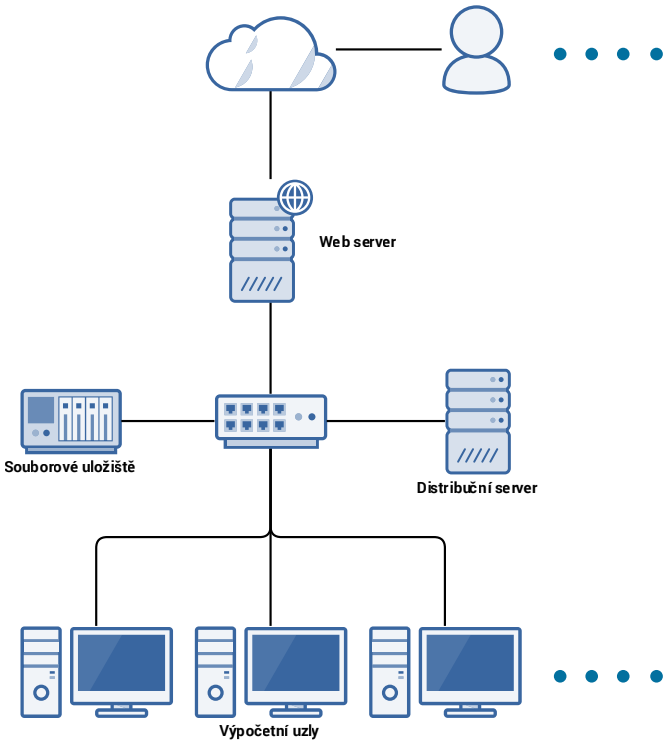
Hlavními prvky jsou již zmíněný *Web server*, *Souborové uložiště* používané pro sdílení souborů scén a výsledků, *Distribuční server*, na kterém je spuštěna serverová část distribučního modulu a samotné *Výpočetní uzly*, na kterých je spuštěna klientská část distribučního modulu.

U Web serveru se předpokládá, že obsahuje vlastní databázi. Dále se předpokládá, že Web server vystavuje buď přímo část Souborového uložiště nebo vlastního řešení pro sdílení koncových řešení s uživateli. Mezi vlastní řešení může patřit buď lokální uložiště nebo například umístění výsledků do cloudového uložiště, jako je například *Amazon S3* [189].

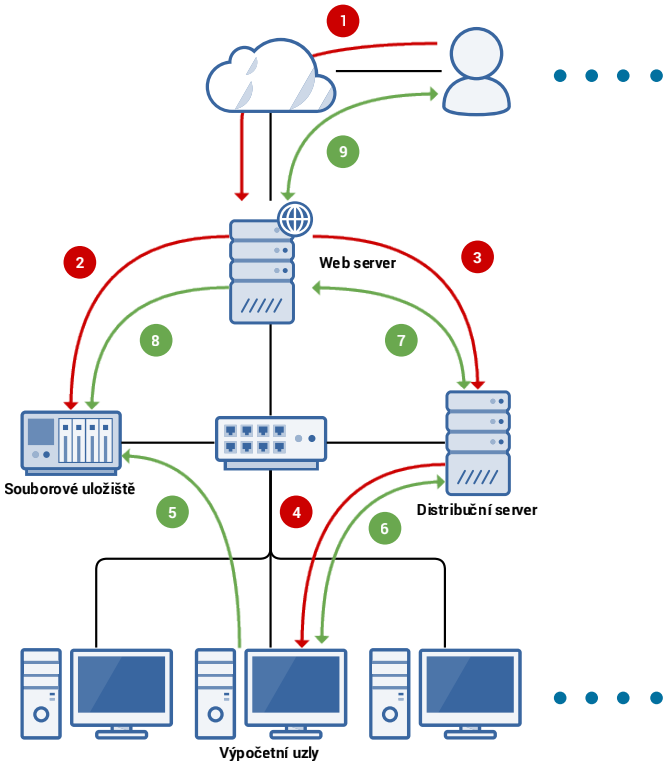
Na druhou stranu Web a Distribuční servery spolu se Souborovým uložištěm je možné podle aktuálních možností a potřeb kombinovat. Extrémní variantu, kde jsou všechny tyto prvky spojeny do jednoho zobrazuje varianta A4, kterou je možné nalézt v příloze E.

Tato varianta počítá s přístupem všech zúčastněných počítačů v lokální síti k souborovému uložišti, což dělá ze sdílení souborů scén a výsledků v rámci sítě triviální úlohu.

Schematicky je možné architekturu zobrazit pomocí diagramu 2.1a. Další 2.1b pak naznačuje postup požadavku o vyrenderování scény v rámci této architektury.



(a) Model architektury A1



(b) Zpracování scény v rámci architektury A1

Obrázek 2.1: Architektura A1

Směr šipek v diagramu 2.1b naznačuje odkud je iniciována komunikace. V případě obousměrné šipky je možné, aby komunikaci iniciovala libovolná strana. Kroky 1-9 reprezentují jednotlivé kroky zpracování požadavku scény a znamenají:

1. Vložení scény ke zpracování uživatelem skrze webové rozhraní.
2. Uložení nahrané scény na Souborové uložiště pro sdílení s ostatními počítači.
3. Zaslání požadavku o výpočet vložené scény na Distribuční server.
4. Zvolení jednoho či více Výpočetních uzlů Distribučním serverem pro zpracování úlohy.
5. Vyzvednutí scény pro zpracování ze Souborového uložiště.
6. Zpráva o dokončení od uzlu k serveru nebo zjištění této skutečnosti periodickou kontrolou serveru.
7. Zpráva o dokončení od Distribučního serveru k Web serveru nebo zjištění této skutečnosti na základě periodické nebo uživatelsky iniciované kontroly.
8. Zpřístupnění výsledků na webu. V případě vlastního uložiště pro sdílení se jedná o překopírování souborů na něj.
9. Zpráva o výsledku uživateli například e-mailem nebo kontrola výsledků uživatelem. Vystavení výsledku ve webovém rozhraní.

Výhodou této architektury je, že nabízí jednoducé sdílení souborů mezi jednotlivými účastníky a nevyžaduje vysoké zabezpečení komunikace mezi Web serverem a zbytkem render farmy.

Nevýhodou je potřeba veřejného Web serveru s přímým přístupem do Souborového uložiště, což s sebou může nést komplikace spojené se správou serveru a bezpečnostní rizika.

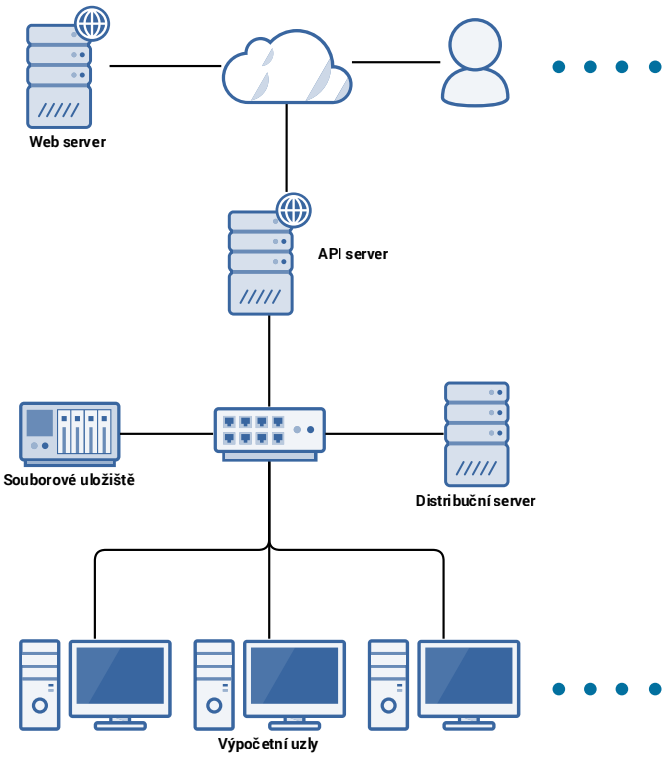
2.1.2 A2 - Samostatný Web server a API server

Další variantou je umístění Web serveru komunikujícího s uživateli mimo lokální síť. To umožňuje rozhraní služby hostovat externě a lépe omezit povolené akce Web serveru na render farmě.

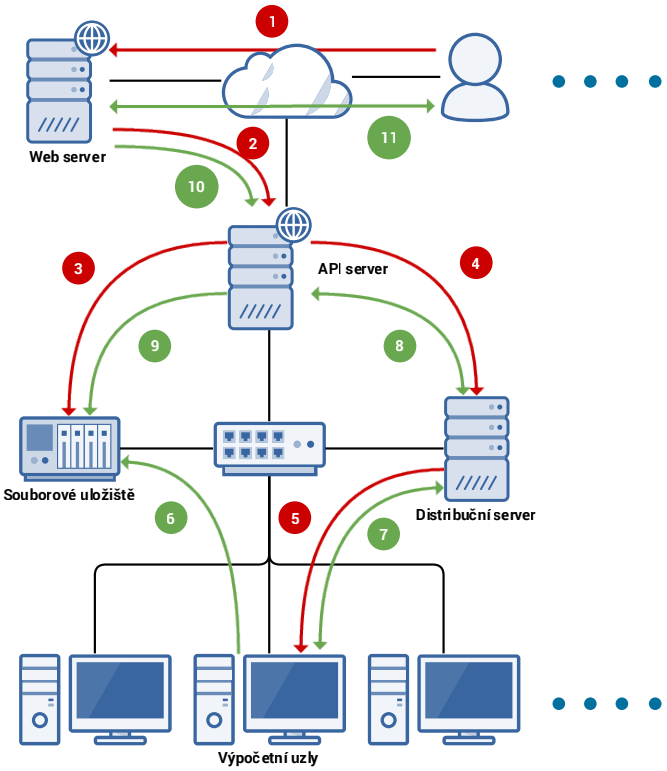
Tato varianta uvádí novou jednotku, API server. Ten vystavuje Web serveru API pro zadávání a vyzvedávání úkolu. Web server tak nemusí mít přímý přístup k souborovému uložišti, například přes FTP, což umožňuje například provádět kontrolu přijámaných souborů. API server přejíma zodpovědnosti za komunikaci v rámci sítě od Web serveru z minulé varianty.

Web server v této variantě logicky není možné fyzicky zkombinovat s ostatními jednotkami. Tato možnost se převádí na API server.

Diagram 2.2a nabízí schéma této varianty. V dalším 2.2b je prezentováno zpracování úlohy v rámci této architektury.



(a) Model architektury A2



(b) Zpracování scény v rámci architektury A2

Obrázek 2.2: Architektura A2

Stejně jako u minulé varianty reprezentují šipky v diagramu 2.2b směr, odkud je započata komunikace. Kroky 1-11 reprezentují:

1. Vložení scény ke zpracování uživatelem skrze webové rozhraní.
2. Odeslání úlohy včetně souboru scény do render farmy za využití API serveru.
3. Uložení nahrané scény na Souborové uložiště pro sdílení s ostatními počítači.
4. Zaslání požadavku o výpočet vložené scény na Distribuční server.
5. Zvolení jednoho či více Výpočetních uzlů Distribučním serverem pro zpracování úlohy.
6. Vyzvednutí scény pro zpracování ze Souborového uložiště.
7. Zpráva o dokončení od uzlu k serveru nebo zjištění této skutečnosti periodickou kontrolou serveru.
8. Zpráva o dokončení od Distribučního serveru k Web serveru nebo zjištění této skutečnosti na základě periodické nebo uživatelsky iniciované kontroly.
9. Zpřístupnění výsledků na skrze API. V případě vlastního uložiště pro sdílení se jedná o překopírování souborů na něj.
10. Vyzvednutí výsledku na základě periodické kontroly nebo uživatelské aktivity.
11. Zpráva o výsledku uživateli například e-mailem nebo kontrola výsledků uživatelem. Vystavení výsledku ve webovém rozhraní.

V této variantě je Web server jasně umístěn mimo síť (může se jednat i o pouze i jinou podsíť v rámci podnikové struktury) a se sítí render farmy komunikuje pouze skrze kontrolované API. To umožňuje vyšší adaptabilitu na různé síťové struktury.

2.1.3 Závěr

Další návrh je potřeba provést tak, aby bylo možné podporovat tyto základní typy architektur.

2.2 Návrh komponent

Při návrhu komponent jsem kladl důraz na *zachování modularity systému*. Tedy co největší možné omezení provázanosti jednotlivých modulů tak, aby byl každý jeden vždy nahraditelný bez větších zásahů do systému. To je nutnou vlastností například vzhledem k tomu, že z analýzy provedené v minulé sekci jsem došel k závěru, že volba backendu nemusí být finální. Mám na mysli možnou eventuální změnu za systém *Clondike* nebo například vhodnější volbu *CGRU Afanasy* při instalaci ve více kontrolovaném, například studiovém prostředí.

Tento záměr se projevil v největší míře v rozdělení zodpovědností a návrhu jednotných komunikačních rozhraní.

Zároveň takto modulární návrh umožňuje podporu obou architektur A1 a A2, ale i mnohých dalších například implementaci agregačních adaptérů, které transparentně využívají zcela rozdílné backendy. Například kombinaci lokálního clusteru, připojeného gridu a vypočetního cloudu.

Návrh celého systému jsem rozdělil do tří vrstev s jasně definovanými zodpovědnostmi. Frontend zodpovědný za komunikaci s vnějším světem, Výpočetní backend zodpovědný za samotné výpočty a Adaptér - komunikační mezivrstvu zodpovědnou za odstínění konkrétní implementace či síťového prostředí backendu od Frontendu pomocí vhodné abstrakce a API.

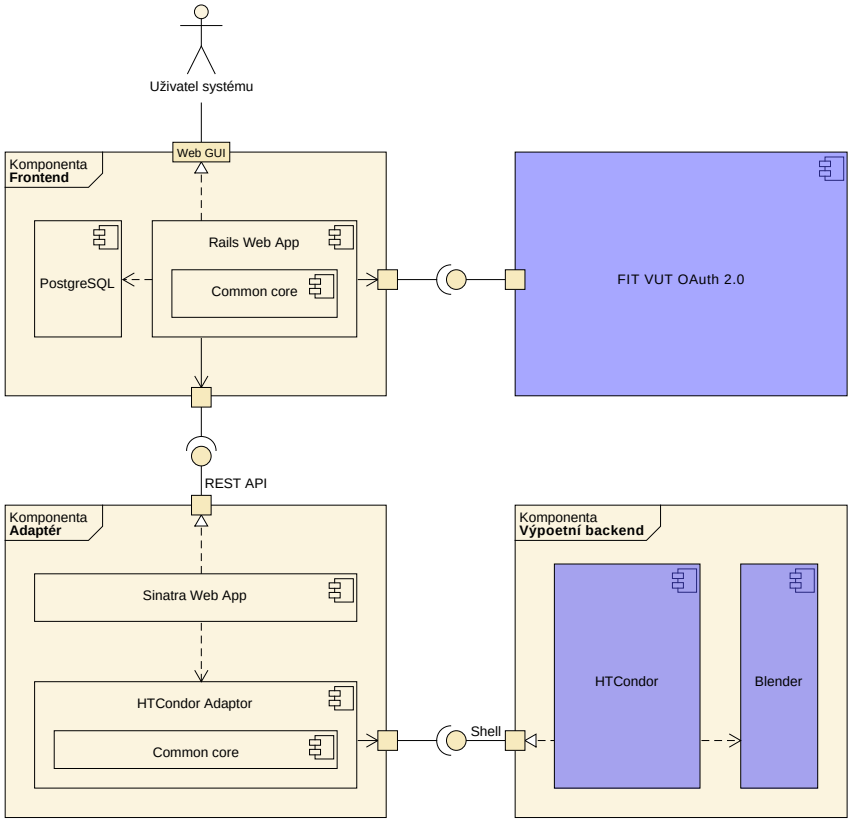
Bližší podrobnosti a odůvodnění jednotlivých rozhodnutí uvádím u každé navržené komponenty.

Diagram 2.3 prezentuje přehled navržených komponent a komunikace mezi nimi. Modře jsou vyznačeny již hotové součásti.

2.2.1 Frontend

Část Frontendu je, jak již bylo zmíněno, zodpovědná za komunikaci s vnějším světem. Jedná se o kompletní webovou aplikaci, která pomocí RESTového API komunikuje s nižší vrstvou. Obsahem této komunikace je zejména předávání souborů scén a vyzvedávání výsledků. Toto API jsem nazval *FITRender Compute API* a jeho popis je k nalezení v příloze C.

Mezi specifické zodpovědnosti této komponenty tedy spadá autentizace uživatele za pomoci externího autentizačního modulu. Zejména pak správa scén, tedy umožnění uživateli nahrát scénu, zajistit její odeslání Adaptéru a vyzvednutí výsledků. Při nahrání scény ji Frontend může sám umístit na Souborové uložiště, pokud k němu má přístup, nebo samotný soubor odeslat skrze API Adaptéru, který se o jeho uložení postará. Od Frontendu jde také signál ke smázání scény, buď tím, že ho přímo provede a nebo k tomu dá příkaz skrze API. Smazání se provádí po uživatelském schválení po kontrole výsledků a nebo po stanoveném časovém intervalu. Dále tato komponenta zpřístupňuje zbytek API, který se věnuje správě uzlů a globální konfiguraci farmy.



Obrázek 2.3: Návrh komponent systému vycházející z modulů a požadavků

Interně se jedná o webovou aplikaci, pro kterou jsem s ohledem na analýzu provedenou v sekci Technologie pro dodatečnou implementaci **??** zvolil framework *Ruby on Rails*. Ten podporuje jak dobré propojení s autentizačním modulem, tak nutnou správu databáze a komunikaci skrze RESTové rozhraní Adaptéru.

Pro sdílení funkcionality a kódu mezi Frontendem a Adaptérem se předpokládá použití jedné komponenty, v diagramu zobrazené jako *Common core*.

2.2.2 WS Adaptér

Tato vrstva se z vnějšího hlediska stará o překlad konkrétního API zvoleného výpočetního backendu na webovou službu poskytující *FITRender Compute API*, které je využito při komunikaci s Frontendem. Podrobnosti tohoto API jsou k dispozici v příloze C.

Zodpovědností této komponenty je přijmout scénu do Frontendu a odeslat ji ke zpracování na Výpočetní backend. Scénu může přijmout jako soubor, který uloží na Souborové uložiště a nebo pouze jako cestu, kterou pouze předá

dál. Pokud backend sám nepodporuje generování úloh ze souboru scénu, jako tomu je například u specializovaných CG distribučních systému, tak je zodpovědností Adaptéru i připravit ze scény zadání srozumitelné pro výpočetní backend. Dále pak Adaptér musí umožňovat sledování stavu jednotlivých úkolů, správu uzlů a nastavení. V podstatě implementovat celé FITRender Compute API. Od Adaptéru se neočekává použití správa scén ve smyslu vedení databáze.

Součástí tedy komponenty je vzhledem ke zvolenému REST API znovu webová aplikace a pak součást zodpovědná za komunikaci s Výpočetním backendem. Při tomto návrhu je úkolem webové aplikace pouze překládat komunikaci mezi konkrétním adaptérem a RESTovým API. Proto zde volím, znovu s odvoláním na sekci Technologie pro dodatečnou implementaci **??**, nástroj pro rychlou a minimalistickou tvorbu webových aplikací *Sinatra* [188].

Při tomto návrhu je možné při změně výpočetního backendu pouze upravit konkrétní adaptér. Webová aplikace poskytující API a Frontendová část zůstanou beze změny.

Pro sdílení funkcionality a kódu mezi Frontendem a Adaptérem se předpokládá použití jedné komponenty, v diagramu zobrazené jako *Common core*.

2.2.3 Výpočetní backend

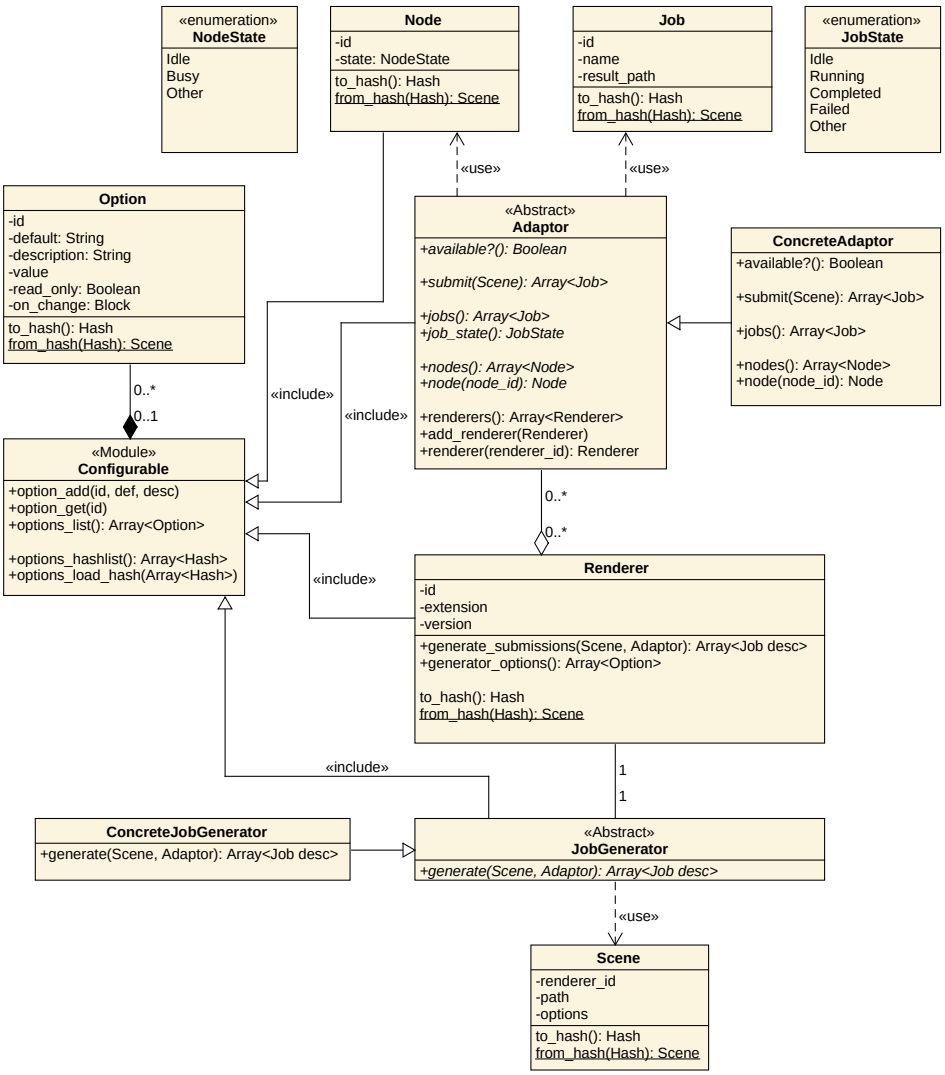
Kombinace distribučního systému a instalovaných rendererů dává dohromady *Výpočetní backend* (VB). Jeho konkrétní architektura záleží na použité distribuční technologii.

Jediným požadavkem je rozhraní, které může Adaptér využít pro naplnění požadavků *FITRender Compute API*. To znamená, že VB musí umožňovat zadání a zpracování nových úloh, informaci o jejich stavu a sdílení výsledku uložením na sdílené uložiště nebo jinou cestou.

2.2.4 FITRender Compute API

Smyslem tohoto API je nabídnout *komunikační rohzraní* mezi Frontendem a Adaptérem, umožňující Frontendu plnit požadovanou funkcionalitu bez nutné předchozí znalosti samotného výpočetního backendu či rendererů. To umožňuje v případě potřeby výpočetní backend upravit či vyměnit a přidávat podporu dalších rendererů a jejich nastavení bez nutných úprav v samotném Frontendu. Dále toto API umožňuje přenos souborů v případě, že Frontend a Adaptér nemají přístup ke sdílenému souborovému uložišti.

Výsledkem návrhu je REST rozhraní využívající pro komunikaci formátu JSON. Kompletní seznam nabízených metod a příkladů je k dispozici v příloze C.



Obrázek 2.4: Návrh jednotlivých tříd WS Adaptéru

2.3 Návrh tříd

V této sekci prezentuji návrh tříd pro komponentu Adaptér a nutný základ pro databázové uložiště Frontendu.

2.3.1 WS Adaptér

2.3.1.1 Adaptor

Tato třída reprezentuje výchozí třídu pro implementaci adaptéru, který umožňuje ovládání Výpočetního backendu přímo z jazyka implementace. Je navr-

žena pro využití webovou aplikací, která poskytuje výsledné REST rozhraní.

Její metody jsou voleny tak, aby co nejblíže odpovídali FITRender Compute API. Přidána je pomocná metoda available?, která je určena pro kontrolu spojení s Výpočetním backendem.

Kromě toho je také tato třída vybavena implementací správy rendererů, jelikož je tato funkcionalita nutné pro veškeré konkrétní implementace.

Používá modulu Configurable pro poskytnutí možností konfigurace.

2.3.1.2 ConcreteAdaptor

Implementace Adaptoru, implementuje specifikované abstraktní metody a umožňuje už skutečné ovládání nějakého Výpočetním backendu.

2.3.1.3 Renderer

Představuje obalovou třídu pro podporu specifického rendereru v rámci Adaptoru. Drží si popisné informace o daném rendereru a umožňuje přípravu úkolů ze scény delegací na JobGenerator.

Vystavuje také možné nastavení svého generátoru, upravované při zadávání každé úlohy.

Vzhledem k tomu, že se předpokládá její vystavení webovou službou v rámci metod pro seznam a detail rendererů, je metoda opatřena serializačními a deserializačními metodami to_hash a from_hash.

2.3.1.4 JobGenerator

Tato třída je určena jako základ pro implementaci tříd pro převod zadané scény do seznamu úkolů pro Výpočetním backend dle nastavení systému a scény.

2.3.1.5 ConcreteJobGenerator

Implementace JobGeneratoru. Generuje seznam úkolů pro Výpočetním backend na základě zadané scény aktuálního nastavení.

2.3.1.6 Scene

Reprezentuje jedno zadání scény systému. Nese s sebou identifikaci, pro jaký Renderer a tím pádem JobGenerator je určena a samozřejmě cestu k souboru scény a nastavení renderu od uživatele.

Vzhledem k tomu, že se předpokládá její přenos v rámci webové služby při zadání nové scény, tak je opatřena serializačními a deserializačními metodami to_hash a from_hash.

2.3.1.7 Node

Třída reprezentuje jeden výpočetní uzel v systému. Obsahuje identifikaci v rámci Výpočetního backendu. Udržuje si identifikaci v rámci Výpočetního backendu a svůj aktuální stav z možností Neaktivní, Zaneprázdněný a Jiný.

Pro správu nastavení, o nichž poskytuje informance, využívá modulu Configurable. Pro nastavení, u kterých nepovoluje změnu nastavuje atribut read_only na pravdivou hodnotu.

Předpokládá její přenos v rámci webové služby při zjišťování přehledu uzlů a podrobností či nastavení jednoho uzlu. Proto je opatřena serializačními a deserializačními metodami to_hash a from_hash.

2.3.1.8 Job

Tato třída představuje jeden konkrétní úkol v rámci Výpočetního backendu. Ve svých atributech obsahuje identifikaci v rámci Výpočetního backendu, svůj popisný název a cestu k výsledku úkolu. Stav úkolu je možné zjistit pomocí metody job_state třídy Adaptor.

2.3.1.9 Configurable

Tento modul představuje zásuvnou podporu konfigurace, která podporuje přidání a úpravu nastavení, která je možná přenášet v rámci FITRender Compute API.

Poskytuje metody pro přidání, získání a přehledu nastavení. Jejich přehled také nabízí v serializované podobě, kterou je schopna zpět načíst.

Jednotlivá nastavení jsou reprezentována třídou Option.

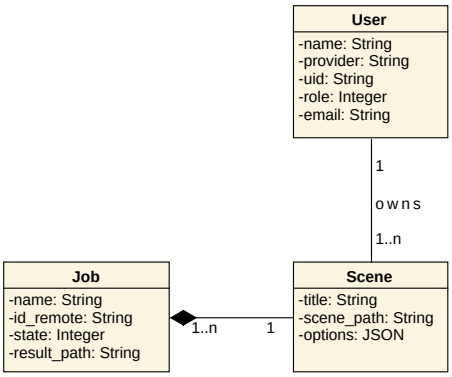
2.3.1.10 Option

Třída ztělesňuje jedno konkrétní nastavení. Drží si svoji aktuální a základní hodnotu, svůj popis a název, informaci o tom, zda je pouze pro čtení a volitelný odkaz na kód, který má vykonat při změně nastavení (například konkrétní příkaz pro konfiguraci samotného výpočetního uzlu).

Vzhledem k tomu, že se očekává její přenos v rámci API ve všech metodách, které zobrazují seznam nastavení, tak obsahuje serializační respektive deserializační metody to_hash respektive from_hash.

2.3.2 Frontend

Pro správnou funkci Frontendu je zapotřebí některá data dlouhodobě uchovávat. Patří mezi ně informace o uživatelích. Informace o scénách, jejich vlastnictví. A o úkolech vytvořených pro jednotlivé scény a jejich identifikace v rámci WS Adaptéru. Pomocné atributy a primární klíče databází nejsou pro přehlednost uvedeny.



Obrázek 2.5: Návrh tříd pro perzistenci Frontendu

2.3.2.1 User

Uživatel. Systém si o nich musí držet data ohledně jména, emailu a samozřejmě roli v systému - User, Elevated User či Admin. Při použití více poskytovatelů autentizace (například při přidání autentizace pomocí systému Shibboleth) je nutné si také udržovat záznam o použitém poskytovateli a o identifikátoru uživatele pro tohoto poskytovatele.

2.3.2.2 Scene

V případech užití byly jednotlivé prvky scény specifikovány jako název, samotná scéna a nastavení renderu. Pro možnost fyzického smazání scény po schválení uživatelem je potřeba si držet cestu k souboru scény nebo její identifikaci v rámci WS Adaptéru.

2.3.2.3 Job

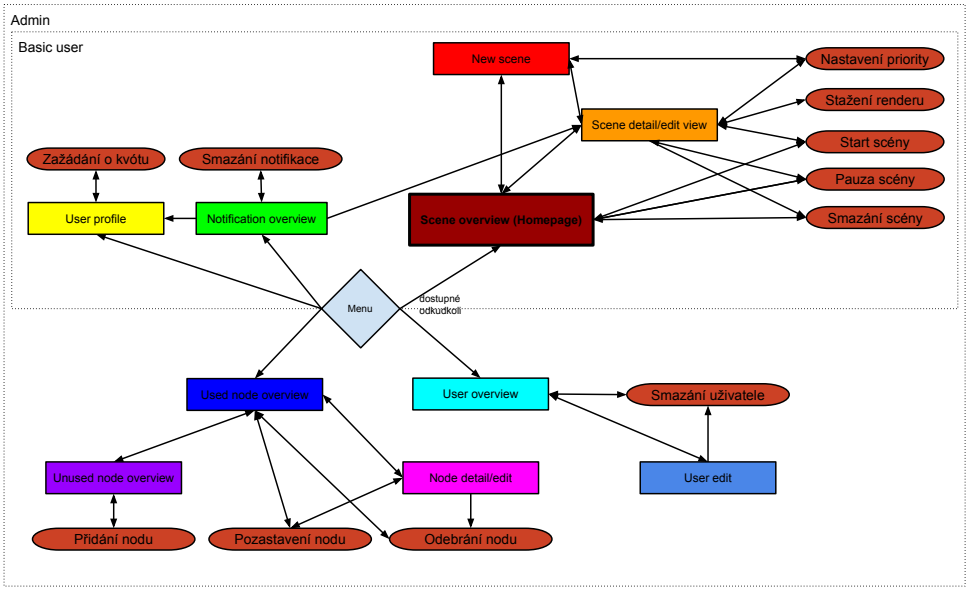
Tato třída reprezentuje jeden atomický úkol v rámci scény. Tento úkol je například vytvořen pro každý snímek či dlaždici renderu, případné spojení obrazu nebo další úlohy prováděné na výpočetních uzlech. Třída si udržuje svoji identifikaci v rámci WS Adaptéru, ze kterého je periodicky aktualizován její stav. Kromě toho se udržuje její název, který umožňuje úlohu opatřit popisem. Zároveň by si úloha měla udržovat cestu ke konečnému výsledku.

2.4 Návrh uživatelského rozhraní

V rámci předmětu NUR jsem ve spolupráci s Jaroslavem Rambou připravil referenční verzi uživatelského rozraní webového rozhraní FITRenderu.

Návrh vycházel ze starší analýzy požadavků a rolí a tak nereflektuje plně zde prezentované požadavaky a role. Mezi hlavní rozdíly patří použití pouze dvou přihlášených roli User a Admin s obdobnými právy jako v aktuálním návrhu. A přidané funkcionality pro přidělování kvót administrátorem. Až na těchto pár změn však návrh vycházel z podobných předpokladů.

Mezi důležité výstupy z tohoto návrhu patří analýza jednotlivých úkonů v rámci uživatelského rozhraní a jejich propojení. Výsledkem byl graf úkonů 2.6.



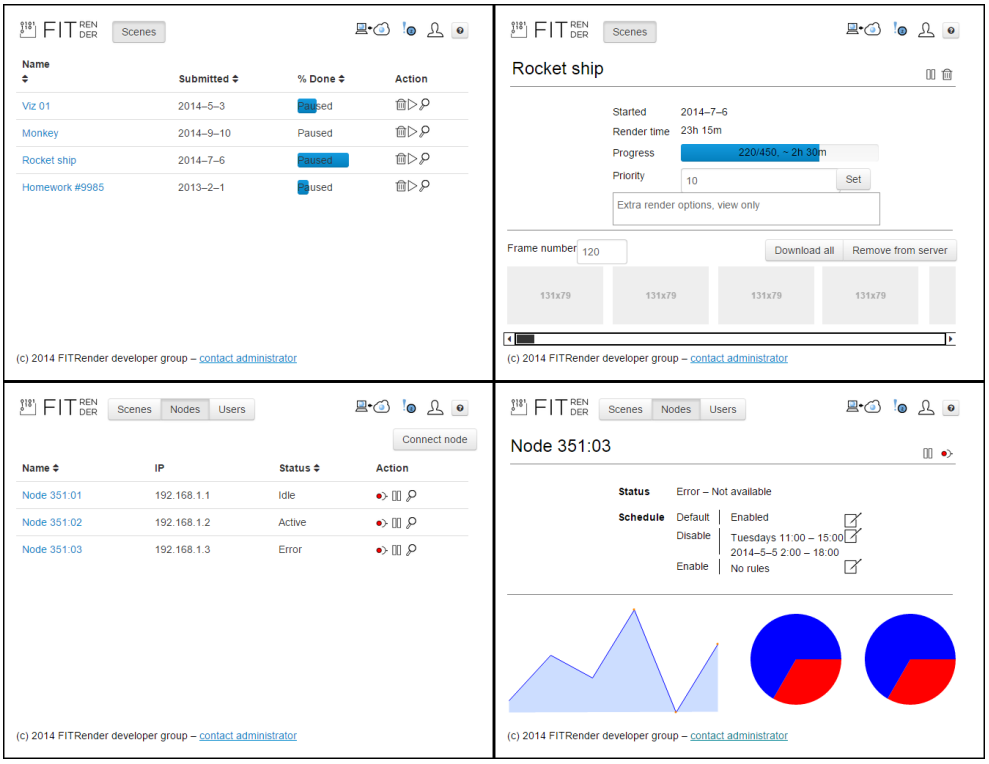
Obrázek 2.6: Graf úkonů v uživatelském rozhraní služby FITRender

Dále byla navržena prvotní verze uživatelského rozhraní, které bylo postupně vylepšováno analýzou uživatelských rozhraní produktů RenderPal V2, SquidNet a Sheep It! Render Farm, heuristickou analýzou jednotlivých prvků a uživatelským testováním na hi-fi prototypu. O provedeném testování je možné nalézt více v kapitole Testování 4.1.

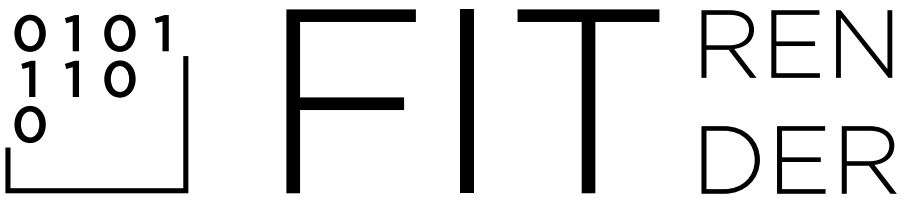
Výsledkem byl sada prototypových obrazovek systému ve formě klikatelného prototypu. Tento prototyp je možné spolu s ostatními výstupy z tohoto návrhu nalézt na přiloženém nosiši. Zde uvádím pouze výběr z obrazovek prototypu 2.7.

Při samotné implementace jsem tento návrh využíval jako vodítko.

Kromě toho jsem pro službu v rámci přípravy uživatelského rozhraní navrhl logo 2.8. Čtverec vlevo reprezentuje převod počítačových instrukcí na



Obrázek 2.7: Výběr obrazovek z návrhu uživatelského rozhraní. Role User, přehled scén. Role User, detail scény. Role Admin, přehled uzlů. Role Admin, detail uzlu obsahující statistiky.



Obrázek 2.8: Návrh loga systému FITRender

obraz, 3D scén do výsledných renderů, tedy primární účel služby.

Kapitola 3

Implementace

Pro implementaci jsem využil zvolených technologií - Ruby on Rails pro komponentu Frontend a framework Sinatra pro WS Adaptér.

Jako podpůrné nástroje jsem při implementaci použil virtualizační řešení VMWare a vývojové prostředí RubyMine 7. Vyvíjel jsem na operačních systémech Microsoft Windows, Ubuntu a OpenSUSE.

3.1 WS Adaptér

Při realizaci WS Adaptéru jsem se řídil návrhem tříd pro tuto komponentu. Většinu tříd a jejich metod jsem implementoval dle této specifikace. Výjimku tvoří třída Node, která v aktuální realizaci neposkytuje možnost konfigurace nastavení, ale pouze zobrazení, ani nevyužívá modul Configurable. Dále třída Option neumožňuje určení read_only atributu a bloku pro vykonání po změně. Tato implementace je však plánována do budoucna.

Implementoval jsem pomocný konfigurační modul Configurable a jeho rozšíření ConfigurableWithFile, který automaticky ukládá upravená nastavení do YAML souboru dle třídy, která je konfigurována. Druhou alternativou uložení konfiguračních hodnot je využít proměnných prostředí. Ty jsou však stejně bez zálohy v souboru smazány při restartu prostředí a jsou myšleny spíše pro statická nastavení trvající po celou dobu běhu systému. Přístup pomocí YAML souboru nabízí možnost zálohované dynamické konfigurace. Při jeho smazání se hodnoty vrátí na základní hodnoty.

Dále jsem pro konfiguraci vytvořil dvě zásuvné sady s konfiguračními hodnotami a pomocnými metodami, moduly Framable a Pathable. Framable přidává nastavení "frames", ze kterého je schopen přečíst seznam čísel snímku k vyrenderování. Druhý Pathable přidává 4 obvyklá nastavení cest: cesty pro scény, cesty pro výsledky, cesty pro logovací soubory a cesty pro textový výstup rendererů.

Vytvořil jsem základní třídy Adaptor, Renderer a JobGenerator, pro které jsem vytvořil konkrétní implementace CondorShellAdaptor, Blender a Con-

dorBlenderJobGenerator.

CondorShellAdaptor vystavuje metody, které komunikují s instalovaným systémem HTCondor pomocí shell příkazů. U některých příkazů používá návratovou hodnotu v XML, pro něž používá parser Nokogiri [190].

Blender reprezentuje jediný podporovaný renderer, který nabízí nastavení granularity dlaždic a generuje snímky za pomocí BlenderCondorJobGenerator.

BlenderCondorJobGenerator podporuje vytváření úkolů pro jednotlivé snímky animace specifikované pomocí seznamu snímku a dělení snímků na dlaždice pro vyšší granularitu úkolů. Při vytváření úkolů se řídí nastavením cest z adaptéru a nastavením granularity z rendereru.

Pro podporu nastavení dlaždic jsem vytvořil Python skript pro Blender, který je možné použít při spouštění programu z příkazové řádky. Skript je k nahlédnutí v příloze G.

Jednotlivé třídy jsou rozdělil do dvou balíčků Gemfile, fitrender_common a fitrender_adaptor_condor. Fitrender_common obsahuje obecnou funkcionalitu a fitrender_adaptor_condor.

Pro chybová hlášení obsahuje sdílený balíček i sadu výjimek InterfaceNotImplementedError, BackendNotAvailableError, NotFoundError, SubmissionsFailedError, RendererNotFoundError, OptionNotFoundError, FileNotFoundError a ConfigError.

Diagram (REF) prezentuje vztahy mezi implementovanými třídami kromě výjimek a jejich rozdělení do balíčků.

Implementace těchto částí WS Adaptéru postupovala z převážné části metodikou Test Driven Development a tak je většina implementované funkcionality pokryta jednotkovými testy typu RSpec. Podrobnější text o provedeném testování je k dispozici v kapitole Testování.

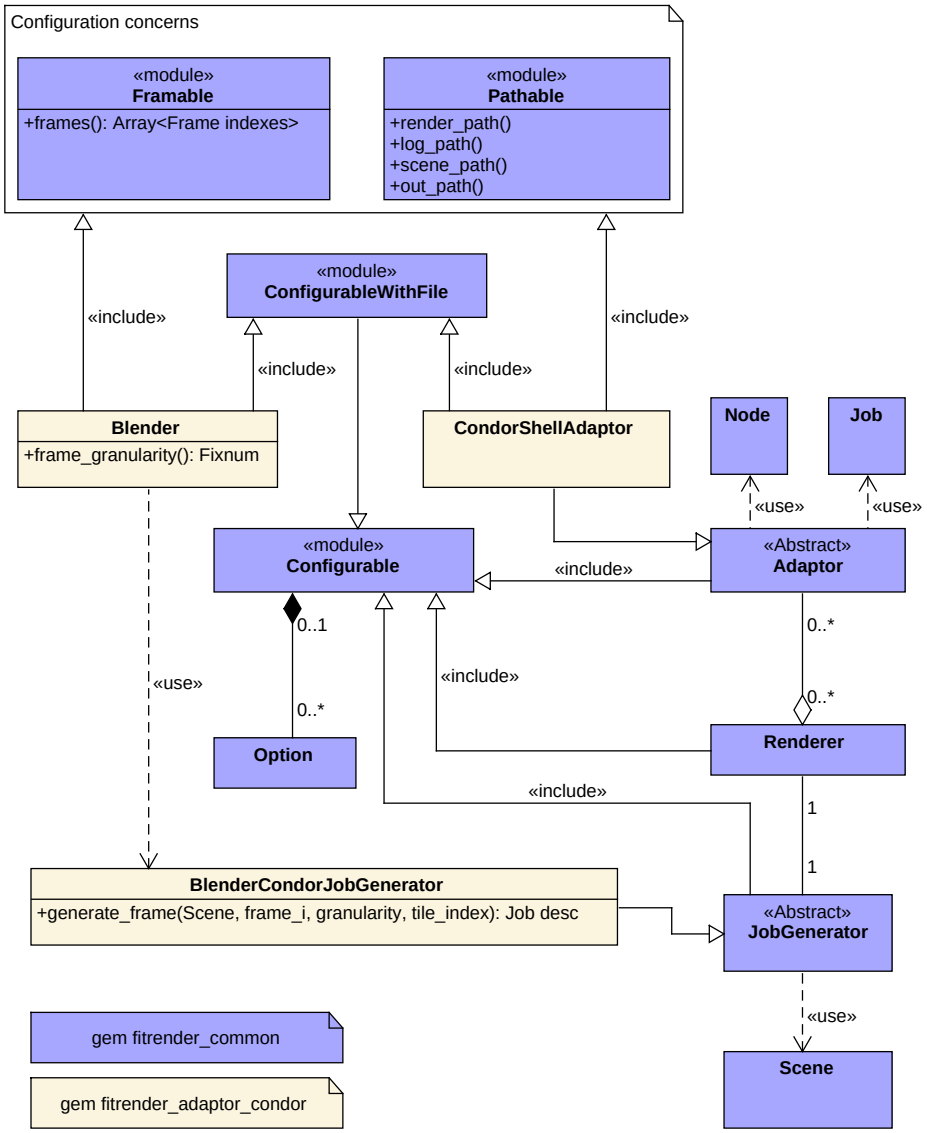
Pomocí frameworku Sinatra jsem implementovaný HTCondor adaptér pro Ruby vystavil jako webovou službu splňující většinu specifikace FITRender Compute API. Implementovány nejsou přenosy souborů a nastavení uzlů.

Funkcionalitu rozhraní jsem v průběhu vývoje zkoušel aplikací Postman [191].

3.2 Frontend

Pro implementaci webového rozhraní jsem využil frameworku Ruby on Rails. Jako databázový stroj jsem zvolil PostgreSQL, který například oproti MySQL podporuje transakce nad migracemi použitého ActiveRecord ORM [192] a nativní podporu JSON typu, který jsem navrhl použít pro uložení nastavení scény [193].

Pro podporu školní autentizace jsem využil knihovnu pro různé autentizační postupy OmniAuth, pro kterou jsem implementoval novou strategii pro přihlašování pomocí OAuth 2.0 serveru ČVUT. Tuto strategie jsem uveřejnil na serveru GitHub pod názvem omniauth-fitcvut-oauth2. Strukturu implementace jsem převzal ze strategie omniauth-soundcloud.



Obrázek 3.1: Náhled stavu tříd v prototypové implementaci. Modře vyznačené třídy jsou součástí Gemu fitrender_common, žluté jsou součástí fitrender_adaptor_condor.

Vzhled aplikace jsem založil na CSS frameworku Foundation v SASS variantě, který nabízí velmi dobrou konfiguraci [194].

Pro generování náhledů obrázku a jejich správu v databázi jsem použil nástroj Paperclip, který umožňuje specifikaci velikostí náhledů, jejich generování pomocí utility ImageMagick a následný přístup k nim [195].

Uživatelské rozhraní jsem implementoval tak, aby dynamicky reagovalo na informace z WS Adaptéru. Formulář pro zadání úlohu v sobě tak obsahuje aktuálně konfigurovatelné nastavení třídy Generator. Nastavení backendu zobrazuje všechna dostupná nastavení třídy Adaptor a dostupných instancí třídy Renderer. Pro podporu této funkcionality byly některé třídy z fitrender_common (Node, Option, Renderer) rozšířeny o chování ActiveRecord modelu. Díky tomu se v aplikaci chovají obdobně jako databázové modely, ale jejich data pocházejí z a jsou zasílána do WS Adaptéru.

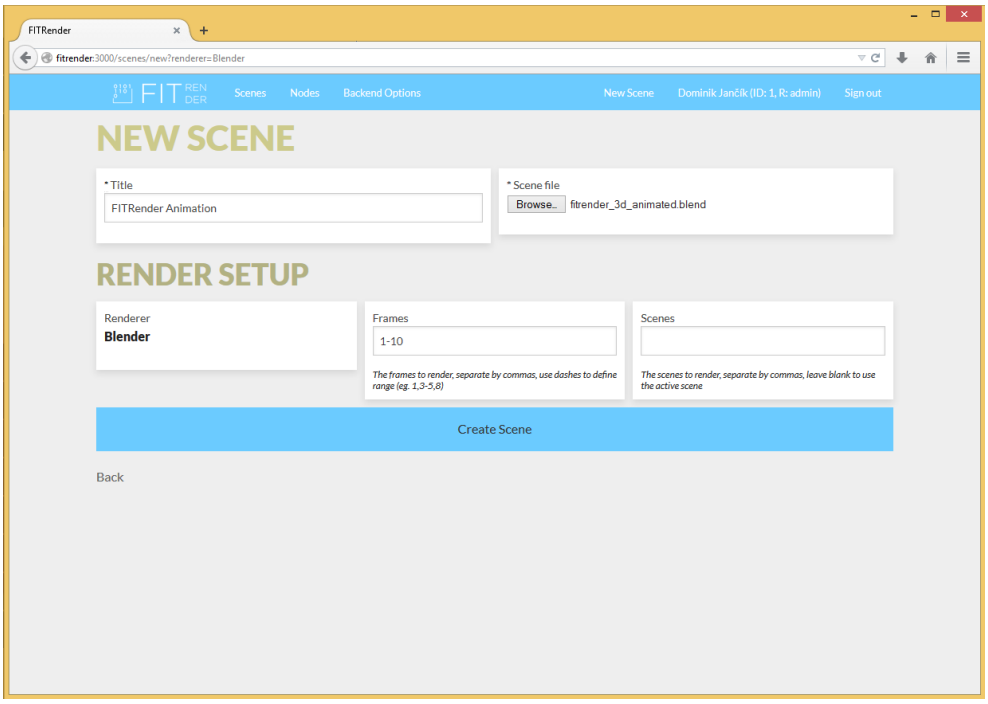
Příkladem může být propagace konkrétního nastavení rendereru 3.2 do formuláře přidání scény 3.3.

Kompletní sada screenshotů je k dispozici v příloze F

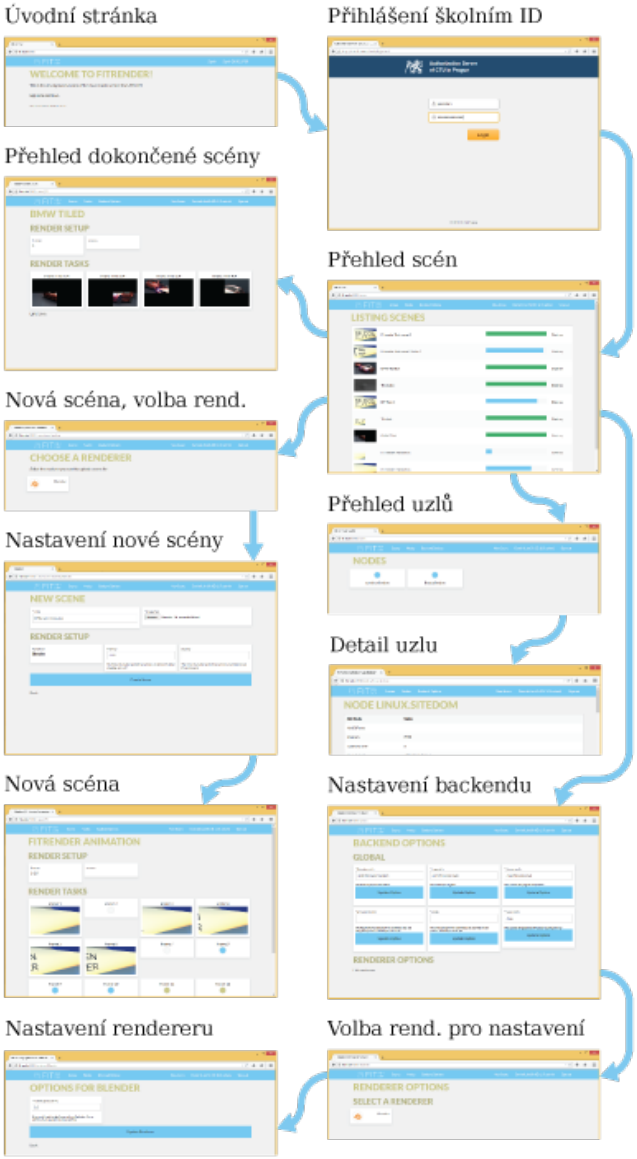
Zdrojový kód je umístěn na přiloženém DVD a také na gitlab repozitářích [196][197][198][199].

```
{
  "id": "Blender",
  ...
  "generator_options": [
    {
      "id": "frames",
      "default": "1",
      "description": "The frames to render...",
      ...
    },
    {
      "id": "scenes",
      "default": "",
      "description": "The scenes to render...",
      ...
    }
  ],
  "renderer_options": [
    {
      "id": "frame_granularity",
      ...
    }
  ]
}
```

Obrázek 3.2: Zkrácená verze konkrétního nastavení rendereru, které je poskytnuto od backendu pomocí FRC API.



Obrázek 3.3: Formulář pro přidání scény, který se přizpůsobuje aktuální konfiguraci backendu.



Obrázek 3.4: Přehled obrazovek webového rozhraní. Modrou šipky jsou naznačeny možné průchody aplikací.

Kapitola 4

Testování

4.1 Uživatelské testování

V rámci návrhu uživatelského rozhraní bylo provedeno uživatelské testování na klikatelném prototypu, který vycházel z navržených mockupů.

Testování probíhalo rozesláním archivu s prototypem a zadáním ve formě Google prezentace, která obsahovala zadání, které vedlo k modelovým scénářům jako zadání nové scény nebo zobrazení přehledu uzlů. Tato prezentace je k dispozici v příloze (REF).

Pro sběr výsledků bylo použito Google Formulářů. Cílem testování bylo zjistit problémy uživatelů s přistupem k jednotlivým funkcím a pochopení použitých ikon. Dotazník tak byl proveden formou série otevřených otázek pro každý scénář.

V rámci testování se vyskytly problémy s méně standardními ikonami a chybějícími odkazy. Na základě těchto poznatek bylo provedeno několik malých úprav v návrhu. Kompletní výsledky a analýzu je možné najít v přílohách (REF).

4.2 Unit testy

Jednotkové testy jsem využil zejména pro ověřování funkcionality komponenty WS Adaptéru. Implementaci jednotlivých vlastností navržených tříd předcházela specifikace testem systému RSpec. Testy pomáhaly také odhalit chyby vytvořené změnami kódu.

Balíčky fitrender_common respektive fitrender_adaptor_condor tak mají 85% respektive 96% pokrytí testy. Výsledky analýzy pokrytí jsou k dispozici v příloze. WS služba a Frontend testy doposud nemají, ale je pro ně připravena infrastruktura. % section unit_testy (end)

4.3 Akceptační testování

Vzhledem k tomu, že se jedná o prototyp, tak tato implementace nesplňuje veškerou navrženou funkcionalitu. Hlavní funkce systému, tedy zasílání scén na render farm a jejich následné vyzvednutí stejně tak jako některé administrační funkce, jsou podporovány. Zároveň je systém napsán tak, aby bylo možné bez větších úprav na práci navázat a dokončit kompletní myšlenku systému.

Následuje seznam požadavků a popis jejich splnění.

4.3.1 Funkční požadavky

4.3.1.1 F-FR1 Poskytnout webové rozhraní pro zadávání a vyzvedávání renderování 3D scén

Služba v prototypové verzi umožňuje zadání scén, které se zpracují na výpočetním clusteru a uživateli se zobrazí výsledek. Tento obecný požadavek je tím splněn.

4.3.1.2 F-FR2 Upozornit uživatele na dokončené scény

Upozornění nebyly dosud implementovány v žádné formě. Tento požadavek prototypová implementace neplní.

4.3.1.3 F-FR3 Podporovat správu nahraných scén

Systém umožňuje jak zobrazování, úpravu, tak i mazání jednotlivých scén. Mazání scén je však povoleno pouze administrátorovi a odstranění scény z databáze nesmaže její fyzické soubory. Prototypová implementace tento požadavek proto splňuje v omezené míře.

4.3.1.4 F-FR4 Podporovat přihlášení pomocí celoškolského přihlášení ČVUT

Aplikace poskytuje možnost využití školní autentizace pomocí fakultního OAuth 2.0 serveru. Tento požadavek je tím splněn.

4.3.1.5 F-FR5 Podporovat render jednotlivých snímků, ale i animací

Při zadání scény je uživateli umožněno zadat sekvenci snímků ke zpracování. Pro každý snímek (nebo dlaždici) pak systém vytvoří úlohu, která je zpracována a později nabídnuta k vyzvednutí uživatelem. Tento požadavek je tím splněn.

4.3.1.6 F-FR6 Podporovat nastavení granularizace úkolů

Implementace třídy Renderer a Generator umožňují specifikovat rozdělení na dlaždice pomocí určení granularizačního faktoru. Toto nastavení je díky dynamickému UI Frontendu dostupné pro Administrátora.

4.3.1.7 F-FR7 Podporovat scény programu Blender

Prototypová implementace pracuje s tímto programem a generuje pro něj validní zadání pro animace a dlaždice. Tím je požadavek splněn.

4.3.1.8 F-FR8 Možnost správy výpočetního clusteru z webové aplikace

Implementace umožňuje sledování stavu uzlů a úpravu některých nastavení pro generování úloh. Není však podporováno nastavení parametrů uzlů. Implementace je tedy v tomto ohledu takzvaně dosud v plenkách. Tím je tento požadavek splněn pouze částečně.

4.3.1.9 F-FR9 Možnost časového rozvrhování využití farmy a jednotlivých počítačů

Distribuce a správa výpočtů je zodpovědností systému HTCondor, který toto nastavení umožňuje. Prototypová implementace však nenabízí toto nastavení upravit přímo z webového rozhraní.

4.3.1.10 F-FR10 Sbírat statistiky ohledně prováděných výpočtů

Sběr statistik nebyl dosud implementován a ani navržen. Na tuto funkcionalitu je nutné se zaměřit při dalším pokračování v projektu.

4.3.1.11 F-FR11 Umožnit správu uživatelů

Správa uživatelů není implementována v požadované podobě. Tento požadavek prototypová implementace neplní.

4.3.2 Nefunkční požadavky

4.3.2.1 N-FR1 Systém by měl zanechat využívané PC použitelné

Distribuce a správa výpočtů je zodpovědností systému HTCondor, který umožňuje reagovat na aktivitu počítače [200]. Při správné konfiguraci tohoto systému tak bude tento požadavek splněn. Nastavení Rank a Requirements z webové aplikace jsou momentálně slepá a tak neexistuje cesta, jak tuto funkcionalitu nastavit z webového rozhraní.

4.3.2.2 N-FR2 Systém by měl nabídnout přehledné uživatelské rozhraní

Systém vycházel z poměrně přehledného návrhu uživatelského rozhraní, který byl při testování přijat kladně. Výsledné rozhraní se v určitých aspektech liší a tak by bylo vhodné uživatelské testování opakovat, aby bylo možné tento požadavek zhodnotit. Vzhledem k prototypové formě implementace však není aktuální rozhraní považováno za finální.

4.3.2.3 N-FR3 Systém by měl být navržen pro více platforem

Frontend spustitelný na Rails je teoreticky multiplatformní, ale není to otestováno. Aktuální implementace WS Adaptér pro zadávání úkolů momentálně využívá pouze .sh skriptů a je tak použitelná pouze pod Unix systémy. Kromě toho by v rámci multiplatformní podpory bylo vhodné zavést mechanismus pro překlad cest. Momentálně tento požadavek není v dostatečné míře splněn.

4.3.2.4 N-FR4 Systém by měl umožnit snadnou výměnu jednotlivých modulů

Balíček s implementací podpory systému HTCondor spolu s podporou rendereru Blender má dle zprávy simpleconv pouze 158 efektivních řádek kódu. Některá funkcionalita, jako třeba změna nastavení uzlů ještě chybí, ale kromě toho je podporováno generování a zasílání úloh na cluster, kontrola jejich stavu, podpora renderování animací a rozdělení na dlaždice. To vše v již zmíněných 158 řádcích a je pravděpodobné, že některá funkcionalita se dá dále vymezit a sdílet. Distribuční a Renderovací modul je tak možné vyměnit s poměrně malým usílím a zbytek systému se změně přizpůsobí.

Posledním modulem kromě samotného implementovaného systému je Autentizační modul. Vzhledem k využití OmniAuth je zde možné jednoduše rozšířit aplikaci dalšími autentizačními službami. Při odstranění některé z nich se však někteří uživatelé již nebudou schopni přihlásit.

Celkově si tak systém zachovává svoji modularitu a s mírným omezením tento požadavek splňuje.

4.3.2.5 N-FR5 Systém by měl být jednoduše rozšiřitelný o podporu dalších rendererů

Základní třída Adaptor má vestavěnou podporu správy rendererů a implementace podpory dalších není náročnou záležitostí (viz předchozí požadavek). Proto tento požadavek považuji za splněný.

4.3.3 Další poznámky

Prototypová verze doposud neimplementuje navrženou podporu zadání scény spolu se souborem potřebnou pro podporu architektury typu A2. Dále pak

neumožňuje zobrazení a správu uživatelských profilů.

Závěr

Zhodnocení

V rámci práce byly analyzovány aktuální možnosti studentů pro zrychlení renderování, byl proveden výzkum o aktuálním stavu studentů architektury vzhledem k této problematice a byly popsány možnosti distribuce renderovacích úloh.

Dále byly analyzovány jednotlivé moduly nutné pro realizaci cloudové renderovací služby. Pro každý modul bylo buď nalezeno hotové řešení vhodné pro konkrétní požadavky - autentizace, distribuce a renderování - a nebo pro něj byla navržena a realizována implementace - webové rozhraní. Dále bylo shledáno, že při nasazení distribučního systému HTCondor škola získá obecnou infrastrukturu využitelnou i pro jiné typy distribuovaných úloh.

Výsledkem práce je primárně funkční prototypová implementace splňující většinu hlavních požadavků. Při realizaci byl kladen důraz na další rozšiřitelnost aplikace. Vedlejším produktem implementace je také knihovna pro využití fakultního autentizačního serveru skrze autentizační framework pro Ruby OmniAuth [201].

Sekundárně práce nabízí vhled do aktuálního stavu studentů architektury a odhad schopnosti služby plnit jejich nároky. Tento odhad je možné využít při odhadu výhodnosti celého řešení.

Budoucnoust

Co se týče budoucnosti projektu FITRender, tak je možné se zaměřit na budoucnost implementace - co je potřeba změnit či zlepšit - a budoucnost projektu samotného - zhodnocení výhodnosti a nasazení do provozu.

Implementace

Implementaci je v první řadě potřeba doplnit o nesplněné požadavky této práce. Konkrétně se jedná o možnost nastavení plánování přímo z webového rozhraní, implementace rolí a správy uživatelů dle specifikace, upozornění, správy životního cyklu souborů vázaných se scénou, sběr a zobrazení statistik, podpory renderování na více platformách, podpory přenosu souborů skrze FITRender Compute API a úpravy uživatelského rozhraní a dokončení konfiguračních tříd dle specifikace.

Dodatečně je potřeba systém doplnit o správu priorit jednotlivých úkolů či studentů, rozšířit API o typy a validační podmínky nastavení, kategorizaci nastavení, stránkování a překlad cest. V systému je také potřeba zavést systém pro kontrolu úspěšného dokončení renderů. Pro efektivní správu uzlů v učebnách by také bylo vhodné zavést podporu skupin a hromadných akcí pro uzly.

Dále se nabízejí možnosti ještě větší modularizace systému pomocí oddělení generačního kódu od komunikátorů s Výpočetním backendem či zvýšení elegance kódu implementováním DSL pro konfigurační třídy.

Pro udržitelný a efektivní vývoj je také nutné zvýšit pokrytí jednotkovými testy a případně zavést další druhy testování. Spolu s tím je také vhodné zavést trackovací systém chyb, podrobně zdokumentovat zdrojový kód a určit konvence jeho psaní. Tím se implementace připraví pro případné příspěvky open source komunitou.

Kromě toho je třeba se v neposlední řadě zaměřit na bezpečnost celé aplikace vzhledem k jejímu nasazení na školní prostředí. Je třeba ji analyzovat vzhledem k webovým útokům. Je potřeba zavést systém kontroly nahraných scén a zavést autentizační mechanizmy do samotného API.

Vzhledem k modularitě systému se může webové rozhraní rozšířit o podporu mnoha dalších Výpočetních backendů, například mnohých CG specializovaných open source řešení. Tím by se mohla zvednout celková úroveň těchto řešení a omezit duplikování práce napříč komunitou.

Projekt

Cílem projektu do budoucna je poskytnout fungující renderovací službu, která využívá volné prostředky školy studentům. Výsledky této práce naznačují, že tyto prostředky nejsou zdaleka zanedbatelné.

Pro splnění tohoto cíle je nutné finalizovat způsob, jak se systém zařadí do školní síťové infrastruktury. Poté ho nasadit v testovacím režimu a postupně rozšiřovat. V rámci toho je však potřeba zhodnotit celkovou výhodnost projektu a identifikovat případné závažnější problémy s jeho nasazením.

Přijde mi, že projekt je zajímavý a jeho realizace s sebou ponese mnoho pozitivních důsledků. Kromě posílení mezifakultní spolupráce a zavedení infrastruktury pro distribuované výpočty je jimi i případná možnost nabídnout využití této služby jako stipendium, čímž by bylo možné zvýšit motivaci studentů.

Proto je i v mém osobním zájmu projekt posouvat dál. Pokud vás tato práce zaujala a máte možnost ji nějakým způsobem podpořit, tak vás tímto způsobem vyzývám tak učinit. Děkuji.

Tímto je ukončen hlavní text práce. Další materiály jsou k dispozici v přílohách.

Literatura

[1] render - definition of render by The Free Dictionary. Dostupné z: http://www.webcitation.org/6ZYntv1PM

[2] Yao, J.; Pan, Z.; Zhang, H.: A Distributed Render Farm System for Animation Production. In *Entertainment Computing–ICEC 2009*, Springer, s. 264–269. Dostupné z: http://link.springer.com/chapter/10.1007/978-3-642-04052-8_31

[3] Chaos Group | V-Ray Render Farms. Dostupné z: http://www.webcitation.org/6ZYnlhEOA

[4] Online render-farm services. Full list of all remote render farms with prices and descriptions. Dostupné z: http://www.webcitation.org/6ZYnmN3U0

[5] Render Farm Managers Comparison. Dostupné z: http://www.webcitation.org/6ZYo1XUuW

[6] render queue // openSourceVFX.org. Dostupné z: http://www.webcitation.org/6ZYo3Ybtj

[7] SquidNet Network Render Manager (. Dostupné z: http://www.webcitation.org/6ZYo9jOle

[8] 4 Easy Ways to Speed Up Blender Cycles - Blender Guru. Dostupné z: http://www.webcitation.org/6ZYnSGWco

[9] why GPU Rendering - FurryBall - Incredibly fast GPU render. Dostupné z: http://www.webcitation.org/6ZYoCO5tl

[10] What is GPU Computing? | High-Performance Computing | NVIDIA | NVIDIA. Dostupné z: http://www.webcitation.org/6ZYoByazC

[11] GPU Rendering vs. CPU Rendering – A method to compare render times with empirical benchmarks | BOXX Technologies Blog. Dostupné z: http://www.webcitation.org/6ZYnk7kwo

[12] Build Your Own Render Farm | ExtremeTech. Dostupné z: http://www.webcitation.org/6ZYnWI2Bi

[13] How To: Building Your Own Render Farm - Introduction. Dostupné z: http://www.webcitation.org/6ZYnl7k0g

[14] Render farms costs & pricing | RenderStreet Blog. Dostupné z: http://www.webcitation.org/6ZYo2N9m4

[15] The best price / performance ratio - Ranch Computing (Cost estimator). Dostupné z: http://www.webcitation.org/6ZYoAkU93

[16] Render-farm service. RANCH Computing contacts. Rendering price. Render farm power. Soft details. Comments and reviews. Dostupné z: http://www.webcitation.org/6ZYo30bPj

[17] The best price / performance ratio - Ranch Computing (Pricing). Dostupné z: http://www.webcitation.org/6ZYoBE4BD

[18] Home - Render Farm | renderspell.com. Dostupné z: http://www.webcitation.org/6ZYnoeXVY

[19] Precios de render online. Dostupné z: http://www.webcitation.org/6ZYnsFtpu

[20] Pricing | GarageFarm.NET – The cheapest render farm, prices from $0.004/Ghz. Dostupné z: http://www.webcitation.org/6ZYntMWVn

[21] Cloud Rendering | Autodesk 360. Dostupné z: http://www.webcitation.org/6ZYnadgjo

[22] Autodesk A360 Cloud Credits FAQ | Account Management | Autodesk Knowledge Network. Dostupné z: http://www.webcitation.org/6ZYnGj9Od

[23] Price - Pearender - Fast Stills Rendering. Dostupné z: http://www.webcitation.org/6ZYnskJVm

[24] About - Pearender - Fast Stills Rendering. Dostupné z: http://www.webcitation.org/6ZYnLdOOL

[25] Getting started. Dostupné z: http://www.webcitation.org/6ZYnbHa23

[26] BoincIntro – BOINC. Dostupné z: http://www.webcitation.org/6ZYnVW5NN

[27] O projektu SETI@home. Dostupné z: http://www.webcitation.org/6ZYnmwdWt

[28] World Community Grid - Research Overview. Dostupné z: http://www.webcitation.org/6ZYoCyNAM

[29] Sheep it Render Farm (Home). Dostupné z: http://www.webcitation.org/6ZYo4Uxbs

[30] Sheep it Render Farm (FAQ). Dostupné z: http://www.webcitation.org/6ZYo58V1i

[31] Sheep it Render Farm (Stats). Dostupné z: http://www.webcitation.org/6ZYo5jFaa

[32] CESNET | Náročné výpočty (MetaCentrum). Dostupné z: http://www.webcitation.org/6ZYnZgV58

[33] AWS | Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting. Dostupné z: http://www.webcitation.org/6ZYnUUBtK

[34] ARK | Intel® Core™ i5-3470 Processor (6M Cache, up to 3.60 GHz). Dostupné z: http://www.webcitation.org/6ZZ8bi2Dg

[35] ARK | Intel® Core™ i5-4570S Processor (6M Cache, up to 3.60 GHz). Dostupné z: http://www.webcitation.org/6ZZ8cevoS

[36] ARK | Intel® Core™ i5-2400 Processor (6M Cache, up to 3.40 GHz). Dostupné z: http://www.webcitation.org/6ZZ8d5MJf

[37] GeForce GTX 480 | Specifications | GeForce. Dostupné z: http://www.webcitation.org/6ZZ8eopEp

[38] GeForce GTX 560 Ti | Specifications | GeForce. Dostupné z: http://www.webcitation.org/6ZZ8fYXgh

[39] Calculate Costs for Rendering - Rebus Render Farm Service. Dostupné z: http://www.webcitation.org/6ZZ8g7jWO

[40] RenderStorm Pricing / Calculator. Dostupné z: http://www.webcitation.org/6ZZ8iJ1dc

[41] Cost Calculator | GarageFarm.NET – The cheapest render farm, prices from $0.004/Ghz. Dostupné z: http://www.webcitation.org/6ZZ8izTtL

[42] Cinebench Ranking Benchmark Results R15 Scores Database. Dostupné z: http://www.webcitation.org/6ZZ9EeaGh

[43] 3D Fluff Maxon Cinebench 15 Score Results. Dostupné z: http://www.webcitation.org/6ZZ9F74Xv

[44] 2.7x Cycles benchmark (Updated BMW). Dostupné z: http://www.webcitation.org/6ZZ8jUqmH

[45] Bathroom Cycles 2 | Blend Swap. Dostupné z: http://www.webcitation.org/6ZZ8lscLE

[46] Bathroom | Blend Swap. Dostupné z: http://www.webcitation.org/6ZZ8mPtZM

[47] The White Room Cycles | Blend Swap. Dostupné z: http://www.webcitation.org/6ZZ8mtV4v

[48] Bedroom With Cycles | Blend Swap. Dostupné z: http://www.webcitation.org/6ZZ8pYvfy

[49] Light Paths — Blender Reference Manual. Dostupné z: http://www.webcitation.org/6ZZ8q2UtK

[50] Redshift. Dostupné z: http://www.webcitation.org/6ZZ8qTdpP

[51] features - FurryBall - Incredibly fast GPU render. Dostupné z: http://www.webcitation.org/6ZZ8spzwZ

[52] GPU Rendering - V-Ray 3.0 for 3ds Max Help - Chaos Group Help. Dostupné z: http://www.webcitation.org/6ZZ8tPHs1

[53] NVIDIA Advanced Rendering: Home. Dostupné z: http://www.webcitation.org/6ZZ8tzRu8

[54] Processing.org. Dostupné z: http://www.webcitation.org/6ZZ8vTRDl

[55] vvvv - a multipurpose toolkit | vvvv. Dostupné z: http://www.webcitation.org/6ZZ8xKDS9

[56] V-Ray for 3ds Max - Testimonials | Chaos Group. Dostupné z: http://www.webcitation.org/6ZZ8y2P7J

[57] V-Ray for 3ds Max - Rendering Software | Chaos Group. Dostupné z: http://www.webcitation.org/6ZZ8yVboA

[58] V-Ray for Maya - Rendering Software | Chaos Group. Dostupné z: http://www.webcitation.org/6ZZ90ZukT

[59] V-Ray for Rhino - Rendering Software | Chaos Group. Dostupné z: http://www.webcitation.org/6ZZ913vFM

[60] V-Ray for Softimage - Rendering Software | Chaos Group. Dostupné z: http://www.webcitation.org/6ZZ91Vu3f

[61] V-Ray for MODO | Chaos Group. Dostupné z: http://www.webcitation.org/6ZZ94LyTK

[62] V-Ray for NUKE | Chaos Group. Dostupné z: http://www.webcitation.org/6ZZ94oHPi

[63] V-Ray Standalone | Chaos Group. Dostupné z: http://www.webcitation.org/6ZZ95pLgu

[64] V-Ray pro školy. Dostupné z: http://www.webcitation.org/6ZZ97wYat

[65] V-Ray Scene Exporter - V-Ray 3.0 for 3ds Max Help - Chaos Group Help. Dostupné z: http://www.webcitation.org/6ZZ99AWdv

[66] V-Ray Common - V-Ray 3.0 for Maya Help - Chaos Group Help. Dostupné z: http://www.webcitation.org/6ZZ9AITxi

[67] Translator - V-Ray 3.0 for Softimage - Chaos Group Help. Dostupné z: http://www.webcitation.org/6ZZ9DRGKV

[68] Installation of V-Ray for SketchUp - V-Ray 2.0 for SketchUp Help - Chaos Group Help. Dostupné z: http://www.webcitation.org/6ZZ9Dy0cc

[69] O2 | Ověřte si dostupnost internetu a vyhrajte. Dostupné z: http://www.webcitation.org/6ZZ9FXfQs

[70] Horn, D. R.; Sugerman, J.; Houston, M.; aj.: Interactive kd tree GPU raytracing. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, ACM, s. 167–174. Dostupné z: http://dl.acm.org/citation.cfm?id=1230129

[71] Improved V-Ray RT - V-Ray 3.0 for Maya. Dostupné z: http://www.webcitation.org/6ZZ9GDILC

[72] Cycles - BlenderWiki. Dostupné z: http://www.webcitation.org/6ZZ9GraSy

[73] Arion stand-alone. Dostupné z: http://www.webcitation.org/6ZZ9HQKUw

[74] Why no GPU? Dostupné z: http://www.webcitation.org/6ZZ9IE3F2

[75] ArchiCAD 18 System Requirements | GRAPHISOFT Australia. Dostupné z: http://www.webcitation.org/6ZZ9Iphok

[76] MAXON | 3D FOR THE REAL WORLD: System Requirements. Dostupné z: http://www.webcitation.org/6ZZ9JH8BU

[77] Maya User's Guide: Network rendering with Backburner. Dostupné z: http://www.webcitation.org/6ZZA8IAQg

[78] 3ds Max Help: Network Rendering. Dostupné z: http://www.webcitation.org/6ZZA8pR4Y

[79] Backburner 2014 Windows version for 3ds Max/3ds Max Design 2014 | 3ds Max | Autodesk Knowledge Network. Dostupné z: http://www.webcitation.org/6ZZA9JBrz

[80] Installation and Configuration Guide for Linux Workstations: Backburner. Dostupné z: http://www.webcitation.org/6ZZA9t3rj

[81] Backburner Command-Line Control | 3ds Max | Autodesk Knowledge Network. Dostupné z: http://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/3DSMax/files/GUID-F23992DB-19BD-423A-A97C-5CC157328E63-htm.html

[82] SquidNet - Purchase. Dostupné z: http://www.squidnetsoftware.com/purchases.html

[83] SquidNet - DownLoads. Dostupné z: http://www.squidnetsoftware.com/downloads.html

[84] SquidNet User's Guide, str. 50, 135. Dostupné z: http://bit.ly/squidnet_manual

[85] SquidNet - OverView. Dostupné z: http://www.squidnetsoftware.com/overview.html

[86] SquidNet - Documentation. Dostupné z: http://www.squidnetsoftware.com/documentation.html

[87] DrQueue - DrQueue development (First commits). Dostupné z: https://ssl.drqueue.org/redmine/projects/drqueue/repository/revisions?page=291

[88] Jupyter and the future of IPython — IPython. Dostupné z: http://ipython.org/

[89] Revize 8ab38c04 - DrQueueIPython - DrQueue development. Dostupné z: https://ssl.drqueue.org/redmine/projects/drqueueipython/repository/revisions/8ab38c046584c3e9d6f54fc5e3b4b848b06acee0

[90] SomeDrQueueUsers - DrQueue - DrQueue development. Dostupné z: https://ssl.drqueue.org/redmine/projects/drqueue/wiki/SomeDrQueueUsers

[91] kaazoo DrQueueIPython - GitHub. Dostupné z: https://github.com/kaazoo/DrQueueIPython

[92] SetupAndConfiguration - DrQueueIPython - DrQueue development. Dostupné z: https://ssl.drqueue.org/redmine/projects/drqueueipython/wiki/SetupAndConfiguration

[93] Startup - DrQueueIPython - DrQueue development. Dostupné z: https://ssl.drqueue.org/redmine/projects/drqueueipython/wiki/Startup

[94] DrQueueIPython Command Templates. Dostupné z: https://github.com/kaazoo/DrQueueIPython/tree/master/etc

[95] Command line - DrQueueIPython - DrQueue development. Dostupné z: https://ssl.drqueue.org/redmine/projects/drqueueipython/wiki/Command_line

[96] ssl.drqueue.org Git - DrQueueOnRails.git/log. Dostupné z: https://ssl.drqueue.org/gitweb?p=DrQueueOnRails.git;a=log;h=refs/heads/rails3_mongodb

[97] Wiki - DrQueueIPython - DrQueue development. Dostupné z: https://ssl.drqueue.org/redmine/projects/drqueueipython/wiki

[98] Introduction · mikrosimage/OpenRenderManagement Wiki. Dostupné z: https://github.com/mikrosimage/OpenRenderManagement/wiki/Introduction

[99] OpenRenderManagement/LICENSE at master · mikrosimage/OpenRenderManagement. Dostupné z: https://github.com/mikrosimage/OpenRenderManagement/blob/master/LICENSE

[100] Contributors to mikrosimage/OpenRenderManagement. Dostupné z: https://github.com/mikrosimage/OpenRenderManagement/graphs/contributors

[101] Tornado Web Server — Tornado 4.2 documentation. Dostupné z: http://www.tornadoweb.org/en/stable/

[102] mikrosimage OpenRenderManagement - GitHub. Dostupné z: https://github.com/mikrosimage/OpenRenderManagement

[103] Open Mikros Image - Puli. Dostupné z: http://opensource.mikrosimage.eu/puli.html

[104] CronHowto - Community Help Wiki. Dostupné z: https://help.ubuntu.com/community/CronHowto

[105] If no pools specified for worker then register to default pool · morevnaproject/OpenRenderManagement@6b3befd. Dostupné z: https://github.com/morevnaproject/OpenRenderManagement/commit/6b3befd44a43cde910625f514d893a63d00cbc02

[106] CGRU - Home. Dostupné z: http://cgru.info/

[107] CGRU - About. Dostupné z: http://cgru.info/about

[108] Contributors to CGRU/cgru. Dostupné z: https://github.com/CGRU/cgru/graphs/contributors

[109] CGRU - License. Dostupné z: http://cgru.info/license

[110] GNU Lesser General Public License v3.0 - GNU Project - Free Software Foundation. Dostupné z: http://www.gnu.org/licenses/lgpl.html

[111] CGRU - Downloads. Dostupné z: http://cgru.info/downloads

[112] CGRU - Afanasy Server. Dostupné z: http://cgru.info/afanasy/server

[113] CGRU - Afanasy Render Host. Dostupné z: http://cgru.info/afanasy/render

[114] CGRU - Afanasy API. Dostupné z: http://cgru.info/afanasy/api

[115] HTCondor - What is HTCondor? Dostupné z: http://research.cs.wisc.edu/htcondor/description.html

[116] HTCondor - Home. Dostupné z: http://research.cs.wisc.edu/htcondor/index.html

[117] Coleman, N.: Distributed policy specification and interpretation with classified advertisements. In *Practical Aspects of Declarative Languages*, Springer, s. 198–211. Dostupné z: http://link.springer.com/chapter/10.1007/978-3-642-27694-1_15

[118] HTCondor - Classified Advertisements. Dostupné z: http://research.cs.wisc.edu/htcondor/classad/classad.html

[119] HTCondor - License Information. Dostupné z: http://research.cs.wisc.edu/htcondor/license.html

[120] HTCondor - 7.2 Microsoft Windows. Dostupné z: http://research.cs.wisc.edu/htcondor/manual/v8.2/7_2Microsoft_Windows.html

[121] Condor Administration. Dostupné z: http://research.cs.wisc.edu/htcondor/tutorials/fermi-2005/admin/admin_handout/

[122] HTCondor - 5.3 The Grid Universe. Dostupné z: http://research.cs.wisc.edu/htcondor/manual/v7.9/5_3Grid_Universe.html

[123] HTCondor - Directed Acyclic Graph Manager. Dostupné z: http://research.cs.wisc.edu/htcondor/dagman/dagman.html

[124] Couvares, P.; Kosar, T.; Roy, A.; aj.: Workflow management in condor. In *Workflows for e-Science*, Springer, s. 357–375. Dostupné z: http://link.springer.com/chapter/10.1007/978-1-84628-757-2_22

[125] Meehean, J.; Livny, M.: A service migration case study: Migrating the Condor schedd. In *Midwest Instruction and Computing Symposium*. Dostupné z: http://micsymposium.org/mics_2005/papers/paper58.pdf

[126] HTCondor - 2.8 Java Applications. Dostupné z: http://research.cs.wisc.edu/htcondor/manual/v7.8/2_8Java_Applications.html

[127] Raman, R.: Overview of the ClassAd Language. Dostupné z: http://research.cs.wisc.edu/htcondor/slides/classad-talk.ps

[128] HTCondor - Dynamic Slots - Distributed and Parallel Computing Lab. Dostupné z: http://dpl.cs.uwec.edu/blog/2012/08/dynamics_slots.html

[129] HTCondor Quick Start Guide. Dostupné z: http://research.cs.wisc.edu/htcondor/manual/quickstart.html

[130] HTCondor - 11. Command Reference Manual (man pages). Dostupné z: http://research.cs.wisc.edu/htcondor/manual/current/11_Command_Reference.html

[131] HTCondor - 6.1 Web Service. Dostupné z: http://research.cs.wisc.edu/htcondor/manual/v8.0/6_1Web_Service.html

[132] HTCondor - 3.2 Installation. Dostupné z: http://research.cs.wisc.edu/htcondor/manual/v7.8/3_2Installation.html

[133] Gkion, M.; Patoli, M. Z.; Al-Barakati, A.; aj.: Collaborative 3D digital content creation exploiting a Grid network. In *Information and Communication Technologies, 2009. ICICT'09. International Conference on*, IEEE, s. 35–39. Dostupné z: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5267221

[134] Chan, L.; Stowe, J. A.: Dopis od C.O.R.E. Feature Animation pro Condor o využití systému pro film The Wild. Dostupné z: http://research.cs.wisc.edu/htcondor/images/WildLetter.pdf

[135] clondike/README.md at master · FIT-CVUT/clondike. Dostupné z: https://github.com/FIT-CVUT/clondike/blob/master/README.md

[136] Contributors to FIT-CVUT/clondike. Dostupné z: https://github.com/FIT-CVUT/clondike/graphs/contributors

[137] Rákosník, J.: Úpravy Clondike pro představení open source komunitě. Dostupné z: https://dip.felk.cvut.cz/browse/pdfcache/rakosjir_2013bach.pdf

[138] Vicher, T.: Výroba produkční verze Clondike. Dostupné z: https://dip.felk.cvut.cz/browse/details.php?f=F8&d=K104&y=2013&a=vichetom&t=bach

[139] Nový, Z.: Výroba sítové verze Clondike. Dostupné z: https://dip.felk.cvut.cz/browse/details.php?f=F8&d=K104&y=2013&a=novyzde3&t=bach

[140] Tvrdík, P.: Implementace BitTorrent discovery protokolu do Clondike. Dostupné z: http://storm2.fit.cvut.cz:9000/theses/359.pdf

[141] Clondike - Production version. Dostupné z: http://clondike.fit.cvut.cz/production-version/

[142] Kacer, M.; Langr, D.; Tvrdik, P.: Clondike: Linux cluster of non-dedicated workstations. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, ročník 1, IEEE, s. 574–581. Dostupné z: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1558605

[143] LuxRender - description. Dostupné z: http://www.webcitation.org/6ZZ9JpBpi

[144] Home | YafaRay. Dostupné z: http://www.webcitation.org/6ZZ9KMbe8

[145] V-Ray Standalone Pricing & Lincensing | Chaos Group. Dostupné z: http://www.webcitation.org/6ZZ9KvpXA

[146] Buy or Upgrade mental ray Standalone 2015 | Subscription Options | Autodesk. Dostupné z: http://www.webcitation.org/6ZZ9Lv3WD

[147] NVIDIA Advanced Rendering: NVIDIA mental ray. Dostupné z: http://www.webcitation.org/6ZZ9N9LJV

[148] License - blender.org - Home of the Blender project - Free and Open 3D Creation Software. Dostupné z: http://www.webcitation.org/6ZZ9Nb8fe

[149] GPU Rendering — Blender Reference Manual. Dostupné z: http://www.webcitation.org/6ZZ9O34oe

[150] Download - blender.org - Home of the Blender project - Free and Open 3D Creation Software. Dostupné z: http://www.webcitation.org/6ZZ9ORs6d

[151] Artlantis. Dostupné z: http://www.webcitation.org/6ZZ9P0AkJ

[152] mental ray's GPU accelerated GI engine prototype | Inside mental ray. Dostupné z: http://www.webcitation.org/6ZZ9Mb47l

[153] blender.org - Movies. Dostupné z: http://www.webcitation.org/6ZZ9PVQlu

[154] Blender Code | developer musings on Blender. Dostupné z: http://www.webcitation.org/6ZZ9Q1PVI

[155] Command Line — Blender Reference Manual. Dostupné z: http://www.webcitation.org/6ZZA7mfgU

[156] SSO Benefits | CCS. Dostupné z: http://www.webcitation.org/6ZZ9QRBtY

[157] ČVUT v Praze – Výhody a nevýhody přihlášení SSO. Dostupné z: http://www.webcitation.org/6ZZ9lrVVk

[158] SSO (Single Sign-On) Simplified. Dostupné z: http://www.webcitation.org/6ZZ9RIxqb

[159] ČVUT v Praze – Single Sign On (SSO) - Shibboleth. Dostupné z: http://www.webcitation.org/6ZZ9Rr4eE

[160] Shibboleth Consortium - How Shibboleth Works. Dostupné z: http://www.webcitation.org/6ZZ9ShaSZ

[161] ČVUT v Praze – federace cvutID. Dostupné z: http://www.webcitation.org/6ZZ9T8KjY

[162] Členové [eduID.cz]. Dostupné z: http://www.webcitation.org/6ZZ9TccSN

[163] IEEE Xplore - Athens / Shibboleth. Dostupné z: http://www.webcitation.org/6ZZ9UBNPd

[164] OAuth 2.0 (Main.oauth2). Dostupné z: http://www.webcitation.org/6ZZ9Uhqvc

[165] Apps Manager BETA. Dostupné z: http://www.webcitation.org/6ZZ9VTSlw

[166] Using OAuth 2.0 to Access Google APIs. Dostupné z: http://www.webcitation.org/6ZZ9VuSb4

[167] HTTP API - Guide - SoundCloud Developers. Dostupné z: http://www.webcitation.org/6ZZ9WMDGq

[168] RFC 6749 - The OAuth 2.0 Authorization Framework. Dostupné z: http://www.webcitation.org/6ZZ9WuGAG

[169] RFC 6749 - The OAuth 2.0 Authorization Framework (Section-4.1). Dostupné z: http://www.webcitation.org/6ZZ9XdeA4

[170] An Introduction to OAuth 2 | DigitalOcean. Dostupné z: http://www.webcitation.org/6ZZ9Y65ia

[171] Usermap API (Main.usermap-api). Dostupné z: http://www.webcitation.org/6ZZ9Z9Qbg

[172] Top 5 Web Application Languages – Red27. Dostupné z: http://www.webcitation.org/6ZZ9ZuFRQ

[173] AngularJS — Superheroic JavaScript MVW Framework. Dostupné z: http://www.webcitation.org/6ZZ9aYaZQ

[174] Web application framework - DocForge. Dostupné z: http://docforge.com/wiki/Framework

[175] Laravel - The PHP Framework For Web Artisans. Dostupné z: http://www.webcitation.org/6ZZ9brH9f

[176] Web framework rankings | HotFrameworks. Dostupné z: http://www.webcitation.org/6ZZ9xPSbz

[177] Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework. Dostupné z: http://www.webcitation.org/6ZZ9y3FqL

[178] ASP.NET | The ASP.NET Site. Dostupné z: http://www.asp.net/

[179] Web framework rankings | HotFrameworks. Dostupné z: http://www.webcitation.org/6ZZ9zIzbG

[180] Madeyski, L.: *Test-Driven Development, str. 200*. Springer Berlin Heidelberg, ISBN 978-3-642-04287-4, 978-3-642-04288-1. Dostupné z: http://link.springer.com/10.1007/978-3-642-04288-1

[181] Winter Arena 2015. Dostupné z: http://www.webcitation.org/6ZZA0IN11

[182] BASS DROP. Dostupné z: http://www.webcitation.org/6ZZ9zoMOW

[183] RSpec: Behaviour Driven Development for Ruby. Dostupné z: http://www.webcitation.org/6ZZA0lAT2

[184] A Guide to Testing Rails Applications — Ruby on Rails Guides. Dostupné z: http://www.webcitation.org/6ZZA1HIUD

[185] The Ruby Toolbox - SQL Database Adapters. Dostupné z: http://www.webcitation.org/6ZZA1m8yM

[186] The Ruby Toolbox - Rails Authentication. Dostupné z: http://www.webcitation.org/6ZZA29G7U

[187] The Ruby Toolbox - Rails File Uploads. Dostupné z: http://www.webcitation.org/6ZZA2WyCq

[188] Sinatra: README. Dostupné z: http://www.webcitation.org/6ZZA2zwBX

[189] AWS | Amazon Simple Storage Service (S3) - Online Cloud Storage for Data & Files. Dostupné z: http://www.webcitation.org/6ZZA3TSCr

[190] Parsing an HTML/XML document - Nokogiri. Dostupné z: http://www.webcitation.org/6ZZA3ucGB

[191] Postman | Supercharge your API workflow. Dostupné z: http://www.webcitation.org/6ZZA4sEk0

[192] Transactional DDL in PostgreSQL: A Competitive Analysis - PostgreSQL wiki. Dostupné z: http://www.webcitation.org/6ZZA5KOmE

[193] PostgreSQL: Documentation: 9.4: JSON Types. Dostupné z: http://www.webcitation.org/6ZZA5qNuB

[194] How to Use Foundation Sass | Foundation Docs. Dostupné z: http://www.webcitation.org/6ZZA6HWw6

[195] Thoughtbot Paperclip. Dostupné z: http://www.webcitation.org/6ZZA6pDfe

[196] Dominik Jančík / fitrender_compute-adaptor | GitLab. Dostupné z: https://gitlab.fit.cvut.cz/jancidom/fitrender_compute-adaptor

[197] Dominik Jančík / fitrender_frontend | GitLab. Dostupné z: https://gitlab.fit.cvut.cz/jancidom/fitrender_frontend

[198] Dominik Jančík / fitrender_common | GitLab. Dostupné z: https://gitlab.fit.cvut.cz/jancidom/fitrender_common

[199] Dominik Jančík / fitrender_adaptor_condor | GitLab. Dostupné z: https://gitlab.fit.cvut.cz/jancidom/fitrender_adaptor_condor

[200] HTCondor - 4.1 HTCondor's ClassAd Mechanism. Dostupné z: http://research.cs.wisc.edu/htcondor/manual/v7.8/4_1HTCondor_s_ClassAd.html

[201] omniauth-fitcvut-oauth2 | RubyGems.org | your community gem host. Dostupné z: https://rubygems.org/gems/omniauth-fitcvut-oauth2

[202] JSON. Dostupné z: http://json.org/

[203] The Official YAML Web Site. Dostupné z: http://yaml.org/

[204] SOAP Specifications. Dostupné z: http://www.w3.org/TR/soap/

[205] How REST replaced SOAP on the Web: What it means to you. Dostupné z: http://www.infoq.com/articles/rest-soap

[206] What does CLI stand for? Dostupné z: http://www.abbreviations.com/CLI

[207] Creating DSLs with Ruby. Dostupné z: http://www.artima.com/rubycs/articles/ruby_as_dsl3.html

[208] View Frustum Culling » Lighthouse3d.com. Dostupné z: http://www.lighthouse3d.com/tutorials/view-frustum-culling/

[209] GPU Gems - Chapter 3. Inside Geometry Instancing. Dostupné z: http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter03.html

[210] HTCondor - 2.5 Submitting a Job. Dostupné z: http://research.cs.wisc.edu/htcondor/manual/v7.8/2_5Submitting_Job.html

Příloha A

Seznam použitých zkratek

XML Extensible markup language. Textový formát pro reprezentaci dat.

JSON JavaScript Object Notation. Textový formát pro reprezentaci dat.[202]

YAML YAML Ain't Markup Language. Textový formát pro reprezentaci dat.[203]

SOAP Stavově orientovaný způsob vystavení webových služeb. Pro přenos dat využívá primárně formát XML.[204]

REST Resource orientovaný způsob vystavení webových služeb. Zkratka znamanáá Representational State Transfer. Vyznačuje se jednoduchostí implementace a jednoznačností metod.[205]

CG Computer Graphics. Počítačová grafika. V práci používáno v kontextu renderování.

TDD Test Driven Development. Metodika vyvíjení softwaru, která klade důraz na využívání testovacích mechanik.[180]

CLI Command Line Interface, rozhraní příkazové řádky.[206]

DSL Domain Specific Language. Specifické rozšíření programovacího jazyka o prvky potřebné pro danou oblast.[207]

HW Hardware

SW Software

Příloha **B**

Seznam použitých odborných termínů

Frustum culling Omezení renderování na viditelnou oblast.[208]

Instancování Využití jednou uložené geometrie na více místech.[209]

Cluster Sada spolupracujících počítačů.

Uzel Člen clusteru.

Příloha C

Specifikace FITRender Compute API

Toto API jsem navrh tak, aby poskytovalo Frontendu možnost zadávání a správy úloh, správu jednotlivých výpočetních uzlů a konfiguraci celého clusteru a to bez předchozí znalosti konkrétního systému použitého pro výpočetní backendu.

Jedná se o REST API, které ctí význam jednotlivých HTTP metod a pro reprezentaci dat využívá formátu JSON.

Tento návrh počítá se sdíleným souborovým uložištěm. Bylo by ho však možné jednoduše rozšířit o metody podporující explicitní přenos souborů pro případ, že Frontend a Adaptér nemají přístup ke sdílenému uložišti.

C.1 Stavy

Uzly a úlohy mohou v průběhu času nabývat různých stavů.

API definuje následující stavy pro Nodes (Uzly):

- Idle (Neaktivní)
- Busy (Zaneprázdněn)
- Other (Jiný)

A tyto pro Jobs (Úlohy):

- Idle (Neaktivní, čeká)
- Running (Spuštěna)
- Completed (Dokončena)

- Failed (Selhala)
- Other (Jiný)

C.2 Metody

Metody spadají do čtyř kategorií: *Správa uzlů*, *Správa rendererů*, *Správa úloh* a *Globální nastavení*.

C.2.1 Přehled metod

Tabulka C.1 prezentuje přehled jednotlivých metod tohoto API.

Tabulka C.1: Přehled metod FITRender Compute API

Metoda	Endpoint	Význam
Správa uzlů		
GET	/nodes	Přehled uzlů
GET	/nodes/<id>	Informace o uzlu s daným ID, zejména dostupná nastavení
PATCH	/nodes/<id>	Změna nastavení uzlu s daným ID
Správa rendererů		
GET	/renderers	Přehled podporovaných rendererů
GET	/renderers/<id>	Informace o rendereru s daným ID, zejména dostupná nastavení pro renderer a scény.
PATCH	/renderers/<id>	Změna nastavení rendereru s daným ID
Správa úloh		
POST	/submit	Odeslání úlohy do clusteru
GET	/jobs/<id>	Informace o úloze s daným ID, zejména stav.
DELETE	/jobs/<id>	Zrušení úlohy s daným ID.
Globální nastavení		
GET	/options	Přehled dostupných globálních nastavení
PATCH	/options	Změna nastavení clusteru

Dále popisuji účel jednotlivých metod a formáty požadavků a odpovědí. U metod také uvádím možné chyby, pokud se nejedná pouze o interní chyby serveru s kódy formátu 5XX.

Odpovědi na upravující metody jsou následujícího formátu:

```
{
  "status": "200",
  "message": "Action completed successfuly."
}
```

C.2.1.1 Metoda GET /nodes

Účelem této metody je zobrazit stručný seznam všech dostupných uzlů zobrazující jejich identifikátor a aktuální stav.

Příklad odpovědí

```
[
  {
    "id": "node_id_1",
    "state": "idle"
  },
  {
    "id": "node_id_2",
    "state": "busy"
  }
]
```

C.2.1.2 Metoda GET /nodes/<id>

Tato metoda nabízí detailní pohled na uzel.

Příklad požadavku

```
GET https://render.fit.cz/nodes/1
```

Příklad odpovědi

```
[
  {
    "id": "node_id_1",
    "state": "idle",
    "options":
      [
        {
          "id": "scheduling",
          "value": "0-4,20-24",
          "default": "0-24",
          "description": "When to render",
          "read_only": false
        },
        {
          "id": "location",
          "value": "T9:350",
          "read_only":
        }
      ]
  }
]
```

Možné chyby

- 404 - Uzel nenalezen

C.2.1.3 Metoda PATCH /nodes/<id>

Tato metoda umožňuje změnu nastavení daného uzlu.

Příklad požadavku

```
PATCH https://render.fit.cz/nodes/1
scheduling=0-12
```

Možné chyby

- 404 - Uzel nenalezen
- 400 - Špatný požadavek. Nevalidní hodnota nastavení nebo nenalezené nastavení.

C.2.1.4 Metoda GET /renderers

Tato metoda nabízí přehled podporovaných rendererů.

Příklad odpovědi

```
[
  {
    "id": "Blender",
    "extension": "blend",
    "version": "2.74"
  }
]
```

C.2.1.5 Metoda GET /renderers/<id>

Tato metoda nabízí detailní pohled na daný renderer.

Příklad odpovědi

```
{
  "id": "Blender",
  "extension": "blend",
  "version": "2.74",
  "options":
    [
      {
        "id": "frame_granularity",
        "value": "0.5",
        "default": "1",
        "description": "Set a lower number for more tiles.",
        "read_only": false
      }
    ]
  "generator_options":
    [
      {
        "id": "frames",
        "default": "1",
        "description": "Frames to render."
      }
    ]
}
```

Možné chyby

- 404 - Renderer nenalezen

C.2.1.6 Metoda PATCH /renderers/<id>

Tato metoda umožňuje změnu nastavení daného rendereru. Funguje obdobně jako PATCH metoda pro uzel.

Možné chyby

- 404 - Renderer nenalezen
- 400 - Špatný požadavek. Nevalidní hodnota nastavení nebo nenalezené nastavení.

C.2.1.7 Metoda POST /submit

Tato metoda umožňuje zadání nové úlohy do výpočetního clusteru. Při zadávání je nutné zadat typ rendereru, cestu k souboru scény na sdíleném uložišti nebo tento soubor poslat s požadavkem a případná nastavení scény.

Jako návratová hodnota se očekává seznam jednoho či více úkolů, které byly na clusteru pro tuto scénu vytvořeny včetně cest k výsledným souborům nebo jiného identifikátoru.

Příklad požadavku

```
POST https://render.fit.cz/submit
renderer_id=Blender
path=/mnt/shared/scene.blend
options=
  {
    "frames": "4,6"
  }
```

Příklad odpovědi

```
[
  {
        "id": "23.2",
    "result_path": "/mnt/shared/renders/result_004.png",
    "name": "Frame 4"
  },
  {
        "id": "23.3",
    "result_path": "/mnt/shared/renders/result_006.png",
    "name": "Frame 6"
  },
]
```

Možné chyby

- 400 - Špatný požadavek. Nekompletní nastavení, nedosažitelná cesta nebo neexisutjící renderer.

C.2.1.8 Metoda GET /jobs

Tato metoda nabízí přehled spuštěných úloh.

```
[
  {
        "id": "23.2",
    "state": "completed"
  },
  {
        "id": "23.3",
    "state": "running"
  },
  {
        "id": "23.6",
    "state": "failed"
  },
  {
        "id": "24.2",
    "state": "completed"
  }
]
```

C.2.1.9 Metoda GET /jobs/<id>

Tato metoda nabízí pohled na jednu úlohu.

Příklad odpovědi

```
{
  "id": "23.3",
  "state": "running"
},
```

Možné chyby

- 404 - Úloha nenalezena

C.2.1.10 Metoda DELETE /jobs/<id>

Tato metoda nabízí možnost vznesení požadavku o zrušení úlohy.

Možné chyby

- 404 - Úloha nenalezena

C.2.1.11 Metoda GET/options

Tato metoda nabízí přehled možností nastavení clusteru.

Příklad odpovědi

```
[
  {
    "id": "render_path",
    "value": "/mnt/shared/renders_summer",
    "default": "/mnt/shared/renders",
    "description": "Where to save finished renders.",
    "read_only": false
  },
  {
    "id": "render_path",
    "value": "/mnt/shared/renders_summer",
    "default": "/mnt/shared/renders",
    "description": "Where to save finished renders.",
    "read_only": false
  },
]
```

C.2.1.12 Metoda PATCH /options

Tato metoda nabízí možnost změnit hodnotu nastavení clusteru, funguje obdobně jako patch metoda Uzlu.

Možné chyby

- 400 - Špatný požadavek. Nevalidní hodnota nastavení nebo nenalezené nastavení.

Příloha D

Uživatelský návod

D.1 Spuštění z dodaného obrazu VM

Tento obraz obsahuje předkonfigurované prostředí potřebné pro běh systému. Nabízí se tak možnost rychlého otestování.

Obraz je ve formátu pro virtualizační programy VMWare a je postaven na Linuxové distribuci openSUSE 13.2 Harlequin.

Root heslo je *condor*.

Pro běh systému je nutné spustit

- Službu databáze PostgreSQL, pomocí příkazu (jako root):

  ```
  /etc/init.d/postgresql start
  ```

- HTCondor, pomocí příkazu (jako root):

  ```
  condor_master
  ```

- Může chvíli trvat, než se HTCondor inicializuje. Běh systému je možné ověřit příkazem:

  ```
  condor_status
  ```

- FITRender Compute API implementaci:

  ```
  /home/condor/fitrender/01-start-adaptor.sh
  ```

- FITRender Compute API implementaci:

  ```
  /home/condor/fitrender/02-start-adaptor.sh
  ```

- A poté vstoupit do aplikace pomocí adresy:

  ```
  http://fitrender:3000
  ```

Na tuto adresu je nastaven callback autentizačního modulu a proto je nutné používat fitrender a localhost. Nastavení je provedeno v souboru /etc/hosts.

API implementace je umístěna na portu 9292.

Pro administrátorský přístup zvolte Sign In DEVELOPER a do obou polí vyplňte *admin*.

D.2 Vlastní konfigurace

Pokud nespouštíte program z přiloženého obrazu, tak je nutné si nakonfigurovat vhodné prostředí.

Je vyžadováno mít

- Ruby verze 2.2.2
- Bundler, pomocí něj se nainstalují další Ruby závislosti příkazem

  ```
  bundle install
  ```

  v adresáři každé komponenty.
- Bundler nastavený tak, aby poskytoval gemy implementace lokálně.
- Přístup k lokálnímu nebo vzdálenému PostgreSQL serveru
- V síti zprovozněn HTCondor a počítači s Adaptorem musí být povoleno zasílat úlohy do clusteru.
- Instalovaný Blender. Uživatel, pod kterým jsou spouštěný HTCondor úlohy ho musí mít ve své PATH proměnné.

D.2.1 Proměnné prostředí

Adaptér i Frontend očekávájí pro svůj běh nastavené následující hodnoty.

D.2.1.1 Adaptér

- FITRENDER_RENDER_PATH Cesta pro výsledné rendery. Uživatel, pod kterým se vykonává výpočet musí mít do této složky přístup k zápisu.
- FITRENDER_LOG_PATH Cesta pro HTCondor logování.
- FITRENDER_OUT_PATH Cesta pro výstup spuštěného rendereru.

D.2.1.2 Frontend

- FITRENDER_SCENE_PATH Cesta k uložení nahraných scén.
- FITRENDER_ADAPTOR_URL Adresa, na které je vystaveno FIT-Render Compute API.
- FIT_APP_ID App ID z ČVUT OAuth 2.0 aplikace.
- FIT_APP_SECRET App secret z ČVUT OAuth 2.0 aplikace.

OAuth aplikaci je možné si vytvořit pomocí AppsManageru [165].

Po splnění těchto požadavků je možné spustit službu. Fitrender_compute-adaptor je mpžné spustit příkazem

```
rackup
```

z podadresáře lib.

Frontend je pak možné spustit pomocí

```
rails s
```

odkudkoli v adresáři.

Příloha E

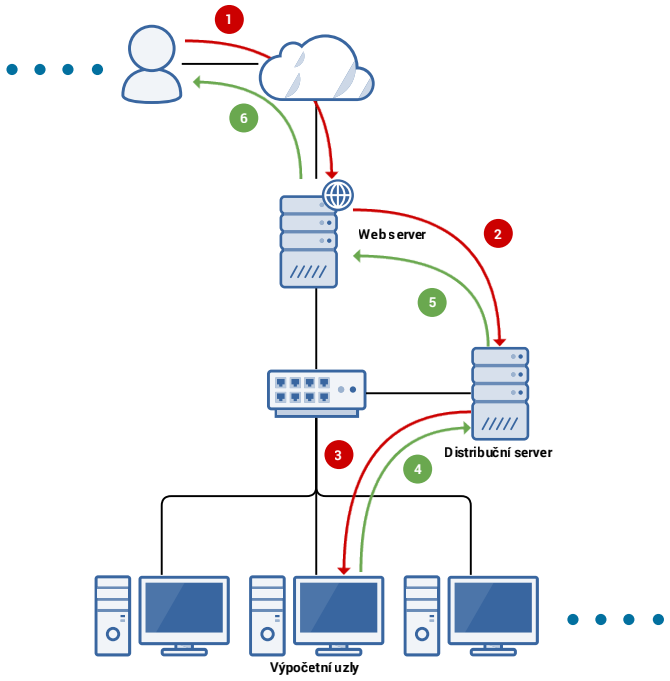
Dodatečné architektury

E.1 Architektura A3

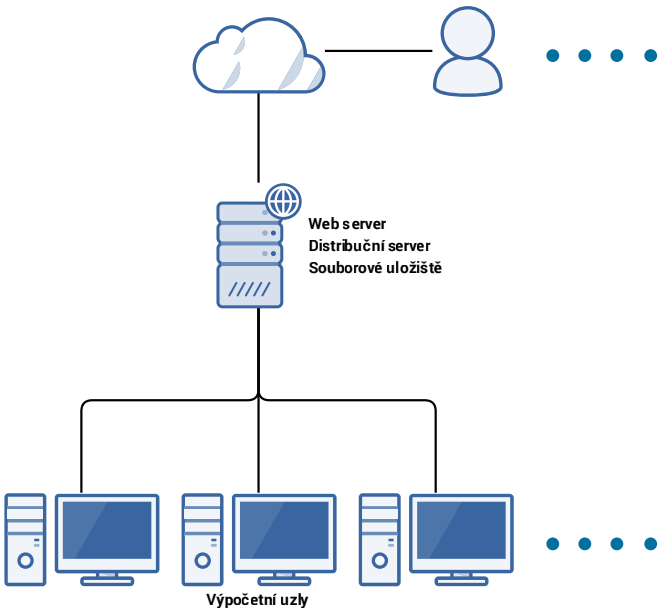
Tato architektura modeluje systém, který nemá mezi počítači sdílené nebo dostatečně velké uložiště. To vyžaduje explicitní kopírování souborů mezi jednotlivými součásti sítě. Se zvoleným backendem HTCondor je toto možné realizovat vzhledem k podporovanému přenosu souborů [210]. Pro účely render farmy je ale tento přístup vysoce nepraktický, jelikož výrazně zvyšuje režii a klade vyšší kapacitní nároky na jednotlivé uzly.

E.2 Architektura A4

Tato architektura zobrauje případ, kdy je veškerá funkcionalita poskytována jedním počítačem.



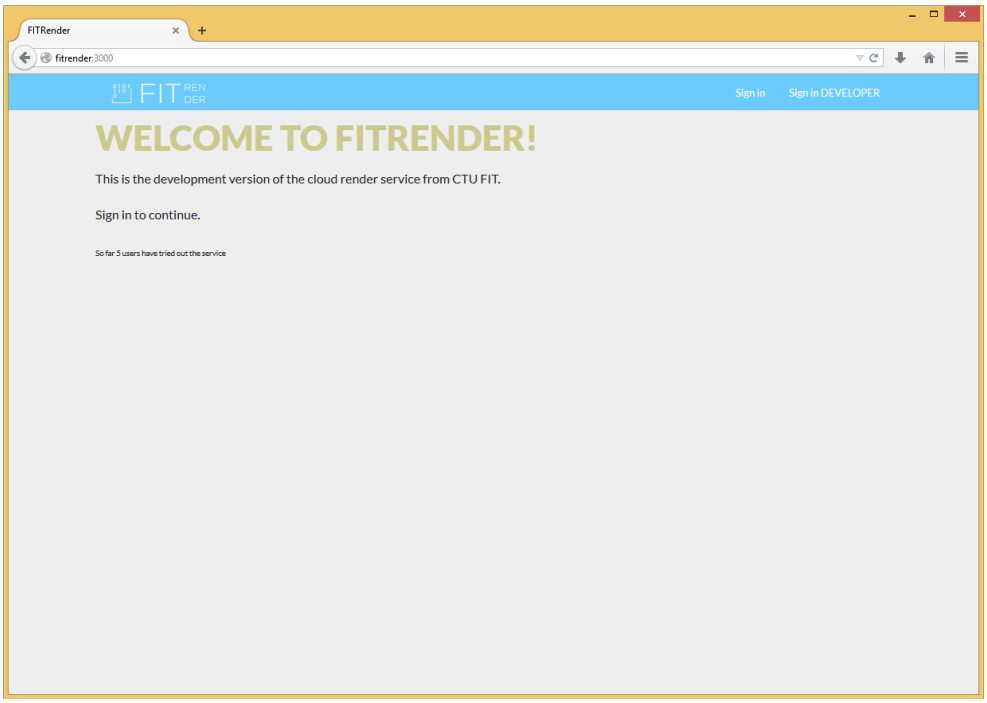
Obrázek E.1: Model architektury A3 s reprezentací zpracování scény.

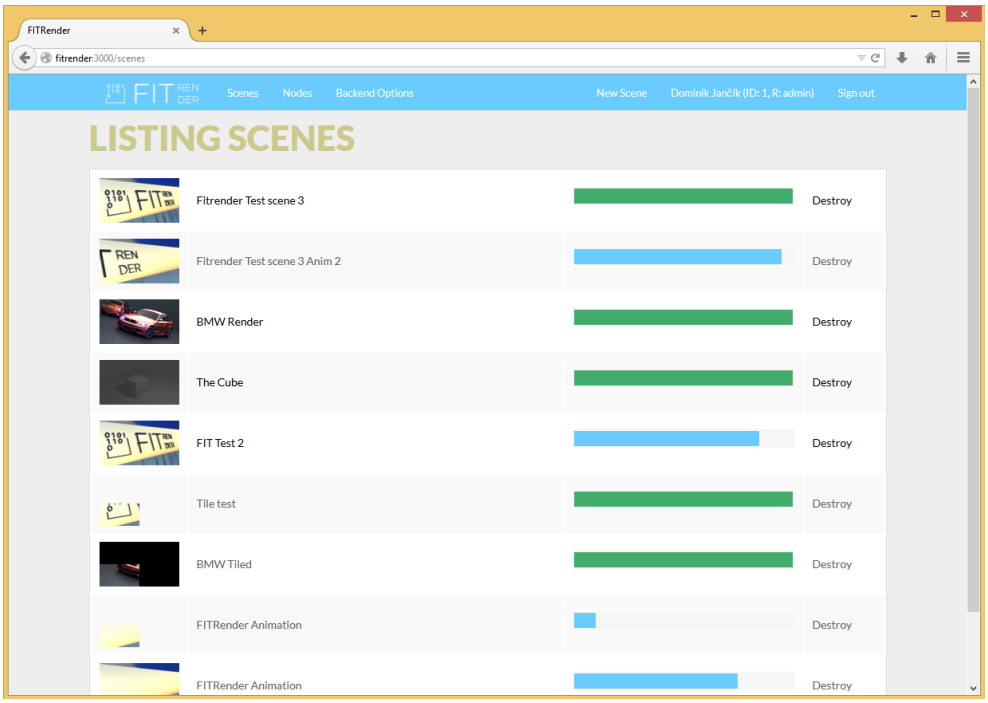


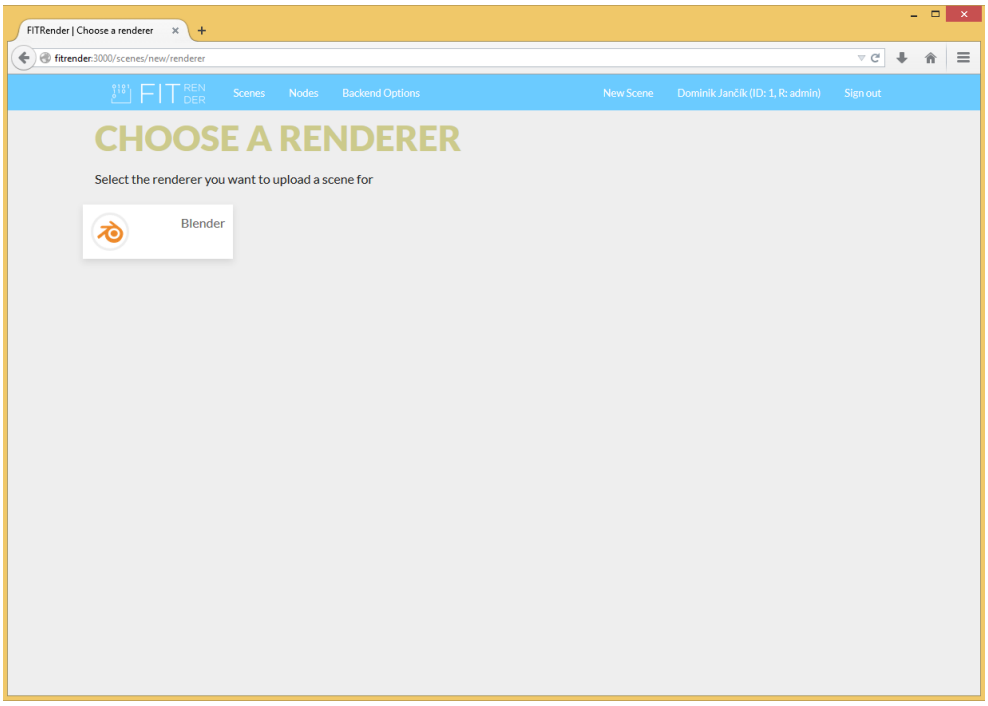
Obrázek E.2: Model architektury A4

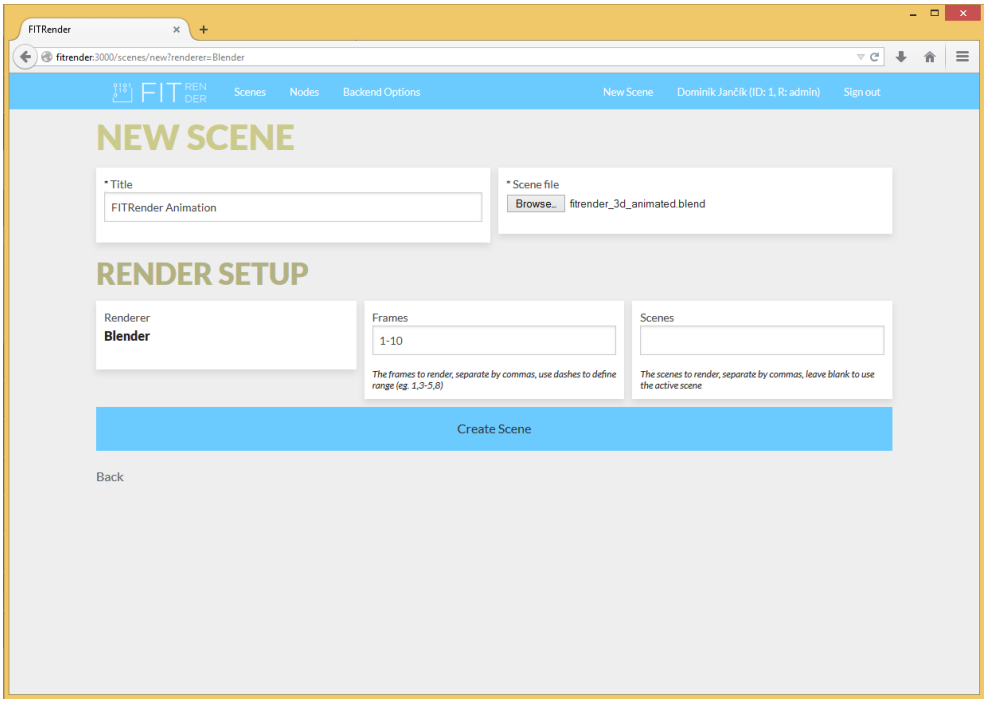
Příloha F

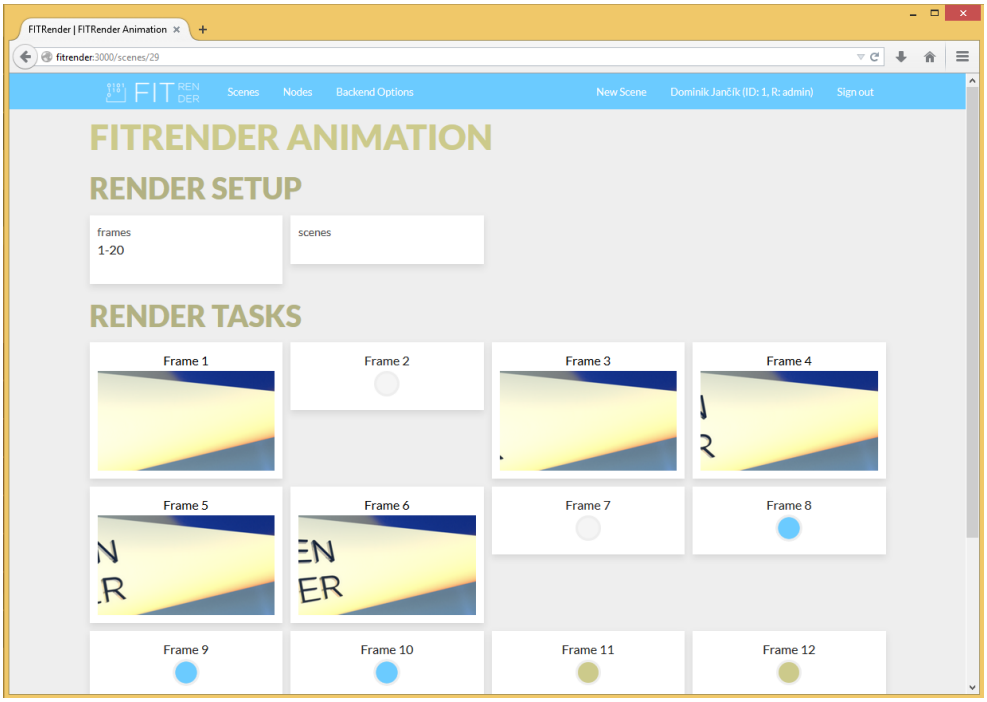
Screenshoty prototypové implementace

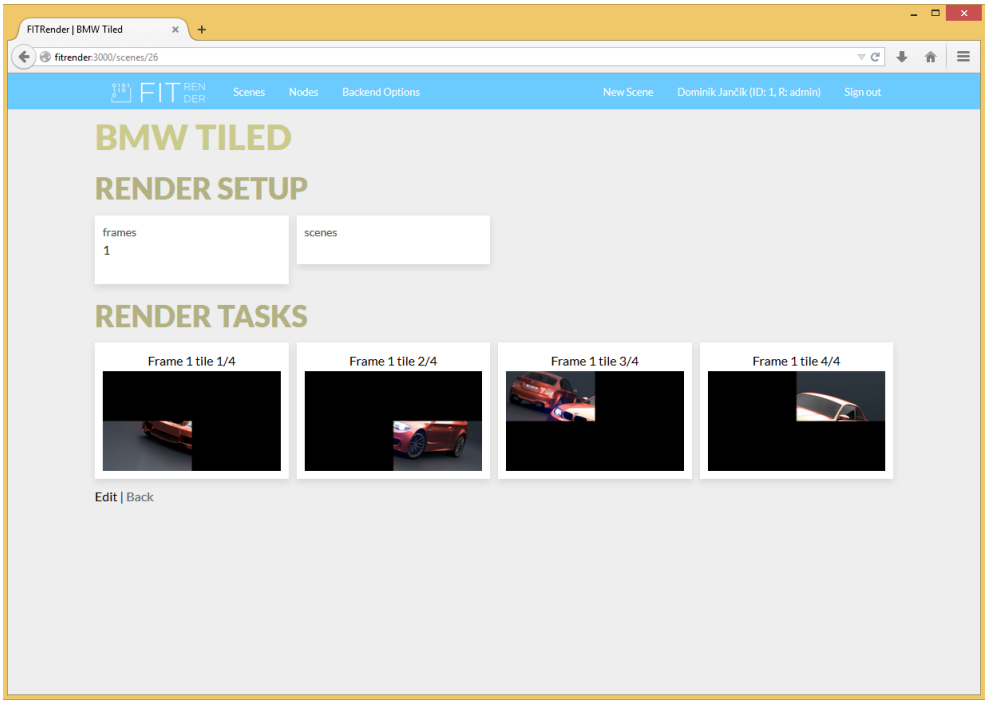


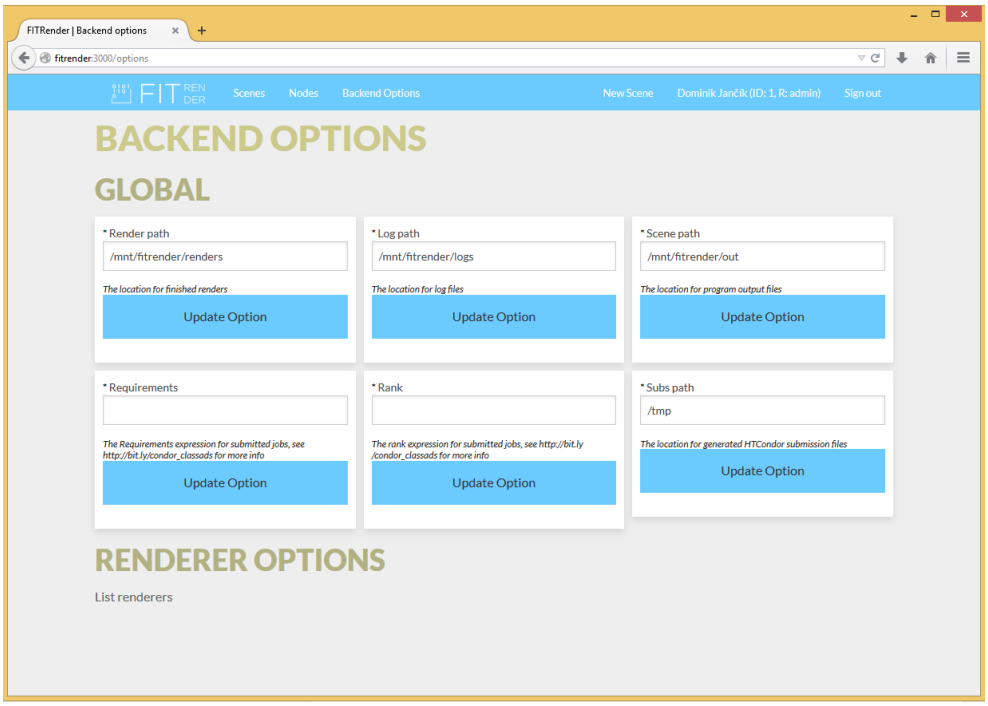

FITRender
fitrender:3000/scenes
Scenes
Nodes
Backend Options
New Scene
Dominik Jančík (ID: 1, R: admin)
Sign out
LISTING SCENES
Fitrender Test scene 3
Destroy
Fitrender Test scene 3 Anim 2
Destroy
BMW Render
Destroy
The Cube
Destroy
FIT Test 2
Destroy
Tile test
Destroy
BMW Tiled
Destroy
FITRender Animation
Destroy
FITRender Animation
Destroy

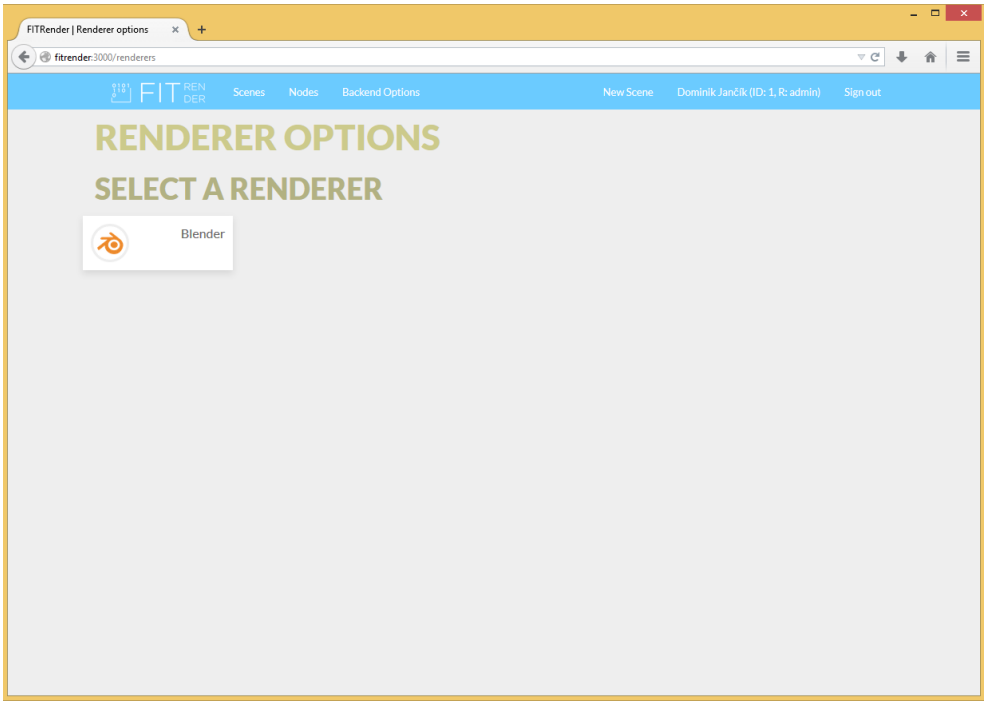

FITRender | Choose a renderer
fitrender:3000/scenes/new/renderer
Scenes
Nodes
Backend Options
New Scene
Dominik Jančík (ID: 1, R: admin)
Sign out
CHOOSE A RENDERER
Select the renderer you want to upload a scene for
Blender

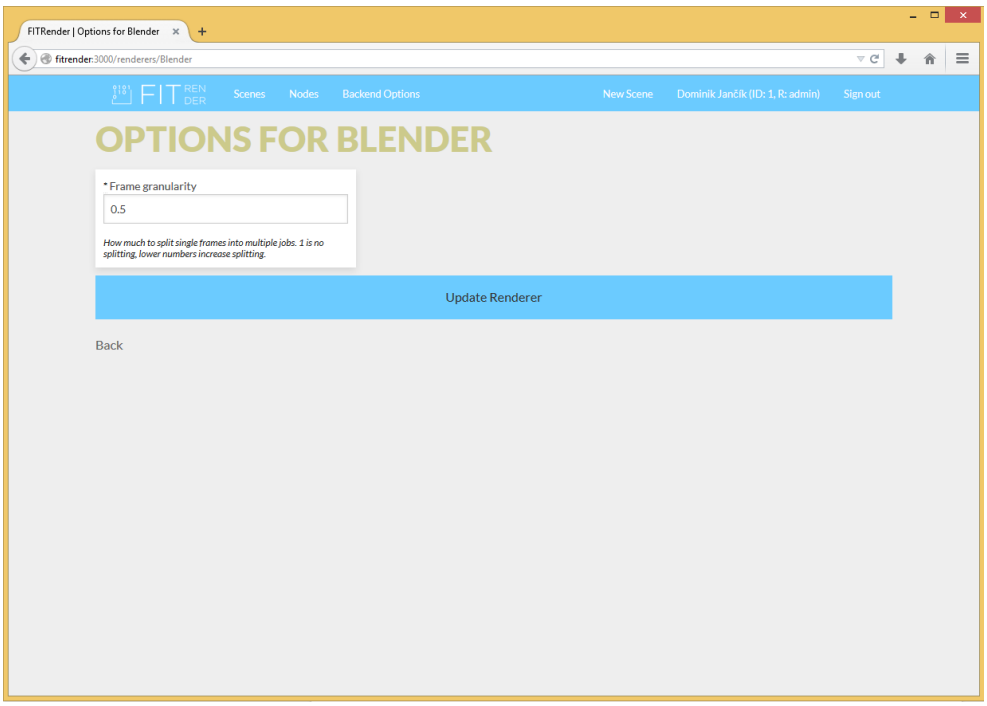

FITRender
fitrender:3000/scenes/new?renderer=Blender
FIT REN DER
Scenes
Nodes
Backend Options
New Scene
Dominik Jančík (ID: 1, R: admin)
Sign out
NEW SCENE
* Title
FITRender Animation
* Scene file
Browse..
fitrender_3d_animated.blend
RENDER SETUP
Renderer
Blender
Frames
1-10
The frames to render, separate by commas, use dashes to define range (eg. 1,3-5,8)
Scenes
The scenes to render, separate by commas, leave blank to use the active scene
Create Scene
Back

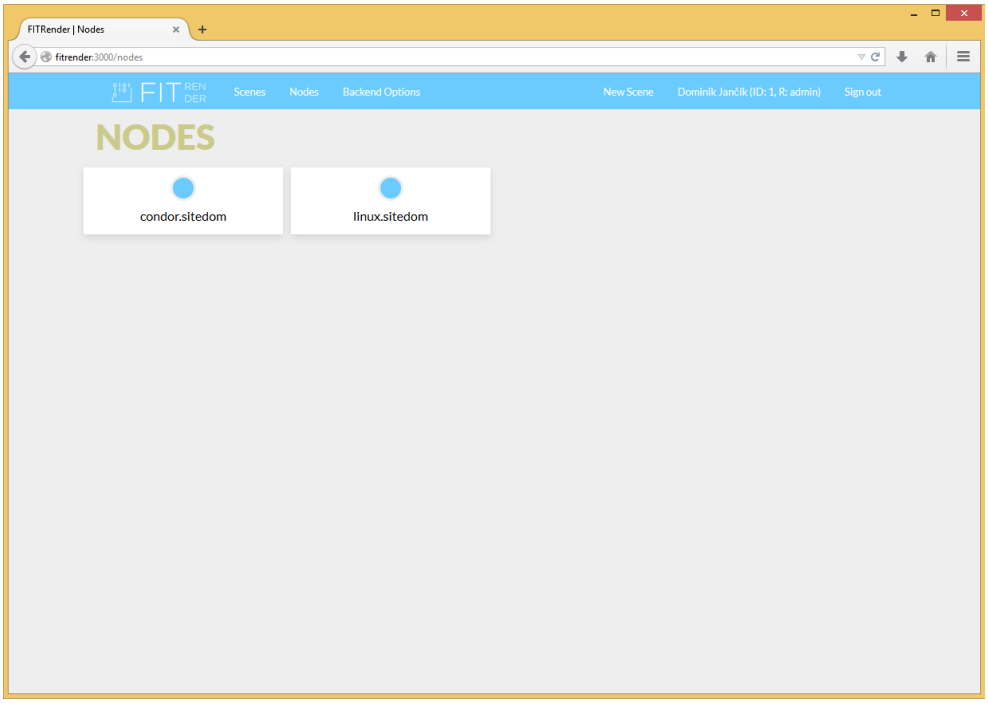

FITRender | FITRender Animation
fitrender:3000/scenes/29
FIT REN DER
Scenes
Nodes
Backend Options
New Scene
Dominik Jančík (ID: 1, R: admin)
Sign out
FITRENDER ANIMATION
RENDER SETUP
frames
1-20
scenes
RENDER TASKS
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Frame 7
Frame 8
Frame 9
Frame 10
Frame 11
Frame 12


FITRender | BMW Tiled
fitrender:3000/scenes/26
FIT RENDER
Scenes
Nodes
Backend Options
New Scene
Dominik Jančík (ID: 1, R: admin)
Sign out
BMW TILED
RENDER SETUP
frames
1
scenes
RENDER TASKS
Frame 1 tile 1/4
Frame 1 tile 2/4
Frame 1 tile 3/4
Frame 1 tile 4/4
Edit | Back


FITRender | Backend options
fitrender:3000/options
FIT RENDER
Scenes
Nodes
Backend Options
New Scene
Dominik Jančík (ID: 1, R: admin)
Sign out
BACKEND OPTIONS
GLOBAL
* Render path
/mnt/fitrender/renders
The location for finished renders
Update Option
* Log path
/mnt/fitrender/logs
The location for log files
Update Option
* Scene path
/mnt/fitrender/out
The location for program output files
Update Option
* Requirements
The Requirements expression for submitted jobs, see http://bit.ly/condor_classads for more info
Update Option
* Rank
The rank expression for submitted jobs, see http://bit.ly/condor_classads for more info
Update Option
* Subs path
/tmp
The location for generated HTCondor submission files
Update Option
RENDERER OPTIONS
List renderers


FITRender | Renderer options
fitrender:3000/renderers
FIT RENDER
Scenes
Nodes
Backend Options
New Scene
Dominik Jančík (ID: 1, R: admin)
Sign out
RENDERER OPTIONS
SELECT A RENDERER
Blender


FITRender | Options for Blender
fitrender:3000/renderers/Blender
FIT RENDER
Scenes
Nodes
Backend Options
New Scene
Dominik Jančík (ID: 1, R: admin)
Sign out
OPTIONS FOR BLENDER
* Frame granularity
0.5
How much to split single frames into multiple jobs. 1 is no splitting, lower numbers increase splitting.
Update Renderer
Back


FITRender | Nodes
fitrender:3000/nodes
FIT REN DER
Scenes
Nodes
Backend Options
New Scene
Dominik Jančík (ID: 1, R: admin)
Sign out
NODES
condor.sitedom
linux.sitedom

Příloha G

Blender skript pro nastavení dlaždic z příkazové řádky

```
# blender_cli_render_border.py

# Render border setting for Blender CLI rendering
# (c) 2015 Dominik Jancik, jancidom@fit.cvut.cz
#
# Usage:
# Try to keep the order and border values between 0 and 1.
# Do not use spaces after commas.
#
# blender -b SCENEFILE \
        -p blender_cli_render_border.py \
        -o //OUTPUT -f FRAMES \
        -border MIN_X,MIN_Y,MAX_X,MAX_Y
# eg. blender -b scene.blend \
        -p blender_cli_render_border.py \
        -o //image.png -f 1 -border 0,0,0.2,0.2

import bpy
import sys

# Border arguments
arg_index = sys.argv.index('-border')
border_args_string = sys.argv[arg_index + 1]
border_args_arr = border_args_string.split(',')
# Convert values to floats
border_args_arr = map(float, border_args_arr)
```

```
def set_border(min_x, min_y, max_x, max_y):
    render = bpy.context.scene.render
    render.use_border = 1
    render.border_min_x = min_x
    render.border_min_y = min_y
    render.border_max_x = max_x
    render.border_max_y = max_y

# Unpack to the set_border method
set_border(*border_args_arr)
```

Příloha H

Nápověda CmdJob

Toto je kopie výstupu nápovědy programu CmdJob, který je součástí správce distribuce Autodesk Backburner. Nápověda sloužila jako reference poskytovaných služeb utilitou.s

```
CMDJOB <options> executable_to_run <executable parameters>

Submits a command line job to Backburner.


                          -OPTIONS-

-?                              - Show this help file.
-cmdFile:<files>                - Semicolon separated list of text files
 OR                               that contains any of the options below.
@<files>                          Can be used alongside these options.

                        -JOB OPTIONS-

-jobName:<name>                 - Job name.
                                  Default is 'cmdJob'.
-description:<string>           - Sets a description for the job.
-priority:<number>              - Sets job's priority.
                                  Default is 50.
-workPath:<folder>              - Working folder for cmdjob.exe and servers.
                                  Used to resolve relative paths for running
                                  the executable.
                                  Default for cmdjob.exe is the current path.
                                  Default for the Servers is Backburner's.
-logPath:<folder>               - Task log folder.
                                  Default is to not produce a log.
-showOutput:<files>             - Semicolon separated list of output files
                                  to be accessible from the Monitor.

                       -SUBMIT OPTIONS-

-dependencies:<job handles>     - Comma separated list of job dependencies.
-timeout:<minutes>              - Sets a timeout per task.
                                  Default is 60 minutes.
-attach                         - Attaches the executable to the job.
-progress                       - Monitor the job progress.
-suspended                      - Starts the job suspended.
-userRights                     - Run the command line job with submitter's
                                  rights
-perServer                      - Creates separate jobs that are identical
                                  to this job, and assigns one to each
                                  server assigned to this job.  Each server
                                  will perform the same tasks as the others.

                       -NETWORK OPTIONS-
```

```
-manager:<name>              - Manager name, default is automatic search.
-port:<number>               - Port number.
-servers:<servers>           - Comma separated list of servers.
                               (Ignored if a group is used)
-serverCount:<number>        - Max number of servers that can work on this
                               job at any point in time.
-group:<group>               - Group name of servers to use.

                          -PARAMETERS-

-taskList:<file>             - File contains a tab separated table.
                               Use fill-in tokens to reference the table.
-taskName:<index>            - Sets the task name from the task list file
                               0=Unnamed, 1-X=column index in the file
-numTasks:<number>           - Number of tasks to perform.
                               (Ignored if -taskList is used)
-tp_start:<number>           - Specify the starting value of an internally
                               generated table used as a task list file.
                               (Ignored if -taskList is used)
-tp_jump:<number>            - Specify the increment of the internally
                               generated table used as a task list file.
                               (Ignored if -taskList is used)
-jobParamFile:<file>         - File with two tab separated column used to
                               add custom data to the job.
                               The first colum is the parameter's name.
                               The second column is the parameter's value.

                       -NOTIFICATION OPTIONS-

-emailFrom:<address>         - Source email address for notification.
-emailTo:<address>           - Destination email address for notification.
-emailServer:<server>        - SMTP name of email server to use.
-emailCompletion             - Notify by email job completion.
-emailFailure                - Notify by email job failure.
-emailProgress:<number>      - Notify by email the completion of every
                               Nth task

                          -FILL-IN TOKENS-

Placeholder tokens that are replaced while calling executable.
These are evaluated on a per server basis.
These are not recursive.  For example, %tp1 cannot evaluate to
contain a fill-in token itself.

%dsc                         - The job description.
%srv                         - Name of the server executing the task.
%tpX                         - Task parameter X from the task list.
                                 X, column index in the task list file.
                                 Rows correspond to the current task #.
%*tpX                        - Same as %tpX
                                 *, number of 0 padded digits to use.
%tn                          - Task number of the assigned task.
%*tn                         - Same as %tn
                                 *, number of 0 padded digits to use.
%jpX                         - Parameter X from the job parameter file.
                                 X, row index in the job parameter file.
%*jpX                        - Same as %jpX
                                 *, number of 0 padded digits to use.

                             -NOTES-

Options are not case-sensitive.
Only the FIRST occurrence of cmdFile/@ is used, the others are ignored.
Only the LAST occurrence of each other option is used.
```

Příloha I

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `vm` ..... adresář s virtuálním strojem s přednastavenou implementací
- `survey` ..... adresár s kompletními výsledky průzkumu
- `nur` ..... výstupy z návrhu uživatelského rozhraní
  - `hifi-prototype` ..... výstupy z návrhu uživatelského rozhraní
    - `index.html` ..... vstupní soubor do mockupového prototypu
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `thesis-jancidom-fitrender.pdf` ..... text práce ve formátu PDF