VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2018 Martin Bařinka

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

PROGRAMOVÝ MODUL PRO ZOBRAZENÍ MRAČNA BODŮ VE VIRTUÁLNÍ REALITĚ
SOFTWARE MODULE FOR POINT CLOUDS DISPLAY IN VIRTUAL REALITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Martin Bařinka**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. Luděk Žalud, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Martin Bařinka ***ID:*** 159656
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Programový modul pro zobrazení mračna bodů ve virtuální realitě

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s moderními technologiemi pro zobrazení virtuální reality (VR) a rozšířené reality (AR), především s tzv. helmami/brýlemi pro VR a AR.
2. Zabývejte se optimalizací zobrazení dat tak, aby bylo možné plynule zobrazit co největší množství bodů (řádově minimálně miliony).
3. Realizujte rychlé načítání ve formátu vybraném na základě Semestrálního projektu. Uživatel musí mít možnost přímo v prostředí vybírat a načítat jednotlivé soubory – a to buď jako nezávislé měření nebo přidání nových dat do existujícího zobrazení.
4. Realizujte rovněž podporu načtení a zobrazení mesh modelů.
5. Realizujte možnost průhlednosti objektů.
6. Vytvořený program otestujte a změřte závislost zobrazovaných snímku za sekundu na počtu bodů. Testování proveďte na počítači vybraném společně se školitelem.

DOPORUČENÁ LITERATURA:

Sky Nite; Virtual Reality Insider: Guidebook for the VR Industry; November 1, 2014; New Dimension Entertainment; ISBN-13: 978-0990999928

Termín zadání: 5. 2. 2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: prof. Ing. Luděk Žalud, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta elektrotechniky a komunikačních technologií
Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

ABSTRAKT

Cílem této bakalářské práce je vývoj aplikace pro vykreslování multispektrálních mračen bodů, meshů a průhledných meshů ve virtuální realitě. Vstupní data mohou pocházet z různých typů skenerů. Tato práce popisuje druhy zařízení pro zobrazení virtuální reality, obsahuje základní data o historii virtuální reality a porovnává populární formáty pro ukládání 3D modelů. Jsou zde popsány metody zobrazování dat a jejich optimalizace. Popisuje program, jeho funkce a ovládání. Je provedena analýza a vyhodnocení efektů jednotlivých optimalizací vykreslování za definovaných podmínek.

KLÍČOVÁ SLOVA

Virtuální realita, vykreslování, multispektrální data, mračno bodů, mesh, průhledný mesh, OpenGL, OpenVR.

ABSTRACT

Goal of this bachelor's thesis is development of application for rendering multispectral pointclouds, meshes and transparent meshes. Input data can be generated by different types of scaners. This thesis describes types of equipment for virtual reality display. There are basic data about virtual reality history, compares popular formats for 3D model storage. Thesis describes program features and contains user guide. Finally there is analysis and evaluation of each optimalisation at defined conditions.

KEYWORDS

Virtual reality, multispectral data, rendering, pointcloud, mesh, transparent mesh, OpenGL, OpenVR.

BAŘINKA, Martin. *Programový modul pro zobrazení mračna bodů ve virtuální realitě*. Brno, 2018, 85 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: prof. Ing. Luděk Žalud, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Programový modul pro zobrazení mračna bodů ve virtuální realitě“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu prof. Ing. Luďkovi Žaludovi, PH.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci, a Ing. Tomáši Miletovi za odbornou pomoc při optimalizaci vykreslování.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Virtuální realita a zobrazování dat v 3D prostoru je rychle se rozvíjející obor a pomáhá v různých odvětvích. Cílem této práce je vytvoření programu pro prohlížení medicínských multispektrálních dat ve virtuální realitě. Tato data budou převážně pocházet ze skeneru RoScan, který je vyvíjen na CEITEC VUT. RoScan je zařízení, které dokáže rychle nasnímat část pacienta a ukázat povrch, barvu, teplotu a hrubost povrchu. *"Během několika desítek sekund lze pořídit velmi přesný trojrozměrný počítačový model vybrané části pacientova těla." "Přesné termovizní modely povrchu lidského těla umožňují v některých případech nahradit velmi drahé zobrazovací metody, jako je MRI nebo CT."*[23] Nyní se používá zobrazování dat pouze na monitoru, díky tomuto programu bude moci uživatel vidět zobrazené data jako plastický obraz, což v medicíně může umožnit zkvalitnění diagnostiky.

Tato práce je pokračováním mé semestrání práce[19], která byla zaměřena na porozumění hardwaru i softwaru virtuální reality, akcelerovanému vykreslování v OpenGL. Nově je přidána podpora meshů, průhledných meshů a optimalizace vykreslování mračen bodů.

V první kapitole jsou základní informace o virtuální a rozšířené realitě, její historii i současných zařízeních, které mohou být použity s tímto programem. Je zde popsán výběr vhodného formátu 3D modelů. Druhá kapitola popisuje načítání a vykreslování modelů a jejich optimalizaci. Informace jsou řazeny chronologicky, tak, jak probíhala optimalizace během této práce. Třetí kapitola je věnována návrhu grafického uživatelského rozhraní, popisuje jednotlivé části i vnitřní logiku programu. Čtvrtá kapitola popisuje samotný program VR-pointcloud-viewer, se zaměřením na uživatele. Poslední kapitola shrnuje výsledky výkonového srovnání jednotlivých optimalizací a popisuje metodiku měření.

1 Virtuální a rozšířená realita

Virtuální realita (VR) je počítačově generované prostředí, které simuluje realistický zážitek.[2] Jde o vytváření vizuálního, sluchového, hmatového či jiného vjemu, budícího subjektivní dojem skutečnosti pomocí počítače, speciální audiovizuální helmy, brýlí, popřípadě obleku.[12] Současné VR technologie nejčastěji používají náhlavní sadu, někdy doplněnou o pomůcky pro zlepšení zážitku. Uživatel takového zařízení je schopen se v takovém virtuálním světě rozhlížet, procházet a interagovat s ním.[2]

VR můžeme dělit na základě technických prvků:

Pohlcující virtuální realita (immersive VR) uživatel je v maximální možné míře oddělen od vjemů skutečného světa, často v simulátoru, který navíc vytváří pocit pohybu, pádu či odstředivé síly. Dále se používají různá zařízení pro hmatovou (*tactile*) či silovou odezvu (*force feedback*), která jsou schopna vyvíjet tlak proti uživateli.[60]

Pohlcující VR nejvíce známe z PC her, ale je také důležitá pro výcvik vojáků nebo pilotů. Své uplatnění nachází i v medicíně, kde se používá například pro léčení různých fobií.

Rozšířená virtuální realita (AR) je označení, používané pro reálný obraz světa, doplněný počítačem vytvořenými objekty. Tyto přidané objekty nebo informace zpravidla zlepšují vnímání reálného světa. Uživatel má obvykle na hlavě kameru a polopropustné brýle.

Rozšířená virtuální realita se používá například ve vojenských vozech nebo letadlech, kde se nasnímaný obraz doplní o různé symboly, které pomáhají např. s orientací.

Jednoduchá virtuální realita. Zde se nepoužívají žádné speciální technické zařízení. Provozuje se na klasickém PC, může se jednat o hry, simulace.

Jiným kritériem, jak pohlížet na virtuální realitu je množství umělých a skutečných prvků, prezentovaných uživateli. Používá se stupnice: reálné prostředí – rozšířená realita – rozšířená virtualita – virtuální prostředí. Kombinace prvků rozšířené reality a rozšířené virtuality se nazývá *smíšená realita (Mixed Reality)*.

1.1 Historie virtuální reality

Přesné počátky virtuální reality je těžké určit, i proto, že je obtížné definovat, co můžeme za virtuální realitu považovat. Pokud se na virtuální realitu zaměříme pouze z hlediska vytváření iluze přítomnosti na jiném místě, můžeme za počátky virtuální reality považovat 360° panoramatické malby z osmnáctého století. V roce 1838 výzkum Charlese Wheatstona demonstroval, že mozek vytváří iluzi prostoru tím, že

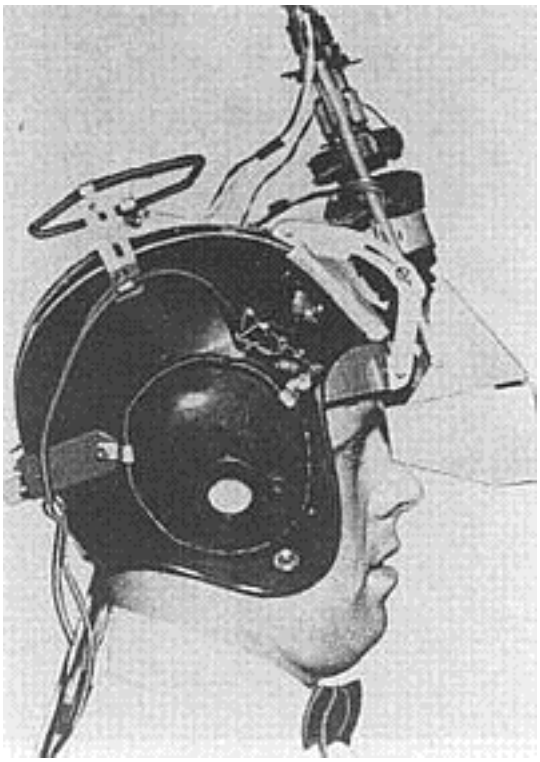
Obr. 1.1: Sensorama [58]

zpracovává dva různé dvourozměrné obrazy z očí. Prohlížení dvou obrázků s mírně posunutým pohledem pomocí stereoskopu vytváří iluzi hloubky.[58]

První známý příklad VR je Sensorama viz obrázek 1.1, sestrojená Mortonem Heiligem v roce 1962. Bylo to mechanické zařízení, využívající stereoskopické zobrazení, ventilátory, rozprašovače pachů, stereo zvuk a pohybující se křeslo. Zařízení sloužilo k promítání filmů, které působily velmi realisticky. Uživatel byl ale jen v pasivní roli, a k dispozici bylo několik málo krátkých filmů. Bohužel Heilig nedokázal najít investora pro tento projekt a projekt ukončil.[58, 11]

1.1.1 První náhlavní sada se sledováním pohybu – Headsight

V roce 1961 dva inženýři ve společnosti Philco vyvinuli první náhlavní sadu Headsight viz obrázek 1.2. Obsahovala displej pro každé oko a magnetický systém sledování pohybu. Systém sledování pohybu byl spojen s kamerovým systémem. Kamery se otáčely podle otáčení uživatele, což umožňovalo přirozené rozhlížení. Headsight byl vyvinut pro vojenské aplikace, pro vzdálené sledování nebezpečných situací[17, 58].



Obr. 1.2: Headsight [17, 46]

1.1.2 První náhlavní sada virtuální reality – The Sword of Damocles

Za první náhlavní sadu se dá považovat The Sword of Damocles[51], kterou sestavil počítačový vědec Ivan Shuterland s pomocí svého studenta Boba Sproula.

Celé zařízení bylo primitivní, jak z pohledu použitelnosti, tak realismu. Grafické modely byly pouze drátové. Obraz v sadě se měnil podle uživatelova pohledu, což vyžadovalo nějaký způsob sledování polohy. Poloha byla snímána pomocí ramene upevněného ve stropě laboratoře.

Název zařízení vznikl z podobnosti konstrukce s mýtickou zbraní, která visela nad královským trůnem.[50]

Přestože byla primitivní, určila směr vývoje pro budoucí zařízení, jako například Oculus Rift nebo HTC Vive.[50]

1.1.3 První rozšířená realita – Virtual fixture

Virtual fixture byla vyvinuta v roce 1992. Její podstatou bylo, že uživatel byl oblečen ve speciálním obleku (exoskeletu) a svými pohyby přímo ovládal robota, který byl na jiném místě a prováděl požadované pokyny. Kamerou se přenášel obraz zpět k uživateli. Vylepšení spočívalo v přidávání umělých hranic, ať už formou vizuální,

Obr. 1.3: VFX1 Headgear[6]

zvukovou nebo mechanickou. Tato mechanická hranice byla implementována pomocí silové zpětné vazby v exoskeletu. Protože byla pouze virtuální, mohla se prolínat s reálnými objekty, aniž by s nimi fyzicky kolidovala. Díky těmto hranicím mohl uživatel provádět přesnější práci, než jaké by byl člověk schopen ručně.[49]

1.1.4 VFX1 Headgear

Byl vyvinut v devadesátých letech firmou Forte Technologies. Sada obsahovala helmu viz obrázek 1.3, ovladač a ISA rozšiřující kartu do počítače. Umožňovala snímání rotace hlavy. Helma měla dva 0.7" 263 × 230 LCD displeje, čočky s nastavitelným ohniskem a zorným polem 45°. Helma měla zabudovány sluchátka a mikrofon.[6]

1.2 Současná virtuální realita

Moderní éra VR začala v době, když v roce 2012 ohlásil Occulus vývoj headsetu Rift.

1.2.1 Oculus

Oculus byla původně samostatná firma, která v roce 2012 ohlásila vývoj headsetu. V průběhu vývoje existovalo mnoho prototypů. Dva z nich byly dodány podporovatelům kampaně. Byly to Development kit 1 v roce 2013 a Development kit 2 v roce

Obr. 1.4: Oculus Rift CV1[8]

2014. V roce 2014 byla společnost Oculus koupena Facebookem. V březnu 2016 se začaly dodávat první kusy finální verze (consumer version 1).[8]

Rift CV1

Oculus Rift (obrázek 1.4) byl vydán v březnu 2016 za oficiální cenu 599,99$, která postupně klesala až na aktuálních 399$, má displej s rozlišením 2160x1200 (1080x1200 pro každé oko), obnovovací frekvenci 90Hz a zorný úhel 110°. Má integrovaná sluchátka a sledovací systém nazvaný Constellation. Tento systém používá dvě infračervené kamery, které snímají infračervené LED zabudované v headsetu a ostatních zařízeních, které blikají definovaným vzorem. Pokud se nepoužívají ovladače, systém může být použit pouze s jednou kamerou. V tomto případě totiž uživatel nezastíní kameru pohybem ovladače ve vzduchu. Pokud si uživatel koupí také ovladače, dostane k sadě druhou kameru, což sníží riziko zastínění. Se dvěma kamerami je také systém schopný sledovat celou místnost.

Původně byl headset vyvíjen bez ovladačů, a protože Oculus spolupracoval s Microsoftem, první kusy CV1 byly dodávány s Xbox One ovladačem. Ovladače pro Rift nakonec byly vydány v říjnu 2016 jako samostatné příslušenství. V srpnu 2017 se začaly dodávat ve standardní sadě místo Xbox One ovladače.[8]

Obr. 1.5: HTC Vive[29]

1.2.2 HTC

HTC společně s Valve corporation vyvíjí headsety s názvem Vive. Vývoj HTC Vive byl oznámen v březnu 2015, vývojové sady byly rozesílány v srpnu a září 2015 a první uživatelská verze byla vydána v dubnu 2016[7]

Vive

HTC Vive (obrázek 1.5) byl vydán za 799$, cena se ale postupně snižovala až na aktuálních 499$. Headset používá dva LCD displeje o rozlišení 1080x1200, obnovovací frekvencí 90 Hz a zorným úhlem 100°. Zepředu je instalovaná kamera, kterou může uživatel vidět skrz brýle, nebo se používá v kombinaci se systémem virtuálních hranic Chaperone. Na povrchu náhlavní sady je několik IR senzorů, které se používají pro sledování pozice headsetu. Jako pomocné senzory jsou v náhlavní sadě zabudovány akcelerometr a gyroskop.

K headsetu jsou přibaleny dva ovladače. Tyto ovladače mají dotykovou plochu, grip a analogovou páčku jako spoušť. Na předu ovladače je 24 IR snezorů, které snímají jejich polohu.

Pro sledování polohy jsou nutné tzv. základnové stanice, někdy nazývané majáky. Tyto stanice jsou dvě, umisťují se do úhlopříčky sledovaného prostoru, vzdálenost mezi stanicemi může být až 5 metrů[29]. Tyto stanice promítají speciální vzor, který je přijímán všemi sledovanými zařízeními. Podle prodlevy mezi zaznamenáním jednotlivých záblesků počítač zjistí pozice všech zařízení s přesností na zlomky milimetrů.

Dále existuje tzv. Vive Tracker, je to zařízení, které v sobě má pouze IR senzory pro snímání polohy. Toto zařízení se dá použít pro sledování libovolných předmětů

Obr. 1.6: Náhlavní sady pro Windows Mixed Reality[33]

ve virtuální realitě.[7]

1.2.3 Windows Mixed Reality

Microsoft vydal v říjnu 2017 svůj vlastní systém nazvaný *Windows Mixed Reality*. Vyvinul sofware a referenční návrh náhlavní sady, tyto dodávají ostatní výrobci jako Acer, Dell, Samsung, HP a Lenovo, a ceny se pohybují v rozmezí 399$ až 499$. Všechny náhlavní sady jsou si podobné, viz obrázek 1.6, mají 2880×1440, 90 Hz LCD displeje, 105° zorný úhel, s vyjímkou Samsung HMD Odyssey, ten má 2880×1600, 90 Hz AMOLED displeje a 110° zorný úhel.[33] Ovladače k těmto náhlavním sadám jsou navrženy Microsoftem, takže jsou všechny stejné.[32]

Tyto sady používají pro sledování pozice kamery uvnitř náhlavní sady, což na jednu stranu ulehčuje nastavení, protože není nutné nastavovat žádné základnové stanice jako u HTC Vive nebo Oculus Rift, na druhou stranu toto sledování není tak precizní a jde sledovat pouze předměty před uživatelem.[26, 32]

I když tomu název napovídá, nejedá se o smíšenou realitu. Jsou to spíše obyčejné náhlavní sady pro virtuální realitu. Chybí jakýkoli způsob snímání reality, aby mohlo dojít k prolnutí reálného a virtuálního světa. Toto mají údajně napravit nadcházející zařízení.[26, 32]

Nahlavní sady pro windows mixed reality oficiálně vyžadují ke svému chodu aplikace z Windows storu, nicméně existuje i podpora OpenVR, zatím ale pouze v beta verzi.[22]

1.2.4 Ostatní náhlavní sady

Existuje řada dalších méně populárních náhlavních sad. Jednou z nich je *Open Source Virtual Reality (OSVR)*. Je to open-source softwarový projekt, jehož cílem je umožnit používání náhlavních sad a ovladačů všech výrobců v jakékoli aplikaci.[34] K tomuto projektu patří i náhlavní sada *OSVR Hacker Development Kit*, kterou vyvíjí Razer a Sensics, o které říkají, že je také open source, ale stále nebyly zveřejněny všechny zdrojové kódy viz stížnosti na stránce projektu[10] a licence je daleko od open-source viz Readme[47].

1.3 Aplikace virtuální reality

Virtuální realita má celé spektrum aplikací. Asi nejpoužívanější je v zábavním průmyslu. Nejde jen o videohry, ale i zpřístupnění výstav, jak to dříve nebylo možné. Návštěvník má díky virtuální realitě možnost se procházet místy, kde je to reálně nemožné a prohlédnout si objekty z různých stran.

V robotice se virtuální realita používá pro teleprezenci.

Široké použití má virtuální realita i ve zdravotnictví. Používá se pro školení, trénink operací, ale i při samotných operacích, kdy operaci provádí robot, ovládaný dálkově lékařem. V psychologii se virtuální realita používá pro terapie zaměřené na léčení fóbii nebo posttraumatických stavů.

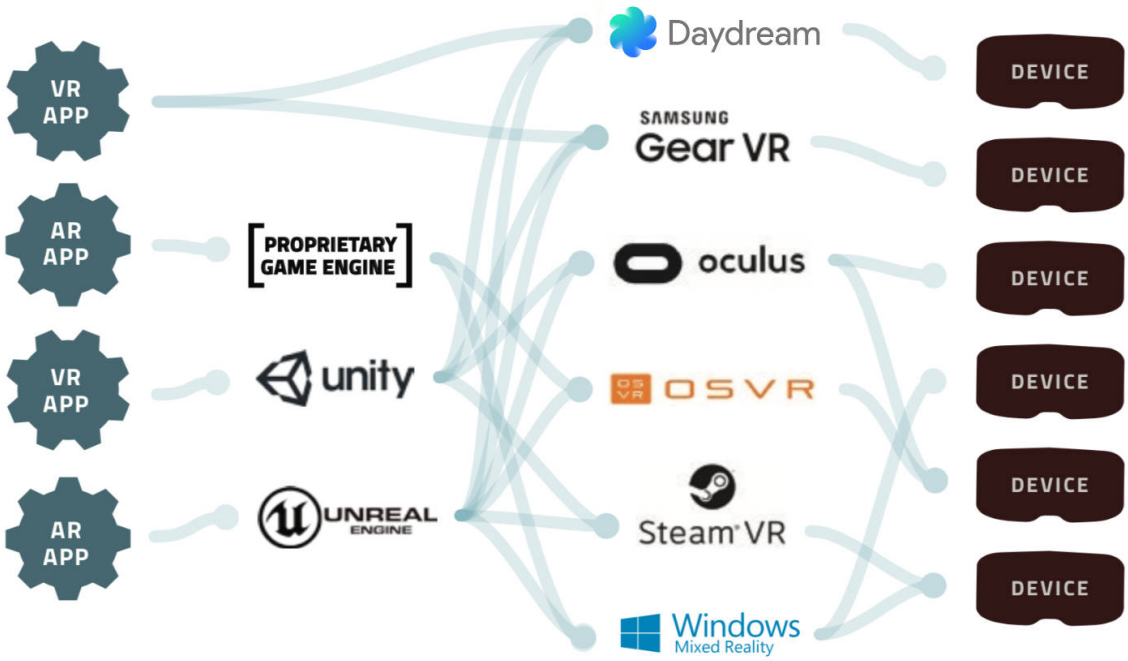
Virtuální realita se používá i v simulátorech při tréninku pilotů, kosmonautů, parašutistů. I pro výcvik vojáků se jako doplňkový trénink používají programy virtuální reality. Nenahradí ovšem plně reálný výcvik.

Virtuální realita se začíná více používat v nejrůznějších vědních oborech. Pomáhá hlavně s vizualizací mechanických součástek, staveb nebo jiných složitých struktur.

Další využití virtuální reality je ve vzdělávání. Pomáhá žákům nejen znázornit, ale i interagovat s komplexními objekty. Například v astronomii se může promítat celá sluneční soustava nebo galaxie a studenti můžou s tímto virtuálním vesmírem interagovat.[57]

1.4 Nebezpečí Virtuální reality

Dlouhodobé účinky virtuální reality nejsou zatím potvrzeny. Je popsáno větší množství krátkodobých nechtěných symptomů, způsobených dlouhým časem, stráveným ve virtuální realitě. Může se jednat o bolesti hlavy, závratě, nevolnosti, stres, napětí ale také ztrátu orientace, rozmazané vidění.[2]



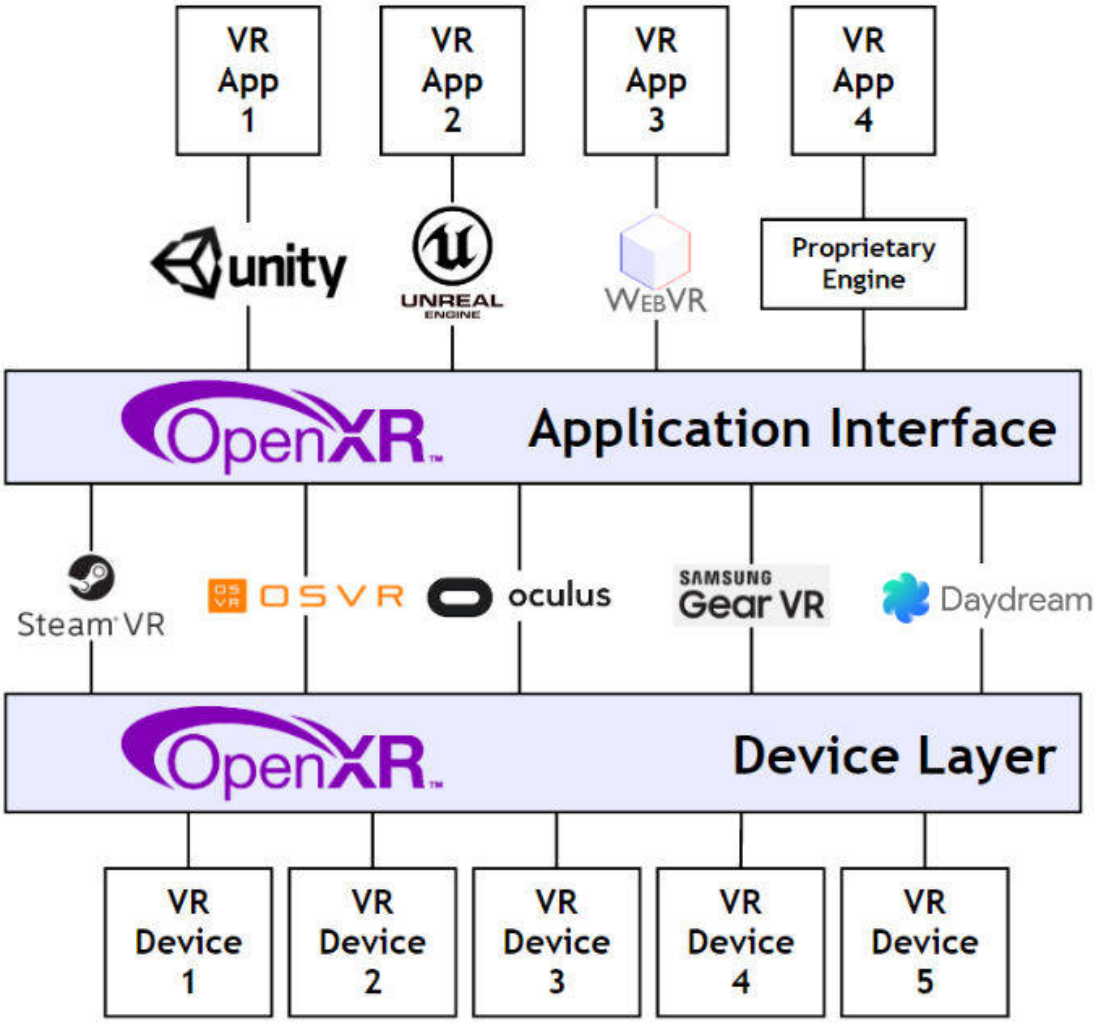
Obr. 1.7: Situace na trhu podle Khronos group[54]

1.5 Vývojový software pro virtuální a rozšířenou realitu

Aktuálně je trh roztříštěný, viz obrázek 1.7, každý výrobce si vyvinul své vlastní API a runtime, které jsou nutné k funkci jejich zařízení. Valve se toto snaží redukovat a je ochotné implementovat podporu cizích headsetů ve svém OpenVR. OpenVR nyní podporuje HTC Vive, Oculus Rift a OSVR headsety.

OpenVR je ale bohužel velmi úzce spjatý s herní platformou Steam, je ovládaný firmou Valve. Dostupná dokumentace není příliš detailní a nereflektuje poslední aktualizace.

V únoru 2017 ohlásila Khronos Group vývoj otevřeného standardu OpenXR, který má za úkol sjednotit softwarovou platformu pro virtuální realitu[52], viz obrázek 1.8. Vydání první verze standardu je plánováno na polovinu roku 2018[31]. Aktuálně jsou mezi přispěvateli do OpenXR i všichni, kteří vyvíjejí vlastní řešení.[54]



Obr. 1.8: OpenXR[54]

1.5.1 OpenVR

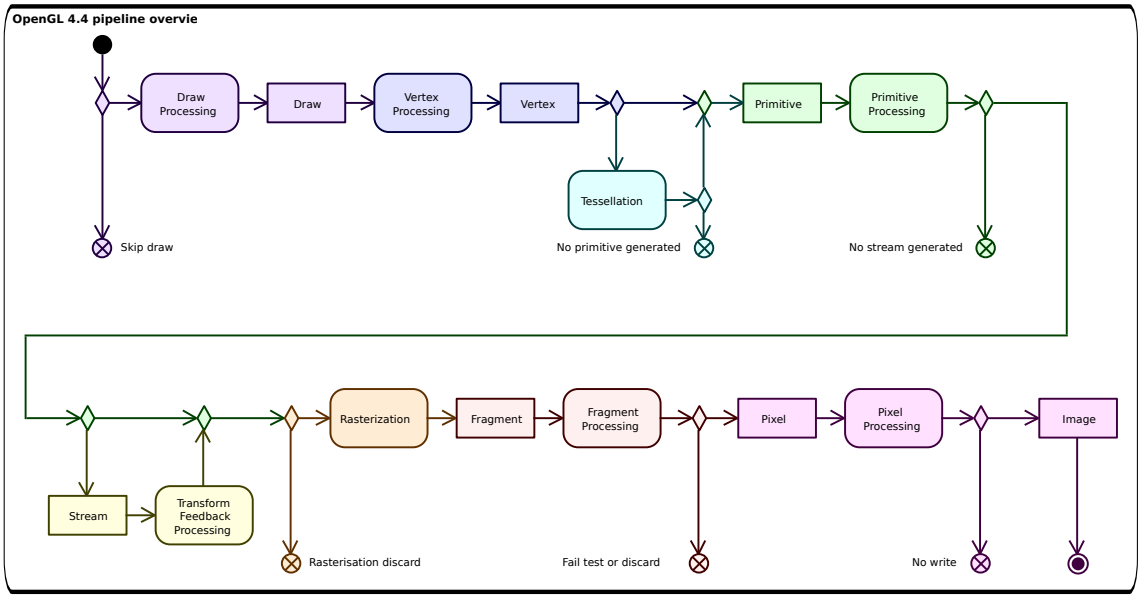
Ve této práci je použit pro interakci s náhlavní sadou OpenVR runtime. Byl vyvinut společností Valve, pro jejich herní platformu Steam. Byl vybrán, protože podporuje jak HTC Vive tak i Oculus Rift a Razer HDK.

V principu runtime funguje tak, že dodává pozice sledovaných zařízení a přijímá vykreslený obraz pro každé oko.

1.6 OpenGL

OpenGL je rozšířené multiplatformní rozhraní pro akceleraci vykreslování 2D a 3D grafiky. Rozhraní je založeno na architektuře klient – server. Program (klient) vydává příkazy a grafický akcelerátor (server) vykonává. OpenGL je multiplatformní software, je nezávislé na jazyce. Specifikace OpenGL neříká nic o získávání a řízení kontextu, o toto se stará okenní systém.[9]

Dnešní grafické akcelerátory jsou velmi flexibilní, existuje v nich řada programovatelných fází, které jsou v OpenGL přístupné pomocí jazyka GLSL (OpenGL Shading Language). Přehled celého zřetězení v grafickém akcelerátoru dostupné v OpenGL je na obrázku 1.9.



Obr. 1.9: OpenGL pipeline[48]

1.7 Formát pro ukládání 3D modelů

Má semestrální práce[19] se zabývala porovnáním formátů vhodných pro mračna bodů, nyní přibyl pro podporovaný formát ještě jeden požadavek, a to podpora meshů.

1.7.1 Stereo Litography STL

Formát byl vyvinut Albert Consulting Group pro 3D systems[1] v roce 1987 a sloužil pro jejich první komerční 3D tiskárnu. Od té doby se téměř nezměnil, až v roce 2009 byl navržen nástupce STL 2.0[28], který se ale na trhu neujal.

Formát STL je často používaný v 3D tisku, podporuje pouze geometrická data. Existují neoficiální nadstavby tohoto formátu, které navíc podporují barvy, ale nejsou moc rozšířené a jsou navzájem nekompatibilní.

Dalším problémem je nejednotná práce s normálami v různých programech. Normály stěn by měly být jednotkové vektory směřující ven z objektu. Většina softwarů ale tyto normály dopočítává dodatečně podle pravidla pravé ruky. Některé programy porovnávají uložené normály s vypočtenými, jiné uložené normály ignorují úplně a používají vypočtené normály. Aby byl soubor maximálně přenositelný, měl by mít uložené normály spočítané pomocí pravidla pravé ruky a vrcholy uspořádané tak, aby normála směřovala správným směrem.[1]

Protože tento formát nemá žádný standardní způsob definování barev a dalších vlastností, je pro tuto aplikaci nevhodný.

1.7.2 Wavefront OBJ

Formát OBJ byl vyvinut Wavefront Technologies pro jejich animační balíček Advanced Vizualizer. Je to jednoduchý textový formát definujíci geometrii, texturovací souřadnice, normály vrcholů a stěny. Tento formát může být doplněn pomocí souboru Material Template Library (MTL), který definuje povrch. Povrch může mít definované odrazové vlastnosti podle Phongova osvětlovacího modelu, texturu a jiné vlastnosti.[13]

Samotný OBJ je pro tuto aplikaci nevhodný, protože neumožňuje ukládat informace o barvě nebo skalárním poli. Přidáním souboru MTL se absence barevných dat eliminuje, ale stále zde není standardní způsob, jak uložit skalární pole.

Navíc jsou oba soubory textové, což znamená, že jsou zbytečně velké a budou se zbytečně dlouho načítat.

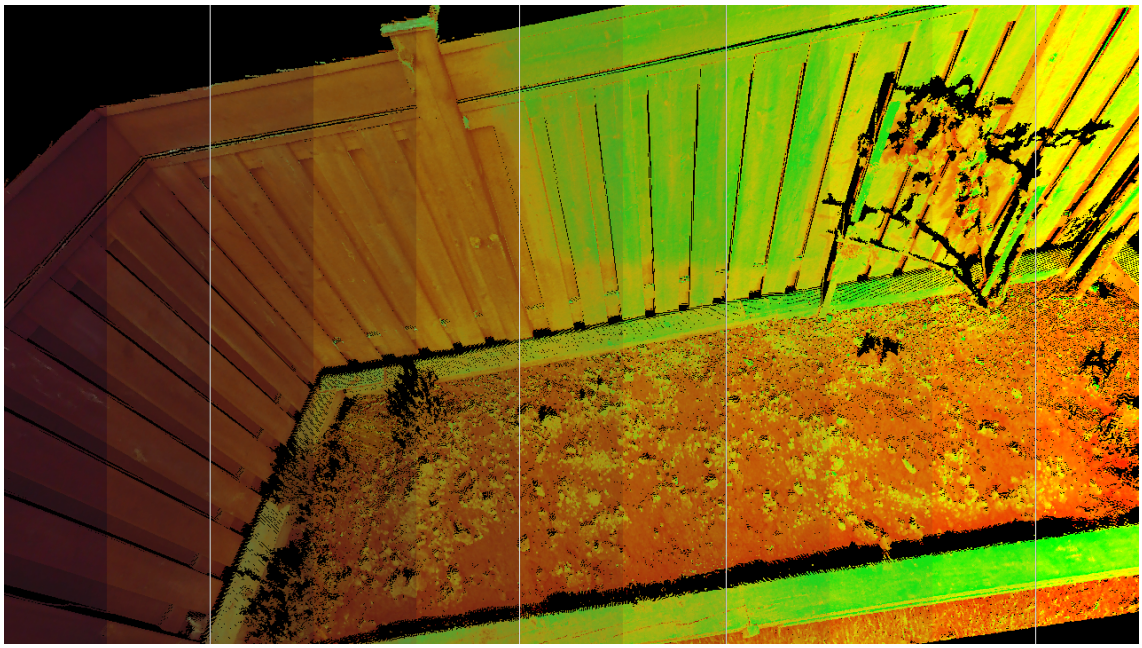
1.7.3 Stanford triangle format PLY

Formát Stanford triangle PLY byl navržen jako flexibilní formát pro ukládání 3D dat. Tento formát má jak textovou, tak binární verzi. Podporuje relativně jednoduchý popis jediného objektu jako seznam plochých mnohoúhelníků. Hlavní výhodou tohoto formátu je možnost definovat pro jednotlivé vrcholy téměř jakékoli vlastní hodnoty. Sice existují zavedené názvy pro nejčastěji používané vlastnosti, ale tyto nejsou povinné, což může způsobit problém přenositelnosti těchto souborů.[56]

Formát podporuje jak meshe, tak i mračna bodů.

1.7.4 Výběr

Už v minulé práci jsem vybral jako vhodného kandidáta formát PLY, protože je flexibilní a jednoduchý na implementaci.



Obr. 2.1: Prolínání barvy (vlevo) skalárního pole (vpravo)[19]

2 Načítání a vykreslování modelů

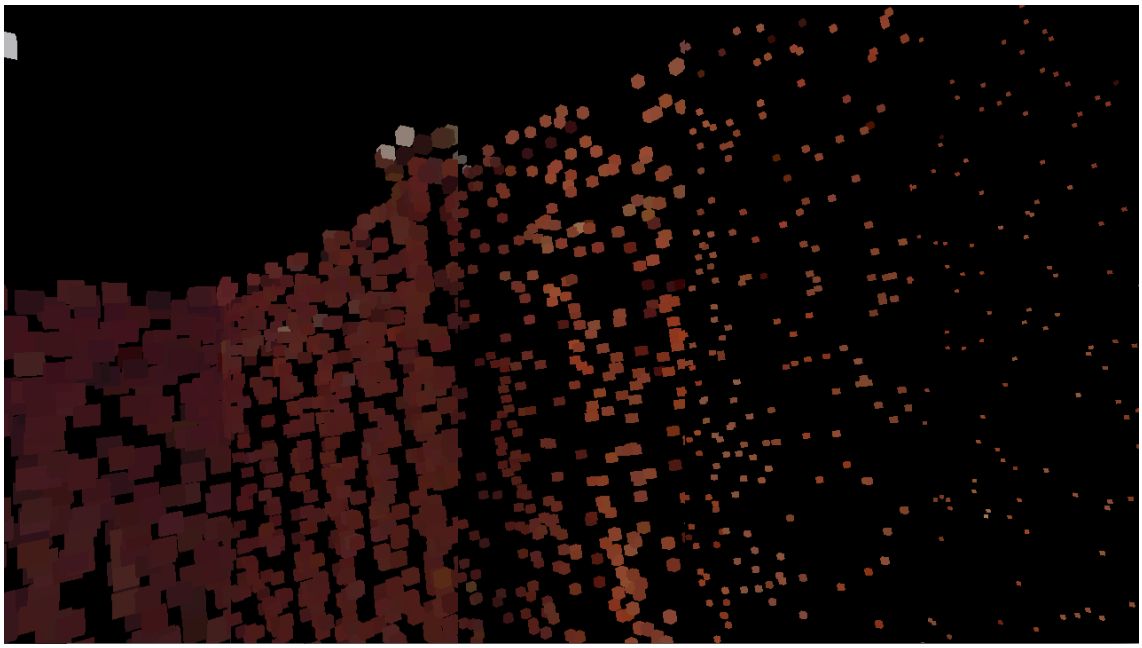
Modely se načítají pomocí dialogu uvnitř aplikace, tento dialog umožňuje uživateli procházet všechny soubory na lokálních discích nebo na namapovaných síťových úložištích.

Aktuálně jsou podporovány pouze soubory PLY. Samotné načtení souboru provádí knihovna tinyply[25]. Knihovna má API, které dovolí uživateli načíst hlavičku souboru a pak vybrat, které vlastnosti načíst.

Mračno bodů je definováno jako množina bodů. Meshe jsou definovány množinou bodů a indexy, které sestavují stěny.

Program dokáže vykreslovat barvu objektů, v případě, že barva není dostupná, vykresluje se objekt bíle. Dále dokáže vykreslovat jedno skalární pole, to je jednorozměrná hodnota přiřazená ke každému bodu. Tato hodnota nemá žádný konkrétní význam. Může to být například teplota. Skalární pole se může zobrazovat místo barvy. Je převedeno na barevnou škálu od červené po zelenou barvu. Toto skalární pole se dá plynule prolnout s barvou objektu viz obrázek 2.1.

V případě mračna bodů lze měnit velikost bodů viz obrázek 2.2.

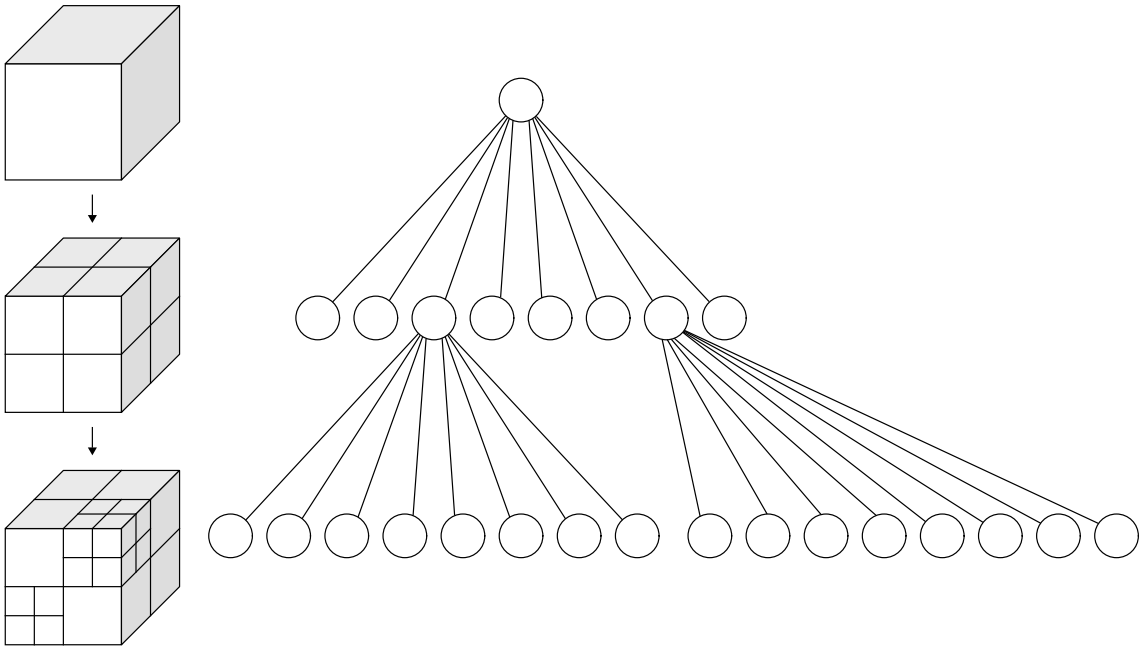


Obr. 2.2: Změna velikosti bodů[19]

2.1 Vykreslování mračna bodů

Pro vykreslování jednotlivých bodů byly zvoleny malé krychle, což je dost nezvyklé. Obvykle se používají tzv. body, jsou to čtverce, které rasterizér vygeneruje na příslušné místo na obrazovce. Krychle byla zvolena, protože čtverce dokáží měnit velikost pouze na celé pixely a to kazí celkový vjem prostoru. Použití krychlí má ale jeden zásadní nedostatek a to je mnohonásobná geometrická složitost. Zatímco bod je definován pouze jedním vrcholem a tudíž se vertex shader vyvolá pouze jednou, krychle se sice skládá z 8 vrcholů, ale musí se kreslit každá stěna zvlášť. Kreslení čtverců se nedoporučuje[3], takže každá stěna je navíc rozdělena na 2 trojúhelníky. To máme 3 vrcholy na trojúhelník, 2 trojúhelníky na stěnu a 6 stěn na krychli, to je efektivně 36 vrcholů. Toto číslo vypadá hrozivě, ale protože grafické karty mají *Post Transform Cache*[40], výsledný výkon není tolik snížen, klesne pouze 8 krát, protože krychle má 8 vrcholů. Výsledky porovnání výsledného výkonu jsou v příloze A.

Body se kreslí pomocí *geometry instancing*[45]. Tato metoda je vhodná, když je potřeba kreslit velké množství opakujících se objektů, v tomto případě krychle, a měnit pouze některé vlastnosti, jako polohu a barvu. Pokud by se totiž jednotlivé krychle kreslily jednotlivými příkazy v cyklu, režie ovladače grafické karty by byla extrémní.



Obr. 2.3: Vlevo: Rekurzivní rozdělení krychle na oktanty, vpravo: odpovídající strom[4]

2.1.1 Optimalizace pomocí oktalového stromu

Mračna bodů mohou obsahovat miliony bodů, což znamená velkou náročnost při vykreslování. Proto je nutné nejdříve vyhodnotit, které body jsou v zorném poli a musí se zobrazit. Body mimo zorné pole se mohou přeskakovat. Aby toto bylo možné, je nutné mračno bodů rozdělit na menší segmenty. Pro rozdělení byl použit oktalový strom viz obrázek 2.3. Mračno se uzavře do $AABB$[1], a poté se body roztřídí do 8 oktantů. Tyto podskupiny se dále rekurzivně dělí dokud mají více bodů, než je předvolená hranice. Hranice byla zvolena jako 100 000 bodů experimentálně v příloze C.

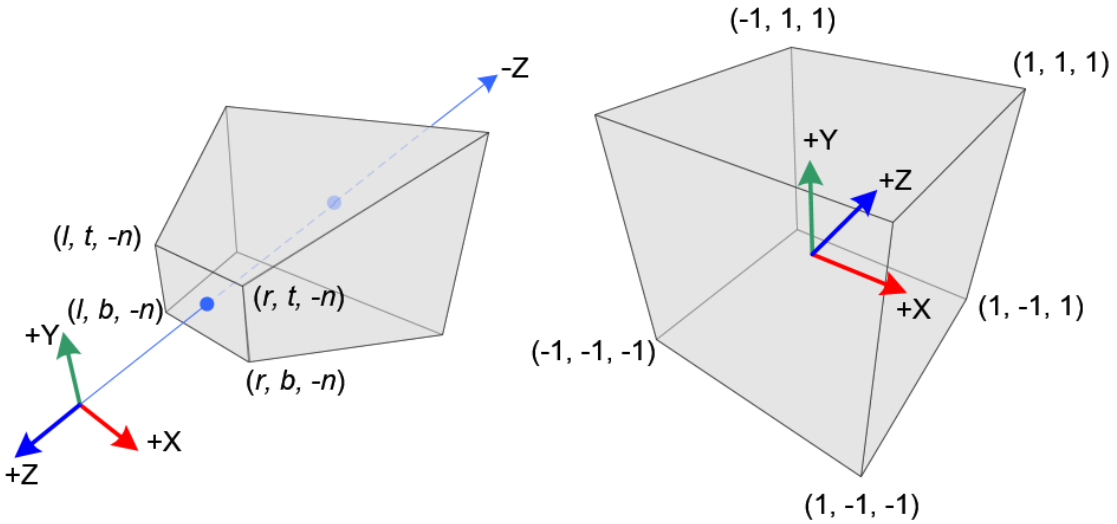
Výsledkem je oktalový strom, kde jsou všechny body roztříděny do uzlů v nejnižší úrovni. Dále se pak pracuje pouze s uzly nejnižší úrovně.

2.1.2 Ořezávání pohledovým objemem

Pohledový objem je oblast v prostoru, která bude vidět na obrazovce[60]. Tvar pohledového objemu je závislý na typu použité projekce.

[1] (*Axis-aligned bounding box*) Osově zarovnaný kvádr – stěny kvádru jsou kolmé na souřadnicové osy, je to současně min-max obálka bodů objektu[60]

[2] *(Normalized Device Coordinates)* je to souřadný systém, do kterého se převede scéna po projekci podělením čtvrtou složkou vektorů (w). Vše co bude vidět na scéně se pak nachází v prostoru ohraničeném krychlí se středem v počátku a délkou hrany 2.[15]



Obr. 2.4: Pohledový objem vlevo, pohledový objem transformovaný do NDC[2]vpravo [14]

Ořezávání pohledovým objemem je operace, kdy předem otestujeme, zda se objekt nachází alespoň částečně v pohledu, a tedy má smysl ho kreslit.

Tato metoda je použita na odfiltrování segmentů mračna bodů. Ořezávání pohledovým objemem má i jednu nevýhodu, přidává režii navíc, to znamená, že pokud tento test trvá déle, než vykreslení optimalizované části, může naopak výkon snížit. Proto je potřeba zvolit minimální množství práce, kdy je ještě výhodné ořezávat segmenty touto metodou viz srovnání v příloze C.

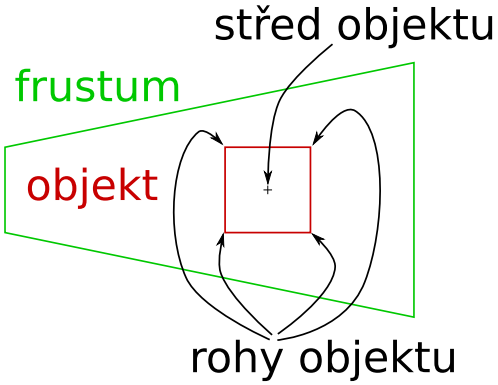
Důležitá vlastnost pohledového objemu je, že se po projekci transformuje do krychle se středem v počátku souřadnic a s délkou strany 2 viz obrázek 2.4. Tato vlastnost zásadně ulehčuje určení zda je libovolný bod v pohledu.

Může nastat několik případů průniků pohledového objemu a segmentu.

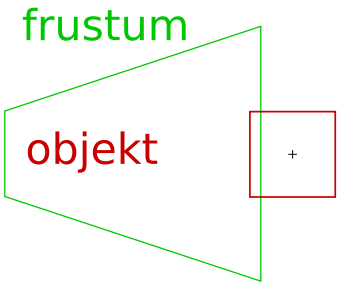
- Všechny rohy segmentu jsou v pohledu viz obrázek 2.5(a).
- Pouze několik rohů ze segmentu nebo jeho střed je v pohledu viz obrázek 2.5(b).
- Žádný z rohů nebo střed je v pohledu, ale některý z rohů pohledového objemu je v segmentu viz obrázek 2.5(c).
- Pohledový objem protíná segment tak, že rohy pohledového objemu nejsou uvnitř segmentu, ani rohy nebo střed segmentu uvnitř pohledového objemu viz obrázek 2.5(d).

Ověřování průniku pohledového objemu a segmentu

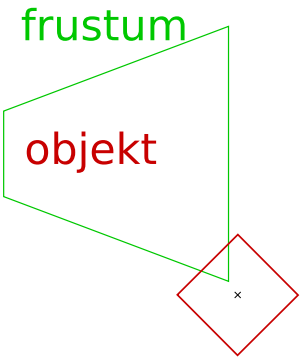
Nejprve se vypočítají souřadnice rohů segmentu jak v modelspace, tak clipspace souřadnicích, poté se ověří, zda jsou rohy i střed segmentu v pohledu. Pokud jsou



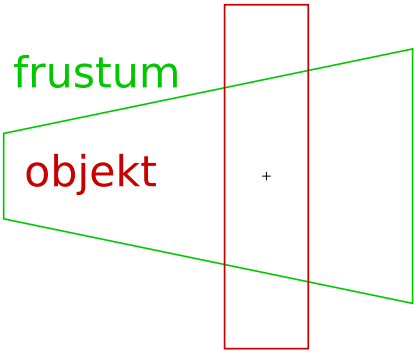
(a) všechny rohy jsou v pohledu



(b) některé rohy jsou v pohledu

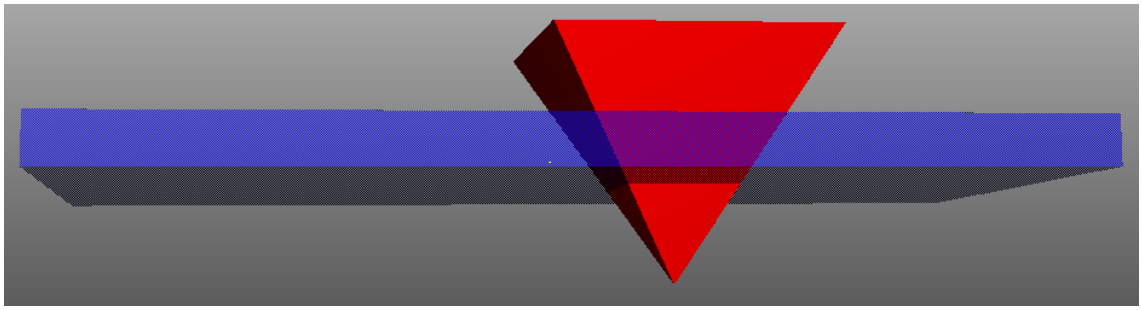


(c) roh pohledového objemu je uvnitř segmentu



(d) průnik bez společných rohů

Obr. 2.5: Možné průniky pohledového objemu a segmentu



Obr. 2.6: Problémový průnik

v pohledu všechny rohy, znamená to, že celý segment je v pohledu, tudíž se musí vykreslit celý. V případě, že je v pohledu alespoň jeden roh nebo střed, znamená to, že je segment vidět jen částečně.

Pokud není vidět žádný bod, je třeba pokračovat s testy. Ověří se, zda stěny segmentu neobepínají některý z rohů pohledového objemu. Pokud ano, je segment částečně v pohledu.

Pokud ne, pokračuje se dalším testem. Ověří se, zda některá z hran segmentu neprotíná pohledový objem. Pokud ano, je segment částečně v pohledu, pokud ne, segment není v pohledu.

Později byl objeven ještě jeden možný způsob průniku viz obrázek 2.6. Tento průnik není v rovině možný, protože když v rovině hrany segmentu protínají pohledový objem, hrany pohledového objemu zároveň protínají segment, což v prostoru neplatí. Proto se při prvotním hledání možností tato nenašla.

Výsledkem tohoto ověření jsou 3 možné varianty:

Segment plně v pohledu - segment se vykreslí celý.

Segment částečně v pohledu - segment je částečně v pohledu, to znamená že je vhodné tento segment dále optimalizovat.

Segment mimo pohled - segment je úplně mimo pohled, to znamená, že se může celý přeskočit.

2.1.3 Optimalizace pomocí geometry shader

Aby bylo možné vynechat ještě více bodů mimo zorné pole, bylo využito výpočetních schopností grafické karty. Geometry shader je volitelná fáze grafického řetězce, která má proměnné množství výstupních primitiv[37]. Pomocí něj se testuje, zda jsou jednotlivé body uvnitř zorného pole, pokud ne, zahodí se.

Nabízí se minimálně 2 možnosti jak toto implementovat.

Generování krychle v geometry shader

Jednotlivé body jsou otestovány v geometry shaderu, zda jsou v pohledu, pokud ano, vygeneruje se v tomto shaderu krychle, jinak se bod zahodí.

Generování krychlí z nezahozených bodů se ukázalo viz příloha B, jako neefektivní, protože geometry shader trpí na problémy se synchronizací a datovou propustností[18]. Proto byl přístup změněn.

Využití transform feedback

Do grafické karty se načtou pouze body, v geometry shaderu se otestuje, zda jsou v pohledu a pokud ano, uloží se do *transform feedback buffer*[43] a vynechají se další fáze grafického řetězce, zároveň se výsledný počet bodů zachytí pomocí *query object*[42]. Tento buffer se poté použije v druhém běhu místo originálního bufferu.

Původně se hodnota query načítala zpátky do paměti RAM, pak se sestavil draw call a zavolal se. Toto je ale velmi neefektivní způsob, protože se vykonávání programu zastaví a čeká na dokončení první fáze, poté sestavuje vykreslovací příkaz a teprve poté se opět používá grafická karta. V době, kdy se čte query a sestavuje vykreslovací příkaz, grafická karta nevykonává žádnou činnost. Tento jev se někdy označuje pipeline stall.

Asynchronní čtení transform feedback query

Po prostudování internetových zdrojů jsem narazil na diskuzi[5], ve které se tento problém řešil. Řešením je použití *query buffer object*[41] a *indirect rendering*[38]. *Query buffer object* je rozšíření, které umožní grafické kartě přikázat, aby obsah query překopírovala na jiné místo v paměti grafické karty. Protože tento příkaz na rozdíl od čtení do paměti programu nevyžaduje vyprázdnění fronty grafické karty, nečeká program na tomto místě na dokončení *transform feedback*, uloží ho do fronty příkazů a pokračuje dále. Jako další příkaz je vykreslovací příkaz `glDrawElements Indirect`. Tento příkaz vyžaduje, aby parametry vykreslování byly předem uloženy v grafické paměti. Do grafické paměti se předchystají všechny parametry kromě počtu bodů, které se mají vykreslit, počet bodů poté doplní příkaz z query.

Takováto je teorie, v mém případě bylo zlepšení patrné pouze na grafických kartách Intel, všechny ostatní grafické karty vykazovaly spíše mírné zhoršení viz kapitola 5.3.2.

Po konzultaci s Ing. Miletem jsem se dozvěděl, že geometry shader není optimální přístup viz [18], lepší je použít compute shader. Nakonec jsem optimalizoval segmentaci a ořezávání pohledovým objemem natolik, že tato fáze nebyla potřeba – viz následující kapitola.

2.1.4 Druhá iterace optimalizací

Na konzultaci mě Ing. Milet přivedl na nápad vylepšení segmentace octree.

Původní algoritmus segmentace fungoval tak, že po rozdělení všech segmentů nechal pouze ty nejnižší uzly.

Nyní se segmenty dělí stále stejně, ale body se nakonec uloží zpět do původního pole a poté se v jednotlivých segmentech pouze zanechá informace o počátku a délce

úseků bodů v tomto poli. OpenGL volání využívá tyto dvě informace a uložení do jednoho pole umožňuje obsáhnout v jednom volání více segmentů.

Díky této úpravě segmentů je možné provádět ořezávání pohledovým objemem hierarchicky, to znamená, že pokud se hlavní segment nevejde do pohledu celý, zkontrolují se podsegmenty. Naopak pokud se segment celý vejde, může se jedním příkazem obsáhnout podřazené segmenty. Pokud už segment nemá potomka, ale stále se nevejde do pohledu, tak se už nevykresluje přes geometry shader a transform feedback, ale kreslí se celý. Důvodem je to, že je možné nyní rozdělit mračno na mnohem menší segmenty a tyto segmenty jsou poté tak malé, že se geometrická fáze nevyplatí. Nová velikost byla zvolena experimentálně 1000 bodů, detaily jsou popsány v příloze D.

Nynější metoda má jednu nevýhodu a to proměnný počet volání, podle toho, kolik segmentů se bude vykreslovat. Toto se dá vyřešit pomocí *indirect multi draw call*. Je to indirect call, který navíc vykreslí seznam příkazů. Při ořezávání pohledovým objemem se generuje seznam segmentů k vykreslení, který se použije ve volání grafické karty.

Tento nový způsob je lepší jak při velké části objektu v pohledu, kde se sníží počet volání grafické karty, tak při malé, protože se může optimalizovat s větší přesností. Výsledky této optimalizace jsou popsány v kapitole 5.3.3.

2.1.5 Další možná zlepšení

Vykreslování mračna se dá stále vylepšovat.

Dalším vylepšením by mohlo být využití toho, že mračno bodů může být tak husté, že přední části plně zakrývají zadní části, což znamená, že se zadní části nemusí kreslit. Této metodě se říká *Occlusion culling*[16].

Protože mračno bodů je při vykreslování limitováno geometrickým výkonem, nabízí se toto řešení: Použije se *occlusion query*[39] a *conditional rendering*[35].

Occlusion query je podskupinou query objektů. Query objekty jsou určeny pro asynchronní dotazování na různé informace z grafického ovladače. Occlusion query slouží k určení, zda vykreslený objekt nějak přispěl do výsledné scény.

V praxi se tato metoda používá tak, že se místo objektu, který je jak geometricky, tak rasterizačně složitý, vykreslí nějaká jednoduchá náhrada, většinou jeho obálka a kontroluje se zda by některý z fragmentů ovlivnil výsledný obraz, pokud ano, vykreslí se originální objekt.

Segmenty k vykreslení by se nejprve seřadily od nejbližšího k pozorovateli, poté by se vždy nejprve vykreslila pouze obálka a pokud by tato obálka přispívala k výslednému obrazu, vykreslil by se segment.

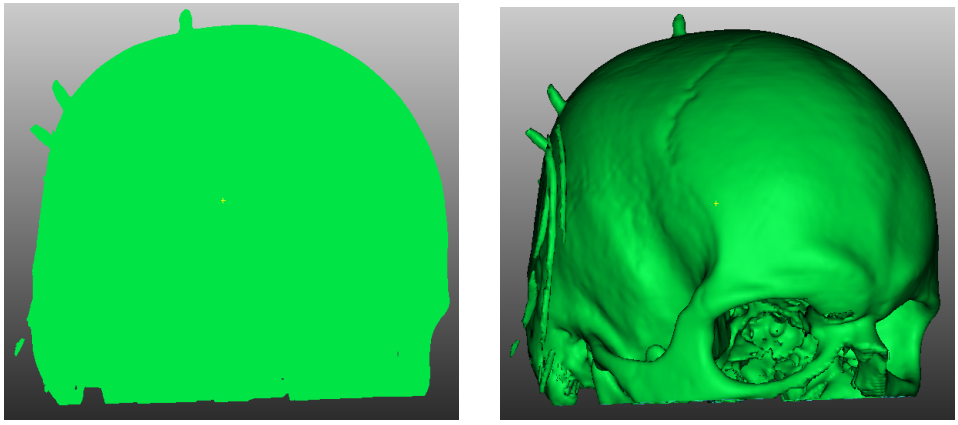
Tomuto by se ale muselo přizpůsobit i ořezávání pohledovým objemem. Ten totiž pokud zjistí, že je celý velký segment vidět, dále ho nedělí a nechá vykreslit, což ale může být zbytečné, protože je velmi vzdálený. Kdyby se rozdělil na menší segmenty, mohly by se vykreslit pouze přední a zbytek zahodit. Možné řešení by bylo nastavit hranici maximálního počtu bodů vykreslených jedním příkazem na vhodnou hodnotu.

2.2 Vykreslování mesh modelů

Formát souboru PLY je přímo kompatibilní s OpenGL, protože mesh je definován jednou posloupností indexů do množiny vrcholů, každý vrchol musí mít svou pozici, barvu, normálu i skalární pole. To znamená že není potřeba v mezikroku celý mesh přeorganizovat. Toto uspořádání je výhodné proto, že šetří místo díky neopakování totožných vrcholů a také je možné využít *post transform cache*[40]. Jediné na co je potřeba si dát pozor je rozdílný formát barev. Zatímco PLY používá celočíselné hodnoty od 0 do 255, OpenGL používá hodnoty s plovoucí desetinnou čárkou v rozsahu od 0 do 1.

Mesh se vykresluje jako posloupnost trojúhelníků, které se skládají z indexů do množiny vrcholů. Normály meshe jsou velmi důležité, protože se díky nim dá dělat osvětlení povrchu pomocí *Phongova osvětlovacího modelu*[60]. Bez osvětlovacího modelu by se mesh vykreslil pouze jako silueta viz obrázek 2.7.

Všechny meshe, které byly při vývoji k dispozici, nebyly dostatečně geometricky náročné viz graf 5.6, takže jakákoli optimalizace byla zbytečná.



(a) Neosvětlený mesh (b) Osvětlený mesh

Obr. 2.7: Rozdíl ve vykreslování mezi osvětleným a neosvětleným meshem

2.3 Vykreslování průhledných mesh modelů

Průhledný mesh je takový mesh, který má pro vrcholy nejenom barvu (RGB) ale i průhlednost (RGBA). Na první pohled se může zdát, že to bude pouze triviální rozšíření neprůhledného meshe, ale je třeba si uvědomit jakým způsobem grafické karty vykreslují objekty.

Pokud kreslíme neprůhledné objekty, viditelnost stěn za nás vyřeší grafická karta pomocí *paměti hloubky*[60]. Poloprůhledné stěny nezakrývají vzdálenější stěny, naopak je třeba barvu poloprůhledné plochy smíchat s barvou, objektů v pozadí. Barva se často míchá pomocí *alfa míchání*[60]. Toto míchání ale není komutativní, proto musíme dodržet pořadí. Dodržení pořadí kreslení je složité, objekt nejde seřadit předem, protože seřazení závisí na směru pohledu.

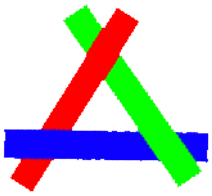
2.3.1 Možná řešení průhledného vykreslování

Existuje mnoho způsobů průhledného vykreslování. Z časových důvodů nebylo možné se všemi zabývat. Nakonec byla zvolen malířův algoritmus se stromem BSP.

Malířův algoritmus

Tento algoritmus řeší průhledné vykreslování právě tak, že seřazuje primitiva od nejvzdálenějšího k nejbližšímu z pohledu kamery. Řešení viditelnosti je tedy převedeno na úlohu seřazení stěn podle vzdálenosti od pozorovatele.

Problémem tohoto algoritmu je nejen časová náročnost samotného seřazování, ale i fakt, že stěny jsou trojrozměrné objekty. Existuje celá řada algoritmů, které řeší seřazování, přesto existují případy, které nelze vyřešit přímo. Pokud mají stěny nekonvexní tvar, mohou být vzájemně propleteny a tyto stěny nejde správně seřadit. Jediné řešení je rozdělit nekonvexní stěny na konvexní. Nicméně ani existence pouze konvexních stěn nezaručuje bezproblémovost. Můžou se například cyklicky překrývat 3 stěny viz obrázek 2.8. Toto se dá zase vyřešit jedině rozdělením stěn.[60]



Obr. 2.8: Příklad cyklického překrytí tří stěn[36]

Malířův algoritmus se stromem BSP

Aby se předešlo problémům s řazením stěn, využije se *binary space partitioning tree*. Při vytváření tohoto stromu se používají jednotlivé stěny jako řezné roviny, které rozdělí prostor na dva poloprostory. Pokud tato rovina protíná některou stěnu, je nutné v tomto protnutí stěnu rozdělit. Poté se vezmou plošky z každého poloprostoru a opět se dělí.[60]

Dělení jednotlivých stěn se provádí modifikovaným Sutherland-Hodgman algoritmem, převzaným z [27]. Výstupem originálního algoritmu byl pouze mnohoúhelník na jedné straně řezné roviny, proto se modifikoval tak, že nyní generuje mnohoúhelník pro obě strany. Druhou modifikací je numerická robustnost. Pokud by byl totiž jeden vrchol velmi těsně u řezné roviny, nemusela by se jeho poloha kvůli nepřesnosti výpočtů s plovoucí desetinnou čárkou správně vyhodnotit. Proto se zavedla *tlustá řezná rovina*. Toto opatření způsobí, že rohy mnohoúhelníků, které jsou jen nepatrně za řeznou rovinou, nebudou odřezány.

Hlavní nevýhodou tohoto algoritmu je, že výsledné množství stěn je velmi závislé na pořadí, ve kterém se kontrolují jednotlivé stěny. Velký počet stěn nejenom, že výrazně zvýší geometrickou náročnost vykreslování, ale také náročnost při seřazování. Seřazování poté bude trvat tak dlouho, že se nestihne v přípustné době. Pokud bychom v každém snímku čekali na seřazení, snímková frekvence by byla nízká a výsledný efekt by se nedal považovat za plynulý. Proto byl zvolen jiný způsob. Seřazování probíhá asynchronně a jakmile jsou data znovu seřazeny, tyto nové data se použijí k vykreslování. Snímková frekvence je nyní přijatelná, ale při pohybu se objevují artefakty. Výsledný vjem je ale mnohem lepší, než jaký by byl při nízké snímkové frekvenci. Celkové vylepšení by znamenalo použít pravděpodobně kompletní změnu přístupů k průhledným meshům.

3 Návrh grafického uživatelského rozhraní

Pro ovládání programu bylo nutné vytvořit uživatelské rozhraní. OpenGL ale nepodporuje gui. Bylo nutné si vlastní rozhraní navrhnout. Rozhraní bylo navrženo pro ovládání klávesnicí a pote bylo rozšířeno o ovládání pomocí ovladačů viz obrázky.

Grafické rozhraní se skládá ze dvou skupin objektů, scény a prvků.

3.1 Scéna

Scéna je seskupení prvků, zobrazena a aktivní je vždy jen jedna. Scéna zpracovává povely z klávesnice jako výběr prvku, aktivaci prvku nebo návrat na předchozí scénu.

3.2 Prvek

Prvek může mít více významů, může odkazovat na jinou scénu nebo může provádět nějakou akci. Podle chování se dají rozdělit do tří skupin:

- pasivní (popisek)
- tlačítko (po aktivaci přesměruje na jinou scénu)
- posuvník (pro nastavení hodnot)

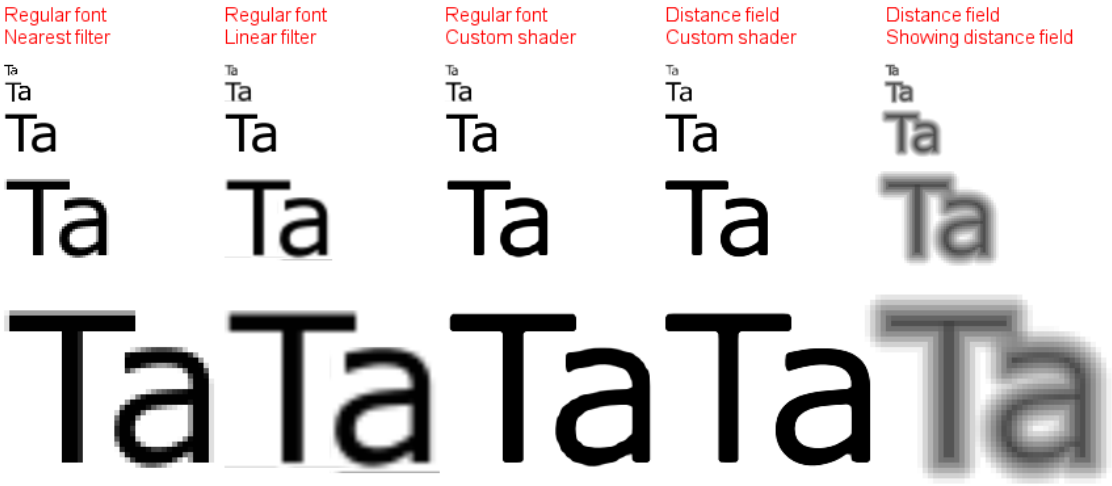
3.3 Vnitřní logika

Grafické rozhraní pracuje jako stavový automat. Program má ukazatel na aktuálně aktivní scénu. Při stisku kláves, kterými se rozhraní ovládá, se pomocí tohoto ukazatele pošle rozhraní zpráva, toto rozhraní podle toho zareaguje a jako výsledek vrátí ukazatel na nově aktivní scénu. Při stisku `enter` se aktivuje aktuálně vybraný prvek a provede se požadovaná operace. Může to být přesun do jiné scény nebo v případě načítacího dialogu se spustí načítání souboru.

Při stisku `escape` se zobrazí nadřazená scéna.

Při stisku šipky nahoru a dolů se vybírá aktivní prvek, tento výběr se provádí změnou indexu do pole prvků. Scéna ověří, aby nedošlo k přetečení nebo podtečení tohoto indexu. V případě velkého množství prvků ve scéně se při přetečení označí první a při podtečení poslední prvek. Jinak se při přetečení označí poslední a při podtečení první prvek.

Šipky doleva a doprava fungují jen v případě posuvníku. Stiskem se nastavuje hodnota nastavená posuvníkem (např. velikost bodů nebo prolnutí skalárního pole).



Obr. 3.1: Porovnání technik kreslení bitmapových textů[21]

3.4 Vykreslování textu

Toto grafické rozhraní vyžaduje text. OpenGL nemá žádné funkce pro vykreslování textu. Text se dá vykreslovat texturováním obdélníku texturami jednotlivých znaků. Toto není ideální způsob, protože pokud tyto textury nejsou mapovány pixel na pixel obrazovky, jsou neostré. Tento nedostatek se většinou řeší pomocí TrueType fontů, tyto fonty jsou definovány pomocí křivek. Textura se poté generuje vždy podle toho, jak velká je potřeba. Toto řešení je možné v případě, že text je zobrazen kolmo na pozorovatele a je předem známo, kolik pixelů bude mít tento text na výšku. To ale není náš případ. Nejen, že přesná velikost v pixelech není známa, protože se může měnit a generovat nové textury při každé změně je z hlediska výkonu nemožné. Dokonce v některých případech nemusí být text zobrazený kolmo, tím pádem se z obdélníku stává lichoběžník a s takovým případem knihovna FreeType nepočítá.

Tento problém se dá vyřešit pomocí *Distance field font*. Je to metoda kreslení bitmapových textů s ostrým okrajem dokonce při vysokém přiblížení viz obrázek 3.1.[21] Do textury se uloží vzdálenost od hrany, kde 1 (na obrázku 3.1, pátý sloupec černě) značí plně uvnitř znaku, 0 značí plně mimo text a 0.5 značí hranu znaku. Takovéto uložení způsobí, že lineární interpolace textury nezkreslí informaci o hraně znaku. Ve fragment shaderu se poté provede prahování, to znamená, že hodnoty menší než 0.5 se převedou na barvu pozadí a hodnoty vyšší se převedou na barvu textu.

4 Popis programu VR-pointcloud-viewer

Aplikace je multiplatformní a je napsána v jazyce C++. Pro vytvoření kontextu a okna používá knihovnu GLFW. Protože ukazatele na funkce rozhraní OpenGL je nutné načíst za běhu, používá knihovnu OpenGL Extension Wrangler (GLEW). K načítání modelů slouží knihovna tinyply[25]. Aby byl program schopný načíst obrázky používá Simple OpenGL Image Library (SOIL)[30]. Na procházení souborů na disku používá součást filesystem knihovny Boost[24].

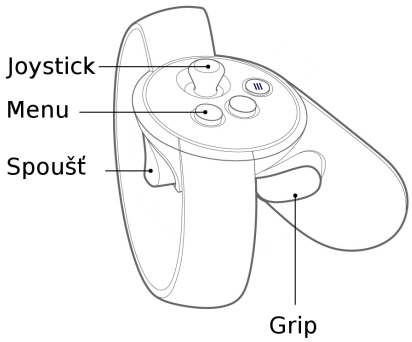
4.1 Uživatelská příručka

Před spuštěním programu je nutné mít spuštěn SteamVR. Program se spouští programem `VR-pointcloud-viewer` v podsložce `bin`. Po spuštění se objeví prázdné okno viz obrázek 4.1. Vlevo nahoře je vidět ukazatel pozice kamery, vlevo dole je vidět orientace kamery, kde červená osa vyznačuje osu x, modrá osu y a zelená osu z.

Obr. 4.1: Úvodní obrazovka

4.1.1 Ovládání

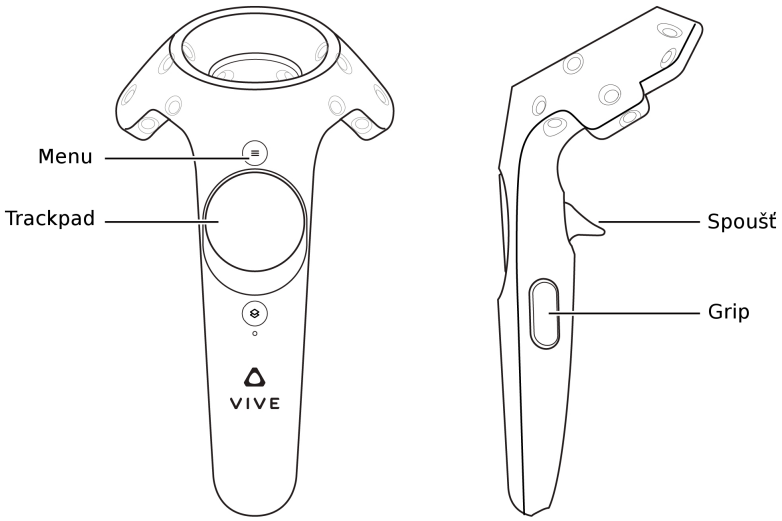
Program se dá ovládat pomocí klávesnice nebo ovladačů viz obrázky 4.2 a 4.3.



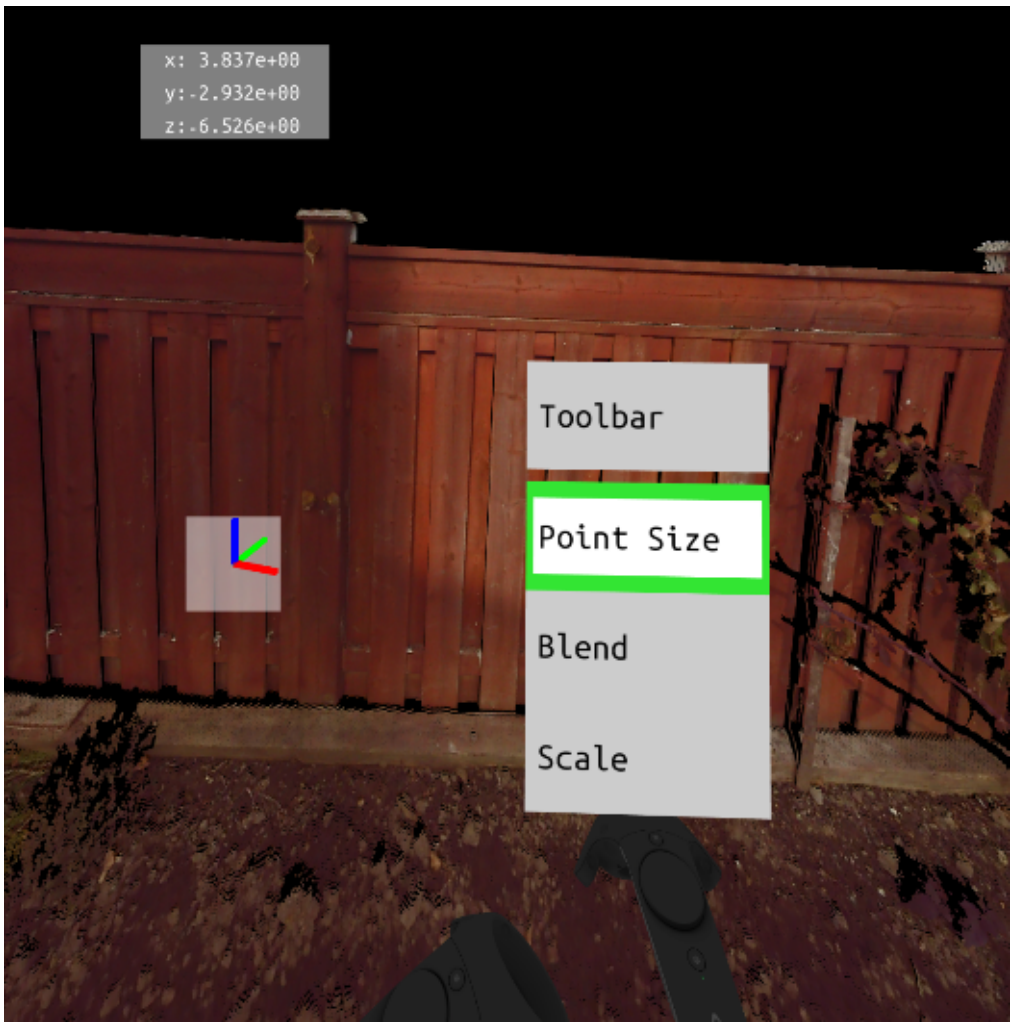
Obr. 4.2: Ovládání pomocí ovladačů k Oculus Rift

Pomocí klávesnice se pohled na objekt ovládá následovně: Klávesami `W` a `S` se pohybuje dopředu a dozadu. Klávesami `A` a `D` se pohybuje doleva a doprava. Klávesami `Space` a `Alt` se pohybuje nahoru a dolu. Klávesami ↑ a ↓ se rotuje podle osy x. Klávesami ← a → se rotuje podle osy y. Klávesami `Q` a `E` se rotuje podle osy z. Klávesou `R` se zvětšují body mračna bodů, klávesou `F` se zmenšují. Klávesami `T` a `G` prolíná skalární pole.

Pomocí ovladačů se objekt ovládá následovně: První možnost je použití pravé spouště. Po jejím stisknutí bude objekt kopírovat pohyby pravého ovladače. Druhou možností je požití joysticků nebo trackpadů. Trackpady fungují podobně jako joystick, poloha prstu na jejich povrchu odpovídá natočení joysticku. Pohybem pravého joysticku nahoru a dolů se pohybujeme dopředu a dozadu. Pohybem pravého joysticku doleva a doprava se pohybujeme doleva a doprava. Pohybem levého joysticku nahoru a dolů se pohybujeme nahoru a dolů. Pohybem levého joysticku doleva a doprava se otáčíme doleva a doprava. Velikost bodů a prolnutí skalárního pole se mění v nabídce nástrojů, která se po stisknutí pravého gripu objeví nad pravým ovladačem viz obrázek 4.4. V této nabídce se orientuje stejně jako v menu. Po vybrání některého z nástrojů se objeví posuvník (obrázek 4.5) a pohybem pravého joysticku doleva se hodnota snižuje a doprava zvyšuje. Změna hodnoty je okamžitá, uložení se provede stisknutím tlačítka menu na pravém ovladači.



Obr. 4.3: Ovládání pomocí ovladačů k HTC Vive

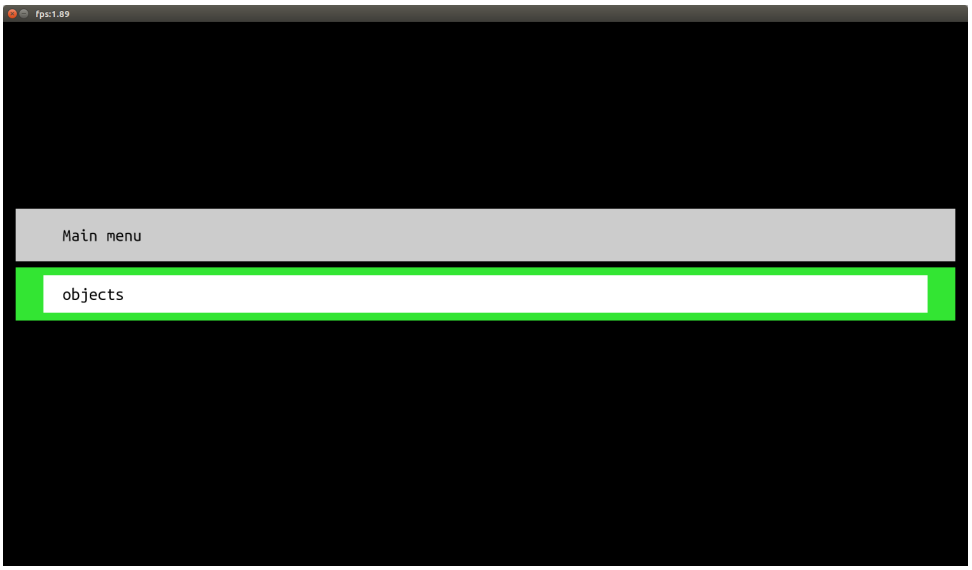


Obr. 4.4: Nabídka nástrojů

Obr. 4.5: Změna velikosti bodů

4.1.2 Načtení objektu

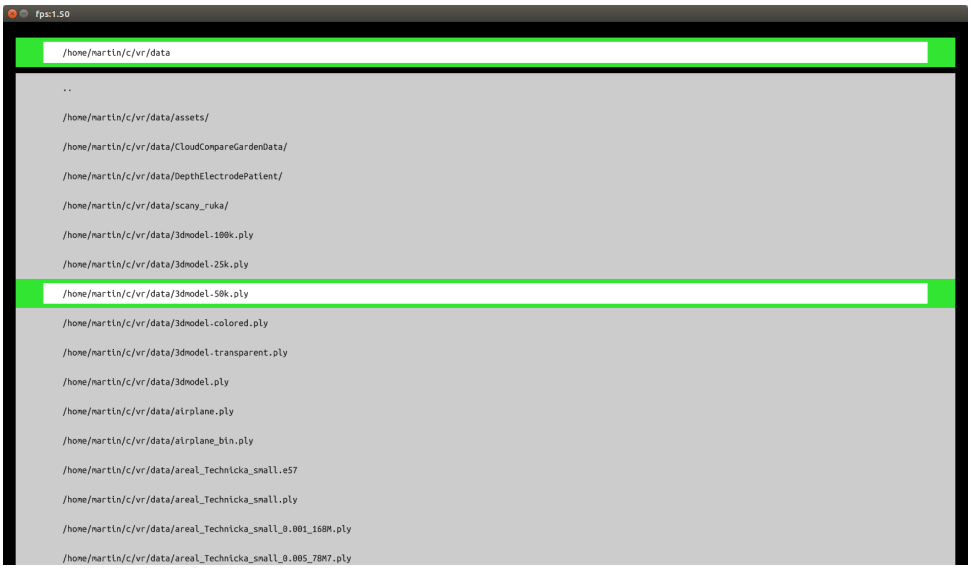
Objekt se načítá vstupem do menu (obrázek 4.6), do kterého se vstoupí stiskem klávesy `M` na klávesnici nebo pomocí příslušného tlačítka na ovladači. V hlavní nabídce je položka `objects`, v ní je možnost načítat další objekty a seznam všech načtených objektů viz obrázek 4.7.



Obr. 4.6: Hlavní menu

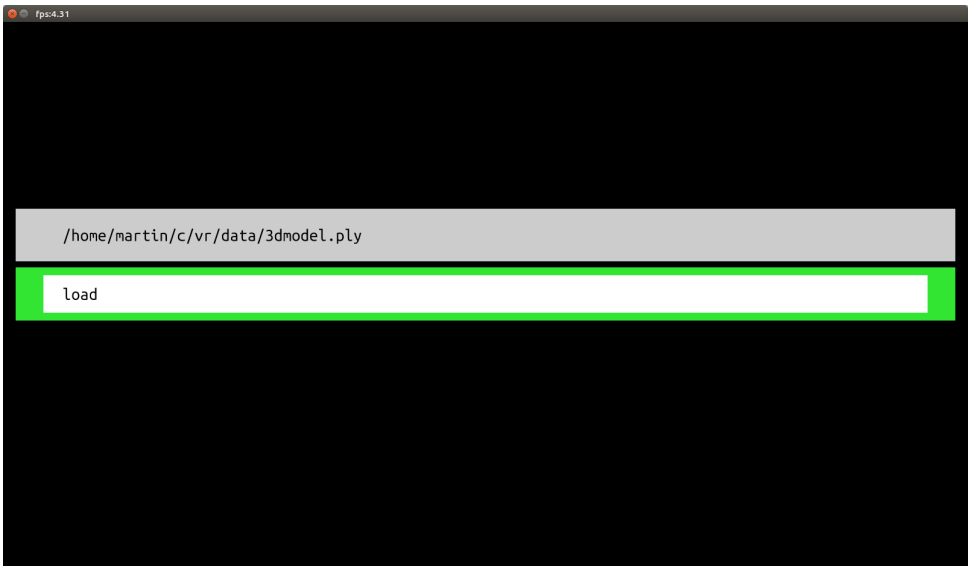
Obr. 4.7: Seznam načtených objektů

Objekty se načítají pomocí nabídky `load`. Zde se vybere soubor s modelem viz obrázek 4.8.



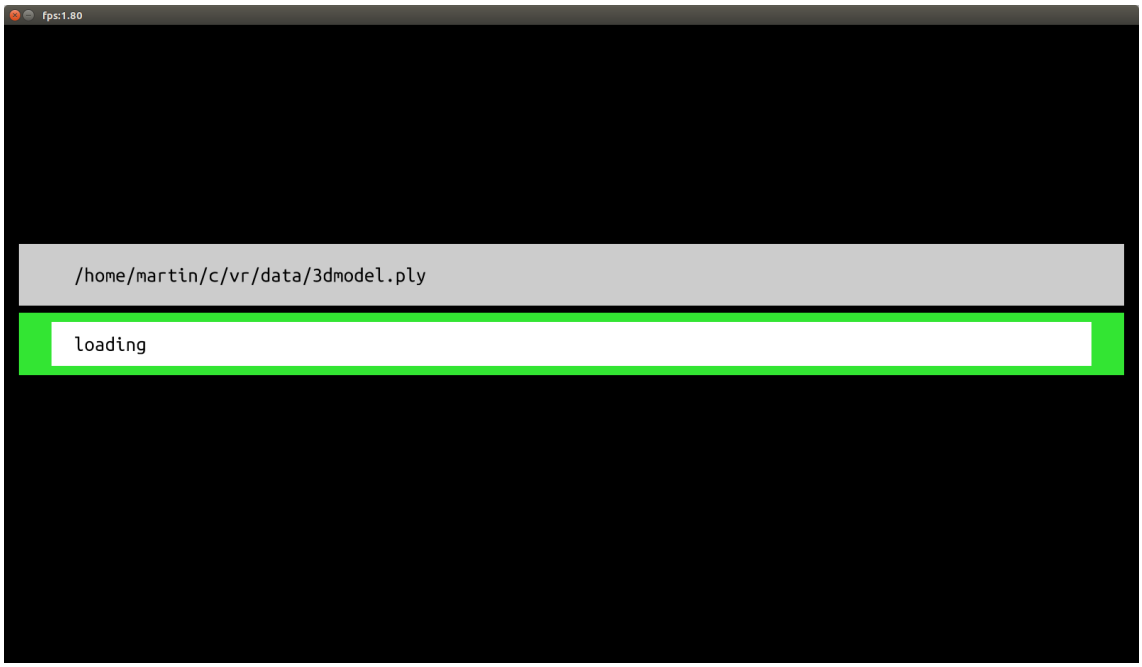
Obr. 4.8: Procházení souborů

Po vybrání se objeví načítací dialog viz obrázek 4.9, načítání se potvrdí stiskem klávesy `enter` nebo příslušným tlačítkem na ovladači. Zrušení načítání se provede stiskem klávesy `escape` nebo příslušným tlačítkem na ovladači.



Obr. 4.9: Načítací dialog

Po potvrzení se změní text tlačítka na `loading` viz obrázek 4.10. Při načítání nelze tento dialog opustit.



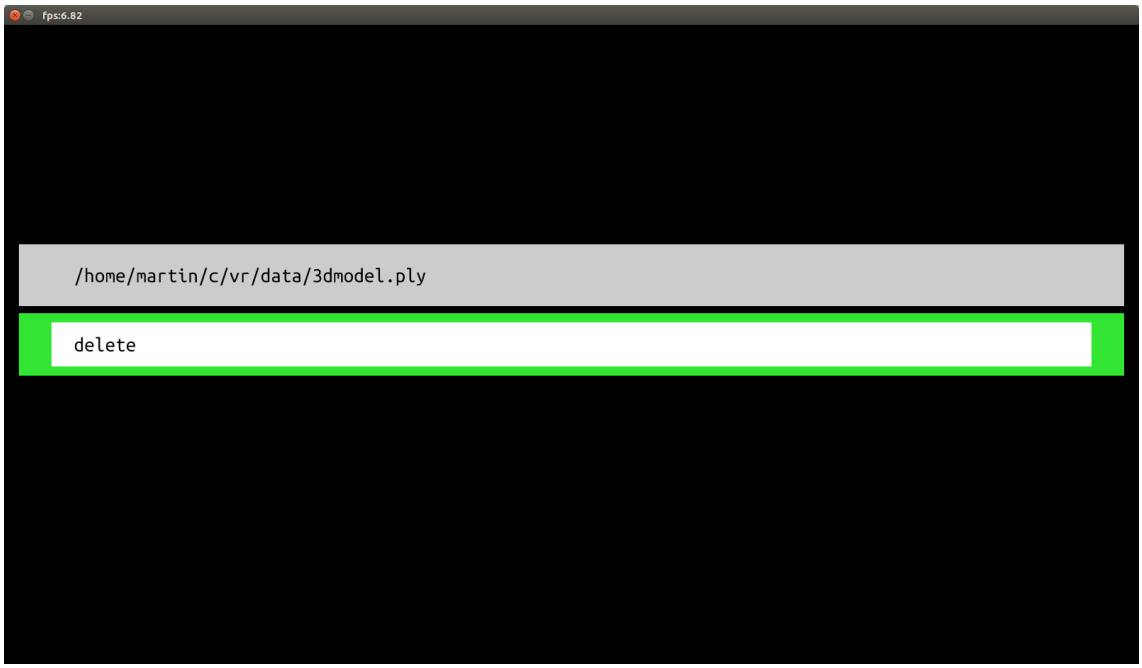
Obr. 4.10: Načítání objektu

Po načtení tlačítko zšedne a text se změní na `done` viz obrázek 4.11. Dialog se dá opustit jak klávesou `enter` tak klávesou `escape` nebo příslušným tlačítkem na ovladači. Menu se pak vrátí na seznam objektů viz 4.7.

Obr. 4.11: Objekt načten

4.1.3 Smazání objektu

Po vybrání objektu v seznamu se objeví nabídka, ve které je možné objekt smazat viz obrázek 4.12. Po stisknutí tlačítka se objekt okamžitě odstraní.



Obr. 4.12: Smazání objektu

4.1.4 Instalace programu

Instalační balíček byl testován na Windows 10, předpokládá se kompatibilita i s Windows 7, 8 i 8.1. Tento balíček ve formátu zip rozbalíme na místo, kde má být program instalován. Poté spustíme instalaci Distribuovatelné součásti Visual C++, která je přibalena pod názvem `vc_redist.x64.exe`.

5 Výkonové srovnání

Aby mohla být ověřena účinnost optimalizací, musely se provést srovnávací testy. Výkon aplikací se hodnotí počtem vykreslených snímků za jednotku času, většinou za jednu sekundu, této jednotce se říká FPS.

Je potřeba si dát pozor na způsob měření snímků. Ovladač grafické karty má totiž v sobě frontu příkazů, to znamená, že když se program vrátí z volání funkce OpenGL, neznamená to, že je práce zadaná touto funkcí hotová, dokonce se nemusela ještě ani začít provádět. Kvůli tomu nelze měřit časovou náročnost jednotlivých operací pomocí systémového času. Nejlepší praktické řešení je měřit časy mezi jednotlivými přepínáními bufferů[44].

Aby toto srovnání bylo vypovídající, je třeba zavést kontrolované a opakovatelné měření.

5.1 Terminologie

Obvykle se výkon grafických karet a aplikací udává v průměrných FPS, toto ale není kompletní hodnocení, protože průměrné FPS nemusí nutně znamenat dobrý zážitek. Tento fakt byl v oblasti měření výkonu grafických karet znám po dlouhou dobu, ale nikdo nevěděl jak lépe vyhodnocovat výsledný zážitek.

S novým způsobem hodnocení přišel šéfredaktor webu TechReport, Scot Wasson v roce 2010[59].

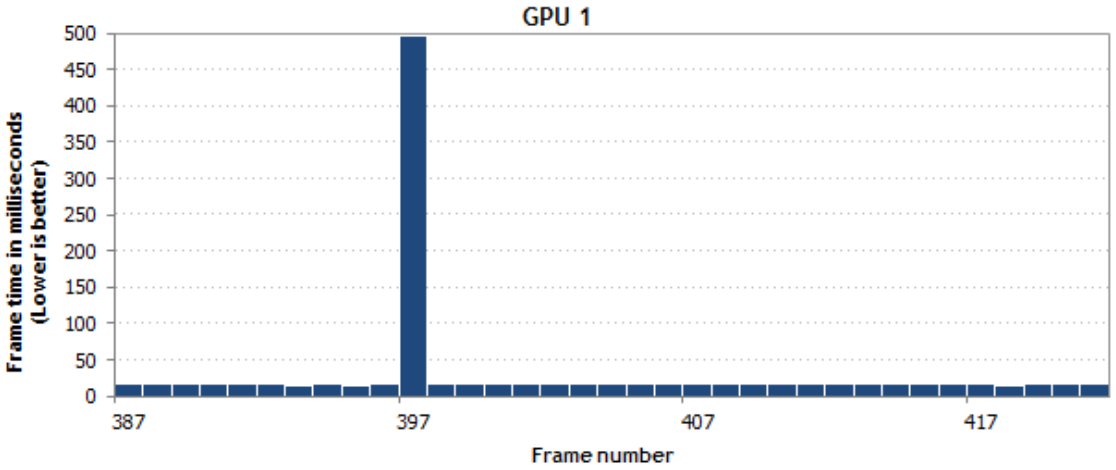
Místo měření celkového průměru FPS a průměru pro každou sekundu si začal ukládat časy všech snímků. Potom tyto časy začal analyzovat a všiml si, že některé sestavy mají sice dobré průměrné FPS, ale občas se vyskytne snímek, který se vykresluje mnohonásobně déle.

Mějme dvě hypotetické grafické karty, karta 1 má průběh časů snímků viz obrázek 5.1. Všechny snímky byly vidět přibližně 16-17 ms až na jeden, ten byl vidět 500 ms, to je v průměru 35 FPS viz obrázek 5.3. Druhá karta má všechny snímky zhruba kolem 33 ms viz obrázek 5.2, to je průměrně 34.7 FPS viz obrázek 5.3. Přestože obě karty dodaly přibližně stejné průměrné FPS karta 2 bude subjektivně lepší.

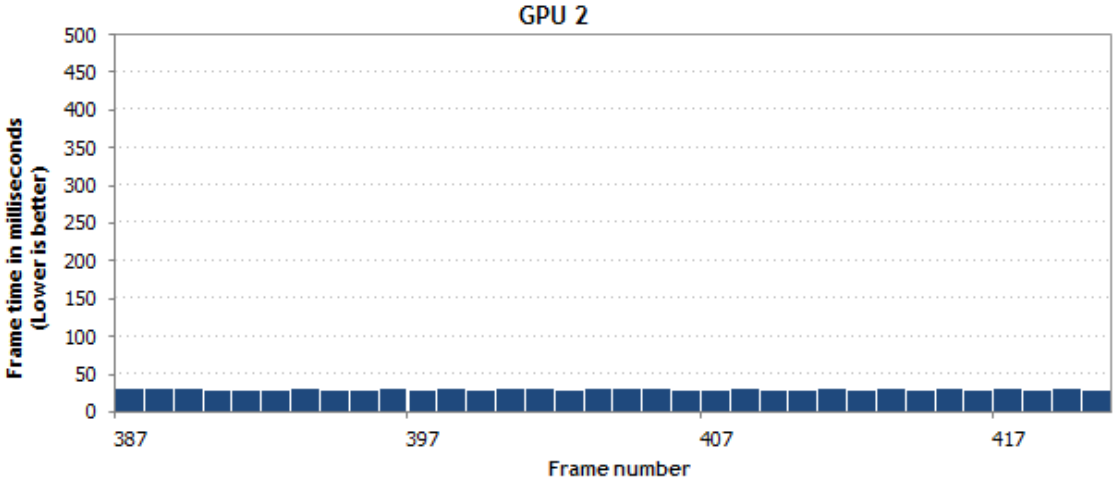
Proto se začal zabývat rozložením časů snímku, experimentoval s metrikami jako počet snímků s časy delšími než určitá hranice.

Nakonec zúročil své zkušenosti s měřením odezev serverů na požadavky a použil podobnou metriku – 99-percentil – tato říká, že časy snímků byly kratší nebo rovny této hodnotě. Čím víc se tato hodnota blíží průměru, tím je zobrazení plynulejší a tím i lepší zážitek.

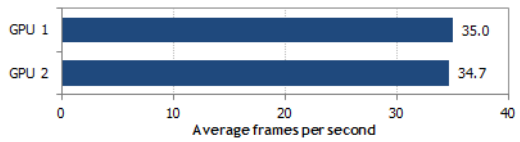
Scott poté začal tyto nové metriky používat ve svých recenzích. Později tuto metriku převzaly další recenzenti. Jedním z nich je web Gamers Nexus[20], který



Obr. 5.1: Příklad časů snímků grafické karty 1[59]



Obr. 5.2: Příklad časů snímků grafické karty 2[59]



Obr. 5.3: Příklad průměrných FPS dvou grafických karet [59]

se vyhodnocováním zabývá do hloubky a velký důraz klade na metodiku měření, tuto metriku ještě upravil. Přidal ještě hodnotu 99.9-percentil a hodnoty nevyjadřuje v časech snímků, ale převrácenou hodnotou, tedy v FPS. Navíc obě hodnoty přejmenoval tak, aby byly pochopitelnější pro širokou veřejnost. Místo 99-percentil používá označení 1% low FPS a místo 99.9-percentil 0.1% low FPS.

Způsob, který je použit v této práci, byl inspirován webem Gamers Nexus.

5.2 Testovací prostředí

Aby bylo měření maximálně opakovatelné, nesmí se používat žádný vstup od uživatele. Proto byla vyvinuta speciální aplikace, která používá předdefinované pohyby objektů.

Jsou použity 3 referenční objekty. Jako mesh je použita lebka, která má 623 728 trojúhelníků, má pouze normály. Jako průhledný mesh je použita tatáž lebka, ale s ručně zadanou bílou poloprůhlednou barvou. Tato lebka má po vygenerování BSP stromu 7 652 455 trojúhelníků. Jako mračno bodů je použit model naší fakulty, který má 1 375 498 bodů, má pouze barvu.

Pro každý objekt byly provedeny 3 testy.

- Statický objekt – objekt je celý vidět a nehýbe se, zde se nejlépe testuje stabilita snímků.
- Rotující objekt – objekt rotuje kolem kamery tak, že chvíli zaplňuje pohled a chvíli není vidět vůbec, tento test prověří efektivitu optimalizací vykreslování.
- Pohybující se objekt – objekt se přibližuje a oddaluje od kamery tak, že je chvíli vidět celý a chvíli není vidět vůbec, tento test také prověří efektivitu optimalizací vykreslování.

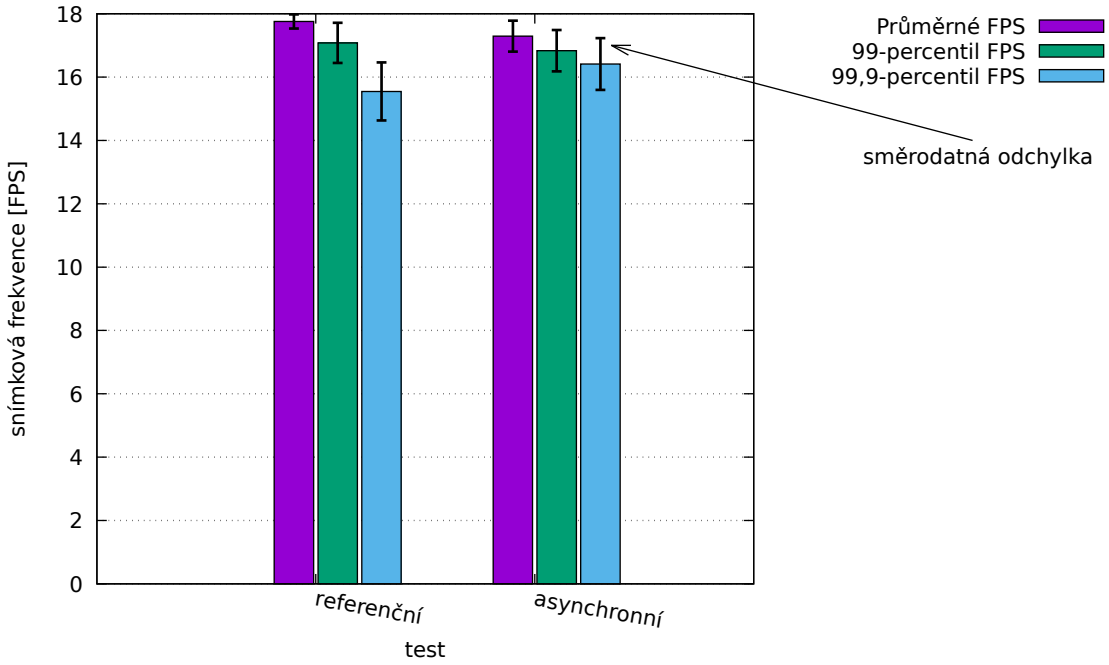
Pro mračno bodů byl proveden ještě jeden test a to je změna velikosti bodů. Celé mračno je vidět na obrazovce a plynule se mění velikost bodů. Tento test měl ukázat od jaké velikosti bodů je úzké hrdlo rasterizace, bohužel taková velikost bodů nebyla nalezena. Měnit maximální velikost bodů nebylo vhodné, protože by bylo znemožněno srovnání jednotlivých verzí programu.

Každý z těchto testů běží jednu minutu a po celou dobu se zaznamenávají všechny časy snímků.

Detailní rozbor byl proveden na referenčním počítači s následující konfigurací.

- Procesor Intel Core i7-6700K @ 4.5GHz
- Paměť 2x8GB Kingston HyperX Fury DDR4-2133 CL14-14-14 (HX421C14FB2K2/16)
- Základní deska ASRock Z170 Extreme4+
- Grafická karta ASUS DUAL RX580 OC 8GB @ 1360/8000MHz

Všechny testy na referenčním počítači byly provedeny alespoň 3x. Pokud nebude řečeno jinak, měření bude prováděno na referenčním PC.

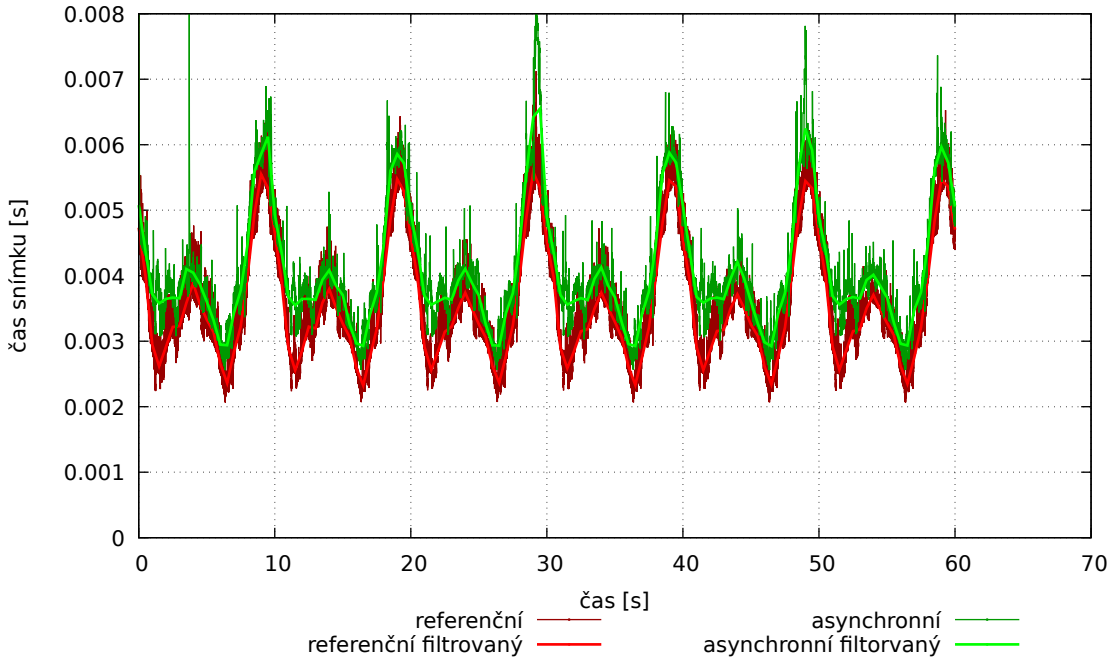


Obr. 5.4: Příklad grafu s průměrnou snímkovou frekvencí, 99-percentilem a 99,9-percentilem

5.3 Výsledky

Pro vyhodnocování jsou použity dva typy grafů. První typ grafu (obrázek 5.4) popisuje průměrnou snímkovou frekvenci, 99 percentil a 99.9 percentil. Protože měření jsou prováděna vícekrát, můžeme určit opakovatelnost měření. Směrodatná odchylka jednotlivých měření je v grafu zobrazena pomocí úsečky.

Druhý typ grafu (obrázek 5.5) znázorňuje průběh časů snímků v průběhu celého měření. V tomto měření vzniká velké množství šumu, který zhoršuje názornost průběhu. Z tohoto důvodu byl použit i filtr a filtrovaný graf je překreslen přes originální data. Tento filtr nejprve zaokrouhlí data na ose x na určitou hodnotu, která byla nastavena podle potřeby v rozmezí od 0,1 do 0,5. Takto se několik vzorků zaokrouhlí na stejný čas a ty se pak zprůměrují. Mezi těmito body se pak vede lomená čára.



Obr. 5.5: Příklad grafu s průměrnou snímkovou frekvencí, 99-percentilem a 99,9-percentilem

5.3.1 Referenční výsledky

Nejdříve bylo nutno stanovit referenční výsledky, vůči kterým se bude optimalizace porovnávat. Jako reference byla zvolena verze ze dne 12. 3. 2018, v této verzi je použit starý způsob segmentování a geometry shader.

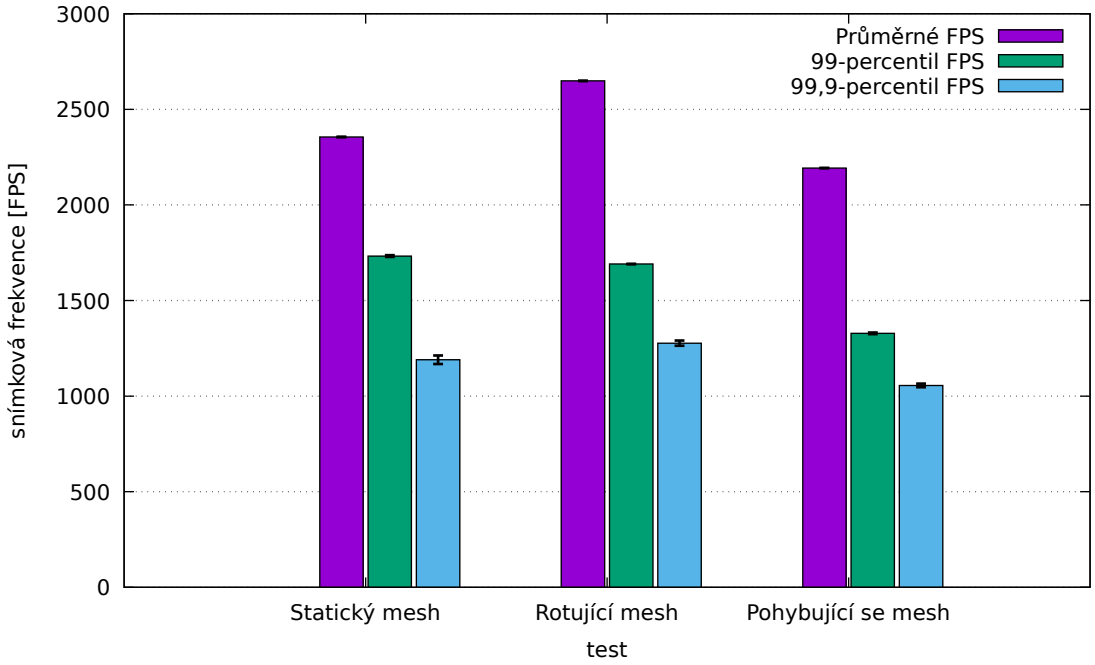
Neprůhledný mesh

Výsledný výkon pro referenční měření neprůhledného meshe viz obrázek 5.6. Statický mesh se kreslí v průměru 2355 FPS, 99-percentil 1733 FPS a 99,9-percentil 1190 FPS. Rotující mesh se kreslí v průměru 2648 FPS, 99-percentil 1691 FPS a 99,9-percentil 1277 FPS. Pohybující se mesh se kreslí v průměru 2193 FPS, 99-percentil 1328 FPS a 99,9-percentil 1056 FPS.

Po těchto testech bylo ihned jasné, že pokud nebude k dispozici složitější mesh, nemá smysl se zabývat jakoukoli optimalizací.

Průhledný mesh

Výsledný výkon pro referenční měření průhledného meshe viz obrázek 5.7. Statický mesh se kreslí v průměru 256 FPS, 99-percentil 238 FPS a 99,9-percentil 24 FPS. Rotující mesh se kreslí v průměru 261 FPS, 99-percentil 232 FPS a 99,9-percentil



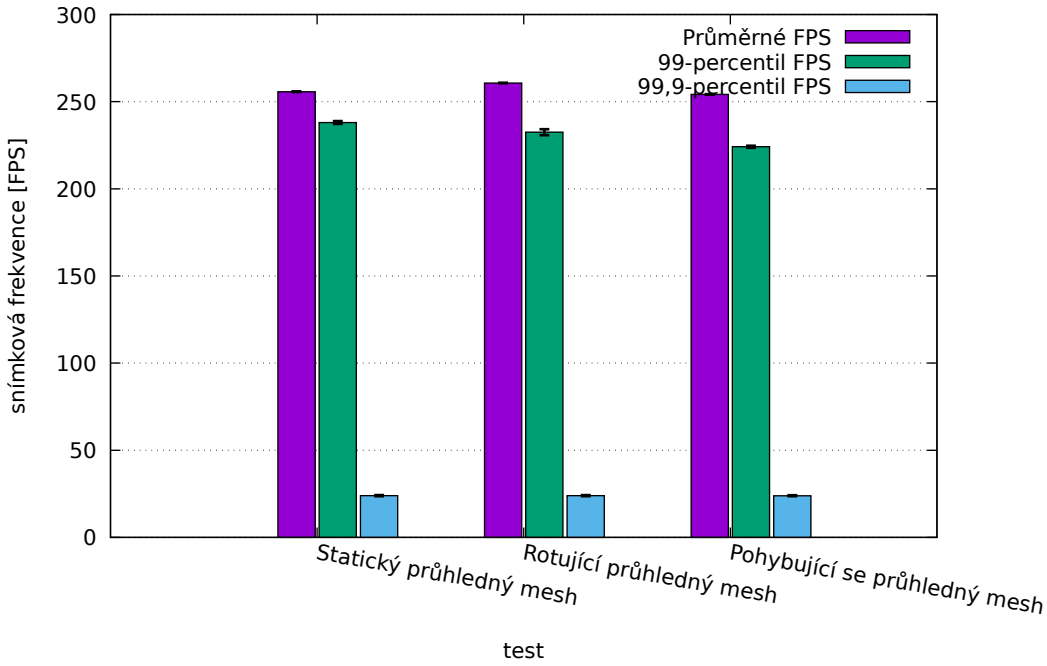
Obr. 5.6: Snímková frekvence neprůhledného meshe

24 FPS. Pohybující se mesh se kreslí v průměru 254 FPS, 99-percentil 224 FPS a 99,9-percentil 24 FPS.

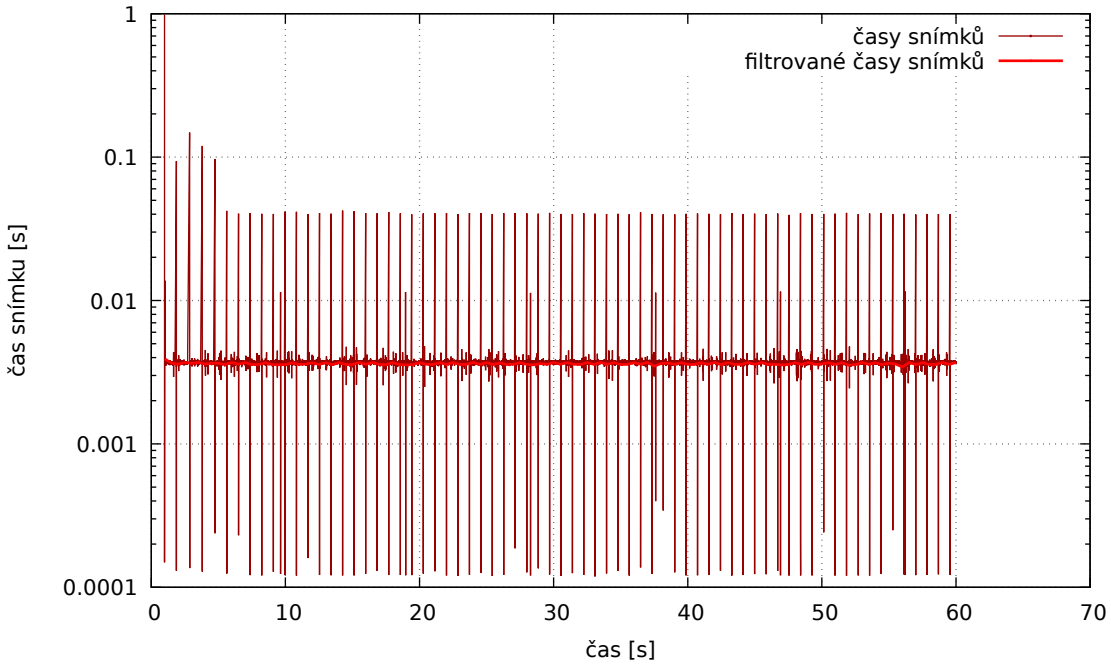
Důvod tak nízkého 99.9-percentilu je lépe vidět na obrázku 5.8, který má pro lepší názornost vertikální osu vykreslenu logaritmicky. Nízký percentil je způsoben aktualizací bufferu, která na chvíli zablokuje vykreslování. Detail na jeden blok je vidět na obrázku 5.9. V okamžiku aktualizace bufferu se právě vykreslovaný snímek zbrzdí asi o 90ms, další dva snímky se vykreslí velmi rychle. Důvod proč se tyto snímky vykreslí tak rychle se nepodařilo zjistit. Jednou z možností je, že ovladač grafické karty používá tzv. *flip queue size, max pre pendered prames* nebo *max frames to render ahead*, toto je v podstatě fronta snímků, kterou připraví procesor grafické kartě. Tato fronta sice zlepšuje plynulost, ale zvyšuje prodlevu mezi vstupem od uživatele a výstupem na monitoru. Aby tato fronta mohla fungovat, volání přepínání bufferů nemůže být synchronizováno se skutečným přepínáním bufferů. To by vysvětlovalo proč jsou dva snímky po zpožděném snímku vykresleny tak rychle. Toto zpoždění vyprázdní frontu snímků a pak se tyto dva můžou ukládat okamžitě do fronty. Podle toho, že jsou ty snímky dva, odhadoval bych, že tato fronta má délku 3 snímky.

Tento test byl pro porovnání proveden i na počítači s procesorem Intel Core i7-6700 a grafickou kartou GTX 1080 Founders Edition.

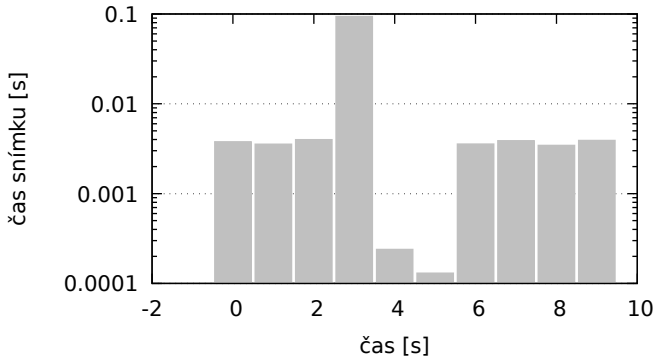
Na obrázku 5.10 je vidět, že 99,9-percentil není výrazně nižší v porovnání s referenčním PC. Obrázek 5.11 ukazuje průběh časů snímků. Kromě toho, že několik



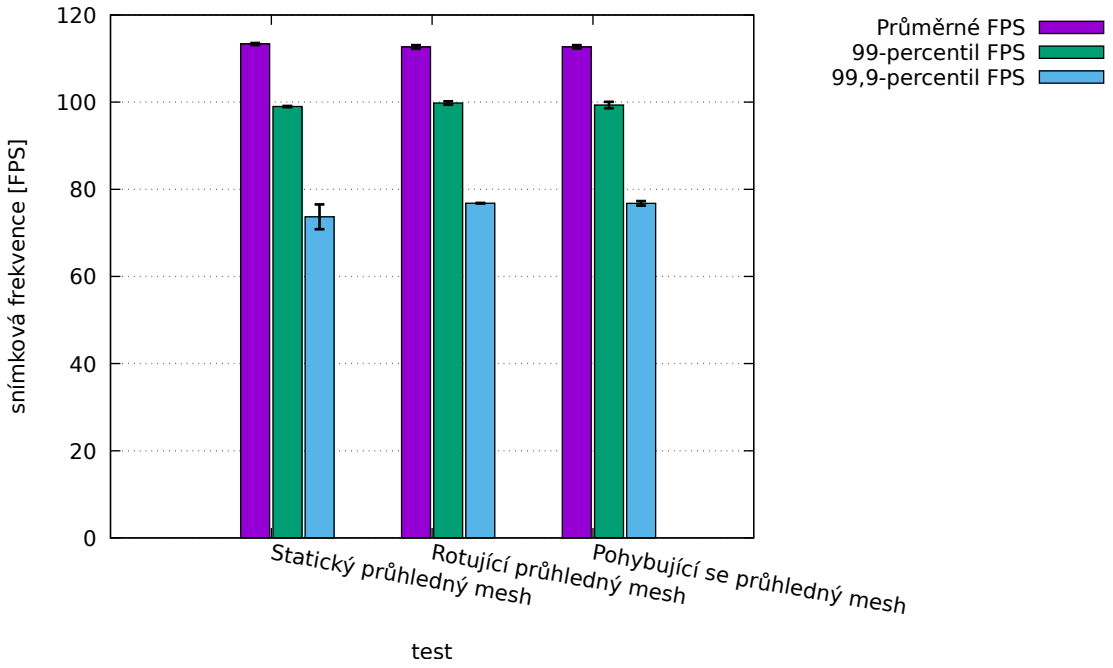
Obr. 5.7: Snímková frekvence průhledného meshe



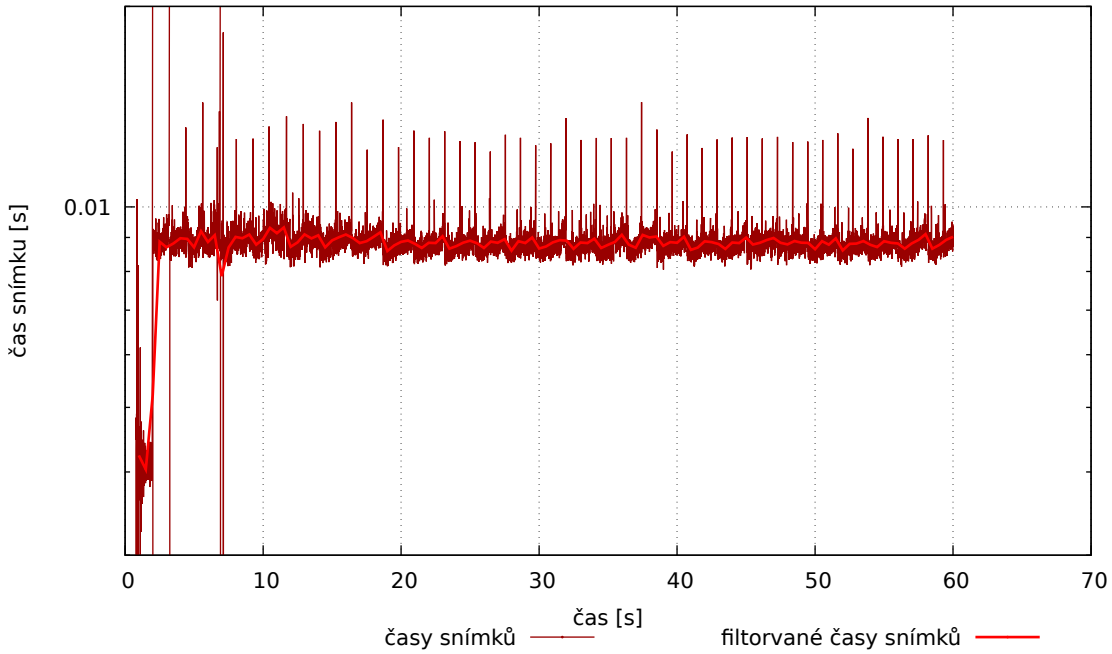
Obr. 5.8: Průběh časů snímku pro průhledný statický mesh



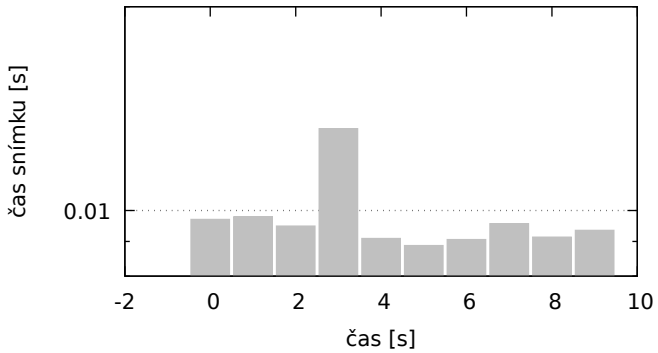
Obr. 5.9: Detail průběhu času snímku pro průhledný statický mesh



Obr. 5.10: Snímková frekvence průhledného meshe na grafické kartě GTX 1080 Founders Edition



Obr. 5.11: Průběh časů snímku pro průhledný statický mesh na grafické kartě GTX 1080 Founders Edition



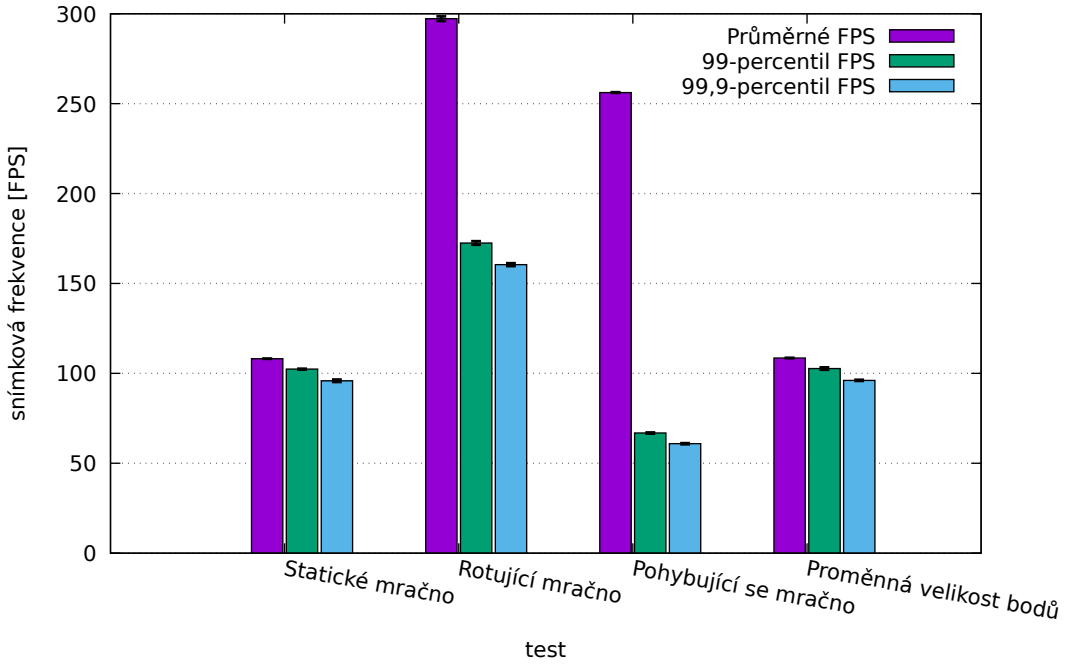
Obr. 5.12: Detail průběhu času snímku pro průhledný statický mesh na grafické kartě GTX 1080 Founders Edition

prvních sekund jsou snímky velmi rychlé, což se nepodařilo odůvodnit, obrázek ukazuje, že tyto prodlevy nejsou tak výrazné, dokonce po prodlevě nenastanou snímky s velmi krátkými časy, na což se můžeme podívat detailněji na obrázku 5.12. Toto chování více odpovídá mému očekávání, nicméně potvrdit přesnou příčinu rozdílu tohoto chování s jistotou nelze, protože při tomto měření nebyla grafická karta jediná proměnná.

Mračno bodů

Výsledný výkon pro referenční měření snímkové frekvence mračna bodů je na obrázku 5.13. Statické mračno se vykresluje průměrně 108 FPS, 99-percentil je 102 FPS a 99,9-percentil 96 FPS. Rotující mračno se vykresluje průměrně 297 FPS, 99-percentil je 172 FPS a 99,9-percentil 160 FPS. Pohybující se mračno se vykresluje průměrně 256 FPS, 99-percentil je 67 FPS a 99,9-percentil 61 FPS. Mračno měnící velikost bodů se vykresluje průměrně 109 FPS, 99-percentil je 103 FPS a 99.9-percentil 96 FPS.

Statické mračno má výbornou konzistenci snímků. Velký rozptyl snímků u rotujícího a pohybujícího se mračna je dán změnou pohledu. Mračno měnící velikost bodů se pro úzké hrdlo v geometrické části řetězce grafické karty chová totožně se statickým mračnem, v dalších testech se jím budu zabývat jen pokud by se nějak výrazně lišilo.



Obr. 5.13: Snímková frekvence pro mračno bodů

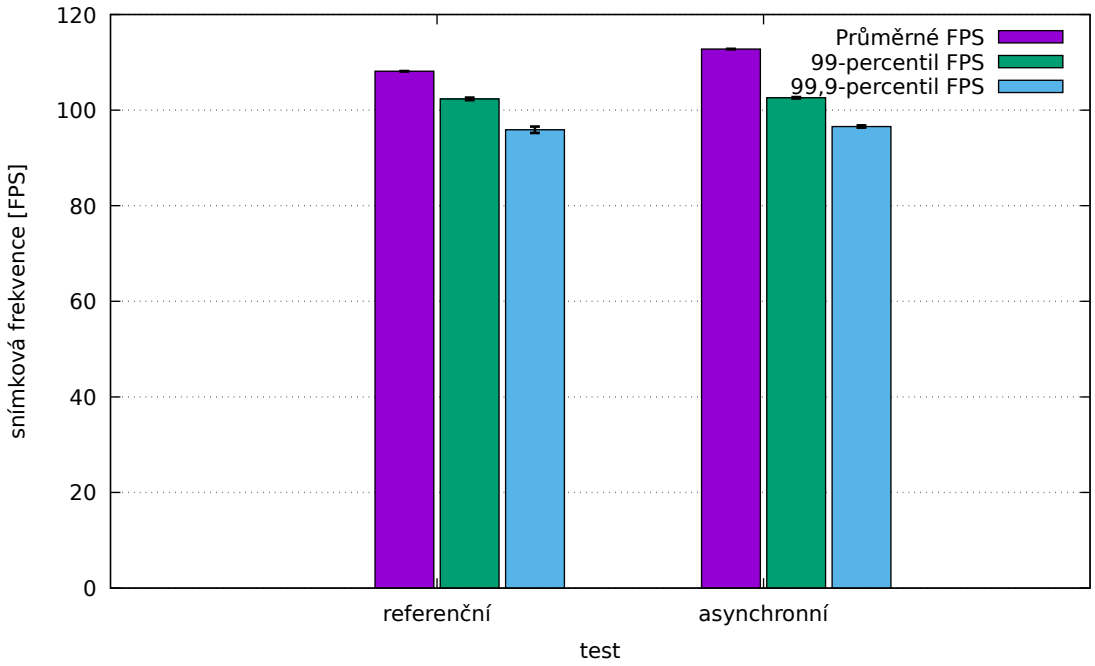
5.3.2 Asynchronní čtení transform feedback query

Zde byly změřen výkon úprav popsaných v kapitole 2.1.3 v sekci Asynchronní čtení transform feedback query.

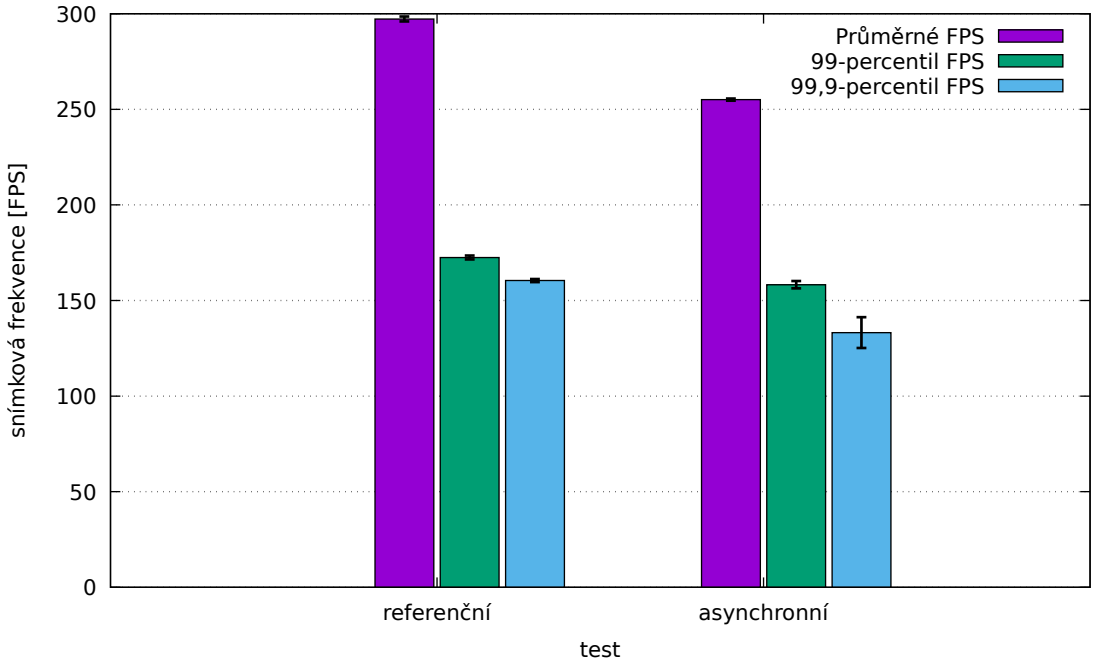
Na obrázku 5.14 je porovnání vykreslování statického mračna. Statické mračno se vykresluje průměrně 113 FPS, 99-percentil je 103 FPS a 99.9-percentil 97 FPS. To je o 4.3 %, 0.22 % a 0.71 % více než v původním případě.

Rotující mračno (viz obrázek 5.15), se vykresluje průměrně 255 FPS, 99-percentil je 158 FPS a 99.9-percentil 133 FPS. To je o 14 %, 8.2 % a 17 % méně než v původním případě. U 99,9-percentilu asynchronního čtení byla zaznamenána značná směrodatná odchylka, a to 8,1 FPS, což je 6,1 % z průměrné hodnoty

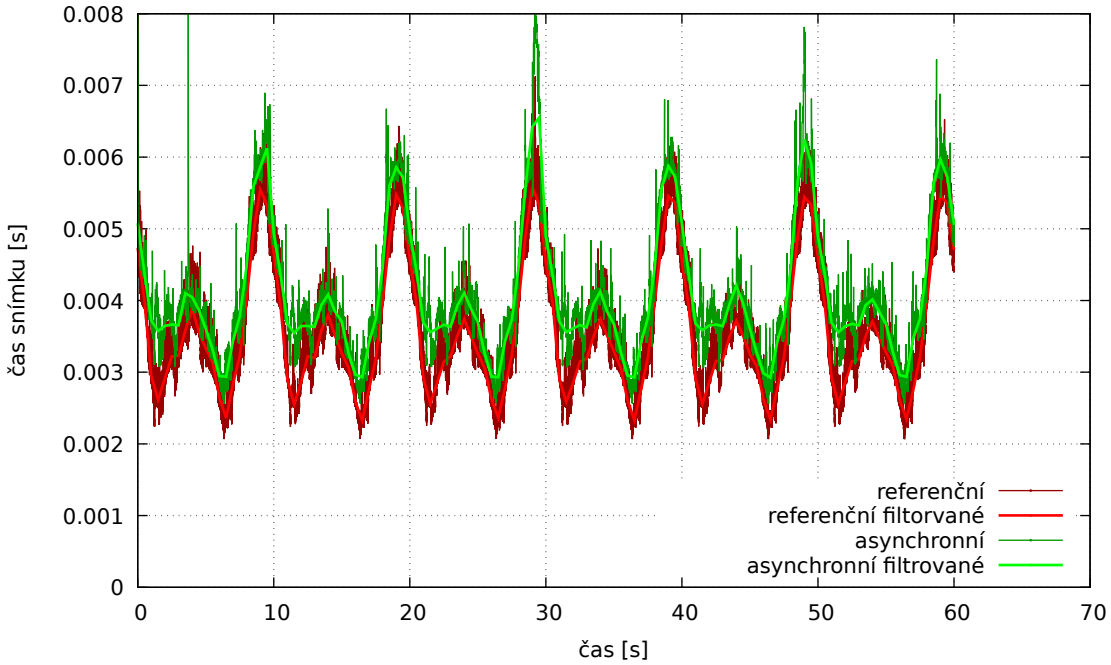
Pokles snímkové frekvence můžeme detailněji rozebrat na grafu časů snímků viz obrázek 5.16. Zde je vidět, že původní verze má většinu času kratší časy snímků. Důvod tohoto poklesu se mi nepodařilo zjistit.



Obr. 5.14: Rozdíl v FPS mezi referenční a asynchronní verzí pro statické mračno

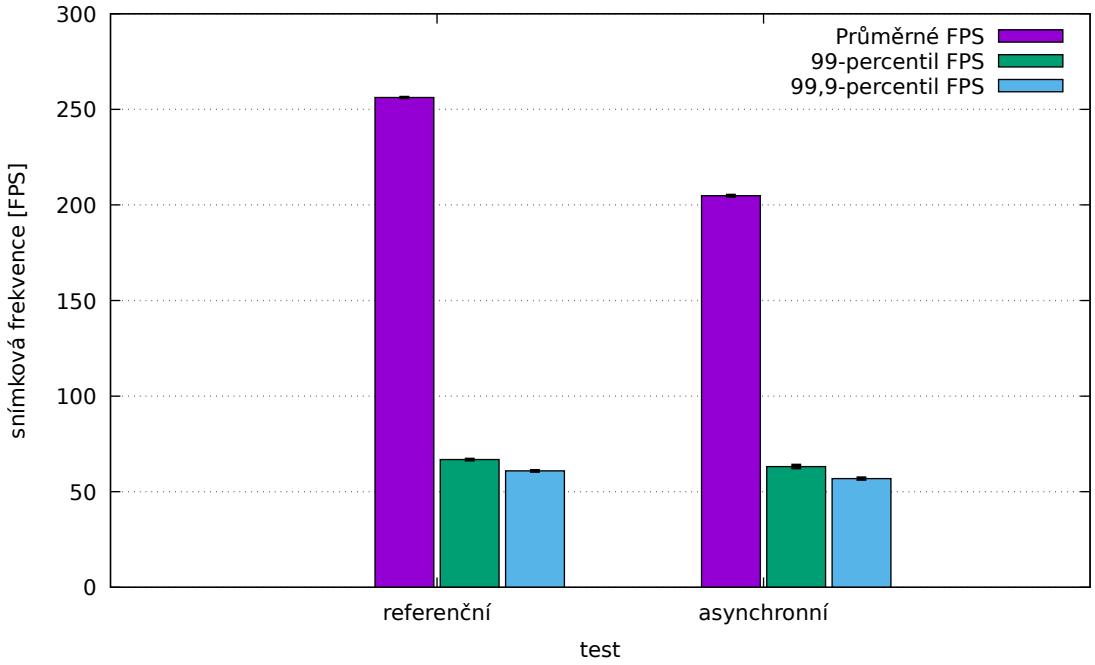


Obr. 5.15: Rozdíl v FPS mezi referenční a asynchronní verzí pro rotující mračno



Obr. 5.16: Rozdíl v časech snímků mezi referenční a asynchronní verzí pro rotující mračno

Totéž platí i pro pohybující se mračno viz obrázek 5.17. Pohybující se mračno se vykresluje průměrně 205 FPS, 99-percentil je 63 FPS a 99.9-percentil 57 FPS. To je o 20 %, 5.67 % a 6.5 % méně, než v původním případě.

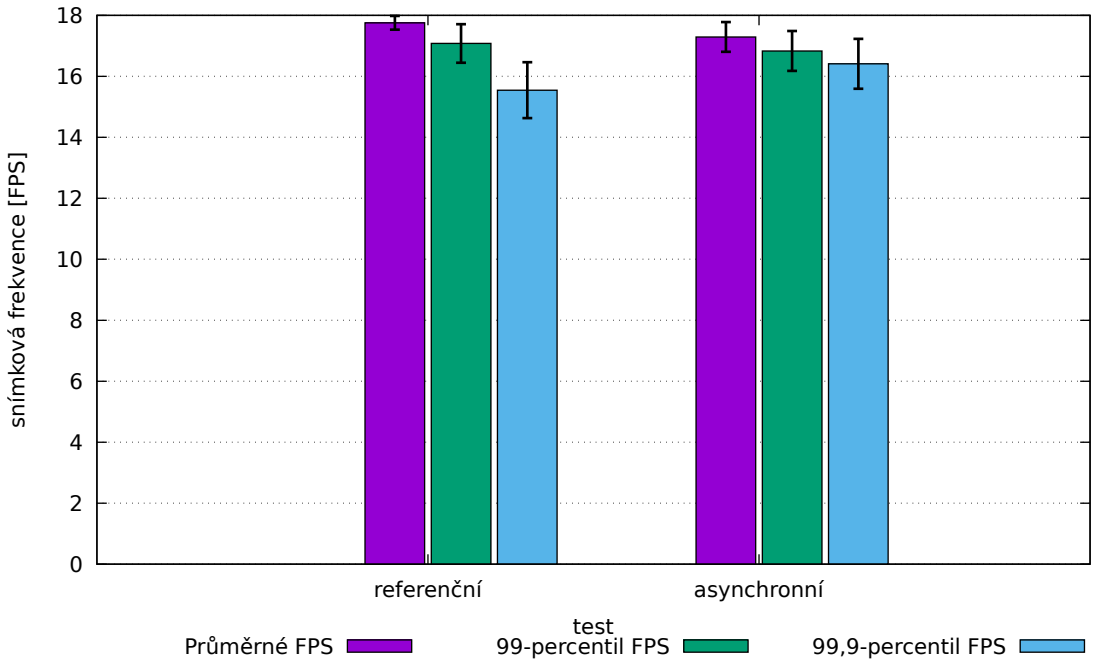


Obr. 5.17: Rozdíl v FPS mezi referenční a asynchronní verzí pro pohybující se mračno

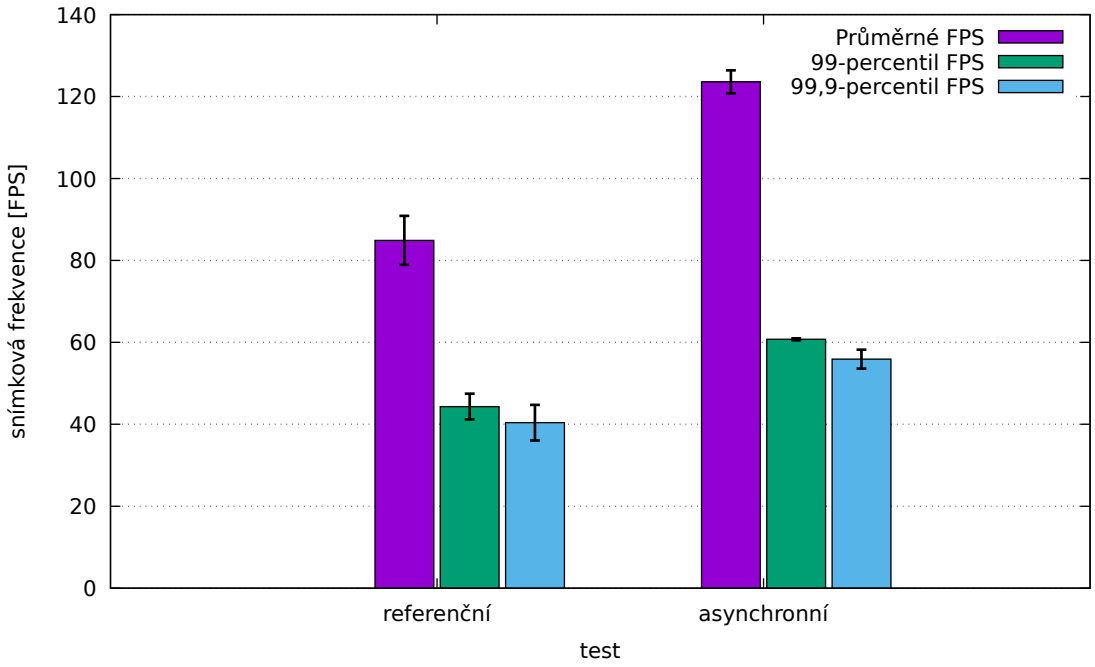
Toto vylepšení se nakonec ukázalo jako kontraproduktivní na většině hardwaru, spíše to vypadá jako by přidávalo další režii a tudíž bylo odstraněno. Zvláštní případ jsou integrované grafické karty od Intelu, ty naopak vykazují zlepšení. Celé porovnání bylo provedeno na referenčním PC s Intel 530, výsledky jsou následující.

Statické mračno (obrázek 5.18) se vykresluje průměrně 17 FPS, 99-percentil je 17 FPS a 99.9-percentil 16 FPS. *Rozdíl v rámci chyby měření?* To je o 2,6 % a 1,4 % méně a o 5,6 % více než v původním případě. Maximální směrodatná odchylka asynchronního vykreslování byla 5,0 % FPS z průměru při 99.9-percentilu. Žádný zřejmý důvod této odchylky nebyl nalezen.

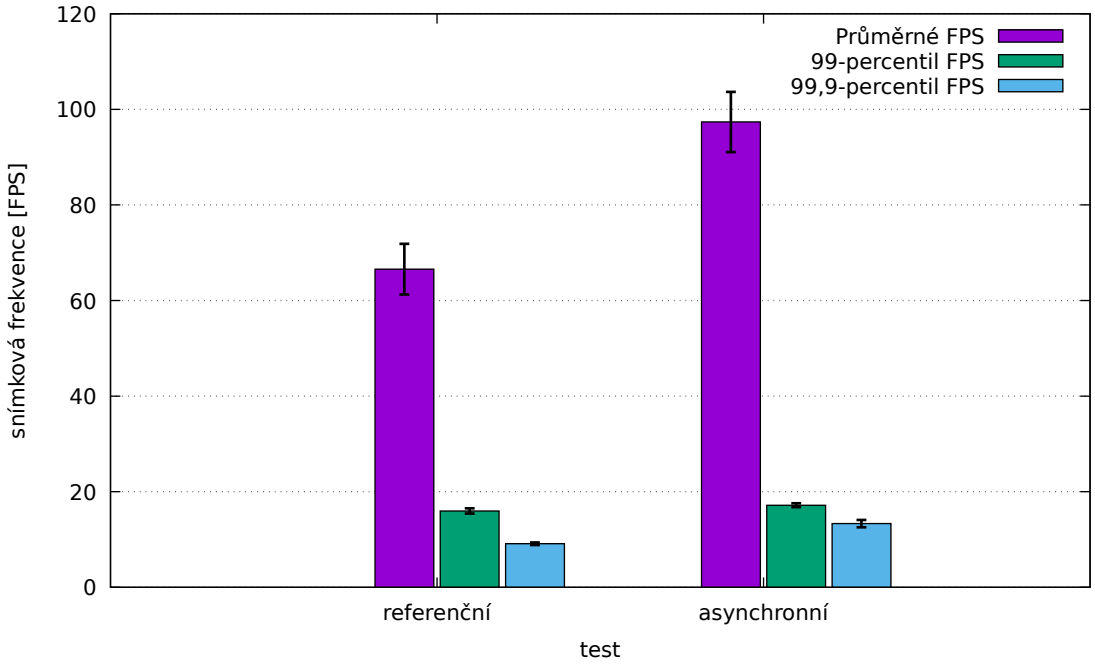
Rotující mračno (obrázek 5.19) se vykresluje průměrně 124 FPS, 99-percentil je 61 FPS a 99.9-percentil 56 FPS. To je o 46 %, 37 % a 38 % více než v původním případě. Povšimněme si více jak sedmináso bného zlepšení výkonu oproti vykreslování statického mračna na grafické kartě Intel 530. Ve stejném případě na grafické kartě RX 580 je toto zlepšení jen zhruba dvakrát vyšší. Důvod takového nárůstu není ihned zřejmý. Teorie je, že nastane situace, kdy se má vykreslovat tak malý počet bodů, že se stane úzkým hrdlem procesor. Tato snímková frekvence je v tomto případě stejná pro všechny grafické karty. Když poté nastane opačná situace, kdy se má vykreslovat naopak velmi velké množství bodů a stane se úzkým hrdlem grafická karta, je snímková frekvence závislá na výkonu této grafické karty. Pak je v případě výkonného procesoru (i7) a pomalé grafické karty (Intel 530) rozdíl těchto frekvencí



Obr. 5.18: Rozdíl v FPS mezi referenční verzí a vylepšenou verzí pro statické mračno na grafické kartě intel 530



Obr. 5.19: Rozdíl v FPS mezi referenční verzí a vylepšenou verzí pro rotující mračno na grafické kartě Intel 530



Obr. 5.20: Rozdíl v FPS mezi referenční verzí a vylepšenou verzí pro pohybující se mračno na grafické kartě intel 530

vysoký, naopak u rychlého procesoru (i7) a rychlé grafické karty (RX 580) je rozdíl těchto snímkových frekvencí nízký.

Tato teorie by se dala ověřit požitím pomalého procesoru, nicméně protože grafická karta Intel 530 se pro toto vykreslování ukázala jako nedostatečně výkonná, testování se z časových důvodů neprovedlo.

Pohybující se mračno (obrázek 5.20) je velmi podobné rotujícímu mračnu. Vykresluje se průměrně 97 FPS, 99-percentil 17 FPS a 99.9-percentil 13 FPS. To je o 46 %, 7,5 % a 46 % více než v původním případě.

Protože grafické čipy integrované v Intel procesorech nejsou dostatečně výkonné pro tuto aplikaci a na ostatních grafických kartách je tato optimalizace spíše kontraproduktivní, byla odstraněna ve prospěch následující optimalizace.

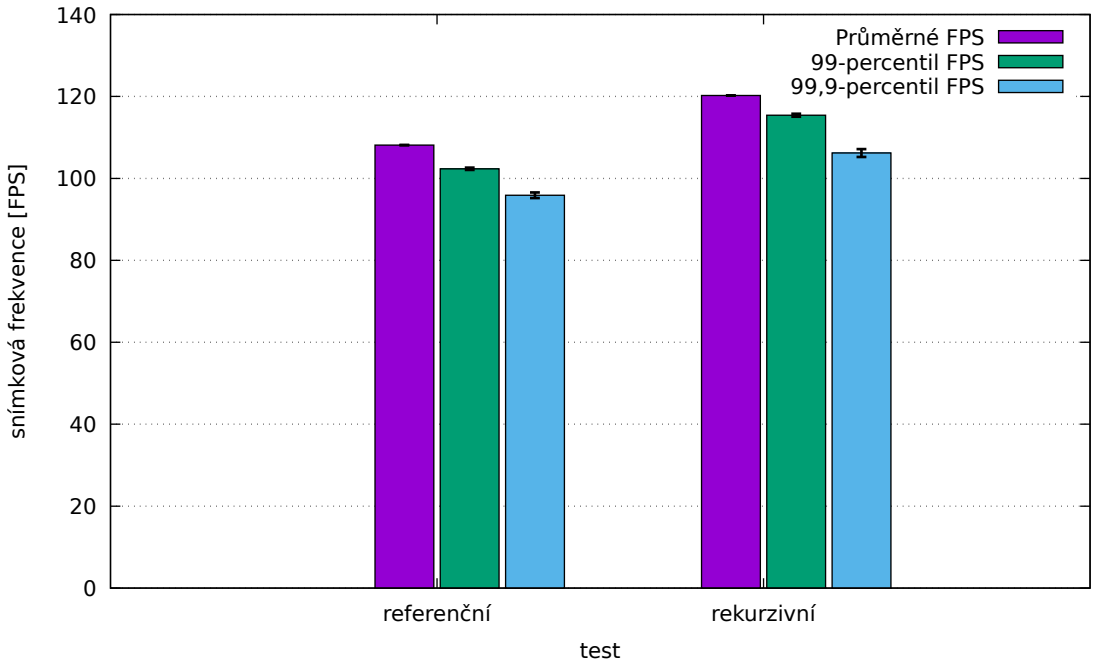
5.3.3 Rekurzivní octree

Zde byl změřen výkon úprav popsaných v kapitole 2.1.4.

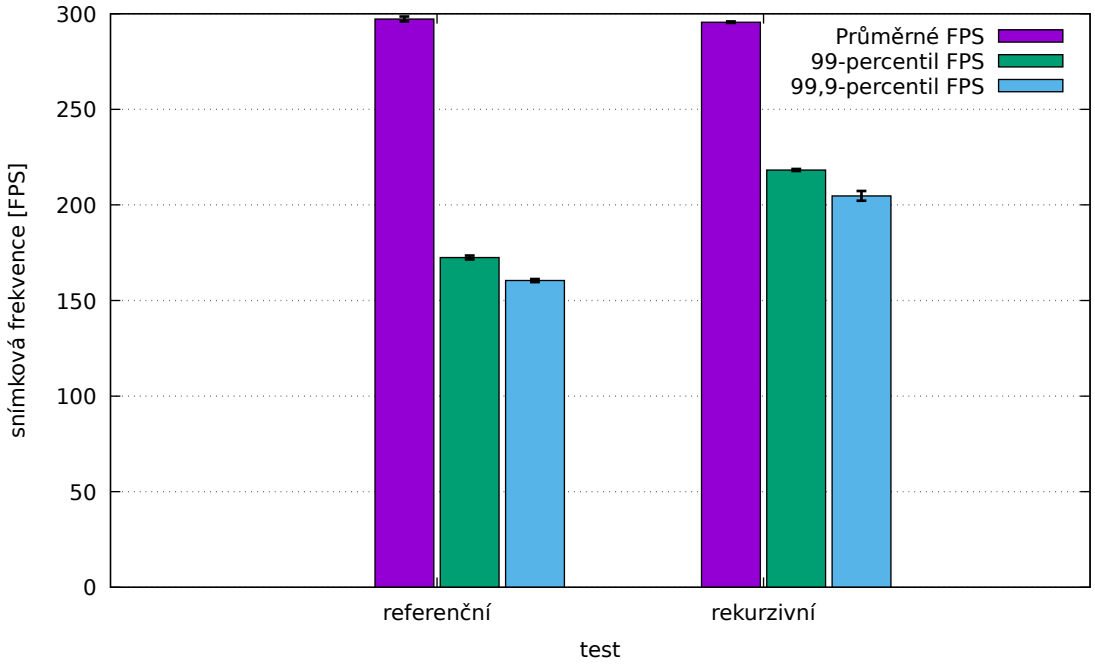
Statické mračno (obrázek 5.21) se vykresluje průměrně 120 FPS, 99-percentil je 115 FPS a 99,9-percentil 106 FPS. To je o 11 %, 13 % a 11 % více než v původním případě.

Zde se projevilo snížení vykreslovacích volání i odstranění geometrické fáze.

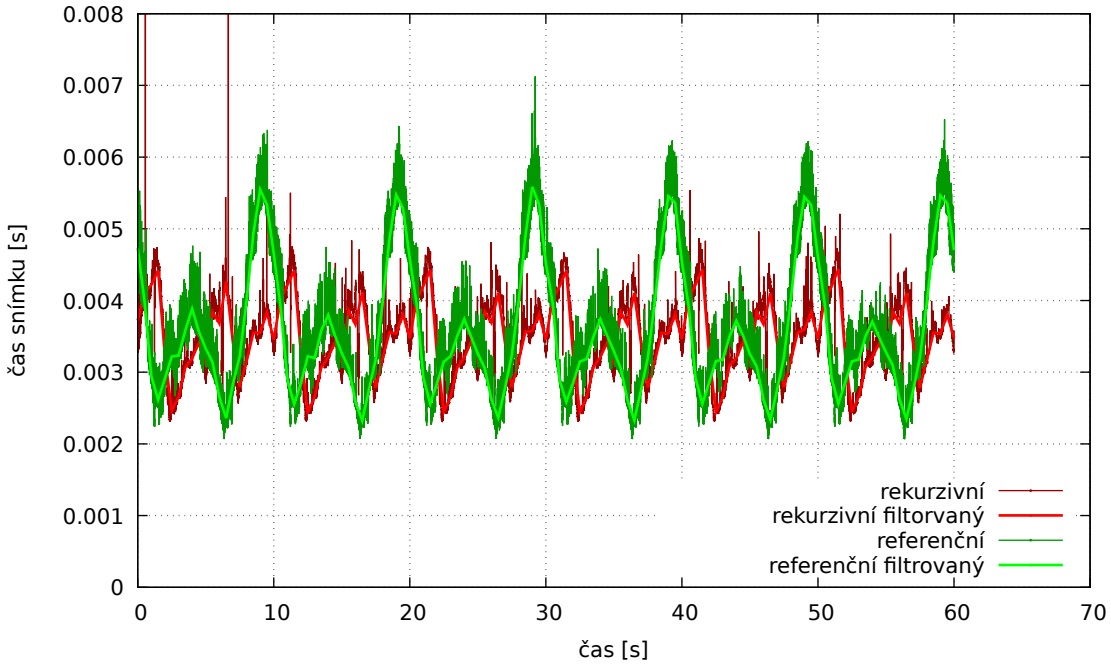
Rotující mračno (obrázek 5.22) se vykresluje průměrně 296 FPS, 99-percentil je



Obr. 5.21: Rozdíl v FPS mezi referenční verzí a vylepšenou verzí pro statické mračno



Obr. 5.22: Rozdíl v FPS mezi referenční verzí a vylepšenou verzí pro rotující mračno



Obr. 5.23: Rozdíl v časech snímků mezi referenční verzí a vylepšenou verzí pro rotující mračno

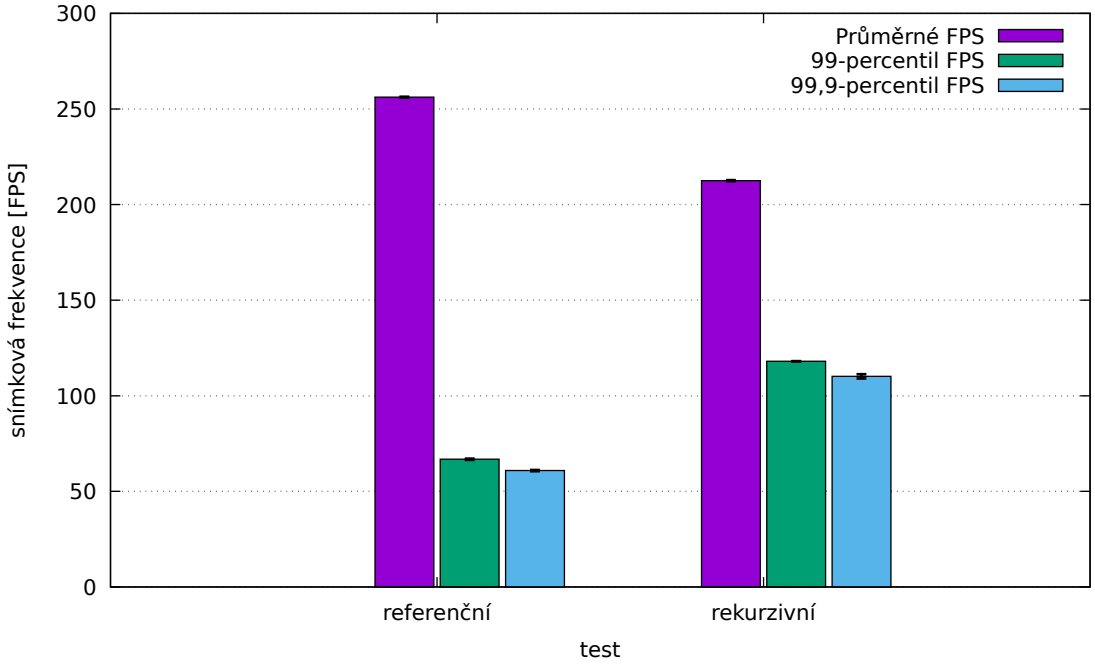
218 FPS a 99,9-percentil 205 FPS. To je o 0,56 % méně a o 27 % a 28 % více než v původním případě.

Na obrázku 5.23 je vidět rozdíl časů snímků mezi referenční verzí a použitím rekurzivního oktalového stromu při rotujícím mračně bodů. Přepracování ořezávání pohledovým objemem změnilo úplně průběh časů snímků, protože se jinak rozhoduje co se bude kreslit.

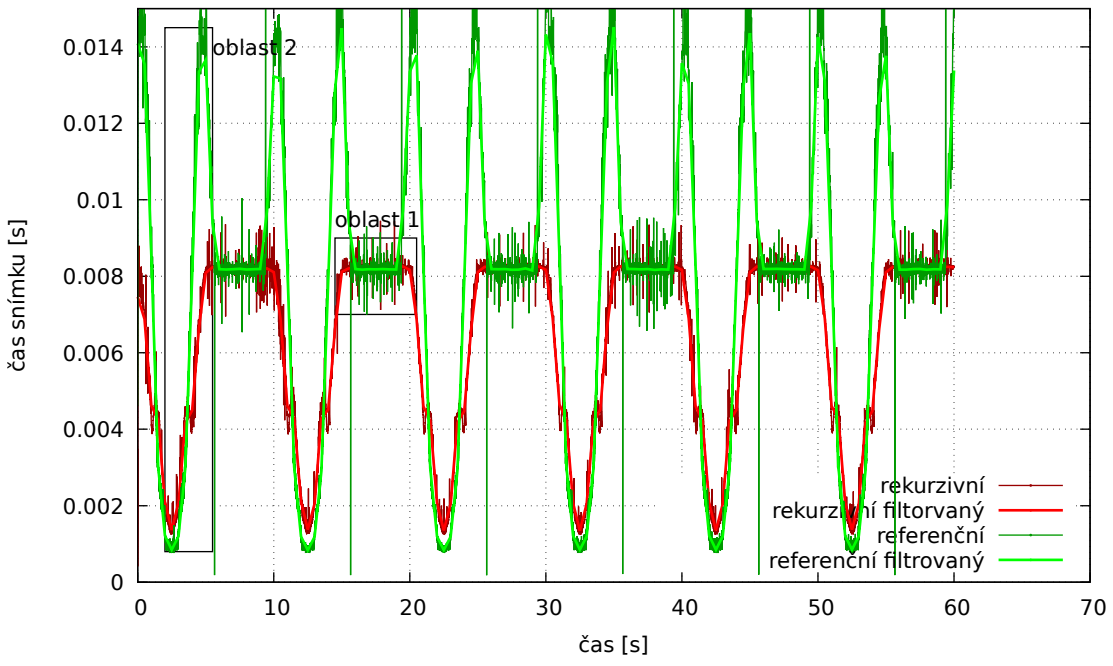
Pohybující se mračno (obrázek 5.24) se vykresluje průměrně 213 FPS, 99-percentil je 118 FPS a 99,9-percentil 110 FPS. To je o 17 % méně a o 78 % a 81 % více než v původním případě.

Na obrázku 5.25 je zobrazen rozdíl časů snímků mezi referenční verzí a použitím rekurzivního oktalového stromu při pohybujícím se mračnu bodů. V oblasti 1 je vidět, že když se mračno oddálí natolik, že je vidět celé, mají obě verze stejné časy snímků, přibližně 8 ms. Takové časy snímků má i statické mračno bodů, což znamená, že rekurzivní ořezávání pohledovým objemem nepenalizuje při vykreslování velkého množství dat. V oblasti 2 je vidět, že nová verze nemá tak výrazný nárůst časů snímků z důvodu vynechání geometrické fáze, která v tomto případě neořezala dostatečné množství bodů a byla spíše kontraproduktivní.

V těchto případech se projevilo efektivnější ořezávání pohledovým objemem i odstranění geometrické fáze.



Obr. 5.24: Rozdíl v FPS mezi referenční verzí a vylepšenou verzí pro pohybující se mračno



Obr. 5.25: Rozdíl v časech snímků mezi referenční verzí a vylepšenou verzí pro pohybující se mračno

6 Závěr

V rámci bakalářské práce byla prostudována problematika zařízení pro virtuální realitu. Bylo nutné se zorientovat na *trhu* a vybrat vhodný formát souboru, aby byl použitelný pro zadání práce. Dále bylo nutné vybrat vhodný formát souboru pro 3D modely tak, aby umožňoval uložit všechny požadované informace. Pro optimalizace vykreslování modelů bylo nutno nastudovat pokročilé techniky vykreslování. Úspěsně provedené optimalizace byly segmentace do oktalového stromu a ořezávání pohledovým objemem měřené v kapitole 5.3.3. Naopak jsem zjistil, že vykreslování průhledného meshe pomocí Malířova algoritmu s BSP stromem není ideální řešení. V další práci se budu zabývat vylepšením vykreslování průhledného meshe.

Protože OpenGL nepodporuje grafické uživatelské rozhraní, muselo být navrženo a implementováno vlastní, aby vyhovovalo přesně požadavkům zadání. Práce popisuje jak program používat, jak modely načítat a prohlížet. Zadání se podařilo splnit – program je otestován a připraven k praktickému využití, je možné prohlížet mračna bodů, meshe i průhledné meshe, model se dá natáčet i se v něm pohybovat. Při použití PC s mainstreamovou grafickou kartou ze současné generace (Nvidia Pascal nebo AMD Polaris) lze vykreslit mračno o milionu bodů zhruba v 50 snímcích za sekundu.

Výkonové srovnání ukázalo, jak obtížné je výkon aplikace měřit, protože na měření má vliv celá řada faktorů. Protože se měřilo pomocí aplikace, která běží v okně, může do měření vstupovat okenní manažer. Další velkou neznámou jsou ovladače grafických karet, ve kterých výrobci používají celou řadu nezdokumentovaných optimalizací. Další práce se bude měřením a vyhodnocováním výkonu dále zabývat.

Další možný vývoj této práce se může zaměřit na dvě oblasti. Jsou možné další výkonnostní optimalizace. Protože VR vyžaduje vykreslování dvou nezávislých pohledů dobře by se využil potenciál dvou grafických karet. Protože ale OpenGL nepodporuje přímý přístup k více grafickým kartám, program by byl přepsán do nového grafického rozhraní Vulkan[55]. Druhým směrem by bylo více spolupráce s potenciálními uživateli. Reálné požadavky od koncových uživatelů na program jako jsou typy zobrazovaných dat (mračna bodů, meshe, objemová data) jsou stěžejní, aby výsledný produkt nabízel to, co se očekává.

Vzhledem k předpokládanému využití v projektu RoScan bude pravděpodobně potřeba zobrazovat tzv. multispektrální data. To je kombinace prostorového modelu s barevnou informací a informací o teplotě a případnými dalšími přídavnými vrstvami jako je odrazivost materiálu apod. Aplikace s tímto požadavkem již počítala a umožňuje zobrazovat modely s barevnou informací a jednou přídavnou vrstvou. Podpora více vrstev bude předmětem dalšího vývoje.

Literatura

[1] STL (file format). [online], [cit. 2018.04.18].
URL https://en.wikipedia.org/wiki/STL_(file_format)

[2] Virtual reality. [online], [cit. 2018.04.18].
URL https://en.wikipedia.org/wiki/Virtual_reality

[3] why are used triangles and not quads? [online], Březen 2008, [cit. 2018.04.22].
URL https://www.opengl.org/discussion_boards/showthread.php/166204-why-are-used-triangles-and-not-quads

[4] Octree. [online], Březen 2010, [cit. 2018.04.22].
URL https://commons.wikimedia.org/wiki/File:Octree2.svg

[5] transform feedback + glDrawElementsInstanced. [online], Listopad 2012, [cit. 2018.04.29].
URL https://www.opengl.org/discussion_boards/showthread.php/178280-transform-feedback-glDrawElementsInstanced?p=1239970&viewfull=1#post1239970

[6] VFX1 Headgear. [online], Březen 2017, [cit. 2018.05.14].
URL https://en.wikipedia.org/wiki/VFX1_Headgear

[7] HTC Vive. [online], Květen 2018, [cit. 2018.05.07].
URL https://en.wikipedia.org/wiki/HTC_Vive

[8] Oculus Rift. [online], Duben 2018, [cit. 2018.05.07].
URL https://en.wikipedia.org/wiki/Oculus_Rift

[9] OpenGL. [online], Leden 2018, [cit. 2018.05.14].
URL https://cs.wikipedia.org/wiki/OpenGL

[10] OSVR-HDK issues. [online], 2018, [cit. 2018.05.08].
URL https://github.com/OSVR/OSVR-HDK/issues?q=is%3Aopen+is%3Aissue

[11] Sensorama. [online], Duben 2018, [cit. 2018.05.08].
URL https://en.wikipedia.org/wiki/Sensorama

[12] Virtuální realita. [online], duben 2018, [cit. 2018.05.05].
URL https://cs.wikipedia.org/wiki/Virtuální_realita

[13] Wavefront .obj file. [online], 2018, [cit. 2018.05.06].
URL https://en.wikipedia.org/wiki/Wavefront_.obj_file

[14] AHN, S. H.: OpenGL Projection Matrix. [online], 2008, [cit. 2018.04.22].
URL http://www.songho.ca/opengl/gl_projectionmatrix.html

[15] AHN, S. H.: OpenGL Transformation. [online], 2018, [cit. 2018.05.14].
URL http://www.songho.ca/opengl/gl_transform.html

[16] AKENINE-MÖLLER, T.; HAINES, E.; HOFFMAN, N.: *Real-Time Rendering*. Natick, Mass. USA: A K Peters, první vydání, 1999, ISBN 1-56881-101-2.
URL https://www.amazon.com/Real-Time-Rendering-Tomas-Akenine-M%C3%B6ller/dp/1568811012?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1568811012

[17] Anon.: 1961 – Headsight. [online], b.r., [cit. 2018-05-05].
URL https://www.sutori.com/item/1961-headsight-developed-by-comeau-and-bryan-the-headsight-projected-screen-f

[18] BARCZAK, J.: Why Geometry Shaders Are Slow (Unless you're Intel). [online], Březen 2015, [cit. 2018.04.28].
URL https://web.archive.org/web/20171213123908/www.joshbarczak.com/blog/?p=667

[19] BAŘINKA, M.: Programový modul pro zobrazení mračna bodů ve virtuální realitě. Semestrální práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017, 25 s. Vedoucí semestrální Práce prof. Ing. Luděk Žalud, Ph.D..

[20] BURKE, S.: Testing Methodology Explanations: 1% & 0.1% Lows, Delta T over Ambient. [online], Červenec 2016, [cit. 2018.04.30].
URL https://www.gamersnexus.net/site-news/2513-testing-methodology-explained-1percent-lows-and-delta-t

[21] ten CATE, T.: Distance field fonts. Online, Leden 2017, [cit. 2018.05.06].
URL https://github.com/libgdx/libgdx/wiki/Distance-field-fonts

[22] CHACOS, B.: SteamVR is coming to Windows Mixed Reality next week. Online, Listopad 2017, [cit. 2018.05.08].
URL https://www.pcworld.com/article/3236202/virtual-reality/steamvr-preview-windows-mixed-reality.html

[23] CHROMÝ, A.: RoScan. [online], 2018, [cit 2018.05.07].
URL http://roscan.ceitec.cz

[24] DAWES, B.; ABRAHAMS, D.; RIVERA, R.: Welcome to Boost.org! [online], 2007, [cit. 2018.05.14].
URL https://www.boost.org

[25] DIAKOPOULOS, D.: C++11 ply 3d mesh format importer & exporter. [online], [cit. 2018.04.18].
URL https://github.com/ddiakopoulos/tinyply

[26] DINGMAN, H.: Don't be fooled: Windows Mixed Reality headsets are just VR headsets. [online], Říjen 2017, [cit. 2018.05.12].
URL https://www.pcworld.com/article/3233247/virtual-reality/dont-be-fooled-windows-mixed-reality-headsets-are-just-vr-headsets.html

[27] ERICSON, C.: *Real-Time Collision Detection*. The Morgan Kaufmann Series in Interactive 3-D Technology, 500 Sansome Street, Suite 400, San Francisco, CA 94111: CRC Press, 2004, ISBN 1558607323.
URL https://www.amazon.com/Real-Time-Collision-Detection-Interactive-Technology/dp/1558607323?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1558607323

[28] HILLER, J. D.; LIPSON, H.: STL 2.0: A PROPOSAL FOR A UNIVERSAL MULTI-MATERIAL ADDITIVE MANUFACTURING FILE FORMAT. Technická zpráva, Mechanical and Aerospace Engineering, Cornell University, Ithaca NY 14853, USA, 2009.
URL https://sffsymposium.engr.utexas.edu/Manuscripts/2009/2009-23-Hiller.pdf

[29] HTC Corporation: Oficiální stránka HTC Vive. [online], 2018, [cit. 2018.05.07].
URL https://www.vive.com/us/product/vive-virtual-reality-system/

[30] kbranigan: Simple-OpenGL-Image-Library. [online], 2017.
URL https://github.com/kbranigan/Simple-OpenGL-Image-Library

[31] LARABEL, M.: A Chat With Khronos President Neil Trevett About Vulkan, OpenXR, SPIR-V In 2017. [online], Březen 2017, [cit. 2018.05.07].
URL https://www.phoronix.com/scan.php?page=news_item&px=Khronos-Post-GDC-2017

[32] Linus Media Group: What is MIXED Reality? – Acer Windows MR Headset. [online], Listopad 2017, [cit. 2018.05.08].
URL https://www.youtube.com/watch?v=VCqfs6yg2XU&t=485s

[33] Microsoft: Immerse yourself in a new reality. [online], 2018, [cit. 2018.05.08].
URL https://www.microsoft.com/en-us/windows/windows-mixed-reality

[34] Open Source Virtual Reality: WHAT IS OSVR? [online], 2016, [cit. 2018.05.06].
URL http://www.osvr.org/what-is-osvr.html

[35] OpenGL Wiki: Conditional Rendering. [online], [cit. 2018.04.28].
URL https://www.khronos.org/opengl/wiki/Vertex_Rendering#Conditional_rendering

[36] OpenGL Wiki: File:Triple overlap.png. [online], [cit. 2018.05.01].
URL https://www.khronos.org/opengl/wiki/File:Triple_overlap.png

[37] OpenGL Wiki: Geometry Shader. [online], [cit. 2018.05.01].
URL https://www.khronos.org/opengl/wiki/Geometry_Shader

[38] OpenGL Wiki: Indirect rendering. [online], [cit. 2018.04.29].
URL https://www.khronos.org/opengl/wiki/Vertex_Rendering#Indirect_rendering

[39] OpenGL Wiki: Occlusion queries. [online], [cit. 2018.04.28].
URL https://www.khronos.org/opengl/wiki/Query_Object#Occlusion_queries

[40] OpenGL Wiki: Post Transform Cache. [online], [cit. 2018.04.18].
URL https://www.khronos.org/opengl/wiki/Post_Transform_Cache

[41] OpenGL Wiki: Query buffer object. [online], [cit. 2018.05.01].
URL https://www.khronos.org/opengl/wiki/Query_Object#Query_buffer_object

[42] OpenGL Wiki: Query Object. [online], [cit. 2018.04.29].
URL https://www.khronos.org/opengl/wiki/Query_Object

[43] OpenGL Wiki: Transform Feedback. [online], [cit. 2018.05.01].
URL https://www.khronos.org/opengl/wiki/Transform_Feedback

[44] OpenGL Wiki: Performance. Online, 2015, [cit. 2018.04.30].
URL https://www.khronos.org/opengl/wiki/Performance

[45] OpenGL Wiki: Instancing. [online], 2017, [cit. 2018.05.14].
URL https://www.khronos.org/opengl/wiki/Vertex_Rendering#Instancing

[46] Rautenberg, M.: History of HCI. Prezentace, Technical University Eindhoven, [2002], [cit. 2018.05.05].
URL http://www.idemployee.id.tue.nl/g.w.m.rauterberg/presentations/hci-history/sld062.htm

[47] Razer: OSVR-HDK Readme. [online], 2018, [cit. 2018.05.08].
URL https://github.com/OSVR/OSVR-HDK#readme

[48] Riccio, C.: OpenGL Pipeline Map. [online], 2014, [cit. 2018.05.12].
URL https://openglinsights.com/pipeline.html

[49] Rosenberg, L. B.: Virtual fixtures: Perceptual tools for telerobotic manipulation. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, Září 1993, s. 76–82, doi:10.1109/VRAIS.1993.380795.

[50] simpublica staff: The Sword of Damocles and the birth of virtual reality. [online], Březen 2014, [cit. 2018.04.18].
URL http://simpublica.com/2014/03/19/the-sword-of-damocles-and-the-birth-of-virtual-reality/

[51] SUTHERLAND, I. E.: A head-mounted three dimensional display. Technická zpráva, The University of Utah, 1968.
URL http://141.84.8.93/lehre/ss09/ar/p757-sutherland.pdf

[52] The Khronos™Group Inc.: The OpenXR Working Group is Here! [online], Únor 2017, GDC, [cit. 2018.05.07].
URL https://www.khronos.org/blog/the-openxr-working-group-is-here

[53] The Khronos™Group Inc.: OpenGL Overview. [online], 2018, [cit. 2018.05.14].
URL https://www.khronos.org/opengl/

[54] The Khronos™Group Inc.: OpenXR Overview. [online], 2018, [cit. 2018.05.07].
URL https://www.khronos.org/openxr

[55] The Khronos™Group Inc.: Vulkan Overview. [online], 2018, [cit. 2018.05.15].
URL https://www.khronos.org/vulkan/

[56] Turk, G.: The PLY Polygon File Format. Technická zpráva, Standford University, 1994.
URL https://web.archive.org/web/20170502135307/http://www.dcs.ed.ac.uk:80/teaching/cs4/www/graphics/Web/ply.html

[57] Virtual Reality Society: Applications Of Virtual Reality. [online], 2017, [cit, 2018.05.08].
URL https://www.vrs.org.uk/virtual-reality-applications/

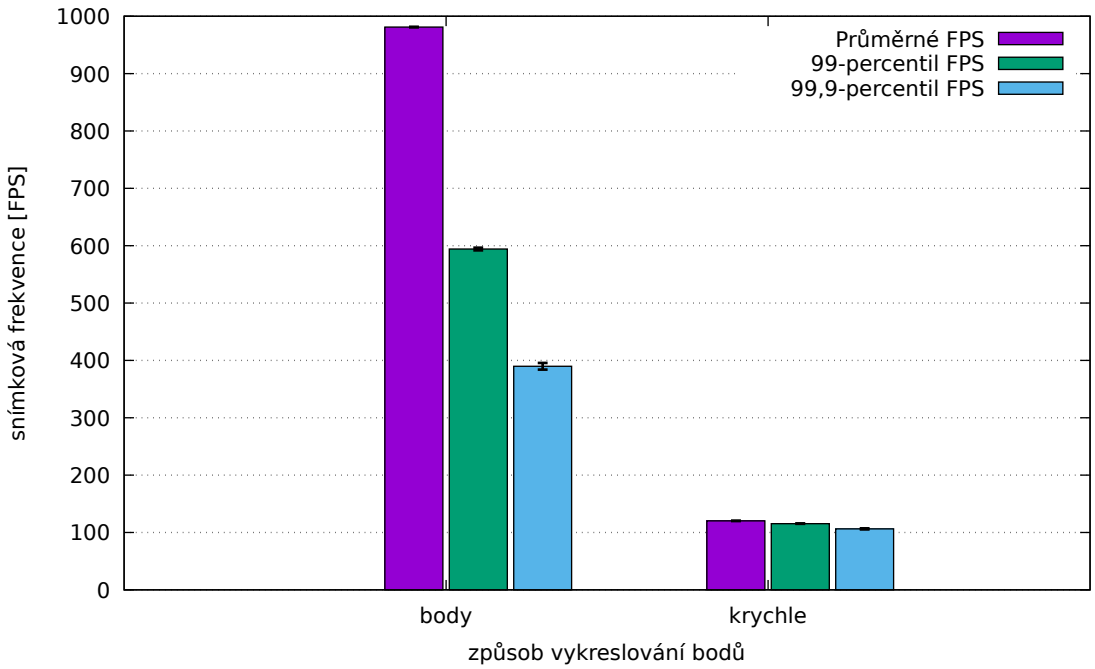
[58] Virtual Reality Society: History Of Virtual Reality. [online], 2017, [cit. 2018.05.05].
URL https://www.vrs.org.uk/virtual-reality/history.html

[59] WASSON, S.: Inside the second: A new look at game benchmarking. [online], Září 2011, [cit 2018.04.30].
URL https://techreport.com/review/21516/inside-the-second-a-new-look-at-game-benchmarking

[60] ŽÁRA, J.; BENEŠ, B.; SOCHOR, J.; aj.: *Moderní počítačová grafika*, ročník 2., přeprac. a rozš. vyd. Brno: Computer Press, 2004, ISBN 80-251-0454-0.

Seznam příloh

A Porovnání Snímkových frekvencí kreslení bodů a krychlí

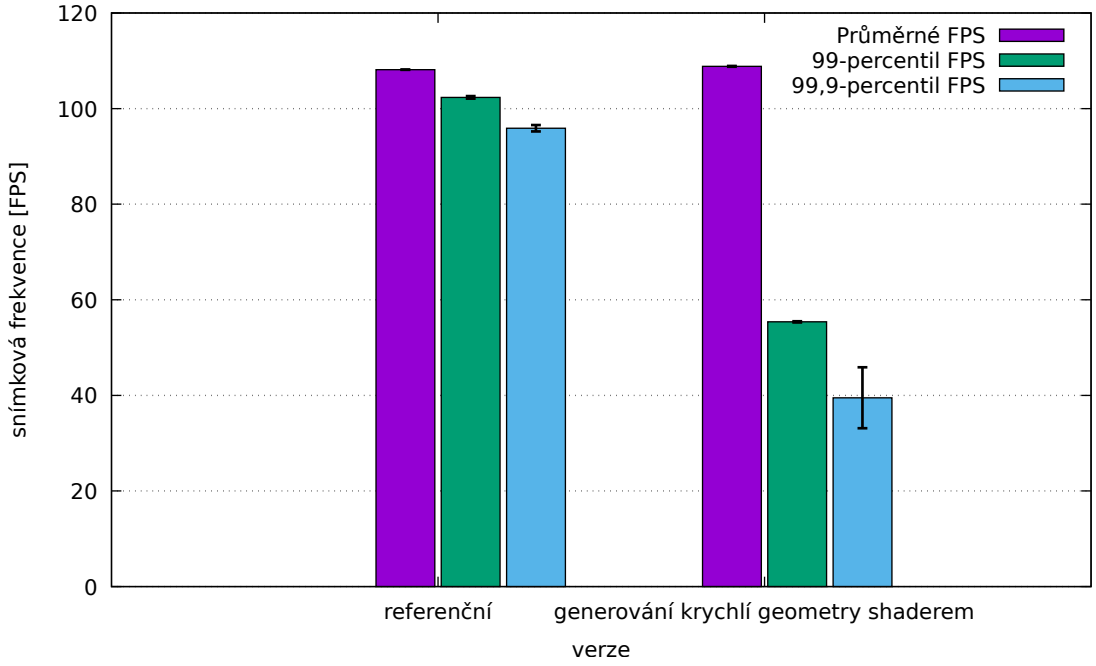


Obr. A.1: Statické mračno

Testovací metodika byla popsána v kapitole 5. Krychle jsou 8 krát geometricky náročnější než body, tudíž je předpokládaný osminásobný propad výkonu přechodem od bodů ke krychlím. Body se kreslí průměrně 981 FPS, 99-percentil je 594 FPS a 99,9-percentil 390 FPS. Zato krychle se kreslí průměrně 120 FPS, 99-percentil je 115 FPS a 99,9-percentil 106 FPS. Průměrná snímková frekvence klesla 8,2 krát, což odpovídá rozdílu v geometrické náročnosti.

B Generování krychlí pomocí geometry shaderu

Testovací metodika byla popsána v kapitole 5.



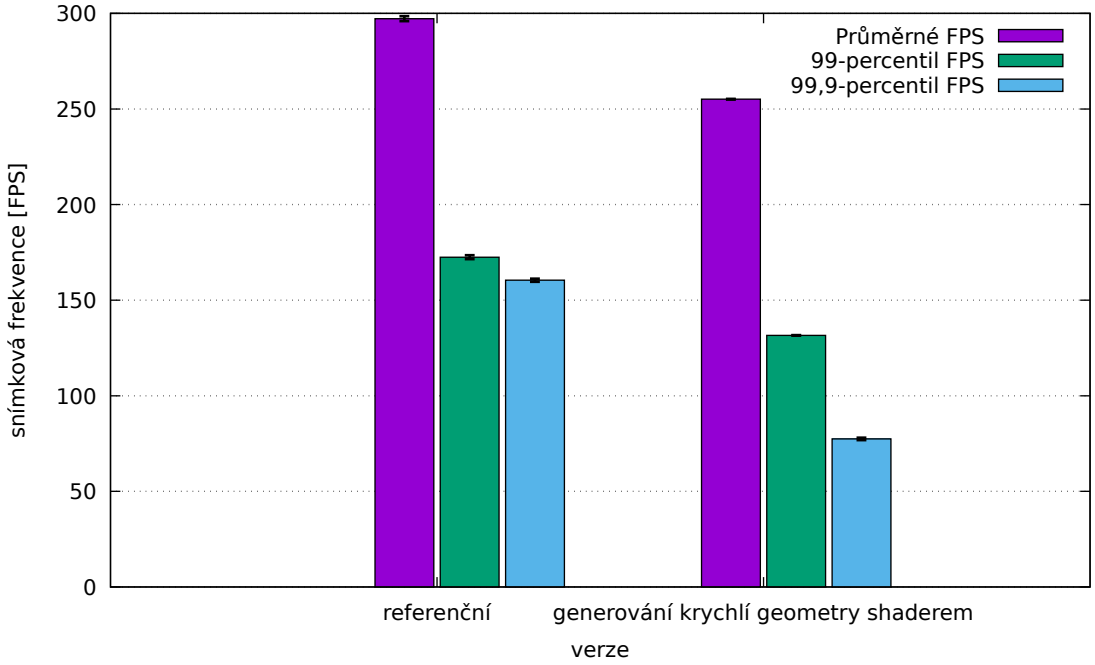
Obr. B.1: Rozdíl snímkové frekvence mezi referenční verzí a verzí s generováním krychlí v geometry shaderu pro statické mračno

Na obrázku B.1 je vidět rozdíl mezi referenční verzí a verzí s generováním krychlí v geometry shaderu pro statické mračno. Nová verze se kreslí v průměru 108 FPS, 99-percentil 55 FPS a 99.9-percentil 39 FPS. To je o 0.6 % více a o 46 % a 59 % méně než v původním případě.

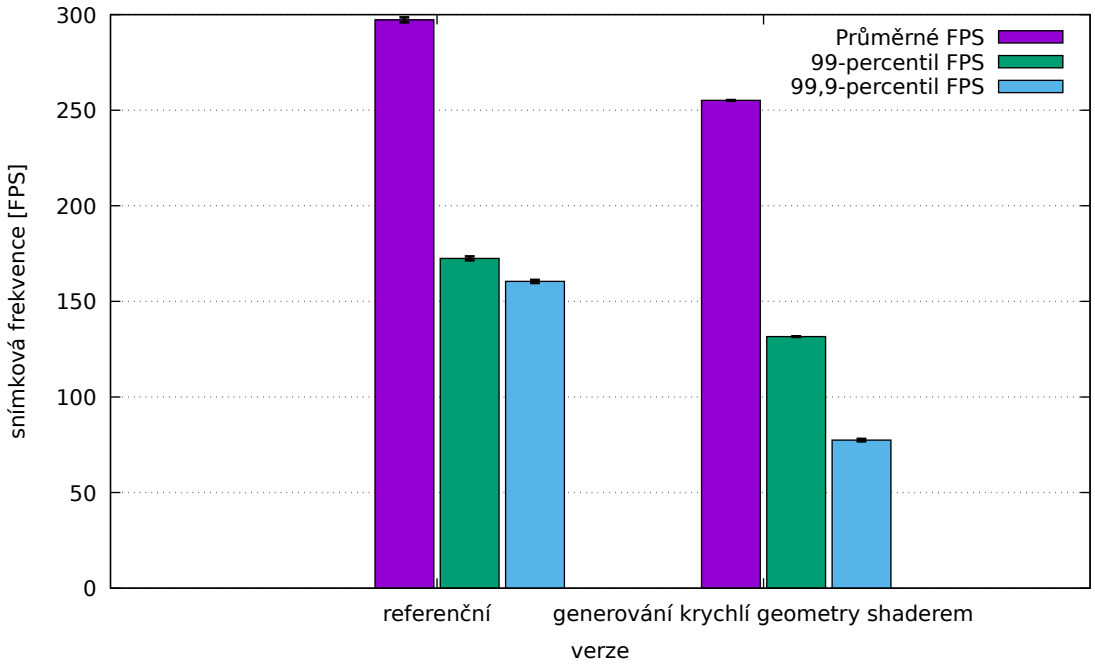
Na obrázku B.2 je vidět rozdíl mezi referenční verzí a verzí s generováním krychlí v geometry shaderu pro rotující mračno. Nová verze se kreslí v průměru 255 FPS, 99-percentil 132 FPS a 99.9-percentil 77 FPS. To je o 14 %, 24 % a 52 % méně než v původním případě.

Na obrázku B.3 je vidět rozdíl mezi referenční verzí a verzí s generováním krychlí v geometry shaderu pro pohybující se mračno. Nová verze se kreslí v průměru 205 FPS, 99-percentil 63 FPS a 99.9-percentil 57 FPS. To je o 20 %, 5.5 % a 6.6 % méně než v původním případě.

Je vidět že ve všech případech je generování krychlí pomocí geometry shaderu horší než referenční způsob.



Obr. B.2: Rozdíl snímkové frekvence mezi referenční verzí a verzí s generováním krychlí v geometry shaderu pro rotující mračno



Obr. B.3: Rozdíl snímkové frekvence mezi referenční verzí a verzí s generováním krychlí v geometry shaderu pro pohybující se mračno

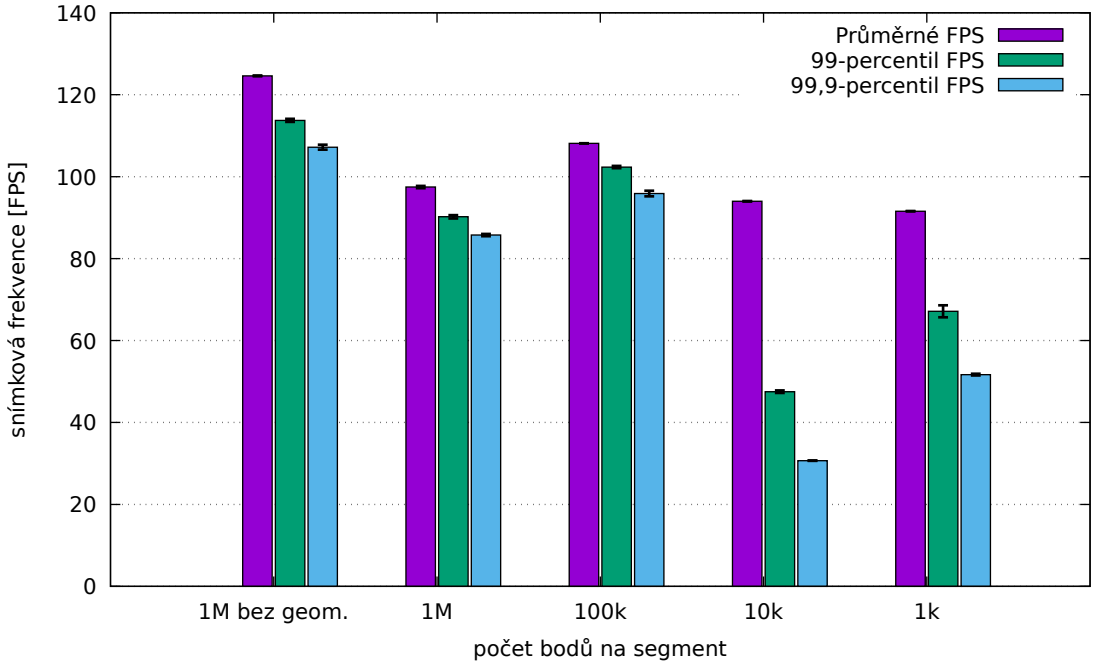
C Porovnání velikostí originálních segmentů

Bylo třeba zvolit vhodnou velikost segmentu. Pro příliš malé velikosti segmentů se bude negativně projevovat nejen ořezávání pohledovým objemem, ale i režie ovladače grafické karty. Naopak pro příliš velký segment bude zase ořezávání ne příliš efektivní, protože se bude testovat velká oblast najednou a budou se zbytečně vykreslovat body mimo pohled.

Velikost segmentu byla zvolena experimentálně. Jako testovací objekt bylo zvoleno mračno, které je použito pro výkonové srovnání v kapitole 5. Tabulka C.1 ukazuje, na kolik segmentů se toto mračno rozdělí pro různé velikosti segmentů.

Velikost segmentu	Počet segmentů
1 000 000	8
100 000	43
10 000	581
1 000	5474

Tab. C.1: Závislost počtu segmentů na velikosti segmentu



Obr. C.1: Závislost snímkové frekvence na velikosti segmentů pro statické mračno

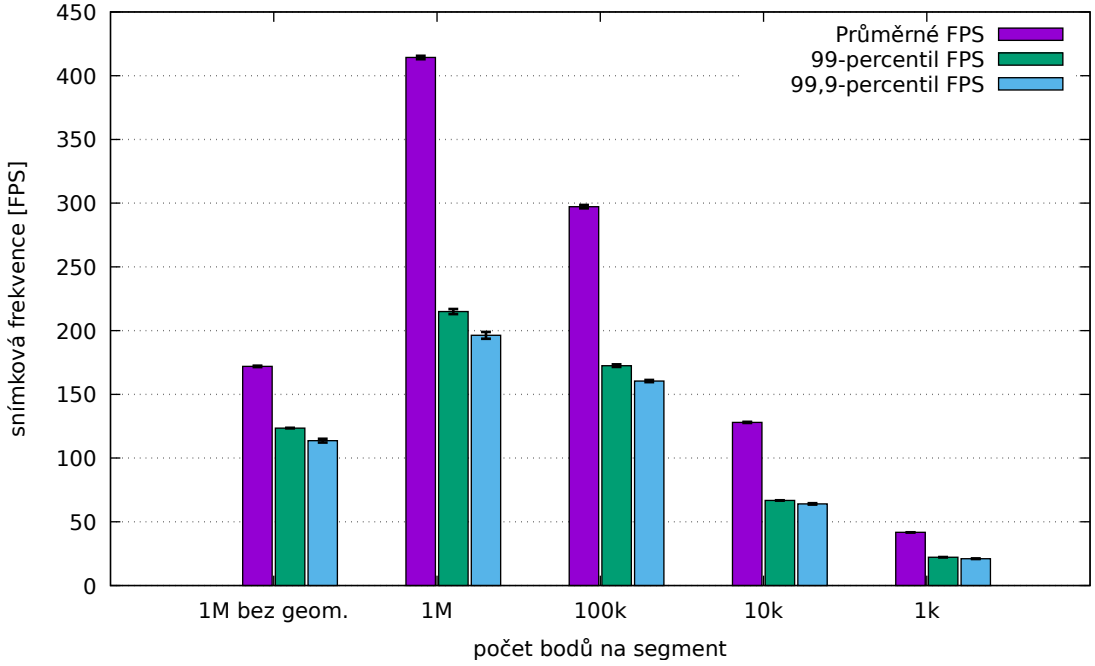
Testovací metodika je popsána v kapitole 5. Na obrázku C.1 je vidět závislost snímkové frekvence na velikosti segmentů pro statické mračno. Segmenty o velikosti

jednoho milionu bodů se kreslí v průměru 97 FPS, 99-percentil 90 FPS a 99.9-percentil 86 FPS. Segmenty o sto tisících bodů se kreslí v průměru 108 FPS, 99-percentil 102 FPS a 99.9-percentil 96 FPS. Segmenty o deseti tisících bodů se kreslí v průměru 94 FPS, 99-percentil 48 FPS a 99.9-percentil 31 FPS. Segmenty o jednom tisíci bodů se kreslí v průměru 92 FPS, 99-percentil 67 FPS a 99.9-percentil 52 FPS.

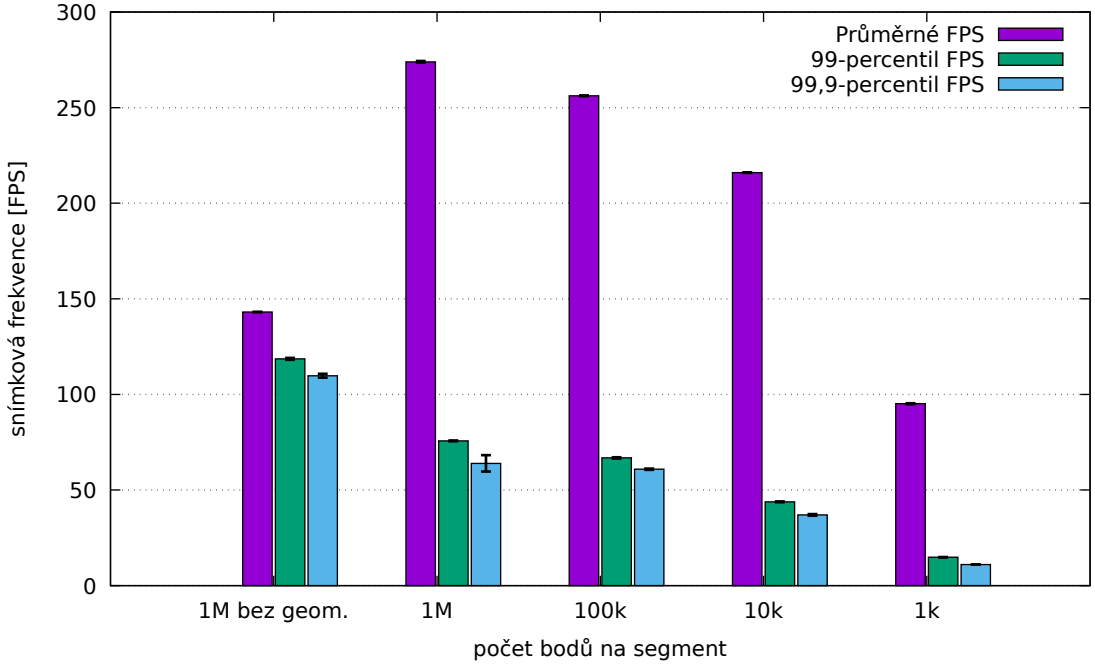
V případě statického mračna, kde se většina segmentů nachází kompletně v pohledu, má ořezávání pohledovým objemem velmi malou režii, protože nemusí procházet všechny testy. Pro segmenty s 10 000 a méně body již dochází k projevování režií hlavně ze strany ovladače grafické karty. Důvod, proč má mračno s milionovými segmenty nižší snímkovou frekvenci než mračno se segmenty stotisícovými je, že ořezávání pohledovým objemem označí menší podíl segmentů jako plně v pohledu, a proto se zbytečně velké množství bodů kreslí přes geometrickou fázi, která v tomto případě nevyřadí dostatečné množství bodů a je kontraproduktivní. Toto bylo ověřeno úpravou programu. V případě segmentu, který byl částečně v pohledu se přeskakovala geometrická fáze. V tomto případě se segmenty o velikosti jednoho milionu bodů kreslily v průměru 125 FPS, 99-percentil 114 FPS a 99.9-percentil 107 FPS.

Na obrázku C.2 je vidět závislost snímkové frekvence na velikosti segmentů pro rotující mračno. Pro pohybující se mračno viz obrázek C.3 s detailním průběhem časů snímků na obrázku C.4. Tyto průběhy závislosti odporovaly očekávání, nicméně jsou vysvětlitelné. Na průběhy se segmenty s 10 000 a 1 000 body se negativně projevila režie nejen ořezávání pohledovým objemem, protože než se prohlásí, že je segment mimo pohled, musí se vykonat všechny testy, ale také režie ovladače. Průběh se segmenty s 1 000 000 body bez geometrické fáze byl výrazně pomalejší než průběhy s 1 000 000 a 100 000 body, protože zakázáním této fáze se znemožnila optimalizace segmentů, které jsou jen z části vidět, což by se v tomto případě projevilo pozitivně. To, že segmenty s 1 000 000 bodů a povolenou geometrickou fází mají nejvyšší snímkovou frekvenci nedokážu vysvětlit.

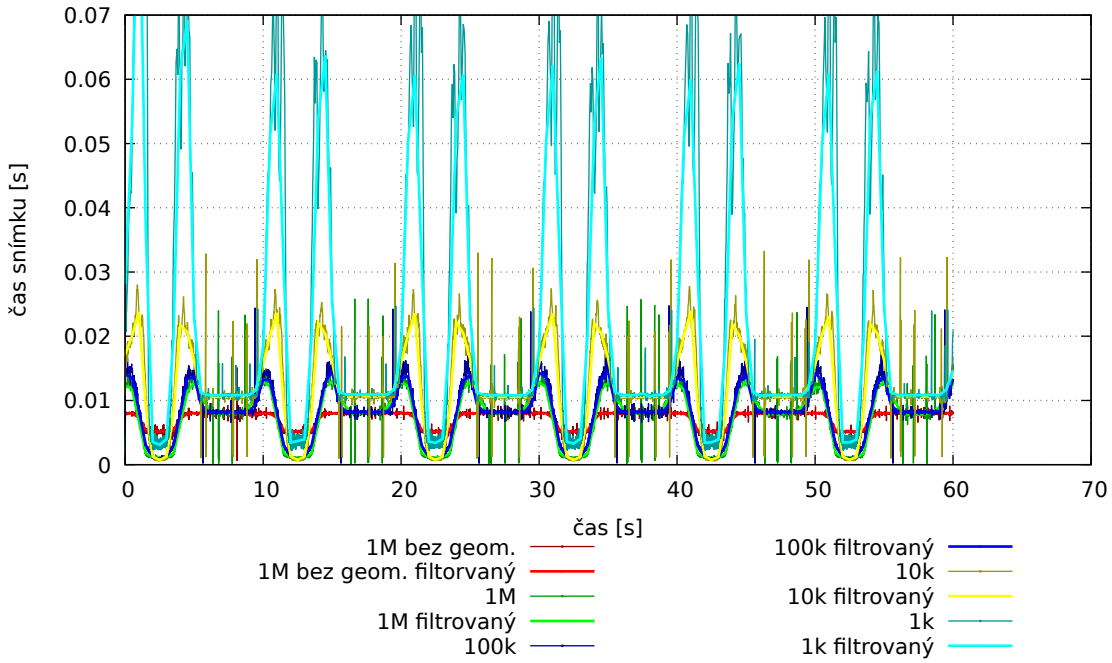
Z těchto dat jsem zvolil jako vhodnou velikost segmentů 100 000 bodů, protože výsledky byly celkově vyvážené.



Obr. C.2: Závislost snímkové frekvence na velikosti segmentů pro rotující mračno



Obr. C.3: Závislost snímkové frekvence na velikosti segmentů pro pohybující se mračno

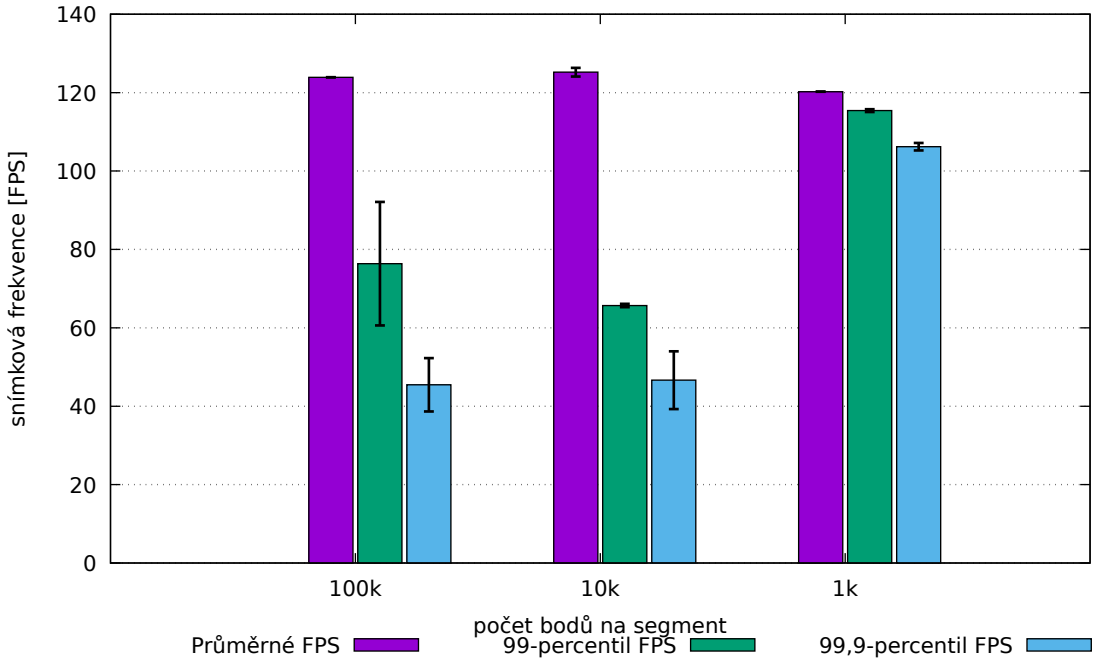


Obr. C.4: Časy snímků pro pohybující se mračno

D Porovnání velikostí nových segmentů

Bylo třeba zvolit vhodnou velikost segmentů ze stejného důvodu jako v příloze C.

Velikost segmentu byla zvolena experimentálně. Protože se ale nyní ořezává pohledovým objemem rekurzivně, může se velikost segmentů výrazně zmenšit. Jako testovací objekt bylo zvoleno mračno, které je používáno pro výkonové srovnání v kapitole 5.

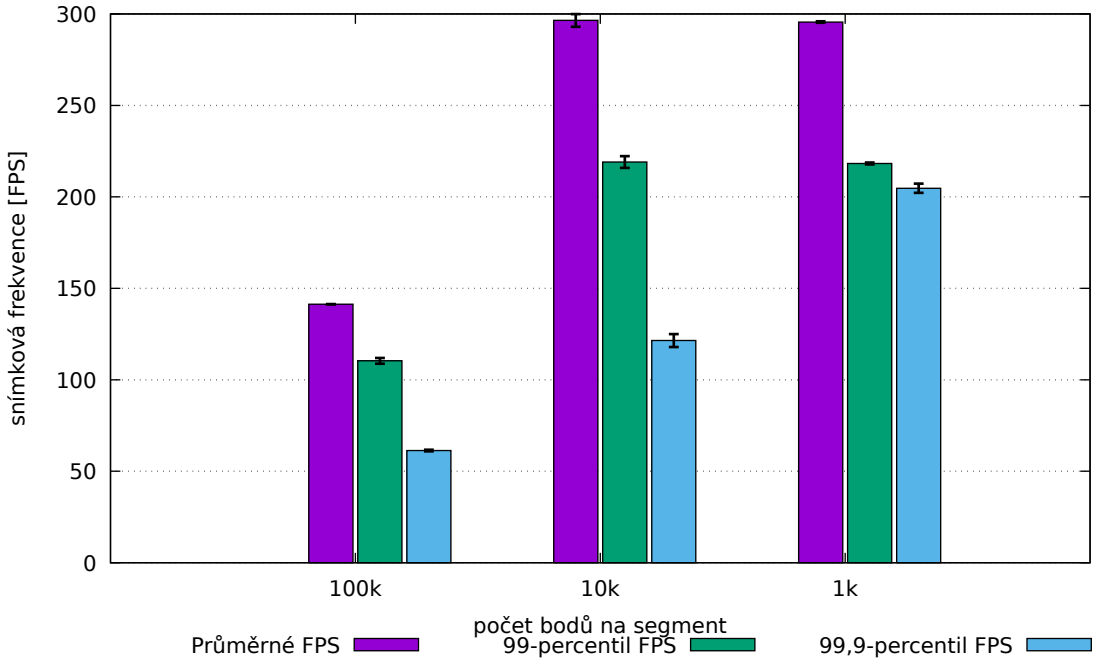


Obr. D.1: Závislost snímkové frekvence na velikosti segmentů pro statické mračno

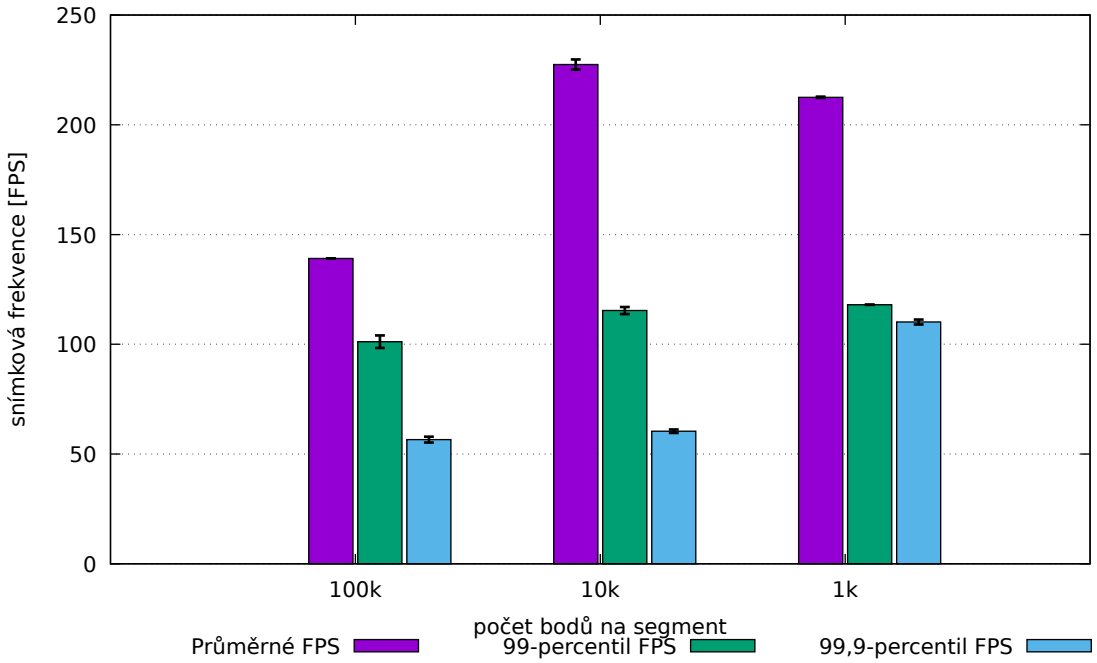
Na obrázku D.1 je vidět závislost snímkové frekvence na velikosti segmentů pro statické mračno. Segmenty o velikosti sto tisíc bodů se kreslí v průměru 124 FPS, 99-percentil 76 FPS a 99.9-percentil 45 FPS. Segmenty o deseti tisících bodů se kreslí v průměru 125 FPS, 99-percentil 66 FPS a 99.9-percentil 47 FPS. Segmenty o jednom tisíci bodů se kreslí v průměru 120 FPS, 99-percentil 115 FPS a 99.9-percentil 106 FPS.

Na obrázku D.1 je vidět závislost snímkové frekvence na velikosti segmentů pro rotující mračno. Segmenty o velikosti sto tisíc bodů se kreslí v průměru 141 FPS, 99-percentil 110 FPS a 99.9-percentil 61 FPS. Segmenty o deseti tisících bodů se kreslí v průměru 296 FPS, 99-percentil 219 FPS a 99.9-percentil 121 FPS. Segmenty o jednom tisíci bodů se kreslí v průměru 296 FPS, 99-percentil 218 FPS a 99.9-percentil 204 FPS.

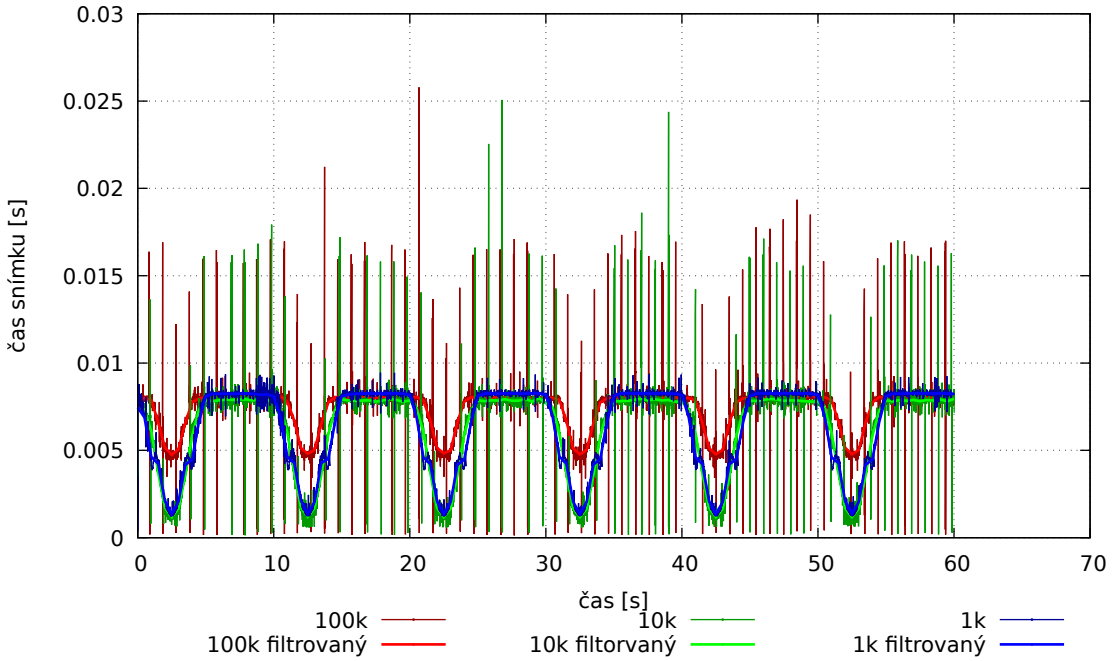
Na obrázku D.1 je vidět závislost snímkové frekvence na velikosti segmentů pro pohybující se mračno. Segmenty o velikosti sto tisíc bodů se kreslí v průměru



Obr. D.2: Závislost snímkové frekvence na velikosti segmentů pro rotující mračno



Obr. D.3: Závislost snímkové frekvence na velikosti segmentů pro pohybující se mračno



Obr. D.4: Průběh časů snímků pro pohybující se mračno s různými velikostmi segmentů

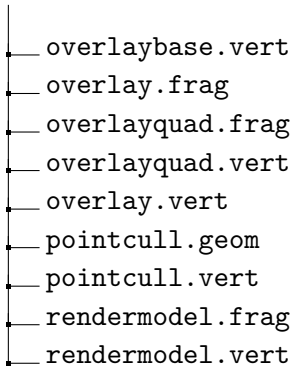
139 FPS, 99-percentil 101 FPS a 99.9-percentil 57 FPS. Segmenty o deseti tisících bodů se kreslí v průměru 227 FPS, 99-percentil 115 FPS a 99.9-percentil 60 FPS. Segmenty o jednom tisíci bodů se kreslí v průměru 213 FPS, 99-percentil 118 FPS a 99.9-percentil 110 FPS.

Na obrázku D.4 je vidět, že mračno se segmenty o sto tisíci bodech výrazně zaostává za ostatními verzemi. Je to dáno hlavně odstraněním geometrické fáze, tudíž se neořeže dostatečné množství bodu.

Nakonec jsem zvolil segmenty o velikosti jednoho tisíce bodů, protože podávaly nejlepší výsledky, hlavně v 99,9-percentilu.

E Obsah přiloženého CD

- md5sum.txt ..... Kontolní součty MD5
- modely ..... Demonstrační modely
  - areal_Technicka_small_1M38.ply ..... Mračno bodů s barvou
  - CloudCompareGarden1.ply ..... Mračno bodů s barvou a skalárním polem
  - lebka.ply ..... Neprůhledný mesh bez barvy a skalárního pole
  - pruhledna_lebka.ply ..... Průhledný mesh s bílou poloprůhlednou barvou
- sha1sum.txt ..... Kontolní součty SHA1
- sha256sum.txt ..... Kontolní součty SHA256
- VR-pointcloud-viewer-1.0
  - bin
    - benchmark.exe ..... Aplikace pro výkonová srovnání
    - boost_filesystem-vc140-mt-gd-1_62.dll
    - boost_filesystem-vc140-mt-1_62.dll
    - boost_system-vc140-mt-gd-1_62.dll
    - boost_system-vc140-mt-1_62.dll
    - glew32.dll
    - glfw3.dll
    - openvr_api.dll
    - tinyplyd.dll
    - tinyply.dll
    - VR-pointcloud-viewer.exe ..... Aplikace VR-pointcloud-viewer-1.0
  - resources ..... Pomocné soubory pro aplikaci
    - benchmarkmesh.ply
    - benchmarkmeshtransparent.ply
    - benchmarkpointcloud.ply
    - cube.frag
    - cube.vert
    - font.fnt
    - font.png
    - frametime.frag
    - frametime.vert
    - guibutton.frag
    - guibutton.vert
    - helpscreen.frag
    - helpscreen.vert
    - HTCViveHelp.png
    - char.frag
    - char.vert
    - mesh.frag
    - mesh.vert
    - OculusRiftHelp.png
    - openfiledialog.frag
    - openfiledialog.vert
    - overlaybase.frag


overlaybase.vert
overlay.frag
overlayquad.frag
overlayquad.vert
overlay.vert
pointcull.geom
pointcull.vert
rendermodel.frag
rendermodel.vert