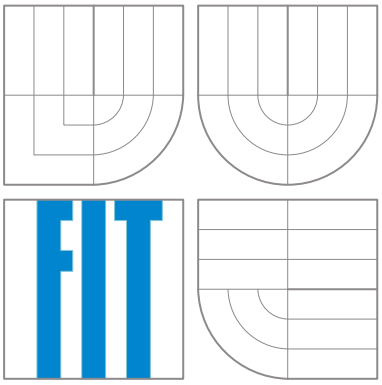

PGR – Počítačová grafika

Rasterizace na CPU

14. prosince 2018

Autoři: Kamil Michl, xmichl02@stud.fit.vutbr.cz
David Skácel, xskace12@stud.fit.vutbr.cz

Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Obsah

Zadání

Tento projekt řeší rasterizaci na CPU implementací v C++. Vzhledem k volnějšímu zadání jsme se rozhodnuli implementovat následující funkcionalitu:

- Zobrazení 3D modelů načtených ze souboru pomocí rasterizace trojuhelníků.
- Optimalizace vykreslování trojuhelníku pomocí 2D obalového tělesa, jednoduchého ořezu scénou a zahazování odvrácených stran.
- Řešení viditelnosti s pomocí jednoduchého hloubkového bufferu a konvence „top-left edge" zahazování hran.
- Texturování modelů obsahujících textury. Jednoduché stínování podle úhlu plochy vůči scéně.
- Zobrazení modelů neobsahujicích textury s pomocí Phongova osvětlovacího modelu.
- Možnost volby mezi tvrdým a hladkým gouraudovým stínováním pokud model obsahuje informace o normálách ve vrcholech.
- K implementaci jsme využili následujících knihoven:
  - GLM pro vektorové počty. [1]
  - Assimp pro načtení modelů. [2]
  - stb_image.h - hlavičková knihovna pro načítání obrazových souborů (textur). [3]
  - SDL pro zobrazení okna a obsluhu klávesnice. [4]

[1] https://glm.g-truc.net/0.9.9/index.html
[2] http://www.assimp.org/
[3] https://github.com/nothings/stb
[4] https://www.sdl.com

Nejdůležitější dosažené výsledky

Následující 3 podkapitoly popisují výslednou funkcionalitu kterou považujeme za stěžejní.

70 000 trojúhelníků v reálném čase i s osvětlením (optimalizace)

Schopnost vykreslit model o bezmála 70 000 trojúhelnících i s osvětlením v reálném čase patří mezi přední vlastnosti tohoto projektu. Na mobilním procesoru intel i7 7. generace s frekvencí 3,5GHz aplikace běží průměrně s 15 snímky za vteřinu v závislosti na vzdálenosti od vykreslovaného objektu[5].

O tuto vlastnost se zasluhují především tyto 3 optimalizace, jejichž použití lze s výjimkou druhého z nich nastavit preprocesorovými direktivami před překladem (`#define BACKFACE_CULLING` a `#define BOUNDING_BOX`):

- 2D minimální obalové těleso [6] zarovnané s osami X a Y – triangle 2D axis aligned bounding box
- Základní ořezání výhledovým tělesem – naive frustum - culling
- Zahazování odvrácených stran trojúhelníků – backface - culling

Obrázek 1: Standford bunny. 69630 trojúhelníků, 2 bodové světla (vlevo) a k nim navíc 1 směrové (vpravo).

[5] Vzhledem k povaze první optimalizace je zřejmé, že výsledný výkon se odvíjí především od toho, jak velkou část obrazu model zabírá.

[6] Jsme si vědomi, že z hlediska kontroly co možná nejmenší plochy je vhodné využít algoritmus DDA namísto 2D obalových těles, ale druhá zmíněná metoda se jevila dostačující.

Scéna	Trojuhelníků	Osvětlení	Textury	Snímky/sekundu
Sponza	13 472	1 Směrové	Ano	~ 4.5
Standford Bunny	69 630	2 bodové, směrové	Ne	~ 15
Blender's Suzanne	15 488	2 bodové, směrové	Ne	~ 15

Tabulka 1: Vzorová tabulka

Scéna s texturami

Rasterizér umožňuje načtení modelů s texturami. Je možno přeložit aplikaci se zapnutým opakováním textur s pomocí makra `#define TEXTURE_REPEAT`. Vykreslování scén s texturami je o něco rychlejší díky absenci bodových světel a Phongova osvětlovacího modelu. Použito je pouze základní stínování v závislosti na úhlu, který plocha trojúhelníku svírá s vektorem rovnoběžným s osou Z v záporném směru, kde souřadnice X a Y odpovídají bodu na trojúhelníku. Tedy $\vec{V} = vec3(Fragment_x, Fragment_y, -1.0)$. Je-li pak normálový vektor v daném bodě N, pro výsledný výpočet barvy platí vztah $Color = Color_{texture} * dot(\vec{V}, \vec{N})$, kde `dot()` je skalární součin vektorů odpovídající cosinu úhlů jimi svíraným.

Obrázek 2: Sponza - 13472 trojúhelníků s texturami.

Nasvětlení pomocí Phongova modelu

V neposlední řadě považujeme jako výsadu Phongovo osvětlení neotexturovaných modelů, které ve spojení s vhodným 3D modelem výrazně zvyšuje líbivost obrazu. Toto osvětlení implementujeme v podobě vyučované v kurzu PGR[1].

Model obsahující potřebné informace má vypočtenu barvu pro fragmet podle následující rovnice $I = I_a + I_d + I_s$, kde písmeno I označuje intenzitu světla, či v našem případě barvu, a indexy a,d a s značí ambientní diffusní a odrazivou („spekulární") složku světla. Jednotlivé složky jsou pak počítány následovně:

$$I_d = I_l \, k_d \, cos(\alpha) = I_l \, k_d \, (\vec{light} \cdot \vec{normal})$$

$$I_s = I_l \, k_s \, cos^{N_s}(\beta) = I_l \, k_s \, (\vec{view} \cdot \vec{reflection})^{N_s}$$

, kde N_s je tzv. „shininess", $k_{s,d}$ označují v tomto případě difuzní a odrazivou barvu materiálu. Nakonec pak lze vyjádřit vektor odrazu jako $\vec{r} = \vec{l} - 2(\vec{l} \cdot \vec{n})\vec{n}$ .Jak dojít k tomuto vztahu jsem čerpal zde[7].
[8]

Obrázek 3: Blender's Suzanne - 15 488 trojúhelníků.

[7] https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/reflection-refraction-fresnel

[8] Všechny prezentované obrázky byly vytvořeny v popisované aplikaci.

Zvláštní použité znalosti

Kamil Michl: Při tvorbě tohoto projektu jsem využil především svých znalostí objektově orientovaného programování a návrhu a optimalizace kódu. Dále mi byly hodně nápomocny C++ knihovny glm [2], zabývající se především složitějšími matematickými strukturami (matice, vektory), a SDL2 [3] zabývající se zobrazováním. Také jsem zde při týmové práci hojně využíval jednotného a přehledného coding style a znalosti čistého kódu.

David Skácel: Z hlediska praktické implmementace a optimálního zápisu matematických výpočtů nad rámec přednášek PGR [1] a PGP[4] oceňuji zdroje The Ryg blog[5] a a Scratchapixel.com [6], které popisují odvozovení interpolací vlastností vrcholů pro vykreslovaný fragment a interpolaci hloubky pomoci barycentrických souřadnic s ohledem na perspektivní korekci. Dále pak taky optimalizaci výpočtu barycentrických souřadnic za pomocí předpočítání základu a přičítání kroku pro souřadnice x a y.

Práce na projektu

Rozdělení práce v týmu

Zodpovědnosti byly rozděleny násldujícím způsobem:

- **Kamil Michl**: Optimalizace kódu, refaktorizace a návrh struktury kódu.
- **David Skácel**: Vykreslovací optimalizace, osvětlovací modely, texturování.

Co bylo nejpracnější

- **Kamil Michl**: Pochopení matematických výpočtů použitých při rasterizaci, optimalizace některých úseků kódu.
- **David Skácel**: Implementace konvencí, optimalizací a jejich pochopení.

Zkušenosti získané řešením projektu

Kamil Michl: Projekt mě obohatil především v oblasti práce s GLM knihovnou a jejími různými datovými typy a funkcemi. Rovněž jsem lépe pochopil některé z grafických algoritmů zabývající se především rasterizací, ale i osvětlováním a texturováním. Také mi tento projekt přinesl nové poznatky ohledně spolupráce a práce v týmu.

David Skácel: Řešení projektu mi umožnilo srovnání CPU rasterizace s CPU raytracingem, který jsem paralelně implementoval do jiného předmětu. Naučil jsem se lépe s C++ a získal praktické zkušenosti s implementací geometrických a matematických operací pro rasterizaci, což považuji za velmi přínosné.

Autoevaluace

Technický návrh (85%): Z hlediska komplexnosti zadání a časové náročnosti jsme usoudili, že míra využití pomocných prostředků byla adekvátní. Navržená struktura programu byla jednoduše rozšiřitelná a nepředstavovala ani v pozdních fázích práce žádné větší překážky.

Programování (75%): Tato část je průměrem dosažených výsledků obou řešitelů, kdy jeden se soustředil na funkční kód a to se odrazilo na kvalitě kódu, kterou se však úspěšně podařilo druhému autoru zvednout. Došlo tím rovněž k prudkému zrychlení vykreslování.

Vzhled vytvořeného řešení (90%): S výsledným výstupem jsme velice spokojení, nicméně si nemůžeme odpustit mínusové body za nedořešené chyby v podobě průsvitných hran a rozbití scény při specifických situacích.

Využití zdrojů (85%): Ač zmiňujeme v literatuře pouze ty největší zdroje, studium zahrnovalo spoustu čtení a potřebu alespoň základního porozumění problematiky. K možnosti experimentace s modely jsme využili externí knihovny

Hospodaření s časem (80%): Projekt jsme začali dělat dostatečně s předstihem, avšak naše cíle byly příliš vysoké a tak jsme na úkor toho nezvládli odstranit některé důležité chyby v zobrazovacím řetězci.

Spolupráce v týmu (65%): Na projektu jsme měli práci rozdělenou nerovnoměrně, ovšem vše na čem jsme se dohodli bylo dodrženo a navzájem jsme doplnili své nedostatky.

Celkový dojem (80%): S výběrem zadání jsme spokojení. Potřebné znalosti ke splnění projektu představují základ, který si myslíme, že by každý studeny počítačové grafiky měl znát. Rádi bychom projektu věnovali více času, především abychom adresovali nedostatky v podobě grafických chyb a pokusili se urychlit vykreslování o něco více.

Ovládání vytvořeného programu

Technologie potřebné pro spuštění programu

Potřebné knihovny Matematická knihovna GLM[1], knihovna pro vytvoření okna pro vykreslování SDL2[4], STB Image[3] pro načítání obrázků ze souboru, knihovna Assimp[2] pro načítání 3D modelů.

Programy a služby použité při tvorbě

Použili jsme některých dostupných grafických modelů [7]. Testování probíhalo především na modelech "Textured F-16", "Blender's Suzanne"a "Stanford Bunny". Kód pro načítání modelů byl převzat z `https://learnopengl.com/Model-Loading/Model`.

Obsluha programu

Po zapnutí zobrazí program model, který byl pevně zvolen ve zdrojovém kódu a umístí kameru do počáteční polohy (která je také definována přímo v kódu). Kamerou lze pohybovat následovně:

- W,S - posun kamery po ose Z (přibližování a oddalování)
- A,D - posun kamery po ose X (doleva a doprava)
- C,space - posun kamery po ose Y (nahoru a dolu)
- I,K - náklon kamery v ose Y
- J,L - náklon kamery v ose X
- R - navrácení kamery do původní pozice a původního náklonu.

Osvětlení lze ovládat omezeným způsobem následovně:

- G - vypnutí/zapnutí globálního osvlětlení
- Nahoru,Dolu,Doleva,Doprava - Posun bodového světla, pokud je použito

Pomocí klévesy ESC se program ukončí po vykreslení následujícího snímku.

Literatura

[1] Prof. Ing. Adam Herout, Ph.D., "Počítačová grafika," Magisterský kurz na FIT VUT v Brně. [Online]. Available: https://www.fit.vutbr.cz/study/courses/index.php?id=12885

[2] Opengl mathematics documentation. [Online]. Available: https://glm.g-truc.net/0.9.4/api/index.html/

[3] Sdl2.0 wiki. [Online]. Available: https://wiki.libsdl.org/FrontPage/

[4] Zemčík Pavel, prof. Dr. Ing., "Pokročilá počítačová grafika," Magisterský kurz na FIT VUT v Brně. [Online]. Available: https://www.fit.vutbr.cz/study/courses/index.php?id=12883

[5] (2018, Nov) Optimizing the basic rasterizer. [Online]. Available: https://fgiesen.wordpress.com/2013/02/10/optimizing-the-basic-rasterizer/

[6] (2018, Nov) Rasterization-practical implementation. [Online]. Available: https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/overview-rasterization-algorithm

[7] W. A. V. Schreüder, *CSCI 4229/5229: Computer Graphics*. University of Colorado at Boulder, objects in OBJ format. [Online]. Available: http://www.prinmath.com/csci5229/OBJ/index.html