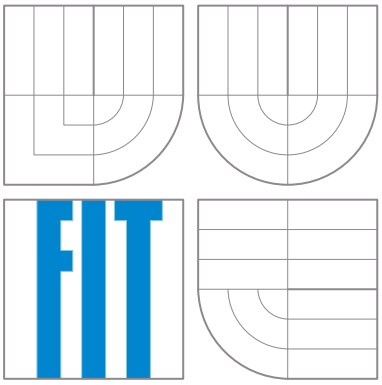

PGR { PoŁítaŁovÆ gra ka

Rasterizace na CPU

14. prosince 2018

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

Fakulta InformaŁních Technologií
VysokØ UŁení TechnickØ v Brnì

Obsah

ZadÆní

Tento projekt łe„í rasterizaci na CPU implementací v C++. Vzhledem k volnìj„ímu zadÆní jsme se rozhodnuli implementovat nÆsledující funkcionalitu:

Zobrazení 3D modelø nałtených ze souboru pomocí rasterizace trojuhelníkø.

Optimalizace vykreslovÆní trojuhelníku pomocí 2D obalovØho tìlesa, jednoduchØho ołezu scØnou a zahazovÆní odvrÆcených stran.

Øe„ení viditelnosti s pomocí jednoduchØho hloubkovØho bu eru a konvence "top-left edge"zahazovÆní hran.

TexturovÆní modelø obsahujících textury. JednoduchØ stínovÆní podle œehlu plochy vøłi scØnì.

Zobrazení modelø neobsahujicích textury s pomocí Phongova osvìtlovacího modelu.

Mo¾nost volby mezi tvrdým a hladkým gouraudovým stínovÆním pokud model obsahuje informace o normÆlÆch ve vrcholech.

K implementaci jsme vyu¾ili nÆsledujících knihoven:

- GLM pro vektorovØ połty.[1]
- Assimp pro nałtení modelø.[2]
- stb_image.h - hlaviłkovÆ knihovna pro nałítÆní obrazových souborø (textur).[3]
- SDL pro zobrazení okna a obsluhu klÆvesnice.[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

Nejdøle¾itìj„í dosa¾enØ výsledky

NÆsledující 3 podkapitoly popisují výslednou funkcionalitu kterou pova¾ujeme za stì¾ejní[5]

70 000 trojœehelníkø v reÆlnØm Łase i s osvìtlením (optimalizace)

Schopnost vykreslit model o bezmÆla 70 000 trojœehelnících i s osvìtlením v reÆlnØm Łase patłí mezi płední vlastnosti tohoto projektu. Na mobilním procesoru intel i7 7. generace s frekvencí 3,5GHz aplikace bì¾prømìrnì s 15 snímky za vtełinu v zÆvislosti na vzdÆlenosti od vykreslovanØho objektu[6].

O tuto vlastnost se zasluhují płedev„ím tyto 3 optimalizace, jejich¾ pou¾ití lze s výjimkou druhØho z nich nastavit preprocesorovými direktivami płed płekladem (`#define BACKFACE_CULLING` a `#define BOUNDING_BOX`):

2D minimÆlní obalovØ tìleso[7] zarovnanØ s osami X a Y { triangle 2D axis aligned bounding box

ZÆkladní ołezÆní výhledovým tìlesem { naive frustum - culling

ZahazovÆní odvrÆcených stran trojœehelníkø { backface - culling

ObrÆzek 1: Standford bunny. 69630 trojœehelníkø, 2 bodovØ svìtla (vlevo) a k nim navíc 1 smìrovØ (vpravo).

[5] V„echny prezentovanØ obrÆzky byly vytvołeny v popisovanØ aplikaci.

[6] Vzhledem k povaze první optimalizace je złejmØ, ¾e výsledný výkon se odvíjí płedev„ím od toho, jak velkou ŁÆst obrazu model zabírÆ.

[7] Jsme si vìdomi, ¾e z hlediska kontroly co mo¾nÆ nejmen„í plochy je vhodnØ vyu¾ít algoritmus DDA namísto 2D obalových tìles, ale druhÆ zmínìnÆ metoda se jevila dostaŁující.

ScØna	Trojuhelníkø	Osvìtlení	Textury	Snímky/sekundu
Sponza	13 472	1 SmìrovØ	Ano	4:5
Standford Bunny	69 630	2 bodovØ, smìrovØ	Ne	15
Blender's Suzanne	15 488	2 bodovØ, smìrovØ	Ne	15

Tabulka 1: VzorovÆ tabulka

ScØna s texturami

RasterizØr umo¾òuje naŁtení modelø s texturami. Je mo¾no płelo¾it aplikaci se zapnutým opakovÆním textur s pomocí makra `#define TEXTURE_REPEAT`. VykreslovÆní scØn s texturami je o nìco rychlej„í díky absenci bodových svìtel a Phongova osvìtlovacího modelu. Pou¾ito je pouze zÆkladní stínovÆní v zÆvislosti na œehlu, který plocha trojœehelníku svírÆ s vektorem rovnobì¾ným s osou Z v zÆpornØm smìru, kde souładnice X a Y odpovídají bodu na trojœehelníku. Tedy $\vec{V} = vec3(Fragment_x; Fragment_y; 1:0)$. Je-li pak normÆlový vektor v danØm bodì $\vec{N}$, pro výsledný výpoŁet barvy platí vztah $Color = Color_{texture} \cdot dot(\vec{V}; \vec{N})$, kde `dot()` je skalÆrní souŁin vektorø odpovídající cosinu œehlø jimi svíraným.

ObrÆzek 2: Sponza - 13472 trojœehelníkø s texturami.

Nasvìtlení pomocí Phongova modelu

V neposlední ładì pova¾ujeme jako výsadu Phongovo osvìtlení neotexturovaných modelø, kterØ ve spojení s vhodným 3D modelem výraznì zvy„uje líbivost obrazu. Toto osvìtlení implementujeme v podobì vyuŁovanØ v kurzu PGR[1].

Model obsahující potłebnØ informace mÆ vypoŁtenu barvu pro fragmet podle nÆsledující rovnice $I = I_a + I_d + I_s$, kde písmeno I oznaŁuje intenzitu svìtla, Łi v na„em płípadì barvu, a indexy a,d a s znaŁí ambientní di usní a odrazivou („spekulÆrní") slo¾ku svìtla. Jednotlivé slo¾ky jsou pak poŁítÆny nÆsledovnì:

$$I_d = I_l k_d \cos(\ ) = I_l k_d (light \cdot normal)$$

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

, kde N_s je tzv. „shininess", $k_{s;d}$ oznaŁují v tomto płípadì difuzní a odrazivou barvu materiÆlu. Nakonec pak lze vyjÆdłit vektor odrazu jako $reflect = light - 2(light \cdot normal)normal$. Jak dojít k tomuto vztahu jsem Łerpal zde[8].

ObrÆzek 3: Blender's Suzanne - 15 488 trojœehelníkø.

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

Zvlá¹tní pou¾ité znalosti

Kamil Michl : Pøi tvorbì tohoto projektu jsem vyu¾il pøedev¹ím svých znalostí objektovì orientovaného programování a návrhu a optimalizace kódu. Dále mi byly hodnì nápomocny C++ knihovny glm [2], zabývající se pøedev¹ím slo¾itìj¹ími matematickými strukturami (matice, vektory), a SDL2 [3] zabývající se zobrazováním. Také jsem zde pøi týmové práci hojnì vyu¾íval jednotného a pøehledného coding style a znalosti èístého kódu.

David Skácel : Z hlediska praktické implmementace a optimálního zápisu matematických výpoètù nad rámec pøedná¹ek PGR [1] a PGP[4] oceòuji zdroje The Ryg blog[5] a a Scratchapixel.com [6], které popisují odvozovení interpolací vlastností vrcholù pro vykreslovaný fragment a interpolaci hloubky pomoci barycentrických souøadnic s ohledem na perspektivní korekci. Dále pak taky optimalizaci výpoètu barycentrických souøadnic za pomocí pøedpoèítání základu a pøièítání kroku pro souøadnice x a y.

Práce na projektu

Rozdìlení práce v týmu

Zodpovìdnosti byly rozdìleny následujícím zpùsobem:

Kamil Michl : Optimalizace kódu, refaktorizace a návrh struktury kódu.

David Skácel : Vykreslovací optimalizace, osvìtlovací modely, texturování.

Co bylo nejpracnìj¹í

Kamil Michl : Pochopení matematických výpoètù pou¾itých pøi rasterizaci, optimalizace nìkterých úsekù kódu.

David Skácel : Implementace konvencí, optimalizací rasterizace a jejich pochopení. Nejvíce práce v¹ak vzalo hledání chyb. Nìkteré z nich se dodnes nepodaøilo odchytit. Aè je výstup gra cký, nìkteré problémy se hledaly velice tì¾ko.

Zku¹enosti získané øe¹ením projektu

Kamil Michl : Projekt mì obohatil pøedev¹ím v oblasti práce s GLM knihovnou a jejími rùznými datovými typy a funkcemi. Rovnì¾ jsem lépe pochopil nìkteré z gra ckých algoritmù zabývající se pøedev¹ím rasterizací, ale i osvìtlováním a texturováním. Také mi tento projekt pøinesl nové poznatky ohlednì spolupráce a práce v týmu.

David Skácel : Øe¹ení projektu mi umo¾nilo srovnání CPU rasterizace s CPU raytracingem, který jsem paralelnì implementoval do jiného pøedmìtu. Nauèil jsem se lépe s C++ a získal praktické zku¹enosti s implementací geometrických a matematických operací pro rasterizaci, co¾ pova¾uji za velmi pøínosné.

Autoevaluace

Technický návrh (85%): Z hlediska komplexnosti zadání a èasové nároènosti jsme usoudili, ¾e míra vyu¾ití pomocných prostøedkù byla adekvátní. Navr¾ená struktura programu byla jednodu¹e roz¹íøitelná a nepøedstavovala ani v pozdních fázích práce ¾ádné vìt¹í pøeká¾ky.

Programování (75%): Tato èást je prùmìrem dosa¾ených výsledkù obou øe¹itelù, kdy jeden se soustøedil na funkèní kód a to se odrazilo na kvalitì kódu, kterou se v¹ak úspì¹nì podaøilo druhému autoru zvednout. Do¹lo tím rovnì¾ k prudkému zrychlení vykreslování.

Vzhled vytvoøeného øe¹ení (90%): S výsledným výstupem jsme velice spokojení, nicménì si nemù¾eme odpustit mínusové body za nedoøe¹ené chyby v podobì prùsvitných hran a rozbití scény pøi speci ckých situacích.

Vyu¾ití zdrojù (85%): Aè zmiòujeme v literatuøe pouze ty nejvìt¹í zdroje, studium zahrnovalo spoustu ètení a potøebu alespoò základního porozumìní problematiky. K mo¾nosti experimentace s modely jsme vyu¾ili externí knihovny

Hospodaøení s èasem (80%): Projekt jsme zaèali dìlat dostateènì s pøedstihem, av¹ak na¹e cíle byly pøíli¹ vysoké a tak jsme na úkor toho nezvládli odstranit nìkteré dùle¾ité chyby v zobrazovacím øetìzci. Dále by bylo vhodné udìlat porovnání s referenèním obrazem vytvoøeným v jiném programu.

Spolupráce v týmu (65%): Na projektu jsme mìli práci rozdìlenou nerovnomìrnì, ov¹em v¹e na èem jsme se dohodli bylo dodr¾eno a navzájem jsme doplnili své nedostatky.

Celkový dojem (80%): S výbìrem zadání jsme spokojení. Potøebné znalosti ke splnìní projektu pøedstavují základ, který si myslíme, ¾e by ka¾dý studeny poèítaèové gra ky mìl znát. Rádi bychom projektu vìnovali více èasu, pøedev¹ím abychom adresovali nedostatky v podobì gra ckých chyb a pokusili se urychlit vykreslování o nìco více.

Ovládání vytvoøeného programu

Technologie potøebné pro spu¹tìní programu

Potøebné knihovny Matematická knihovna GLM[1], knihovna pro vytvoøení okna pro vykreslování SDL2[4], STB Image[3] pro naèítání obrázkù ze souboru, knihovna Assimp[2] pro naèítání 3D modelù.

Programy a slu¾by pou¾ité pøi tvorbì

Pou¾ili jsme nìkterých dostupných gra ckých modelù [7]. Testování probíhalo pøedev¹ím na modelech "Textured F-16", "Blender's Suzanne"a "Stanford Bunny". Kód pro naèítání modelù byl pøevzat zhttps://learnopengl.com/Model-Loading/Model .

Obsluha programu

Po zapnutí zobrazí program model, který byl pevnì zvolen ve zdrojovém kódu a umístí kameru do poèáteèní polohy (která je také de nována pøímo v kódu). Kamerou lze pohybovat následovnì:

W,S - posun kamery po ose Z (pøibli¾ování 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 pùvodní pozice a pùvodního náklonu.

Osvìtlení lze ovládat omezeným zpùsobem následovnì:

G - vypnutí/zapnutí globálního osvlìtlení

Nahoru,Dolu,Doleva,Doprava - Posun bodového svìtla, pokud je pou¾ito

Pomocí klévesy ESC se program ukonèí po vykreslení následujícího snímku.

Literatura

[1] Prof. Ing. Adam Herout, Ph.D., \Poèítaèová gra ka," Magisterský kurz na FIT VUT v Brnì. [Online]. Available: https://www. t.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] Zemèík Pavel, prof. Dr. Ing., \Pokroèilá poèítaèová gra ka," Magisterský kurz na FIT VUT v Brnì. [Online]. Available: https://www. t.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