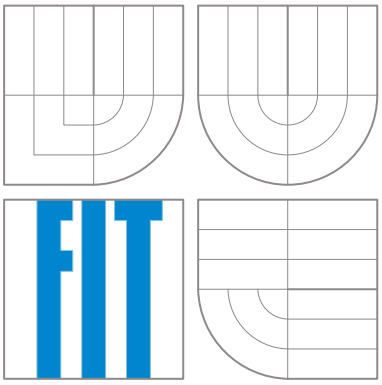


Počítačová grafika

Šachy pomocí ray-tracing

16. prosince 2014

Autoři: Jan Bednařík, xbedna45@stud.fit.vutbr.cz
Jakub Kvita, xkvita01@stud.fit.vutbr.cz

Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Obsah

Zadání

Úkolem semestrální práce bylo vytvořit vlastní program schopný vykreslit scénu s šachovnicí a šachovými figurkami pomocí metody ray tracing. Celou práci lze specifikovat několika významnými požadavky:

- Tvorba programu, který pomocí metody sledování paprsku zobrazí šachovnici s figurami.
  - Získat model šachovnice a figurek.
  - Specifikovat požadavky na obsah modelu tak, aby byl program flexibilní a nebyl vázán na jeden konkrétní model.
  - Upravit model tak, aby při co nejmenším počtu geometrických primitiv dosahoval co nejvyšší vizuální atraktivity.
- Navrhněte jednoduchý způsob uložení stavu šachovnice, který by sloužil ke komunikaci s šachovým strojem nebo modulem pro interakci s uživatelem.
  - Navrhnout jednoduchý člověku srozumitelný textový protokol pro komunikaci mezi šachovým programem a námi implementovaným vykreslovacím programem.
- Demonstrujte kvalitu zobrazení na několika příkladech stavu šachovnice, zaměřte se na jevy, které nejsou jednoduše řešitelné rasterizací na GPU (stíny, odlesky, průhlednost, ...).
  - Implementovat algoritmus ray tracing, jenž bude podporovat předně stíny, odrazivost a lom.
  - Optimalizovat algoritmus pro co nejvyšší rychlost výpočtu.
  - Navrhnout konfigurační soubor, jehož prostřednictvím bude moci uživatel nastavit parametry vykreslování, jako pozice a směr kamery, světla, materiálů modelových objektů nebo míru hloubku algoritmu ray tracing.
- Důraz je kladen na vizuální krásu výsledku
  - Připravit vizuálně atraktivní záběry scén, jež budou demonstrovat podporované vlastnosti vykreslovacího programu.
  - Navrhnout konfigurační soubor, jehož prostřednictvím bude moci uživatel konfigurovat parametry vykreslování, jako pozice a směr kamery, světla, materiálů modelových objektů nebo hloubku rekurze algoritmu ray tracing.

Použité technologie

Kapitola shrnuje technologie, jež byly při vývoji programu využity.

- Program je implementován v jazyce **C++**.
- Pro vývoj byla využita platforma Windows a vývojové prostření Microsoft Visual Studio 2012, tedy pro překlad je preferován **kompilátor MSVC** (projektové soubory obsahující nastavení kompilátoru jsou součástí odevzdaného řešení).
- Pro práci s maticemi je využita knihovna **Eigen** implementující operace lineární algebry [2].
- Program podporuje modely ve formátu **OBJ**[1]
- Pro úpravu a export modelů byl využit open source modelovací program **Blender**[2]
- Výstupní obrázky jsou ukládány ve formátu **PPM**[3]

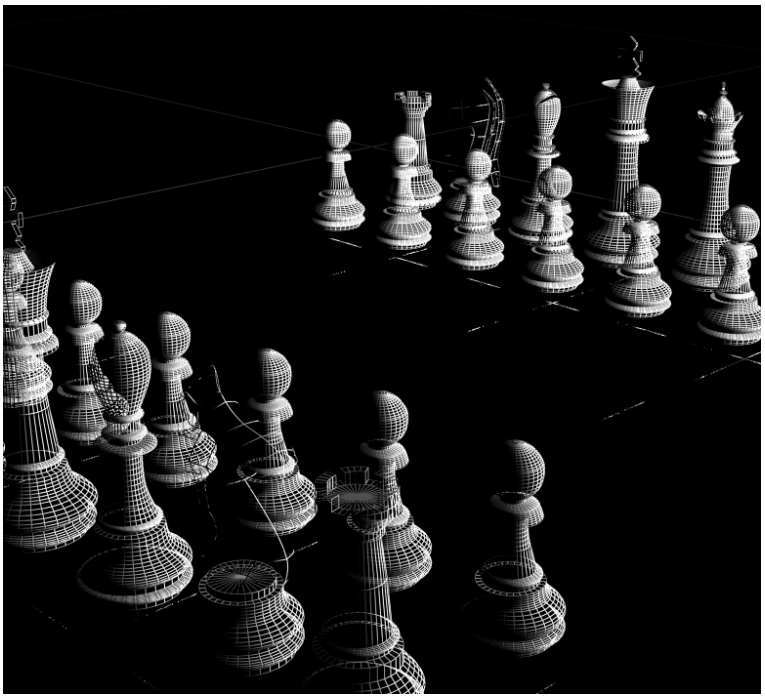
[1] Specifikaci lze nalézt zde: http://people.cs.kuleuven.be/ ares.lagae/libobj/obj-0.1/doc/OBJ.spec
[2] http://www.blender.org/
[3] Portable Pixmap Format, specifikaci lze nalézt zde: http://netpbm.sourceforge.net/doc/ppm.html

Použité zdroje

Tato kapitola uvádí zdroje, z nichž bylo při tvorbě programu čerpáno.

Modely

Volně dostupný model šachovnice a šachových figurek[1] byl získán ze serveru Turbosquid[4].



Obrázek 1: Model šachovnice a šachových figurek.

Studijní zdroje

Znalosti základního principu algoritmu ray tracing jsme získali:

- ve studijní opoře předmětu Počítačová grafika[3],
- z webu Scratchapixel[6] věnujícího se počítačové grafice.

Implementovali jsme rovněž metodu rychlého a paměťově nenáročného výpočtu průsečíku s trojúhelníkem Fast Minimum Storage Ray/Triangle Intersection [4].

[4] http://www.turbosquid.com/

Existující řešení

V implementačních detailech jsme se inspirovali:

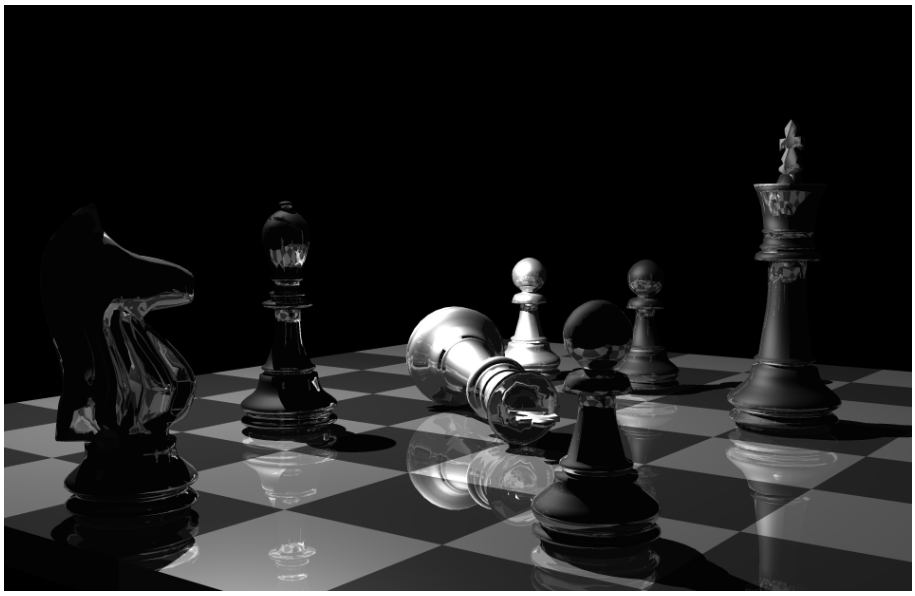
- u programu Aurelius implementujícícho ray tracing, který byl představen v rámci přednášek předmětu Počítačová grafika,
- u demonstračního programu Barebone Ray Tracer[5].

Nejdůležitější dosažené výsledky

Funkční plně parametrizovatlený ray tracing

ray tracing Implementovali jsme funkční algoritmus ray tracing, jenž si poradí s polygonálními (trojúhelníkovými) modely i parametricky zadanými koulemi. Algoritmus podporuje:

- Phongovo stínování
- stíny
- odlesky



Obrázek 2: Při vhodném nastavení dává program krásné výsledky.

modely Program není vázaný pouze na jeden specifický šachový model, nýbrž při splnění formátu OBJ souboru (viz sekci) dovede pracovat s libovolným modelem, včetně parametrizovatelných translací modelů šachových figurek za běhu programu.

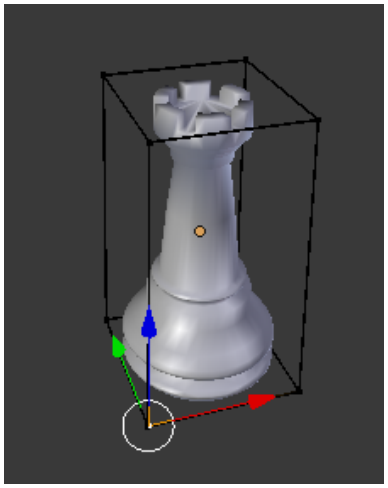
parametrizovatelnost Při implementaci byl kladen důraz na flexibilitu programu. Pomocí intuitivního konfiguračního souboru lze nastavit:

- pozice kamery

- směr kamery
- zorné pole kamery
- rozlišení čipu kamery
- pozice světla
- hloubka rekurze algoritmu ray tracing
- barva pozadí
- barvy šachových figurek
- barvy polí šachovnice

Urychlení pomocí prostorových obálek

Algoritmus ray-tracing je poměrně výpočetně náročný a při vysokých rozlišeních (FullHD a vyšších) by bylo při základní implementaci nutné na výsledek čekat i desítky hodin. Výrazného, více než **50násobného urychlení** bylo dosaženo pomocí prostorových obálek, jež jsou implementovány formou kvádrů (sestavených z 12 trojúhelníků) obalujících každou šachovou figurku. Při výpočtu vrženého paprsku se nejprve zjistí existence průsečíku s obálkami objektů, až následně se samotnými objekty.



Obrázek 3: Prostorová obálka šachové figurky.

Snadná konfigurace šachovnice

Při tvorbě programu bylo zohledněno rovněž možné propojení s šachovým strojem a byl implementován jednoduchý konfigurační soubor rozestavení šachových figurek. Jedná se o užitečnou vlastnost rovněž z hlediska renderování obrázků, neboť si může uživatel jednoduše nastavit pozice jednotlivých figurek bez toho, aniž by musel jakkoliv zasahovat do samotného modelu.

Ovládání vytvořeného programu

Program funguje jako konzolová aplikace. Parametry i formáty modelu a konfiguračních souborů jsou uvedeny v následujících sekcích.

Spuštění

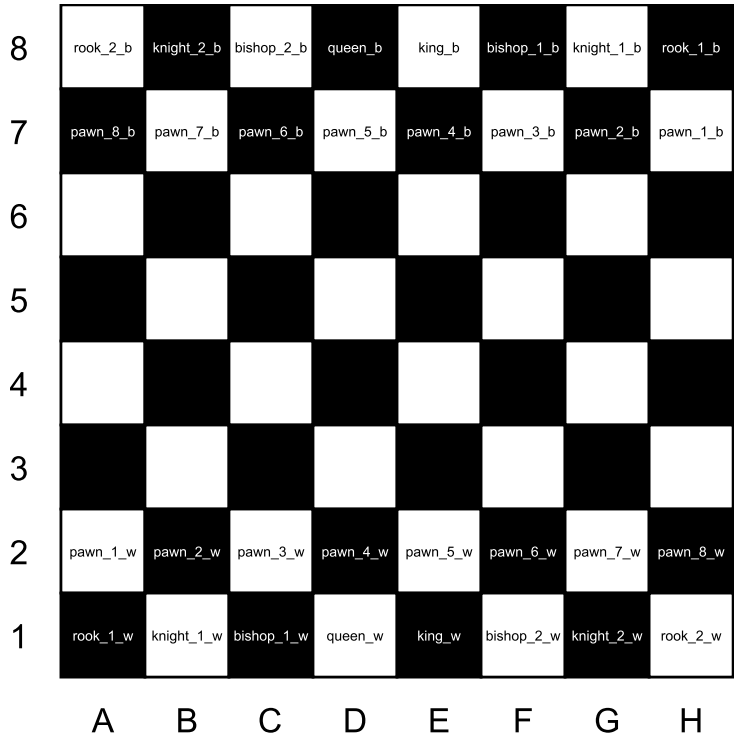
Program se spouští s následujícími parametry:

```
Usage: rtchess model config_chessboard config_ray_tracer output
    model                   model file name (.OBJ)
    config_chessboard       chessboard configuration file
    config_ray_tracer       ray tracer configuration file
    output                  output file (.PPM)
```

Formát modelu (OBJ)

- Program očekává na vstupu šachovnici, jež leží v rovině z, její hrany jsou rovnoběžné s osami x a y a její levý dolní roh políčka *A1* leží v počátku souřadného systému (tedy v bodě *[0,0,0]*).
- Figurky musí být rozmístěny v základní konfiguraci při běžném startu hry (viz obrázek 4).
- Každý objekt (šachovnice nebo figurka) musí být v modelu pojmenována podle vzorů:
  - chessboard_[b|w],
  - pawn_n_[b|w],
  - rook_n_[b|w],
  - knight_n_[b|w],
  - bishop_n_[b|w],
  - queen_[b|w],
  - king_[b|w],

  kde b odpovídá černé barvě, w odpovídá bílé barvě a n odpovídá pořadovému číslu šachové figurky.



Obrázek 4: Základní rozestavení šachovnice.

Formát konfiguračního souboru šachovnice

Názvy šachových figurek jsou stejné, jako v případě modelu. Na jednom řádku se nachází vždy název figurky, bílý znak a pozice na šachovnici ve standardním šachovém značení. Znak # uvozuje komentář.

```
# white
pawn_1_w A2
pawn_2_w B2
pawn_3_w C2
pawn_4_w D2
pawn_5_w E2
pawn_6_w F2
pawn_7_w G2
pawn_8_w H2
rook_1_w A1
knight_1_w B1
bishop_1_w C1
queen_w D1
king_w E1
bishop_2_w F1
knight_2_w G1
rook_2_w H1

# black
```

```
pawn_1_b H7
pawn_2_b G7
pawn_3_b F7
pawn_4_b E7
pawn_5_b D7
pawn_6_b C7
pawn_7_b B7
pawn_8_b A7
rook_1_b H8
knight_1_b G8
bishop_1_b F8
king_b E8
queen_b D8
bishop_2_b C8
knight_2_b B8
rook_2_b A8
```

Formát konfiguračního souboru vykreslování

Na jednom řádku se nachází vždy název parametru, bílý znak a hodnota parametru. Pro hodnoty parametrů je přípustný buď vektor ve formátu *[x,y,z]* nebo číslo. Znak # uvozuje komentář.

```
# camera
camera-position [0.182,2.459,1.930]
direction  [0.8664,0.3339,-0.3714]
width  3840
height  2160
fov  45

# light
light-position [-0.05,8.432,3.769]

# ray tracer
depth 5
bgrd-color [0.0,0.0,0.0]

# model
white-piece-color [0.88,0.2,0.2]
white-piece-reflectivity 0.7
white-piece-shininess 4.0
black-piece-color [0.32,0.2,0.01]
black-piece-reflectivity 0.7
black-piece-shininess 4.0
white-field-color [0.5,0.5,0.5]
white-field-reflectivity 0.4
white-field-shininess 16.0
black-field-color [0.66,0.66,0.44]
```

```
black-field-reflectivity 0.4
black-field-shininess 16.0
```

Zvláštní použité znalosti

Fresnelovy vztahy Pakliže má algoritmus ray tracing podporovat odraz i lom světla, je nutné vhodným způsobem kombinovat barevný příspěvek povrchu objektu v místě dopadu paprsku, příspěvek lomeného paprsku a příspěvek odraženého paprsku. Skládání intenzit světla se řídí Fresnolovými vztahy, jež na základě úhlu dopadu a indexů lomu obou sousedících prostředí určují, v jakém poměru se intenzita světla dělí mezi lomený a odražený paprsek. Pro účely (fyzikálně ne zcela přesného) ray tracingu lze vztah pro míchání intenzit zjednodušit na

$$fr = 0.9(1 - \vec{r}\vec{n})^3 + 0.1, \tag{1}$$

kde $\vec{r}\vec{n}$ značí úhel mezi paprskem a normálou v místě dopadu a fr značí poměr, v jakém se kombinuje odražený a lomený paprsek [5]. Pro závěrečné problémy s implementací lomeného paprsku však bohužel ve výsledném programu nebyly Fresnelovy vztahy implementovány.

Průsečík paprsku s trojúhelníkem v prostoru Jedním z kritických úzkých hrdel programu je výpočet průsečíku přímky s trojúhelníkem v prostoru. Pro urychlení výpočtu bylo nutné nastudovat a implementovat Fast Minimum Storage Ray/Triangle Intersection [4].

Blender Nemalou část času rovněž zabralo seznámit se s prostředím modelovacího nástroje Blender.

Práce na projektu

Kapitola shrnuje rozdělení úkolů mezi autory projektu.

- **Jan Bednařík**:
  - Implementace rekurzivní funkce sledování paprsku.
  - Implementace průsečíku s trojúhelníkem.
  - Osvětlovací model.
  - Dokumentace.
- **Jakub Kvita**:
  - Příprava modelů.
  - Návrh a implementace konfiguračních souborů.
  - Transformace kamery a světla po scéně
  - Příprava renderů.

Co bylo nejpracnější

- **Jan Bednařík**: Zřejmě nejvíce času zabrala implementace funkce pro sledování paprsku z hlediska osvětlovacího modelu a míchání barev. Implementace výpočtu průsečíku s trojúhelníky taktéž zabrala delší dobu, neboť bylo nutné hledat rychlejší implementaci oproti původním.
- **Jakub Kvita**: Pracné bylo zahrnutí parametrizovatelnosti do programu, protože jsme neprovedli dostatečně důkladný návrh a spíše jsme prototypovali, bylo nutné měnit hodně kódu. Náročné bylo také naučit se zacházet s Blenderem.

Zkušenosti získané řešením projektu

Po technické stránce jsme se zdokonalili v návrhu rozsáhlejšího projektu a v programování v C++. Protože je ray tracing obecně náročný na výpočet, museli jsme dbát na efektivitu výpočtu.

Komunikace v týmu probíhala bez problému, problémy by možná nastaly až v početnějším týmu. S časem jsme nicméně nehospodařili zcela uvážlivě, snažili jsme se dotahovat do konce technické detaily, což zapříčinilo fakt, že jsme nestihli implementovat vše, co jsme si zadali.

Autoevaluace

Technický návrh (60%): Kladli jsme důraz na rychlé testování navržených postupů, proto jsme více prototypovali a méně mysleli na robustní návrh.

Programování (80%): Ačkoliv je samotný kód nepříliš komentovaný a někdy nečitelný, spolehlivost běhu je vysoká a rovněž byl kladen velký důraz na znovupoužitelnost a nezávislost na konkrétním modelu.

Vzhled vytvořeného řešení (90%): Při vhodném nastavení parametrů je zobrazení poměrně realistické a vizuálně atraktivní, nicméně chybí implementace refrakce.

Využití zdrojů (80%): Inspirace dvěma existujícími řešeními ray trace programů, použití článku popisujícího rychlý výpočet průsečíku s trojúhelníkem.

Hospodaření s časem (40%): Příliš jsme se zaměřovali na detaily a občas tak ztratili nadhled nad celkovým stavem projektu. Projekt jsme nicméně stihli dokončit včas.

Spolupráce v týmu (100%): Spolupráce probíhala bez sebemenších problémů, což lze od pouze dvoučlenného týmu očekávat.

Celkový dojem (85%): Ačkoliv jsme bohužel nestihli implementovat vše, co jsme si zadali, poměrně dosti jsme se projektem naučili a vizuální výsledek vypadá velmi dobře.

Doporučení pro budoucí zadávání projektů

K zadávání projektů nemáme snad žádnou připomínku, témat bylo k dispozici celá řada, tudíž si mohl vybrat každý. Možná by bylo pouze snazší, kdyby se na konkrétní zadání přihlašovali pouze vedoucí týmů, čímž by odpadla při případných konfliktech starost s vyjednáváním mezi současně registrovanými týmy, kdo které téma zpracuje.

Spuštění programu

Kapitola shrnuje data a software, jež nejsou součástí odevzdaného řešení, avšak jsou potřeba pro korektní běh, a proces překladu i spuštění.

Závislosti

Model šachů Program podporuje obecně libovolný model šachovnice a šachových figur, jenž splňuje formát specifikované v sekci . Při vývoji jsme využili volně dostupný model Glass chess set [1]. Jeho upravená verze splňující formát specifikovaný naším programem je dostupná na http://www.stud.fit.vutbr.cz/ xbedna45/PGR/chess.obj.

Knihovna Eigen Na cílové platformě musí být instalovaná volně dostupná knihovna Eigen [2] a umístěna v kořenovém adresáři `C:\`.

Překlad

Po rozbalení archivu *xbedna45.zip* je nutné ve vývojovém prostředí Visual Studio 2012 (či vyšší verzi) otevřít projektový soubor *rtchess.sln* a přeložit jej (doporučujeme překládat i spouštět v režimu *Release*, jež má povoleny optimalizace a program běží rychleji).

Spuštění

Program přijímá čtyři vstupní parametry (viz sekci), nicméně v projektu jsou výchozí parametry přednastaveny, tudíž postačí pouhé stažení modelu šachovnice *chess.obj* (viz) a jeho umístění do adresáře *rtchess**models*\. Výstupní obrázek je uložen do adresáře *rtchess*\

Literatura

[1] cjx3711. Glass chess set [online].
http://www.turbosquid.com/FullPreview/Index.cfm/ID/544320, 2010.

[2] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3.
http://eigen.tuxfamily.org, 2010.

[3] Adam Herout. Počítačová grafika - studijní opora. prosinec 2008.

[4] Tomas Möller and Ben Trumbore. Fast, minimum storage ray-triangle intersection. *J. Graph. Tools*, 2(1):21–28, říjen 1997.

[5] Scratchapixel. A barebone ray tracer [online].
http://www.scratchapixel.com/old/lessons/3d-basic-lessons/lesson-8-putting-it-all-together-part-1-a-barebone-ray-tracer/a-barebone-ray-tracer/, 2009-2014.

[6] Scratchapixel. Scratchapixel v2.0 - learn computer graphics programming from scratch [online].
http://www.scratchapixel.com, 2009-2014.