Projekt do predmetu **PGR – Počítačová grafika 2019**

gi10 - Šachy pomocou sledovania lúčov

riešiteľ: **Timotej Halás**, xhalas10

Zadanie

- Zobrazenie šachovnice s figúrkami pomocou sledovania lúčov
- Dôraz je kladený na zobrazovanie v reálnom čase
- Taktiež na zobrazenie čo najrealistickejšie vyzerajúcich základných materiálov
- Podpora obrázkových textúr a normálových máp pre textúry
- Možnosť interakcie s figúrkami na šachovnici
- Možnosť meniť materiály figúrok
- Možnosť pridávania a uberania svetelných zdrojov v scéne
- Možnosť meniť optické vlastnosti kamery v scéne
- Zobrazenie environmentálnej mapy ako pozadia a taktiež ako zdroja svetla pre scénu
- Uloženie stavu figúrok po vypnutí a načítanie po opätovnom zapnutí

Použité technologie

- C++
- OpenGL
- OS Windows (Testované iba na Windowse ale Linux by mal fungovať tiež)
- Nvidia GPU (na Inteli nefunguje a AMD som netestoval)
- CMake
- GIT (potrebný pre stiahnutie všetkých knižníc)
- GitHub
- Visual Studio

Použité zdroje

Knižnice a kód:

- GLFW – OpenGL knižnica pre vytváranie okien, kontextov atď...
- geGL – OpenGL C++ Wrapper
- glm – matematická knižnica pre OpenGL
- ImGUI – grafické užívateľské rozhranie
- LodePNG – načítavanie a ukladanie obrázkov vo formáte PNG
- Vars – knižnica pre ukladanie shared pointrov na C++ objekty
- tinyobjloader – knižnica pre načítavanie 3D modelov vo formáte OBJ
- TinyXML-2 – knižnica pre načítavanie a ukladanie súborov vo formáte XML
- Fluctus – path tracer v OpenCL
- GLSL Shader Includes – knižnica pre podporu #include direktív v GLSL
- HDRLoader – knižnica pre načítavanie HDRI environmentálnych máp
- Fast-BVH – implementácia BVH

Vedecké články a online zdroje:

- Megakernels Considered Harmful: Wavefront Path Tracing on GPUs
  - Popis Wavefront Path Tracingu od Nvidie
- Physically Based Rendering:From Theory To Implementation
  - Kniha o PBR
- Scratchapixel 2.0
  - Rôzne teoretické a aj praktické informácie
  - Využíval som algoritmy pre priesečníky s rôznymi geometrickými útvarmi

3D modely a textúry:

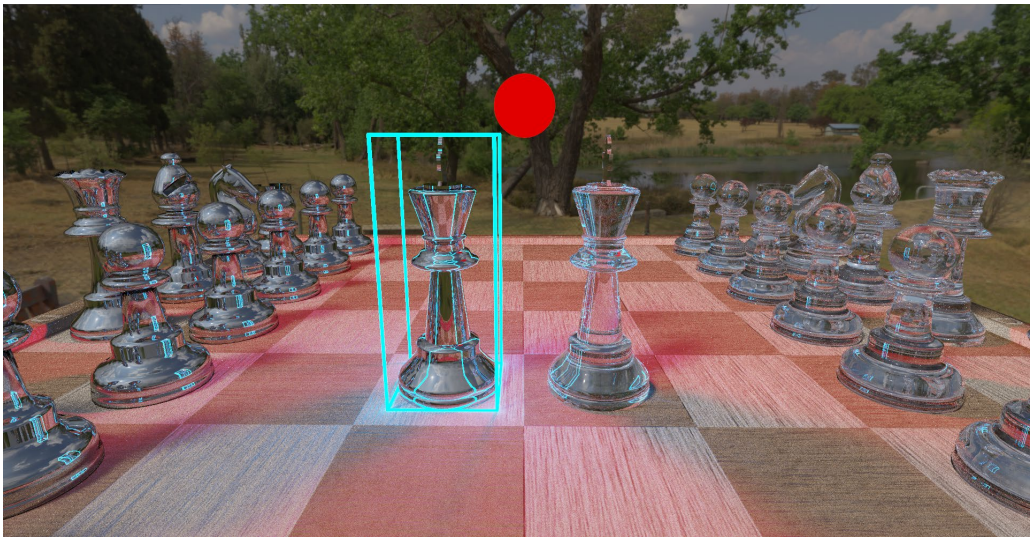
- Figúrky
- Šachovnica

- Stôl a obrus
- Textúry stola a obrusu
- Environmentálna mapa

Najdôležitejšie dosiahnuté výsledky

- Simulácia clony (hĺbka ostrosti)

- Refraktívne (figúrky vpravo), difúzne (šachovnica), reflektívne (figúrky vľavo) a emisívne materiály (červené svetlo a modré hrany kocky ukazujúce práve označený objekt)



- Realistické zobrazenie (ukážka lomu svetla cez refraktívny materiál)

Ovládanie vytvoreného programu

Pri vypnutom užívateľskom rozhraní:

- WASD – pohyb po prostredí
- G – zapnutie užívateľského rozhrania

Pri zapnutom užívateľskom rozhraní:

- Ľavé tlačidlo myši na objekt v scéne – otvorenie nastavení objektu ak je to možné
- Ctrl + Ľavé tlačidlo myši na objekt v scéne – zaostrenie na objekt
- G – vypnutie užívateľského rozhrania

Užívateľské rozhranie:

- App info
  - Uloženie renderu do súboru
  - Pridanie svetla do scény
- Render settings
  - Nastavenie spôsobov rendrovania
  - Nastavenie pozadia (environmentálna mapa alebo jednofarebné pozadie)
  - Nastavenie kamery
    - Uhol pohľadu
    - Priemer clony
    - Rýchlosť pohybu
- Material settings
  - Nastavenie materiálu figúrok
- Light settings
  - Zobrazí sa po ťuknutí na svetlo
  - Nastavenie farby, polohy, intenzity a polomeru svetla (pre jemné tiene)
  - Odstránenie svetla zo scény
- Piece settings
  - Zobrazí sa po ťuknutí na figúrku
  - Nastavenie pozície danej figúrky na šachovnici

Osobitné použité znalosti

Upravoval a textúroval som si modely v programe Blender.

Čo bolo najpracnejšie

Najpracnejšie bolo určite zisťovanie informácií. Zo začiatku mi veľmi veľa času zabralo naučiť sa pracovať s Compute shadermi. Najskôr som skúšal základnú OpenGL knižnicu. Tam ale bolo veľmi pracné urobiť niektoré veci. Napríklad len vytvoriť buffer. Rozhodol som sa pohľadať nejaký objektový wrapper pre OpenGL a veľmi sa mi zapáčil geGL od Tomáša Mileta. Je to veľmi pekná, prehľadná a ľahko pochopiteľná implementácia. Ďalej mi veľmi veľa času zabralo naštudovať ako vôbec funguje ray tracing a path tracing a tiež rozdiely medzi nimi. Zo začiatku som skúšal naimplementovať veľmi jednoduchý ray tracer, ktorý podporoval len gule. Nakoniec som ale našiel vedecký článok (Wavefront Path Tracing) od Nvidie, ktorý som sa rozhodol naimplementovať. Jeho pochopenie mi trvalo zďaleka asi najviac a v aktuálnom stave v mojej implementácií stále chýbajú niektoré veci z článku. Často som nemohol veľmi dlhú dobu zaspať, pretože moja hlava bola plná svetelných lúčov a predstáv o tom ako fungujú a ako to premietnuť do kódu.

Skúsenosti získané riešením projektu

- Implementácia veľmi zaujímavého vedeckého článku
- Použitie Compute shaderov v OpenGL a práca s pamäťou na GPU
- Rôzne paralelizačné optimalizácie na grafickej karte (napr. atomický inkrement premennej v rámci warpu pomocou ballotu)
- Prehľad o optike a fyzike svetla
- Veľmi veľa vedomostí o path tracingu a ray tracingu

Autoevaluácia

Technický návrh: 90% (analýza, dekompozícia problému, voľba vhodných prostriedkov, …)

Program je rozdelený do veľkého počtu tried, kde každá slúži na niečo iné. Vybral som si veľa knižníc, ktoré som použil na to aby program bol interaktívny.

Programovanie: 70% (kvalita a čitateľnosť kódu, spoľahlivosť behu, obecnosť riešenia, znovupoužiteľnosť, …)

Čitateľnosť a komentovanosť kódu by mohla byť oveľa lepšia a taktiež lepšie rozdelenie do tried. Spoľahlivosť je veľmi dobrá a riešenie sa dá pomerne jednoducho upraviť pre niečo iné ako šach.

Vzhľad vytvoreného riešenia: 85% (uveriteľnosť zobrazenia, estetická kvalita, vzhľad GUI, …)

Kvalita výsledkov je veľmi dobrá a uveriteľná pri väčšom počte iterácií rendrovania. Vzhľad GUI je pomerne dobrý ale je obmedzený knižnicou ImGUI.

Využitie zdrojov: 90% (využitie existujúceho kódu a dát, využitie literatúry, …)

Využil som veľmi veľa zdrojov či už kódu alebo, veľmi veľa som toho načítal a pravdepodobne som pri tom strávil viac času ako pri implementácií samotnej.

Hospodárenie s časom: 90% (rovnomerné dotiahnutie častí projektu, miera ponáhľania, chýbajúce časti řiešenia …)

Na projekte som robil konštantne a rovnomerne od naklikania si projektu vo WISe. Miera ponáhľania sa samozrejme zvyšovala, pretože som chcel naimplementovať viac a viac funkcií

Celkový dojem: 100% (pracnosť, získané znalosti, užitočnosť, voľba zadania, hocičo, …)

Na projekte som strávil určite najviac času zo všetkých projektov na FITe. Nie preto, že by som sa s ním tak trápil ale preto, že ma zo všetkých úplne najviac bavil. Naučil som sa veľmi veľa vecí a rád by som v projekte a v rozširovaní znalostí pokračoval napr. ako na diplomovej práci ak by bolo niečo také možné (nemusí to byť konkrétne šach). Hodnotenie celkového dojmu dávam 100% preto, že veľakrát som bol tak ohromený z výsledkov, ktoré z programu vychádzali (aj keď rendrovanie chvíľu trvá), že som nechápal ako sa mi niečo také podarilo. Ďalej dávam 100% preto, že to bol naozaj najzaujímavejší projekt a musím povedať, že nech som už v minulosti pracoval na hocičom (nie len IT veci) tak nikdy ma to nechytilo tak ako tento projekt.

Doporučenie pre budúce zadávanie projektov

Veľmi mi vyhovuje takýto spôsob dokumentácie. Je stručná, povie toho viac a nezanoruje sa zbytočne až moc hlboko do danej problematiky (to je skôr vecou DP). Zadania sú zaujímavé a páči sa mi možnosť vlastného zadania aj keď som ho nevyužil, pretože hneď ma zaujalo práve toto zadanie. Taktiež sa mi páči, že nie je špecifikovaná platforma alebo prostredie, pre ktoré musí projekt byť naimplementovaný.