**projekt** do předmětu **PGR – Počítačová grafika 2022**

**Rasterizace na CPU**

řešitel: Ladislav Dokoupil, xdokou14

# Zadání

Zadaním projektu je softwarový rasterizér na CPU. Zadání jsem pojal jako vytvoření celé vlastní grafické pipeline, tedy od načítání vrcholů ze standartního formátu, až po jeho vykreslení na obrazovku. To zahrnuje zejména načítaní modelu, vertex shader, clipping, transformace, rasterizace a fragment shader. Pro zjednodušení práce bylo využito již existujících knihoven zejména pro načítaní dat a matematické operace.

# Nejdůležitější dosažené výsledky

* načtení generického modelu objektu
* možnost navigace kamery
* aplikace png textury

# Ovládání vytvořeného programu

Pro jednoduchost byla použita pouze rotace okolo počátku, ve kterém je umístěn model. Pomocí kláves W a S se mění vzdálenost k modelu a klávesami A a D je tvořena rotace okolo modelu.

# Použité technologie

Projekt je tvořen v jazyce C++ s kompilačním nástrojem cmake.

Program využívá několika knihoven:

* GLFW – vytvoření grafického uživatelského rozhraní
* GLM – matematická knihovna pro vektorové a maticové operace
* STB – pro načítání modelu ze standartního formátu
* ASSIMP – načítání textur (zejména) z PNG souboru

Výše zmíněné knihovny jsou umístěny v adreáři libs s vyjímkou souboru STB, jejíž hlavičkový soubor musel být umístěn mezi zdrojové soubory.

# Použité zdroje

Modely a textury pro vykreslení byly převzaty z:

* https://github.com/harzival/stanford\_bunny\_obj\_mtl\_jpg

Studijní zdroje byly tvořeny zejména:

* https://github.com/g-truc/glm
* https://assimp-docs.readthedocs.io/en/v5.1.0/
* <http://www.opengl-tutorial.org/>

# Co bylo nejpracnější

Jako nejpracnější se jevilo studium dokumentace všech použitých knihoven a obzvláště jejích linkování s cmake. Aplikace textur je nejvíce problematická, jelikož i po zdlouhavém debugování nebylo možné korektně aplikovat UV souřadníce na model.

# Zkušenosti získané řešením projektu

Naučil jsem se lépe pracovat s cmake a C++, navíc jsem byl odrazen od budoucí práce s nimi kvůli složitosti importování knihoven. Také jsem získal obecný nadhled nad fungování GPU a její pipeline.

# Autoevaluace

**Technický návrh: 100%**

Snaha o použití co nejvíce již existujících řešení, znovupoužitelnost, dekompozici problému na jednotlivé funkce a jejich logická souvislost mi příjdou adekvátní.

**Programování: 70%**

Kritické části kódu mi přijdou adekvátně komentované, struktura kódu je jasná, avšak použití globálních/třídních proměnných by mělo být zredukováno. Možnost zobrazení pouze souborů o jednom modelu a navigace kamery je velice jednoduchá, avšak pro ukázaní konceptu práce na CPU mi přijde přijatelná.

**Vzhled vytvořeného řešení: 30%**

Samotný zajíček navíc se „špatně“ aplikovanou texturou se jeví v GUI celkem smutně.

**Využití zdrojů: 100%**

Množství a kvalita již použitých řešení a literatury mi příjde adekvátní dané problematice.

**Hospodaření s časem: 80%**

Svou práci s časem bych hodnotil jako rozumnou a splnil jsem všechny klíčové body zadání. Samozřejmě by šlo přidávat dalši funkcionality jako ray tracing či shading, avšak to mi příjde spíše jako práce pro OpenGl, jelikož i desítky tisíc polygonů v současném řešení se jeví procesoru jako velmi náročné.

**Celkový dojem: 70%**

Zadání se mi jeví velice volné avšak klíčové body pro zobrazení byly splněny. Ucelil jsem si základní znalosti pipeline GPU s nadhlem pro implementaci vhodného rozhraní k jejímu použití. Dále jsem si procvičil práci s kompilačními nástroji, C++ a snažil jsem se o co nejrychlejší zorientování se v neznámých knihovnách. Projekt mi přišel užitečný, avšak příště bych se raději pustil do práce s OpenGl. Ta by mi umožnila snadnějí vytvořit líbívější modely a odprostit se od mnoha implementačních detailů.