projekt do předmětu **PGR – Počítačová grafika 2020**

Globální viditelnost v systému místností

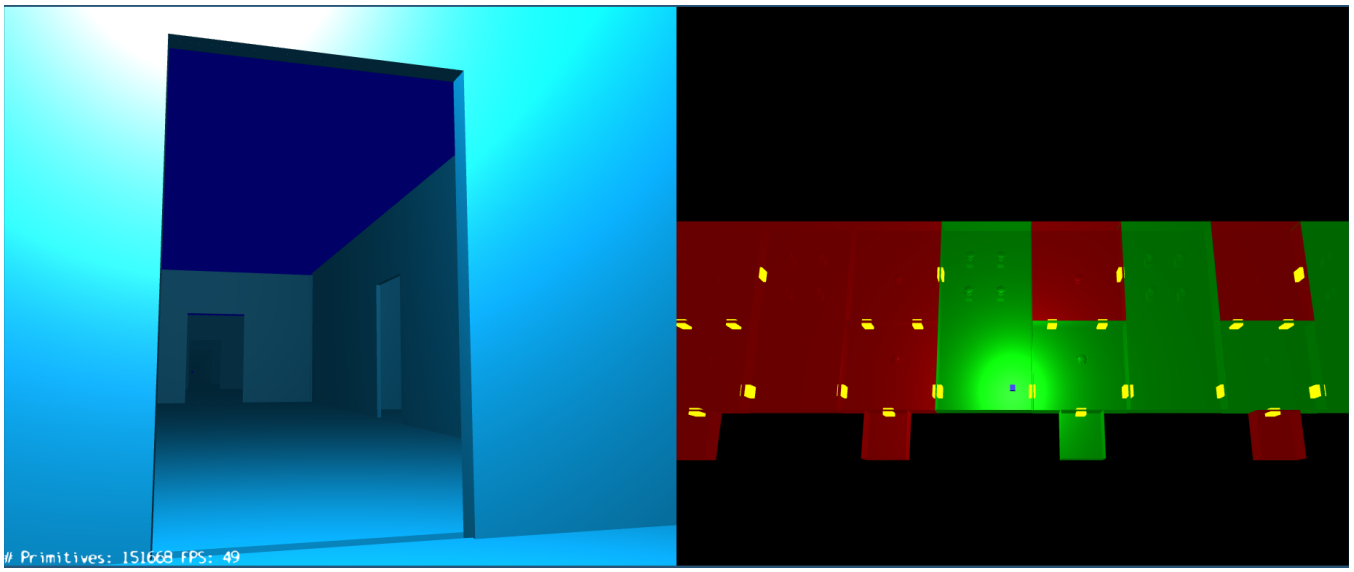
řešitel: **Lukáš Piwowarski**, xpiwow00

Zadání

- Vytvořit model systému místností.
  - Systém místností by měl být dostatečně velký na to, aby na něm byla vidět úspora algoritmu.
- Naprogramovat aplikaci v OpenGL, která demonstruje portálový algoritmus.
- Demonstrovat úsporu, kterou získáme při použití portálového algoritmu na následujících metrikách:
  - FPS,
  - počet renderovaných primitiv za snímek.
- Naprogramovat aplikaci, která ukáže model ze dvou pohledu a to z nadhledu a z pohledu kamery, která modelem prochází.
- Ukázat, jak je důležité správně navrhnout rozložení portálů v modelu (např. pokud je série portálů na přímce, tak se efekt portálového algoritmu snižuje)

Nejdůležitější dosažené výsledky

1. Vysledný program umožňuje nahlédnout na renderovaný model za dvou pohledů, což umožňuje názornou ukázku fungování portálového algoritmu.
2. Program ukazuje počet renderovaných trojúhelníků a také rychlost renderování v jednotkách FPS.
3. Program umožňuje vypnout portálový algoritmus. V takovém případě klesnou FPS z hodnoty 60 na hodnotu ~ 6 (pro model skládající se z 127 milionů trojúhelníků)



Ovládání vytvořeného programu

- Pro pohyb v modelu se používají klávesy **W,A,S,D** a pro otáčení kolem své vlastní osy **myš.**
- Pro vypnutí portálového algoritmu se používá klávesa **O** a pro jeho opětovné zapnutí se používá klávesa **P**.
- Klávesou **1** a **2** se změní renderovaný model:
  - **1** - model bez portálů, které se nacházejí na přímce
  - **2** - model s portály, které se nacházejí na přímce
- Klávesou **ESC** se program ukončí.

Zvláštní použité znalosti

- Portálový algoritmus jako takový jsme probírali na přednášce. Narazil jsem však na dva přístupy, které se používají pro rozhodnutí, zda daný portál je renderovaný:
    1. Využívání AABB pro zjištění, zda daný portál je viditelný. Každý portál se promítne do screen-space, ve kterém je pro každý portál vytvořen AABB. Pro zjištění, zda je portál postupně testujeme vytvořené AABB.
    2. Pro zjištění zda je daný portál viditelný použijeme occlusion query v OpenGL. Occlusion query umožňuje zjistit, počet fragmentů, které byly vyrenderované (**přístup použití v implementaci**)
- Při práci na projektu jsem se také naučil pracovat s programem Blender, se kterým jsem ještě nikdy nepracoval.

Použité technologie

- GLFW (vytvoření a správa okna aplikace),
- GLM (matematická knihovna pro OpenGL),
- GLEW (pro zjištění, které funkcionality OpenGL jsou dostupné)

Použité zdroje

- Pro pochopení portálového algoritmu jsem využil následující zdroje:
    - 1 - C. B. Jones, A new approach to the 'hidden line' problem
    - 2 - David Luebke, Chris Georges, Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets
    - 3 - Jiří Petruželka, Výpočet viditelnosti v 3D bludišti
    - 4 - youtube, How Occlusion Culling Works: Portal-based Occlusion Culling

- Začínal jsem projekt na začátku semestru, proto jsem pro pochopení OpenGL použil následující tutorial www.opengl-tutorial.org. Kód z tutoriálu jsem také použil pro zobrazení modelu, pohyb v modelu, zobrazení textu (soubory úplně převzaté nebo převzaté a částečně upravené jsou označeny v hlavičce souboru)

Co bylo nejpracnější

- Nejpracnější mi přišlo rozchodit OpenGL a pochopit, jak zhruba funguje. Začínal jsem projekt celkem brzy před přednáškama z OpenGL.
- Chvilku mi trvalo přijít na to, jakým způsobem přesně portálový algoritmus funguje a vymyslet, jak jej implementovat.
- Podařilo se mi vytvořit pro mě celkem nepříjemný bug zahrnující pointery a vektory v C++. Chvilky mi trvalo, než jsem ho vyřešil.

Zkušenosti získané řešením projektu

Popište, co jste se řešením projektu naučili. Zahrňte dovednosti obecně programátorské, věci z oblasti počítačové grafiky, ale i spolupráci v týmu, hospodaření s časem, atd.

Rozsah: formulujte stručně, uchopte cca 3-5 věcí

- Naučil jsem se pracovat se základníma funkcema v Blenderu.
- Vykreslovací pipeline v OpenGL
- Některé nové věci v C++ (práce s cmake)

Autoevaluace

Technický návrh: 75% (analýza, dekompozice problému, volba vhodných prostředků, …)

- Myslím si, že jsem navrhl implementaci portálového algoritmu dobře.

Programování: 60% (kvalita a čitelnost kódu, spolehlivost běhu, obecnost řešení, znovupoužitelnost, …)

- Kód implementace portálového algoritmu je dobrý. Určitě by mohl být lepší a zasloužil by si trochu "uklidit". Abych s kódem byl úplně spokojený, tak bych některé části rád přepsal.

Vzhled vytvořeného řešení: 90% (uvěřitelnost zobrazení, estetická kvalita, vhled GUI, …)

- Myslím si, že vzhled výstupu je kvalitní.

Využití zdrojů: 90% (využití existujícího kódu a dat, využití literatury, …)

- Myslím si, že jsem používal dobré zdroje.

Hospodaření s časem: 90% (rovnoměrné dotažení částí projektu, míra spěchu, chybějící části řešení, …)

- Myslím si, že s časem jsem pracoval dobře a na projektu jsem začal pracovat brzy. Necítil jsem žádný spěch.

Celkový dojem: 75% (pracnost, získané dovednosti, užitečnost, volba zadání, cokoliv, …)

- Myslím si, že výsledná aplikace je dobrá. A dobře demonstruje to, co by měla. Možná bych byl rád, kdybych udělal trochu složitější modely a udělal jich více. Kód by mohl být určitě o něco lepší.

Doporučení pro budoucí zadávání projektů

Líbilo se mi možnost konzultací k projektu, na které bylo jednoduché se dostat. Možná bych byl rád za nějaké větší intro k projektům na začátku semestru (co je/není možné). Vybral bych si možná i svoje zadání, ale zčásti i svou vlastní lenností jsem si raději vybral už předpřipravené zadání než abych něco vymýšlel a zkoušel co je a není možné.