STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18. Informatika

Prohlížeč 3D modelů

Dominik Pazdera
Olomoucký kraj

Přerov 2024

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18. Informatika

Prohlížeč 3D modelů

3D model viewer

Autoři: Dominik Pazdera
Škola: Střední průmyslová škola, Přerov, Havlíčkova 2, 750 02 Přerov
Kraj: Olomoucký kraj
Konzultant: Ing. Pavel Cimbálník

Přerov 2024

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Přerově dne 27.3. 2024 Dominik Pazdera

Anotace

Ve své práci SOČ jsem se zabýval projektem v jazyce C++ se zaměřením na vývoj prohlížeče 3D modelů s využitím OpenGL. Cílem bylo vytvořit efektivní jednoduché a uživatelsky přívětivé rozhraní pro prohlížení a manipulaci s 3D modely.

Prohlížeč je schopen načítat různé formáty 3D souborů a zobrazovat je v reálném čase. Uživatelé mohou manipulovat s úhly kamery, což jim umožní prozkoumat je z různých úhlů.

Využití OpenGL umožňuje efektivní renderování a nízkoúrovňový přístup. Projekt zahrnuje jednoduché uživatelské rozhraní, které umožní snadnou navigaci a manipulaci s modely.

Klíčová slova

Programování, C++, OpenGL, Linux, 3D grafika, Prohlížeč 3D modelů, CMake, 3D model

Annotation

In my SOČ thesis, I worked on a C++ project focusing on the development of a 3D model viewer using OpenGL. The goal was to create an efficient simple and user-friendly interface for viewing and manipulating 3D models.

The viewer is capable of loading different 3D file formats and displaying them in real time. Users can manipulate the camera angles, allowing them to explore them from different angles.

The use of OpenGL allows for efficient rendering and low-level access. The project includes a simple user interface that allows for easy navigation and manipulation of the models.

Keywords

Programming, C++, OpenGL, Linux, 3D graphics, 3D model viewer, CMake, 3D model

Seznam zkratek	
API	**Application programming interface** je rozhraní sloužící pro komunikaci dvou nebo více systémů.
MSVC	**Microsoft Visual C++** je překladač pro C/C++ od společnosti Microsoft, který používá Visual Studio.
GCC	**GNU Compiler Collection** je sada překladačů vytvořených v rámci projektu GNU.
ISO	**International Organization for Standardization** je mezinárodní organizace, který se zabývá tvorbou mezunárodních norem.
GPU	**Graphics processing unit** je specializovaný procesor, který slouží k vykreslování grafiky. Také známý jako grafická karta.
X11	**X Window System** je systém na Unix-like operačních systémech, který umožňuje vytvořit uživatelské rozhraní.
DPI	**Dots per inch** je údaj určující, kolik pixelů se vejde do délky jednoho palce.
VRAM	**Video Random Acces Memory** je označení pro paměť, kde se ukládají obrazová data.
GLSL	**OpenGL Shading Language** je programovací jazyk, který se používá pro psaní shaderů pro OpenGL a Vulkan.
HLSL	**High Level Shader Language** je programovací jazyk, který se používá pro psaní shaderů pro DirectX a Vulkan.
WGSL	**WebGPU Shading Language** je programovací jazyk, který se používá pro psaní shaderů pro aplikace používající WebGPU API.
IMGUI	**Immediate mode graphics user interface** je typ uživatelského rozhraní, které vykresluje grafické prvky přímo na obrazovku bez ukládání jejich stavu.
RMGUI	**Retained mode graphics user interface** je typ uživatelského rozhraní, které ukládá stav grafických prvků a vykresluje je na základě uložených dat.

Obsah

Úvod

Cílem této práce bylo vytvořit a popsat aplikaci, která bude sloužit jako prohlížeč 3D modelů. To znamená, že aplikace by měla být schopná dané modely načíst a manipulovat s nimi v 3D prostoru. Aplikace by dále měla být schopná načítat 3D modely různých formátů a následně na ně aplikovat textury. Důležitou součástí by také měla být podpora operačních systémů Linux a Windows. Vytvořený program bude napsaný v C++ a pro vykreslování grafiky bude použito nízkoúrovňové grafické API OpenGL.

1. Seznámení s použitými technologiemi

1.1 Programovací jazyk C++

C++ je výkoný multiparadigmatický programovací jazyk. Tento jazyk se používá pro programování aplikací, které vyžadují vysoký výkon a nízkoúrovňový přístup k paměti.
C++ je staticky kompilovaný jazyk a neobsahuje garbage collector. Správa paměti je na programátorovi, což znamená, že se každá ručně alokovaná paměť musí ručně smazat (Tento problém částečně řeší funkcionalita standardu C++11, která se nazývá chytré ukazatele).

Jazyk C++ vznikl v roce 1985 a vyvinul ho Bjarne Stroustrup. Jazyk C++ je „rozšíření“ jazyka C tzn. kód napsaný v jazyce C lze zkompilovat kompilátorem pro jazyk C++, ale nikoliv opačně.

C++ je velmi komplexní jazyk, který obsahuje oproti jazyku C velké množství funkcí
a má více běžně používaných kompilátorů. Nejčastěji využívané kompilátory jsou Clang, GCC a MSVC, přičemž Clang a GCC jsou multiplatformní a open source kompilátory. Kompilátor MSVC je pouze pro Windows a je closed source.

Programovací jazyk C++ je standardizovaný organizací ISO. Každé tři roky vychází nový standard, který přidává různé funkcionality. Nejnovější standard je C++23. Implementace nových standardů je na programátorech kompilátorů. (STANDARD C++ FOUNDATION, 2023)

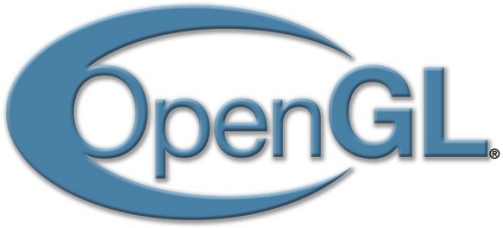
Obrázek 2: Logo C++

Obrázek 1: Tvůrce C++ Bjarne Stroustrup

1.2 Grafické API OpenGL

OpenGL je průmyslový standard pro psaní 2D a 3D grafiky. Je to multiplatformní aplikační rozhraní. OpenGL je používáno pro přímou komunikaci s grafickým procesorem (GPU). OpenGL bylo poprvé vytvořeno, jako otevřená alternativa k Iris GL, která byla proprietárním grafickým API na pracovních stanicích Silicon Graphics. První verze OpenGL, verze 1.0, byla vydána 30. června 1992. Od roku 2006 je OpenGL spravováno neziskovou technologickou konsorcií Khronos Group, která stojí například také za grafickým API Vulkan, jenž je modernější, výkonnější, ale zároveň mnohem více nízkoúrovňový a komplexnější alternativou k OpenGL. (THE KHRONOS® GROUP INC. OpenGL, 2023)

Implementace OpenGL se nachází prakticky na každém zařízení, které je schopné vykreslování grafiky. Samotná implementace OpenGL se nachází v ovladačích grafické karty a je na výrobci grafických karet, aby jej implementoval správně. Mimo implementaci v ovladačích existují i implementace OpenGL, které jsou softwarové. Ty se používají na zařízeních, která nepodporují OpenGL tzn. nemají implementaci v ovladači. Softwarová implementace OpenGL se může používat např. ve virtuálních strojích, kdy běžně ovladače virtuální grafiky nemají moderní implementaci OpenGL.



Obrázek 3: Logo OpenGL

1.2.1 Použití OpenGL

OpenGL je definováno jako sada funkcí, které může aplikace volat. Tyto funkce je třeba získat z ovladače grafického adaptéru. K tomu slouží vytvořené knihovny, jenž získají ukazatele na funkce, které jsou definované v ovladači grafické karty za běhu resp. po inicializaci knihovny. Teoreticky není potřeba knihovna pro práci z OpenGL. Je možné ukazatele na funkce získat z ovladače ručně, ale něco takového by bylo zdlouhavé a záviselo by na určité platformě.

1.2.2 Knihovny pro OpenGL

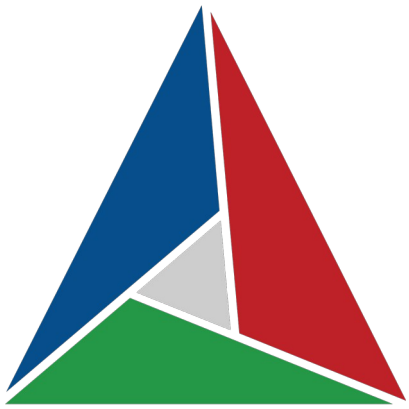
Nejznámější knihovny pro tohle použití jsou GLEW a GLAD. Hlavním rozdílem mezi těmito knihovnami je ten, že GLAD obsahuje pouze hlavičkový soubor s definicemi funkcí, které je možné si při stahování knihovny zvolit, Hlavičkový soubor pak lze jednoduše zahrnout do projektu. GLEW na rozdíl od GLAD je .lib knihovna, která je třeba i nalinkovat. Obě tyto knihovny jsou open source a multiplatformní.

1.3 CMake

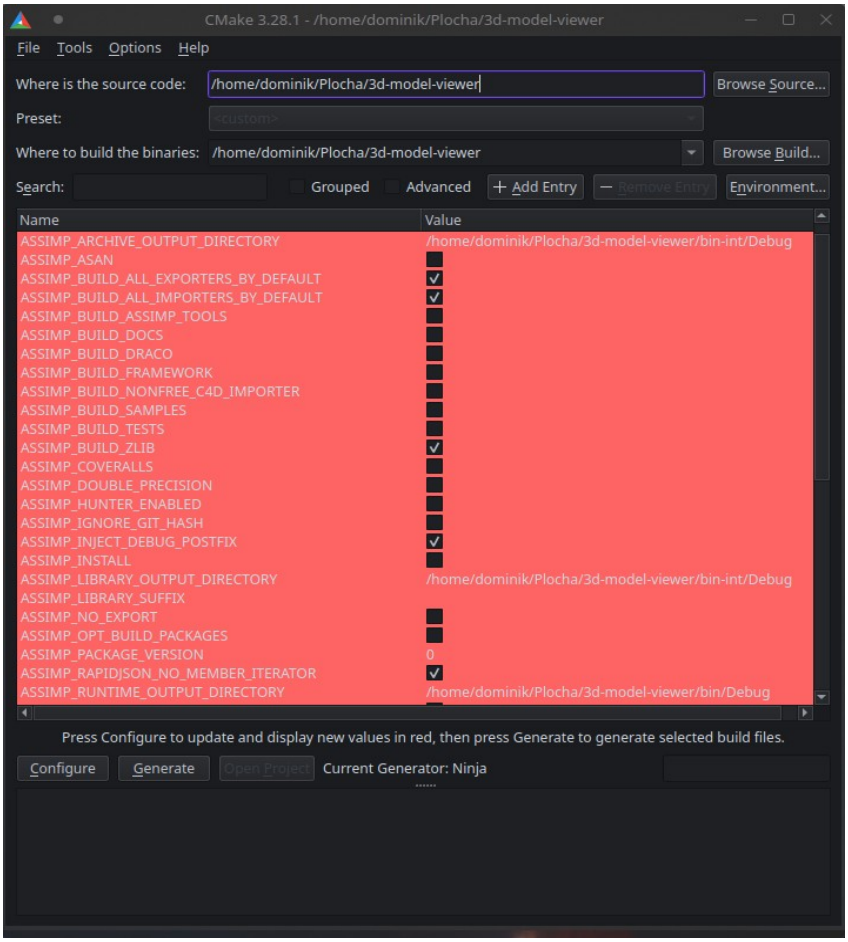
Programovací jazyk C++ nemá žádný oficiální správce balíčků a nemá žádný výchozí systém pro správu závislostí, jako např. mají jiné programovací jazyky, jako Rust – Cargo, C# - NuGet, Python - pip atd…

Pro správu závislostí a zajištění multiplatformnosti C++ aplikace, existuje vícero nástrojů. De-facto standard pro psaní multiplatformního C++ kódu je nástroj CMake, který zjednodušuje správu závislostí a kompilaci napříč platformami. CMake je open source
a multiplatformní nástroj.

Konfigurace CMake se nastavuje pomocí skriptu. Skript se píše skriptovacím jazykem přímo pro tento nástroj, který se nazývá CMake script. V skriptu se běžně konfigurují knihovny resp. jejich lokace v adresářové struktuře, zahrnování hlavičkových souborů knihoven, podmíněné překlady, nastavování definic maker a spoustu dalších...



Obrázek 4: CMake logo



Obrázek 5: CMake-GUI - Aplikace, která poskytuje grafické rozhraní jinak čistě terminálovému cmake

2. Práce s oknem

2.1 Knihovna GLFW

GLFW je open source multiplatformní knihovna pod licencí Zlib, která je napsaná v jazyce C a používá se k vývoji OpenGL, OpenGL ES a Vulkan aplikací. Tato knihovna přináší jednoduché API pro vytváření oken a čtení vstupu ze vstupních zařízeních.

Knihovna GLFW nativně podporuje Windows, macOS a Linux. Na Unix-like systémech (Linux, FreeBSD apod.) GLFW podporuje jak X11, tak i Wayland. Knihovna také má podporu pro otevírání vícero oken, pro více monitorů a také podporu pro vysoká DPI.

2.2 Instalace GLFW

Instalace knihovny lze v C++ provést mnoha způsoby. Pro účely tohoto projektu jsem zvolil možnost kompilace knihovny pomocí CMake, protože knihovna GLFW používá jako hlavní systém sestavování CMake. Kompilace knihovny je proto jednoduchá a stačí jen ve scriptu přidat knihovnu GLFW jako podadresář a zahrnout hlavičkové soubory knihovny do projektu. CMake si poté v podadresáři najde soubor se scriptem pro CMake a knihovnu zkompiluje a nakonec nalinkuje ke spustitelnému souboru. Knihovna se linkuje staticky.

2.3 Použití knihovny GLFW

Před vytvořením okna musí být GLFW inicializována. Pokud dojde k chybě, funkce vrátí *GLFW_FALSE*. Před ukončením aplikace se knihovna GLFW také ukončuje. Toto ukončení zničí jakákoliv zbývající okna a uvolní prostředky alokované knihovnou.

Po inicializací knihovny se vytvoří okno, pomocí funkce *glfwCreateWindow*. Pokud tvorba okna proběhla úspěšně tzn. ukazatel na okno se nerovná nulovému ukazateli, tak spustíme hlavní aplikační cyklus *while (!glfwWindowShouldClose(window)) {}*, uvnitř tohoto cyklus se budou provádět veškeré volání pro vykreslování.

Pro získávání informací ze vstupních zařízeních slouží tzv. „callbacky“ což jsou funkce, nebo také statické metody, které jsou volány knihovnou GLFW s argumenty, jenž popisují danou událost. Pro funkčnost callbacků je třeba každou iteraci aplikačního cyklus zavolat funkci *glfwPollEvents*, ta zajistí, že knihovna zpracuje data ze vstupních zařízeních a zavolá příslušný callback specifikovaný v kódu. (GLFW Documentation, 2022)

3. Vykreslování modelů

3.1 Inicializace OpenGL

Pro začátek práce s grafickým API OpenGL musíme nejdříve knihovnu, která nám poskytuje funkce OpenGL z ovladače grafické karty inicializovat. V aplikaci jsem použil pro získání OpenGL funkcí knihovnu GLAD. Knihovna GLAD se inicializuje voláním funkce *gladLoadGLLoader(...)*, kde funkce má jeden parametr typu *GLADloadproc*, což je ukazatel na funkci, která se používá k načítání adres OpenGL funkcí. (Learn OpenGL, 2014)

Funkce, jenž poskytuje takovou funkci je knihovna GLFW, kterou jsem použil pro práci s okny. Funkce pro inicializaci knihovny vrací celé číslo. Podle něj můžeme zjistit, zda se inicializace povedla, tzn. pokud je výsledek 1, potom se inicializace podařila, pokud je to 0, tak se stala chyba a funkce OpenGL nebyly načteny.

3.2.1 Komponenty potřebné pro vykreslování

OpenGL je nízkoúrovňové grafické API. To znamená, že k vykreslení objektu na obrazovku je potřeba znát a nastavit více věcí. Mezi ně patří – Objekty, které chceme vykreslit, potom také Shadery a Buffery. Mezi nepovinné komponenty patří např. textury, kamera, světla v scéně, či dodatečné buffery.

3.2.2 Objekt, pro vykreslení

Objekt, který můžeme chtít pomocí OpenGL vykreslit může být např. 3D model nebo i jednoduchý tvar např. trojúhelník, ale může to být i text. Takový objekt můžeme do OpenGL aplikace dostat tak, že jej buď přímo specifikujeme v kódu. Což je relevantní jen pro jednoduché tvary. Jinak můžeme například 3d model načíst ze souboru.
Jakékoliv objekty, které vykreslujeme pomocí OpenGL jsou definovány vrcholy a indexy daných vrcholů. Tyto dvě věci představují minimum pro specifikaci modelu. Modely většinou ještě obsahují další data, jako jsou souřadnice pro textury, nebo normály.

3.2.3 Buffery

OpenGL pro vykreslení modelu potřebuje znát jeho vrcholy. Pro předání dat o modelu se v OpenGL používají Buffery (vyrovnávací paměti). Bufferů je v OpenGL mnoho typů, ale pro vykreslení 3D modelu stačí dva typy, vertex buffer a index buffer.

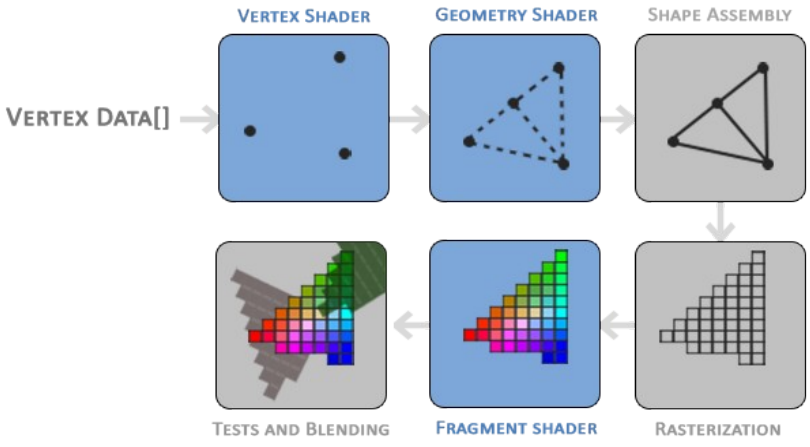
Do vertex bufferu se ukládají data o vrcholech modelu. Mimo vrcholy modelu se do vertex bufferu také běžně ukládají texturové souřadnice, barva, nebo také normály objektu.

Do indexu bufferu se ukládají indexy na vrcholy. Indexy vrcholů jsou při komplexnějších modelech potřeba, kvůli úspoře paměti. Souřadnice vrcholů, se tak nemusí duplikovat, ale existují indexy, které na ně odkazují. Buffery s daty po nastavený ovladač grafické karty nahraje do paměti grafické karty VRAM, pro rychlejší přístup.

3.2.4 Shadery

Shadery jsou většinou malé programy, které běží na grafické kartě. Takové programy jsou psané v různých jazycích. Pro OpenGL se používá jazyk GLSL. Mezi ostatní jazyky patří např. HLSL, Spir-V (Velmi nízkoúrovňový jazyk pro Vulkan API), nebo WGSL. Jazyk GLSL má syntax velmi podobný jazyku C. Shadery se dělí na více základních typů.

- Vertex shader: Program, který se provede na každém vrcholu (vertexu) specifikovaném ve vertex bufferu. (Takový program proběhne tolikrát kolik je specifikováno vrcholů)
- Geometry shader: Umožňuje přidávat nebo odebírat vrcholy a tím ovlivňovat výslednou ge ometrii.
- Fragment shader: Prováděn pro každý pixel (fragment) rasterizované scény.
- Shadery pro teselaci: Umožňují měnit geometrii objektů.
- Compute shader: Slouží k realizaci a možnému urychlení obecných algoritmů na grafickém procesoru.



Obrázek 6: Postupné zpracování vrcholů až k vykreslení

3.2.5 Volitelné komponenty při vykreslování

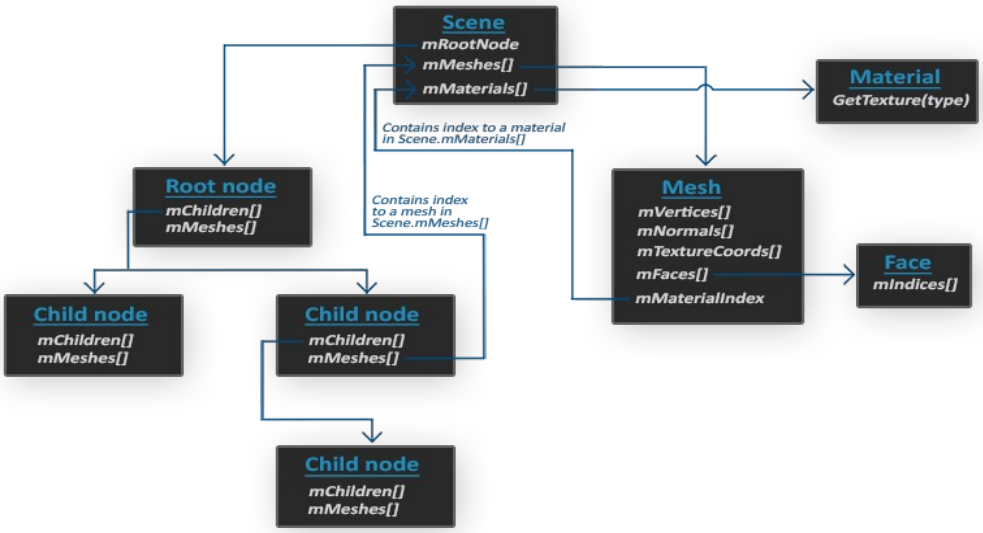
Volitelný komponent může být textura. Textura je obrázek, který je načten (většinou pomocí nějaké externí knihovny jako např. stb-image) a taková textura je potom vytvořena. Podobně jako buffer je následně poslána do paměti grafické karty. Taková textura se poté musí specifikovat pro použití ve fragment shaderu. V shaderu se poté textura namapuje na daný vykreslovaný objekt. Objekt, který by měl mít texturu musí mít specifikované souřadnice textur v vertex bufferu pro správné mapování textury.

Další častý volitelný komponent může být kamera. V OpenGL neexistuje nic jako systém kamer. Jednoduchá kamera je v OpenGL definována dvěma, respektive třemi maticemi. První je pohledová (View) matice se používá k transformací mezi souřadnicemi objektu. Následuje projekční (Projection) matice. Ta je použita k transformaci objektů z prostoru kamery do prostoru zobrazení.

Nejčastější typ takové projekční matice je perspektivní matice, která vytváří perspektivu ve scéně (čím dál je objekt na ose Z, tím je menší). Poslední je modelová (Model) matice se používá k transformaci objektu. Mezi takové transformace patří rotace, posun a škálování.

3.3 Načtení 3D modelu

Knihovna Assimp má pro načtení modelů jednotnou datovou strukturu postavenou na uzlech díky kterým se dají modely jednoduše načíst.
Načtení 3d modelu pomocí knihovny assimp je poměrně přímočaré. Nejprve je třeba vytvořit instanci Assimp::Importer.
Tato instance se pak používá k načtení 3D modelu z disku pomocí metody ReadFile. Tato metoda vrací strukturu aiScene, která obsahuje všechny informace o načteném modelu. (Assimp-docs, 2020)



Obrázek 7: Stromová datová struktura knihovny assimp

Většina komplexnějších 3d modelů není sestavena pouze z jednoho tvaru, ale každý model má obvykle několik dílčích modelů ,ze kterých se skládá. Takovým částem modelu se říká mesh.

Po načtení modelu je třeba zpracovat načtená data a převést je do formátu, který OpenGL potřebuje. To zahrnuje procházení scény a extrakci vertexů, normál, textur a dalších atributů z každého meshe.

4. GUI (Grafické uživatelské rozhraní)

4.1 Knihovna Dear ImGui

Dear ImGui je knihovna pro grafické uživatelské rozhraní napsaný v jazyce C++. Knihovna poskytuje vrcholová (vertex) data, pro jednoduchou integraci s různými grafickými API. Knihovna Dear ImGui je zveřejněna pod svobodnou licencí MIT.

4.2 Immediate Mode GUI

K vykreslování jsou běžně používány dva druhy přístupu k vykreslování Immediate Mode (IMGUI) a Retained Mode (RMGUI). Známé knihovny, které používají způsob Immediate Mode jsou například Dear ImGui nebo například knihovna egui. Známé knihovny, které používají Retained Mode jsou například Qt, WxWidgets a také GTK. (КАРАЏИЋ, Бранимир, 2024)

Hlavní rozdíly mezi IMGUI a RMGUI

IMGUI:

- Překresluje grafické rozhraní každý snímek, tzn. v ideálním případě při zapnuté vertikální synchronizaci a displeji, který má obnovovací frekvenci 60 hz se rozhraní překreslí šedesátkrát za sekundu.
- IMGUI může mít větší nároky na výkon, kvůli častému překreslování, nehledě na interakci od uživatele. Tohle stálé překreslování může zvýšit pocit responsivity rozhraní.
- Práce s IMGUI knihovnou je poměrně jednoduchá, kdy pro vykreslení a zároveň získání stavu daného „widgetu“ stačí jen jednoduchá funkce v podmínkovém bloku.

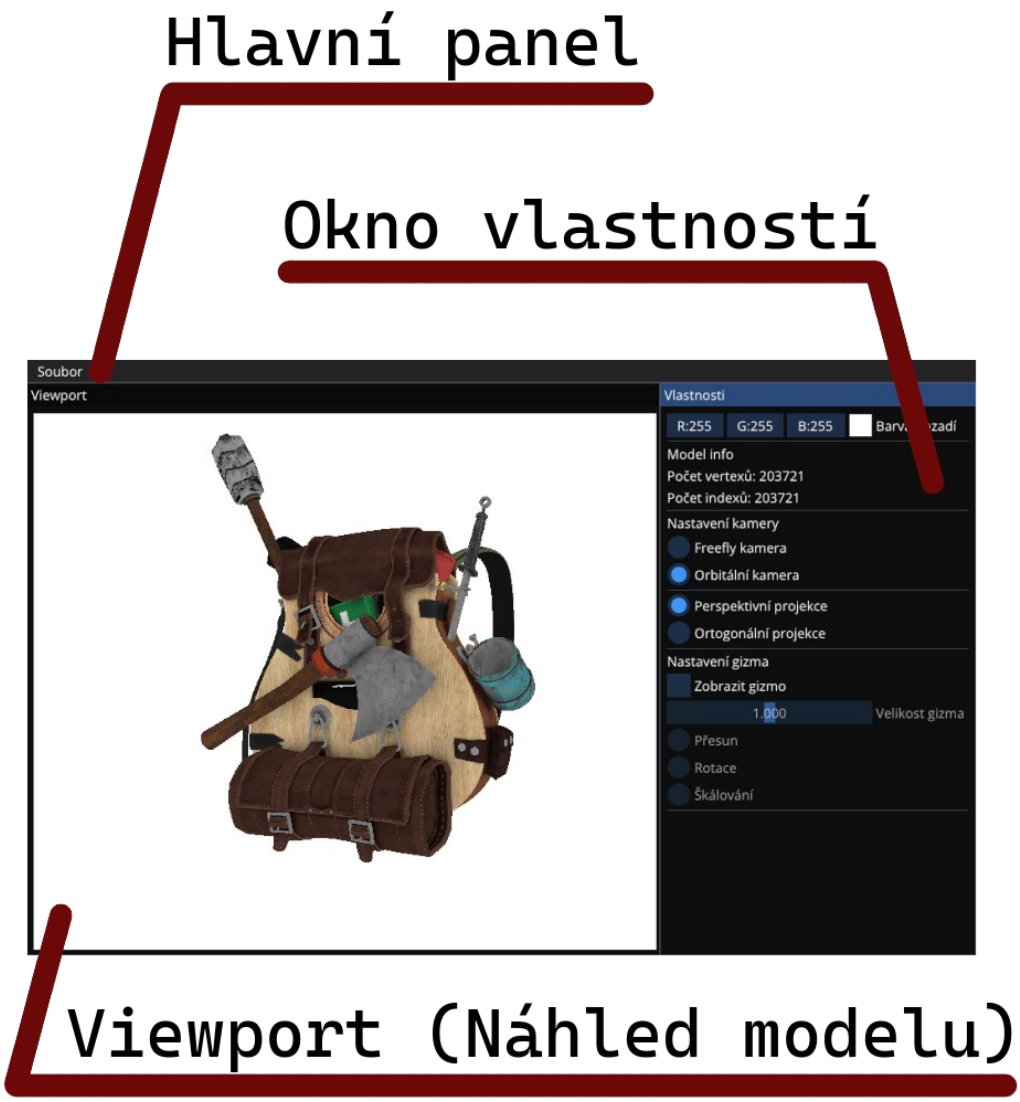
RMGUI:

- Běžné se ui překresluje jen tehdy, když je provedena nějaká interakce s uživatelským rozhraním. To může být efektivnější, kdy se stav uživatelského rozhraní mění jen méně často.
- RMGUI má běžně menší nároky na výkon kvůli tomu, že se rozhraní překreslí jen tehdy, kdy je opravdu potřeba.
- Veškeré „widgety“ jsou běžně reprezentovány jako objekty. Tento způsob je velmi úzce propojen s konceptem objektově orientovaného programování.

4.3 Rozložení grafického rozhraní

Rozložení grafického rozhraní pro aplikaci, která bude sloužit jako prohlížeč 3D modelů, jsem se snažil udělat jednoduché, pro snadné ovládání a orientaci v aplikaci. Velkou část okna pokrývá „Viewport“ tzn. část okna, která obsahuje náhled samotného modelů, pokud nebyl zatím načten žádný 3D model, tak je o tom vypsán text na místo viewportu.

V pravé části okna aplikace se nachází okno, které slouží k nastavení pohledů na model a zobrazuje základní informace o právě načteném modelu.
V horní části okna se nachází hlavní panel, které obsahuje menu s názvem „Soubor“, ve kterém se dá zvolit model, který chce uživatel načíst.



Obrázek 8: Popis uživatelského rozhraní

Závěr

Tato aplikace umožňuje načítat a zobrazovat 3D modely v různých formátech a aplikovat na ně textury pro dosažení realistického vzhledu. Nabízí jednoduché uživatelské rozhraní, které umožňuje intuitivní manipulaci s modelem, jako je otáčení, posunování a změna velikosti.

Aplikace je navržena s ohledem na otevřenost a spolupráci. Zdrojový kód je uložen na veřejném repozitáři na webu Github a je licencován pod open-source licencí MIT, čímž je volně dostupný pro kohokoliv k použití, úpravě a sdílení. Díky tomu je ideální platformou pro další vývoj a rozšíření funkcionalit.

Aplikace je napsána v moderním programovacím jazyce C++ 17, čímž je zajištěna vysoká efektivita a výkon. Pro správu knihoven a usnadnění instalace a konfigurace je použitým nástrojem CMake.

Vytvořená aplikace splnila stanovené cíle a představuje funkční nástroj pro práci s 3D modely. Díky open source přístupu a dostupnému zdrojovému kódu má velký potenciál pro další rozvoj a široké spektrum využití. V budoucnu je možné implementovat podporu pro více grafických efektů, přidat nástroje pro jednoduchou úpravu modelů, integrovat aplikaci s herními enginy platformami. Vytvořit aktivní komunitu uživatelů a vývojářů, kteří by se podíleli na dalším vývoji.

Použité zdroje

1. Learn OpenGL. DE VRIES, Joey. LearnOpenGL [online]. 2014 [cit. 2023-10-05]. Dostupné z: https://learnopengl.com

2. Assimp-docs [online]. 2020, 2022 [cit. 2023-10-05]. Dostupné z: https://assimp-docs.readthedocs.io/en/v5.3.0/index.html

3. GLFW Documentation [online]. 2022 [cit. 2023-10-05]. Dostupné z: https://www.glfw.org/docs/latest/

4. STANDARD C++ FOUNDATION. ISO C++. STANDARD C++ FOUNDATION. ISO C++ [online]. 2023 [cit. 2023-11-09]. Dostupné z: https://isocpp.org

5. THE KHRONOS® GROUP INC. OpenGL [online]. 1997, 2023 [cit. 2023-11-09]. Dostupné z: https://www.opengl.org

6. КАРАЏИЋ, Бранимир. Why I think Immediate Mode GUI is way to go for GameDev tools [online]. In: . [cit. 2024-02-07]. Dostupné z: https://gist.github.com/bkaradzic/853fd21a15542e0ec96f7268150f1b62

Tabulka obrázků