FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

Aplikace pro vizuální skládání transformací 3D scény

Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie

Autor práce: **Jiří Vokřínek**
Vedoucí práce: Ing. Jiří Jeníček, Ph.D.
Ústav informačních technologií a elektroniky

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Aplikace pro vizuální skládání transformací 3D scény

Jméno a příjmení:	**Jiří Vokřínek**
Osobní číslo:	M18000219
Studijní program:	B2646 Informační technologie
Studijní obor:	Informační technologie
Zadávající katedra:	Ústav informačních technologií a elektroniky
Akademický rok:	2022/2023

Zásady pro vypracování:

1. Proveďte rešerši technologií vhodných pro visuální návrh scény pomocí skládání grafických transformací do stromu scény a vyberte vhodnou technologii.
2. Proveďte rešerši grafických transformací a operací a vyberte vhodnou podmnožinu pro realizaci.
3. Implementujte aplikaci a otestujte ji z hlediska výkonu.
4. Zhodnoťte možnosti využití a limity aplikace. Navrhněte případný další vývoj.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] G. Sellers, R. S. Wright, N. Haemel. *OpenGL SuperBible, Seventh Edition*. 2016. ISBN 9780672337475.

[2] Kessenich, J., Sellers, G., Shreiner, D.: OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V, 2016, Addison-Wesley Professional, ISBN 978-0134495491

Vedoucí práce:	Ing. Jiří Jeníček, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	24. října 2022
Předpokládaný termín odevzdání:	22. května 2023

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 24. října 2022

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

13. května 2023 Jiří Vokřínek

Aplikace pro vizuální skládání transformací 3D scény

Abstrakt

Tato bakalářská práce se zaměřuje na nástroje pro vizuální skládání transformací 3D scény. Sepisuje seznam požadavků na nástroj, který by byl schopný přiblížit počítačovou grafiku začátečníkům v této oblasti, včetně matematických objektů potřebných k vytvoření scény.

Dále hledá již existující nástroje, které by pokrývaly požadavky. Po neúspěšném nálezu navrhuje a později implementuje nástroj nový. Nechybí zde ani testování nástroje z hlediska výkonu.

Klíčová slova: OpenGL, skládání transformací, 3D scéna, vizualizace, interaktivní aplikace

Abstract

This bachelor thesis focuses on tools for visual compositing of 3D scene transformations. It lists the requirements for a tool that would be able to bring computer graphics to beginners in this area, including the mathematical objects needed to create the scene.

It also looks for existing tools that would cover the requirements. After an unsuccessful finding, it proposes and later implements a new tool. There is also testing of the tool in terms of performance.

Keywords: OpenGL, composition of transformations, 3D scene, visualization, interactive applications

Poděkování

Děkuji vedoucímu Ing. Jiřímu Jeníčkovi, Ph.D. za cennou pomoct a trpělivost během tvorby této bakalářské práce. Bez jeho směřování a užitečných rad by se výsledek práce na konci mého bakalářského studia nedostal do tak, dle osobního názoru, dobrého stavu, a to i přes nečekané nepříjemné překážky.

Obsah

Seznam obrázků

Seznam zkratek

API	Application Programming Interface
FOSS	Free and Open Source Software
FOV	Field Of View
GO	Game Object
UI	User Interface

1 Úvod

Vizuální skládání transformací 3D scény představuje významnou část v počítačové grafice. Jde o proces umísťování 3D modelů za pomocí transformací jako jsou translace, rotace a škálování. Tyto operace lze za sebe skládat v určitém pořadí a tím tak vytvářet od jednoduchých až po složité scény.

Jedním z hlavních problémů při skládání transformací je právě určení jejich správného pořadí. Jestliže jsou aplikovány v pořadí špatném, může být výsledek nežádoucí a v některých případech i dosti matoucí. Modely mohou být zdeformovány a nebo mezi sebou být nekonzistentní. Takovéto jevy jsou důsledkem vyjádření transformací v 3D grafice maticemi o čtyřech dimenzích, kde poslední sloupec vyjadřuje posunutí po jednotlivých osách: x, y, z. Skládání transformací pak není nic jiného než násobení matic. A protože násobení matic není komutativní operace, je, jak již bylo zmíněno, velmi důležité je aplikovat ve správném pořadí.

S touto a dalšími souvisejícími problematikami týkajících se počítačové grafiky, jako jsou například kvaterniony, se setkává každý, kdo se prvně seznamuje s jejími principy. Proto by bylo vhodné mít k dispozici nástroj pro vizualizaci základních stavebních nebo popřípadě pokročilých technik k sestavování 3D scény.

Tato bakalářská práce se zabývá právě tímto tématem. V první části jsou rozebrány požadavky aplikace pro vizualizaci skládání transformací 3D scény pro začátečníky v oblasti počítačové grafiky vzniklé na základě konzultací s vedoucím práce Ing. Jiřím Jeníčkem, Ph.D.

Ve druhé části jsou rozebírány již existující aplikace, které obsahují implementaci některých požadavků a vypisuje důvody, proč zrovna tyto aplikace nejsou vhodné. Následně dospívá k názoru, že je zapotřebí vytvořit aplikaci novou splňující požadavky. Následuje návrh řešení s několika obrázkovými ilustracemi struktury a vzhledu aplikace. Mezi ilustracemi vzhledu jsou zahrnuty i podstatné bloky částí aplikace jako je například způsob tvorby skládání transformací.

Další dvě části se věnují vlastní implementaci navrženého řešení. Jsou zde popsány zvolené technologie pro vývoj a odůvodnění jejich výběru nad možnými ostatními technologiemi.

Následující kapitola je zaměřena na testování aplikace z hlediska výkonu. Testování bylo prováděno na jediném osobním počítači automatickým skriptem pro tvorbu scény a dalším pro měření průměrných snímků za sekundu. Pro každé testování byly vybrány různé 3D modely, aby byl zachycen výkon nejen z pohledu vykreslování tvorby scény, kdy hraje největší roli rychlost procesoru, ale i z pohledu zátěže grafické karty při zobrazování aplikace skládaných transformací na vybrané modely.

V poslední části se práce zabývá posouzením výsledné aplikace, shrnutím výsledků z testování a dalším možným vývojem.

2 Požadavky

V této kapitole jsou zmíněny všechny požadavky na aplikaci pro vizuální skládání transformací 3D scény, které by měla implementovat, na základě konzultačních hodin s vedoucím práce Ing. Jiřím Jeníčkem, Ph.D.

Především jde o požadavky se zaměřením na nově příchozí do počítačové grafiky. Aplikace by jim měla být schopná znázornit výsledky operací skládaných transformací. Základy v této oblasti tvoří z hlavní části lineární algebra s vektory, transformacemi, kvaterniony a vztahy mezi nimi. Z toho důvodu je matematická stránka aplikace brána s největším důrazem.

2.1 Uživatelské rozhraní

Jelikož transformace jsou tématem ke vstupu do oblasti počítačové grafiky, měla by být aplikace co nejvíce uživatelsky přívětivá a jednoduchá. Neměla by obsahovat složité UI s velkým množstvím tlačítek a dalších vstupů.

Určitě by mělo být rozhraní konzistentní a dobře organizované bez potřeby prozkoumávat funkcionality z přístupu uživatele. Tím se rozumí chybějící klávesové zkratky s více jak dvěma kombinacemi a ovládání aplikace přednostně myší.

Taky byla dohodnuta podmínka tvorba scény v reálném čase. Jestliže uživatel změní hodnoty vstupu ve tvorbě scény, aplikace by měla bez dalšího zásahu okamžitě promítnout tyto změny a aktualizovat svůj stav.

2.2 Matematický obsah

Existují matematické objekty, které jdou ruku v ruce s transformacemi popsány jako matice se čtyřmi dimenzemi, a to jsou skaláry, kvaterniony a třísložkové a čtyřsložkové vektory. V následujících podkapitolách se práce věnuje separátně těmto objektům a přidává ke každému list funkcí, které je rozšiřují nad rámec jejich vlastností a možností.

V 3D grafice se pracuje s číselnými hodnotami s desetinnou čárkou. Je to tedy obor racionálních čísel bez množiny zlomků. Všechny následující matematické objekty jsou tedy vyjádřeny ve svých dimenzích pouze v tomto oboru a žádném jiném. Navíc se v 3D grafice pracuje výhradně v radiánech, nikoliv ve stupních.

2.2.1 Skaláry

Skaláry jsou čistě jednodimenzové objekty. Pro ně by aplikace potřebovala vyjádření s jedním vstupem pro zadání hodnoty.

Skaláry se kromě sčítání, odčítání, násobení a dělení dají ještě dále upravovat funkcemi, jako jsou například trigonometrické funkce, horní a dolní celá část, atp. Pro tyto a další rozšiřující funkce byl sepsán seznam, který by měla aplikace obsahovat:

- Sinus,
- Kosinus,
- Arkus sinus,
- Arkus kosinus,
- Absolutní hodnota,
- Horní celá část,
- Dolní celá část,
- `Clamp` – Funkce definovaná jako:

$$f(x,a,b) = \begin{cases} a, & \text{pokud x} \geq a \\ b, & \text{pokud x} \leq b \\ x, & \text{jinak} \end{cases}$$

- Mocnina,
- Přirozený logaritmus,
- Modulo,
- `Sign` – Funkce definovaná jako:

$$f(x) = \begin{cases} -1, & \text{pokud x} < 0 \\ 0, & \text{pokud x} = 0 \\ 1, & \text{pokud x} > 0 \end{cases}$$

- Lineární interpolace mezi dvěma čísly definovanou jako:

$$f(x,a,b) = a + (b-a) * x$$

2.2.2 Třísložkové vektory

Třísložkové vektory jsou v 3D grafice matematickými objekty se třemi složkami pro popis hodnot na jednotlivých osách (x, y ,z) v souřadnicovém systému. Používají se k vyjádření směru, polohy nebo velikosti objektů. Můžeme nad nimi pracovat s jejich vlastnostmi jako je jejich délka nebo je interpolovat či normalizovat.

Aplikace by tedy měla obsahovat tři hodnoty zabalené do jednoho objektu pro popis třísložkového vektoru a možnost s nimi pracovat nebo je upravovat zmíněnými způsoby. Následně musí umět vektory sčítat, odčítat, násobit, počítat vektorový součin a násobit hodnoty skalárem.

2.2.3 Čtyřsložkové vektory

Čtyřsložkový vektor je velmi podobný třísložkovému až na to, že obsahuje čtvrtou hodnotu pro práci s transformacemi a implementaci perspektivy. Je zde ovšem další patrný rozdíl. Můžeme je násobit s maticemi používané v 3D grafice. Aplikace by toto měla umožňovat společně se stejnými funkcemi jako u třísložkového vektoru.

Po konzultaci ale byla vyřazena možnost zjistit délku vektoru, jelikož se nejedná o důležitou vlastnost ke zjištění. V případě, že by uživatel chtěl tuto vlastnost zjistit, má možnost využít funkce z kapitol 2.3.2 a 2.2.1.

2.2.4 Matice

Nejdůležitějším obsahem jsou afinní matice o velikosti 4x4 k manipulaci transformací. Celá tvorba scény se bude jimi přednostně zabývat. Aplikace musí poskytovat jejich tvorbu ve formě jednoho objektu s šestnácti vstupy pro zadání hodnot. Dále musí obsahovat zvláště tvorbu translační, škálovací a rotační matice zamknutím nepotřebných prvků matice:

- Translační matice – zamknuté všechny vstupy, které neudávají posunutí po osách,
- Škálovací matice – odemknuté pouze vstupy pro prvky matice a_{11}, a_{22} a a_{33},
- Rotační matice – zamknuté vstupy odpovídající translačnímu sloupci a nulovému řádku.

Co se týče transformací, nesmí v aplikaci chybět jejich následující seznam. Ke každé transformaci se bude muset vázat jiný matematický objekt k jejich vytvoření:

- Translace – tvorba pomocí třísložkového vektoru,
- Škálování – stejná tvorba jako u translace (třísložkový vektor),
- Rotace podle os: x, y, z – vstupem takovéto rotace bude čistě skalár,
- Rotace podle osy a úhlu – tvorba třísložkovým vektorem k vyjádření osy a skalárem pro úhel,

- `LookAt` – speciální transformace se třemi třísložkovými vektory jako vstupy. První vektor vyjadřuje polohu, ze které se chceme rozhlížet, druhý polohu, na kterou koukáme a třetí k udání směru vzhůru.

Stejně jako u předchozích matematických objektů nesmí chybět v aplikaci možnost matice sčítat, odčítat, násobit mezi sebou a násobit skalárem.

Další velmi významnou oblastí počítačové grafiky je promítání k definici kamery. Patří mezi ně perspektivní, ortografické a frustum. Všechna tyto promítání se vytvářejí za pomoci několika skalárů. U perspektivního promítání to jsou skaláry pro FOV, aspekt a blízkou a vzdálenou plochu. Ortografické promítání a frustum se od něho liší chybějícím FOV a aspektem. Definují se přes levou, pravou, horní, spodní a vzdálenou a blízkou plochu. Aplikace bude muset umět všechna tyto promítání definovat.

Mezi funkce pracujících nad maticemi nesmí v aplikaci také chybět ty nejdůležitější:

- Transpozice
- Inverze
- Determinant
- Dekompozice – tato funkce je schopná z matice najednou vytáhnout informace o translaci, škálování, rotaci, zkosení a perspektivu.

2.2.5 Kvaterniony

V 3D grafice se k popsání rotace používají kvaterniony namísto Eulerových úhlů. Jedná se o matematickou strukturu s jednou reálnou a třemi imaginárními složkami. Jejich zvolení nad úhly je hlavně z důvodu lepší interpolace mezi rotacemi a zabránění nežádoucím jevům jako je například gimbal lock. Aplikace by měla umět vytvářet kvaterniony, dále s nimi pracovat, upravovat je nebo být schopná zjistit jejich vlastnosti.

Všechny tyto funkce jsou shrnuty v následujícím seznamu:

- Konjugát,
- Inverze,
- Normalizace,
- Délka,
- Lineární interpolace,
- Lineární sférická interpolace.

Kromě předchozích funkcí musí být aplikace schopná násobit kvaterniony mezi sebou a také je násobit skalárem.

2.3 Ostatní požadavky

Mezi ostatní požadavky patří ty, které nejsou zařazeny v hlavních kapitolách 2.2 a 2.1. Jedná se o důležité požadavky aplikace z hlediska ulehčení tvorby scén nebo její funkčnosti nad rámec práce v 3D grafice.

2.3.1 Booleova logika

Určitě by bylo vhodné, kdyby aplikace byla schopná porovnávat hodnoty a s výsledkem porovnání dále pracovat. Proto bylo rozhodnuto, že musí z Booleovy logiky obsahovat následující funkce:

- Logický součet, součin, negace a exkluzivní disjunkce,
- Porovnání skalárů: rovno, nerovno, menší a menší nebo rovno,
- Výběr – Tato funkce má tři vstupy. Prvním vstupem je pravdivá hodnota, podle které se rozhodne, jestli na výstupu funkce bude druhý nebo třetí vstup. Typ druhého a třetího vstupu musí být stejný, tedy lze vybírat jen mezi skaláry, vektory, maticemi a nebo kvaterniony.

2.3.2 Konverze

Protože se jednotlivé matematické objekty o více dimenzích skládají z několika skalárů, měly by v aplikaci existovat mezi nimi navzájem konverze. Například matice lze také vyjádřit ve sloupcích pomocí vektorů nebo Eulerovy úhly převést do kvaternionu a obráceně. V této části jsou zmíněny všechny potřebné konverze k budování scén.

Ze skalárů musí aplikace umět vytvářet po složkách oba typy vektorů, matice a kvaterniony, ze čtyřsložkových vektorů matici a z matice kvaterniony. Poté být schopná převést kvaterniony na Eulerovy úhly nebo na osu s úhlem a speciálně z kvaternionu vytvořit matici. Všechny tyto zmíněné konverze se budou muset vyskytovat v obou směrech.

2.3.3 Animace

Všechny požadavky zmíněné v předchozích kapitolách jsou schopny vytvořit statickou scénu. Pokud by byly implementovány, scéna by nebyla schopná se s časem měnit a animovat objekty. Z toho důvodu bylo v konzultačních hodinách vymyšleno, že by aplikace měla obsahovat nějaký způsob dynamicky upravovat scénu.

Způsobů může být více. Třeba by to mohl být čas od startu aplikace, nebo aktuální čas systému, pod kterým aplikace bude běžet. Hodnota ovšem musí být v podobě skaláru a udávaná v sekundách.

2.3.4 Kamera

Kromě tvorby transformací a jejich následného skládání se v 3D grafice nováčci seznamují s principy kamery. Jakým způsobem se z vertexu umístěným modelační maticí do prostoru, poté přenásobením zleva pohledovou a projekční maticí, stane pixel na 2D obrazovce.

Pro vizualizaci by se velmi hodilo, kdyby aplikace kromě promítání scény uměla promítat další, uživatelem definované, kamery. A pro tyto kamery navíc zobrazovat ve scéně výřez jejich výhledu.

2.3.5 Modely

Výsledek skládaných transformací v 3D scéně se dobře ukazuje na modelech. Aplikace by měla obsahovat některé základní, jako jsou například: kostka, kužel, válec, koule nebo vektor o délce jedna.

Dále by byla příhodná funkce aplikace pro načítání externích souborů s 3D modely. Formát souborů nehraje roli, pokud není proprietární nebo jinak uzavřený pro veřejnost.

2.3.6 Ukládání

Další užitečnou vlastností aplikace musí být ukládání vytvořené scény a její následné načtení. Jestliže toto nebude splňovat, uživatel bude nucen vytvářet stejné scény opakovaně kdykoliv znovu otevře aplikaci, což by mohlo vést k jeho nespokojenosti.

U formátu souboru se scénou bude dávána přednost textovému zápisu oproti binárnímu.

2.3.7 Platforma

Jak bylo zmíněno v kapitole 2.1, aplikace musí být ovladatelná myší a klávesnicí. Podle statcounter.com [1] je v roce 2023 (statistiky dostupné v době publikování baklářské práce) nejvíce používaným desktopovým operačním systémem Windows a na ten by mělo být přednostně mířeno. Nebude záležet, zdali aplikace bude běžet v prohlížeči nebo v nativním kódu.

2.3.8 Otevřenost

Protože, jak již bylo několikrát zmíněno, je hledaná aplikace směřována na začátečníky v oblasti 3D počítačové grafiky, měla by se řadit do FOSS. Takováto vlastnost je chtěná hlavně z toho důvodu, že se začátečníci potřebují seznámit s vnitřním chováním softwaru pracujícím s funkcemi dostupnými pro 3D grafiku. V otevřeném zdrojovém kódu by tak mohli chtít přejmout některé funkcionality aplikace pro svoje potřeby. Jedním takovým příkladem je již zmiňovaná definice kamery v kapitole 2.3.4 a nebo cesta od definice scény až po její vykreslení.

3 Existující aplikace

V rámci této bakalářské práce byl proveden průzkum existujících nástrojů pro práci s 3D grafikou, včetně nástrojů pro skládání transformací 3D scény. Nalezené nástroje obsahují široké množství funkcí. Tato kapitola se jim věnuje, vysvětluje, které požadavky neimplementují, z jakých důvodů nejsou vhodné pro začátečníky v oblasti 3D grafiky, a následně dospívá k názoru, že je zapotřebí vytvořit nový nástroj.

3.1 Blender

Blender je velmi rozšířený a populární nástroj splňující podmínku otevřenosti, FOSS. Lze v něm modelovat, animovat, skládat vizuální scénu, aplikovat textury na modely a obecně lze považovat za profesionální nástroj v oblasti počítačové grafiky [2].

Splňuje požadavky definování kamery, ukládání scény, načítání externích modelů nebo vytváření vlastních a jejich animování. Bohužel, co se týče obsahu po matematické stránce, Blender postrádá skoro všechny požadavky. Sice lze modely umístit do scény, nastavit jim translaci, škálování a rotaci (u rotace je schopný dokonce využít kvaterniony), ale už je není schopen zobrazit v jejich podobě matic.

Dále nabízí hierarchii modelů. Používá kolekce k jejich seskupení s možností celou kolekci posouvat ve scéně jako jeden objekt. Chybí zde ovšem možnost celou kolekci otočit nebo škálovat.

Předchozí nedostatky by šly vyřešit přidáním rozšíření do Blenderu. Rozšíření umožňují do aplikace přidávat další funkce a možnosti. Vezmeme ale v potaz UI, pro někoho, kdo prvně otevře aplikaci Blender, může na něho působit neintuitivně a složitě. UI obsahuje spoustu tlačítek a kontextových nabídek.

Například na obrázku v příloze A.1 je v levé části vidět nabídka s patnácti sekcemi pro jediný model. Určitě se nedá považovat podle požadavků z kapitoly 2.1 za intuitivní a jednoduchou.

Z důvodu chybějících požadavků se nedá Blender považovat za vhodný nástroj.

3.2 Unity

Unity patří mezi nejpopulárnější herní nástroje k vytváření her podle steamdb.info [3]. Zdrojový kód není sice otevřený, ale Unity má bezplatný plán pro osobní použití a studenty [4].

Kromě her může sloužit také v automobilovém, výrobním nebo filmovém průmyslu. Jeho využití nechybí ani v simulačních oblastech a rozšířené realitě [5]. Oproti aplikaci Blender v něm lze jednodušeji skládat scénu. Pracuje s pojmem Game Object (GO), abstraktním herním objektem, s možností obsahovat 3D modely. Tyto objekty jsou schopny mít další objekty jako potomky a tím tak vytvářet od jednoduchých až po složité scény. Pokud se aplikuje nějaká transformace na rodiče, všechny jeho potomci automaticky tuto transformaci přebírají, což je očekávané chování.

Přestože u GO nechybí možnost nastavení pozice, škálování a rotace, jejich maticové vyjádření Unity z pohledu uživatele zcela postrádá, a to i ve skriptovacím jazyce, kde je možné všem těmto vlastnostem přiřadit hodnoty. Také zde úplně chybí možnost mezi sebou konvertovat matematické objekty.

UI je zde sice o něco přehlednější než u Blenderu, ale i tak je velmi patrné, že vývojáři dali přednost především hernímu průmyslu. Tato skutečnost se v rozhraní odráží: obsahuje přílož mnoho funkcí pro nováčky začínající v počítačové grafice. Proto s předešlými chybějícími funkcemi požadovaných z kapitoly 2 není Unity přijatelnou aplikací.

3.3 Unreal Engine

Unreal Engine patří stejně jako Unity k herním vývojářským nástrojům. Vyznačuje se širokou mírou přenositelnosti, protože podporuje širokou škálu platforem pro stolní počítače, mobilní, konzolová zařízení a virtuální realitu [6].

Pomocí schémovacího nástroje Blueprint v něm lze vytvářet hry bez nutnosti znát programovací jazyk [7]. Uživatel propojuje logické bloky ve výsledku tvořící grafovou strukturu. Takovýto systém je přímo perfektní pro vizuální skládání transformací 3D scény.

Blueprint implementuje všechny požadavky týkajících se matematického obsahu z kapitoly 2 a to včetně kvaternionů, Booleavy logiky, definici kamery a konverze mezi matematickými objekty. Dalo by se říci, že se jedná o hledanou aplikaci, kdyby ovšem zdrojový kód nebyl uzavřený. Nově příchozí do počítačové grafiky pak nemají přístup k funkcím používaných Unreal Enginem a nemohou se tak podívat, jakým způsobem je zajištěno vykreslování scény. Navíc bloky s operacemi, jako je třeba násobní matic, neukazují jejich výsledek, což není přívětivé pro snahu pochopit aplikaci transformací.

Bohužel i Unreal Engine má složité UI. Obsahuje spoustu menu a tlačítek, která se nijak nepojí s transformacemi. Určitě se nedá považovat za přívětivé pro nové uživatele.

Z těchto důvodů se nedá považovat Unreal Engine jako hledané řešení.

3.4 Online nástroje

Existují další nástroje pro vizuální skládání transformací a to jsou online nástroje. Běží v dnes optimalizovaných prohlížečích, což přináší několik výhod. Jednou takovou výhodou je zbavení se závislosti na operačních systémech anebo potřeba

instalovat jakékoliv programy nebo podpůrné knihovny. Existujícími nástroji jsou například: Clara.io (clara.io) a Vectary (vectary.com).

Clara.io je zaměřená spíše na editaci modelů a připomíná z velké části Blender. Zvládá umisťovat modely do scény, otáčet a škálovat je. Avšak postrádá z požadavků větší množství. Úplně zde chybí vektory, matice, Booleova logika, konverze a kvaterniony.

Vectary je skvělým nástrojem pro tvorbu scény. Hodí se pro demonstrační účely návrhů 3D scén. Co se týče požadavků, je na tom z předchozích zmíněných nástrojů nejhůře. Z transformací dovoluje akorát posunutí, rotaci a škálování. Skládání zde úplně chybí.

Zmíněné online nástroje jsou sice uživatelsky přívětivé, neobsahují složité UI, ale ani z poloviny nepokrývají množství požadavků. Nejsou tak vhodným hledaným nástrojem.

3.5 Placený software

Další oblastí prozkoumávaných nástrojů je placený software. Hned zkraje je nutné zmínit, že nesplňují požadavek otevřenosti z kapitoly 2.3.8. Přesto stojí některé za uvedení.

Autodesk Maya a 3ds Max patří mezi profesionální nástroje pro 3D modelování, animace, simulace a vizuální efekty [8], [9]. Jsou používány hlavně ve filmovém průmyslu [10], [11]. Pro účely začínajících v 3D grafice jsou ovšem přehnaně mocným, složitým a náročným nástrojem. Pokud někdo začíná v počítačové grafice, určitě vizuální efekty a simulace nejsou v jeho zájmu.

3.6 Závěr průzkumu existujících aplikací

Z průzkumu existujících aplikací pro vizuální skládání transformací 3D scény vyplívá, že většinu nástrojů nelze zdaleka považovat za ideální aplikace implementující požadavky. Ze všech je nejblíže Unreal Engine, avšak i ten nebyl shledán jako ideální řešení. Proto je potřeba vytvořit aplikaci novou, takovou, aby splňovala žádané požadavky.

V následujících kapitolách probíhá rešerše vhodných technologií pro tvorbu ideální aplikace, její návrh a následná implementace.

4 Návrh řešení

Tato kapitola se věnuje prvnímu návrhu nové aplikace. Je rozdělena na dvě části. První popisuje uživatelské rozhraní včetně tvorby scény a druhá celkový vzhled aplikace.

4.1 Návrh UI

Jedním z hlavních cílů nové aplikace byla podmínka jednoduchého UI. Uživatel by neměl být nucen prozkoumávat jeho funkcionality, protože rozhraní by mělo být intuitivní a snadno ovladatelné.

I když se jedná o bakalářskou práci napsanou v češtině, jazykem aplikace byla po konzultaci zvolená angličtina. Je to z toho důvodu, že bylo plánováno psát zdrojový kód také v angličtině. Jedná se spíše o konvenci.

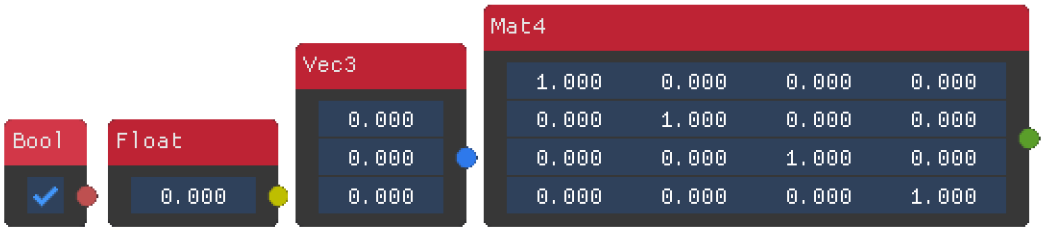
4.1.1 Systém uzlů

S předchozími požadavky a požadavky zmíněné v kapitole 2.1 byl pro tvorbu scény navržen systém propojitelných uzlů.

Systém uzlů je velmi podobný Blueprintu v Unreal Enginu. Jde o interaktivní princip tvorby schémat. Každý uzel má různý počet vstupů a výstupů o jasně předem definovaném typu. Uživatel je poté schopen myší tyto uzly propojovat, řadit je za sebe a bez potřeby umět programovat vytvářet schémata v podobě stromů. Tento systém je extrémně flexibilní a schopný návrhářům poskytovat širokou míru funkcionalit [7].

V rámci bakalářské práce byl pro matematické objekty a funkce navržen vzhled uzlů. Na Obrázku 4.1 jsou zleva vyzobrazeny uzly pro pravdivou hodnotu, skalár, vektor a matici. V ilustraci chybí z matematických objektů čtyřsložkový vektor a kvaternion. Jejich podoba odpovídá uzlu s třísložkovým vektorem s tím, že ve sloupci obsahuje čtyři namísto tří hodnot.

Jak je vidět, každý uzel má v nadpisu svoje písemné označení a v těle číselné vstupy pro zadání hodnot. Výjimkou je pravdivá hodnota, kde je nahrazeno desetinné číslo zaškrtávátkem. Následně na pravém okraji mají uzly výstup odpovídající jejich představované hodnotě. Typ výstupu, tedy skalár, vektor atp., je odlišen barvou, aby uživateli bylo jasné, které vstupy a výstupy lze mezi sebou propojovat. Na Obrázku 4.2 je vidět podoba vstupu v uzlu s funkcí sinus. Vstup vypadá stejně jako výstup, jen je umístěn na levé straně.

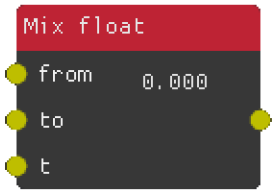


Obrázek 4.1: Návrh uzlů

Obrázek 4.2: Uzel s funkcí sinus

U Obrázku 4.2 stojí za povšimnutí zmínit, že u uzlů se vstupy není možné ručně změnit jejich hodnoty. Tato skutečnost je uživateli předána v podobě chybějícího modrého pozadí u hodnoty desetinného čísla. Takovéto chování je navržené záměrně. Uživatel by neměl mít možnost upravit výstup jakékoliv funkce.

Pro orientaci ve vstupech jsou u vybraných uzlů přidány popisy. Například na u uzlu na Obrázku 4.3 s lineární interpolací skalárů by chybějící popisky vytvářely zmatek.

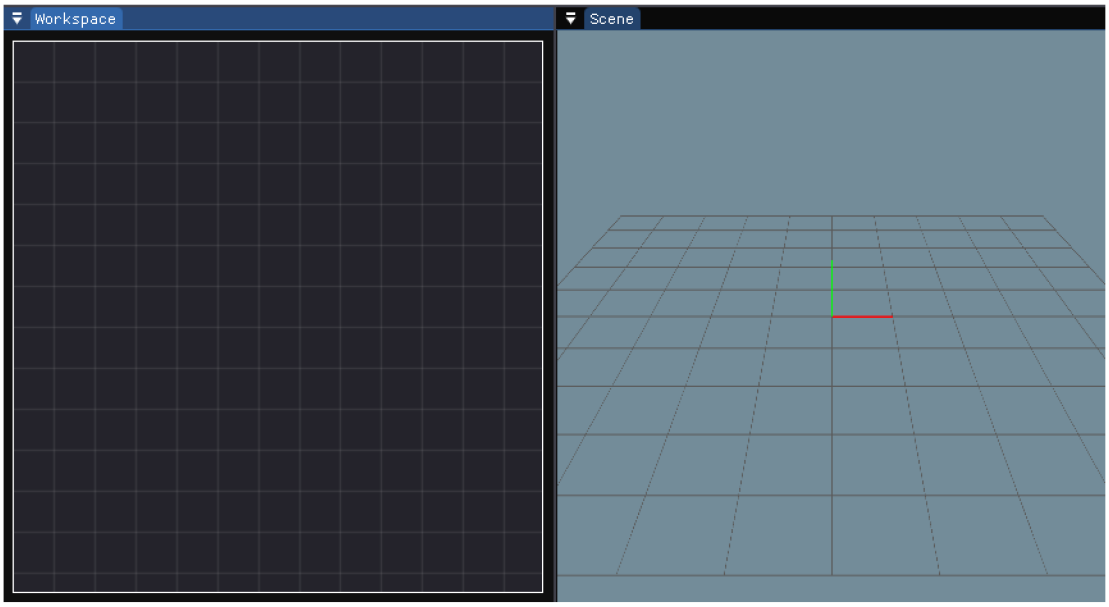


Obrázek 4.3: Caption

4.1.2 Vzhled aplikace

Jelikož má být aplikace interaktivní a provedené změny ve tvorbě scény okamžitě promítnuty, bylo navrženo rozložení aplikace na dvě části. V jedné části bude uživatel schopen upravovat scénu, přidávat další modely, nastavovat jim transformace, a ve druhé bude mít možnost prohlížet vytvořenou scénu. Na Obrázku 4.4 je přesně takové rozvržení ilustrováno.

Oblast s označením „Workspace“ bude sloužit pro tvorbu scény. Zde si bude moci uživatel přes kontextové menu vyvolávané pravým tlačítkem myši definovat scénu prostřednictvím již zmíněných uzlů a v oblasti s názvem „Scene“ prohlížet její výsledek. Do scény byla navíc přidána mříž s jednotkovým rozestupem pro lepší vyznání se v prostoru.



Obrázek 4.4: Rozvržení aplikace

Po konzulacích s vedoucím práce Ing. Jiřím Jeníčkem, Ph.D. byl návrh odsouhlasen a připraven k implementaci.

5 Nástroje pro vývoj

Před implementací bylo ještě zapotřebí nalézt vhodné technologie a nástroje pro tvorbu aplikace.

Prvním krokem bylo rozhodnutí v jakém jazyce bude psána aplikace. Protože k tvorbě bakalářské práce byla čerpána látka z odborné literatury týkající se počítačové grafiky z knih OpenGL SuperBible [12] a OpenGL Programming Guide [13], které obsahují zdrojový kód popisující API OpenGL v jazycích C a C++, byl nakonec zvolen programovací jazyk C++. Další výběr knihoven a technologií potřebných k vývoji aplikace bylo odváděno od tohoto jazyka.

Technologie pro vykreslování byla zvolená stejným způsobem jako programovací jazyk. Odborná literatura se věnuje OpenGL. Jedná se o multiplatformní API pro vykreslování 2D a 3D grafiky.

Sice bylo možné si napsat celou aplikaci kompletně od nuly, ale vzhledem k tomu, že C++ je jazyk skoro čtyřicet let starý [14], byla vybrána množina existujících podpůrných knihoven.

Jedním takovým příkladem podpůrné knihovny je GLFW (www.glfw.org). Ta umožňuje v operačních systémech vytvářet okna a odchytávat uživatelský vstup. Společně s knihovnou GLEW, která poskytuje rozšíření pro OpenGL API [15], tvoří technologii potřebnou k vývoji okenní aplikace se schopností vykreslovat obrazce [16]. Kromě GLFW pro tvorbu oken existuje celá řada náhrad jako jsou například freeGLUT nebo GLAD (www.freeglut.sourceforge.net, www.github.com/Dav1dde/glad. Všechny mají stejný účel: Vytvořit okno a poskytnout API pro jeho další manipulaci. Není mezi nimi velký rozdíl.

Výběr matematické knihovny byl celkem přímočarý. The Khronos Group publikuje standardy v oblasti počítačové grafiky včetně OpenGL a poskytuje podpůrnou knihovnu pro matematiku GLM [17].

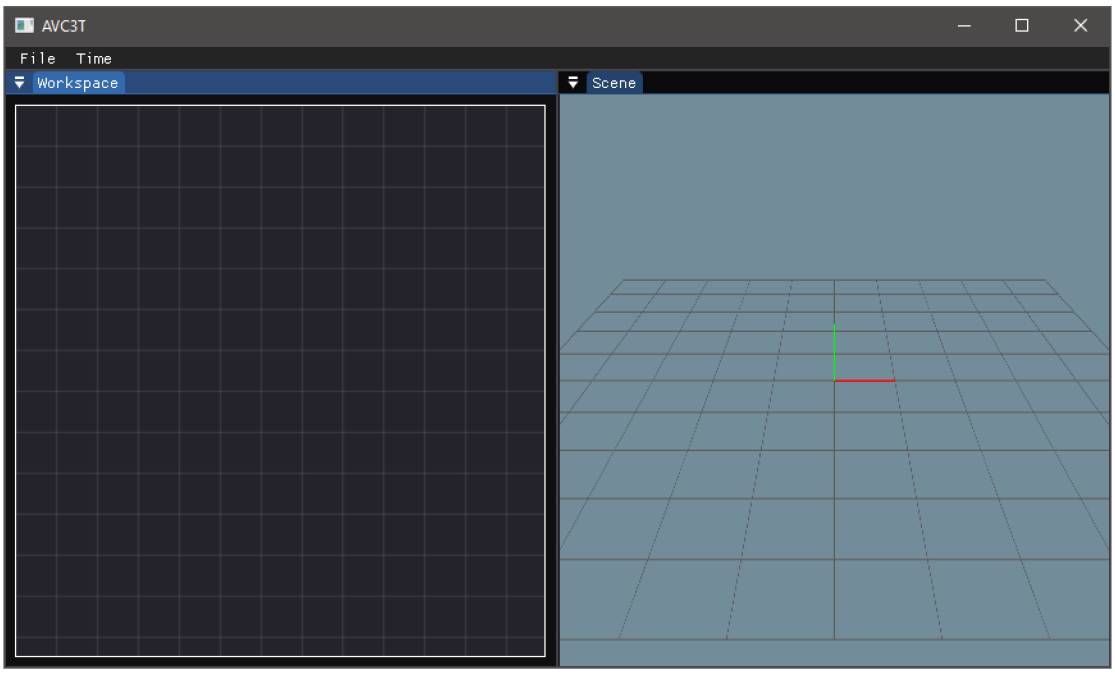
Aby aplikace splňovala požadavky načítání externích modelů (zmíněno v kapitole 2.3.6), byla vybrána knihovna Assimp. Tato knihovna je schopná načíst přes 40 formátů 3D souborů do paměti počítače [18], což byl hlavní důvod k jejímu zvolení, aby mohla být aplikace flexibilní při načítání souborů.

V neposlední řadě bylo zapotřebí vybrat knihovnu pro návrh UI. Mezi známé patří například Qt, GTK, wxWidgets a Dear ImGui (www.qt.io, www.gtk.org, www.wxwidgets.org, www.github.com/ocornut/imgui). Všechny podporují tvorbu tlačítek, kontextových menu, vstupů pro desetinná čísla a zaškrtávátek. Po jejich přezkoumání byla vybrána knihovna Dear ImGui (dále jen ImGui) a to hlavně z důvodu přítomných ukázek v repozitáři zdrojového kódu a jednoduchosti vykreslovat vlastní obrazce.

Tímto jsou shrnuty všechny knihovny, které byly použity pro tvorbu nové aplikace pro vizuální skládání transformací 3D scény. Pro zmíňku je důležité říci, že spluňují požadavek otevřenosti z kapitoly 2.3.8.

6 Implementace

Se schváleným návrhem již zbývala implementace požadavků. V konečné implementaci vypadá aplikace znázorněná na Obrázku 6.1. Při horním okraji je menu pro ukládání a načítání scén a další menu s názvem „Time“ pro ovládání animací. Levá část obsahuje místo pro tvorbu scény a pravá zobrazuje vytvořenou scénu.



Obrázek 6.1: Konečný vzhled aplikace

6.1 Vykreslování obou částí

Protože část „Workspace“ slouží k tvorbě scény, je k jejímu vykreslování použita knihovna ImGui. Ta umožňuje otevírat kontextová menu po kliknutí myší pravým tlačítkem. V této nabídce se podle kliknutého uživatelského prvku (prázdno, uzel nebo spoj uzlu) zobrazí příslušná nabídka.

Zobrazování druhé části „Scene“ je zajištěno přes OpenGL API kreslením do textury. Textura v kontextu OpenGL má přidělen svůj identifikátor, který se poté předloží ImGui a ta společně s přiloženou pozicí umístí texturu na obrazovku. Stejným způsobem funguje vykreslování uživatelem definovaných kamer.

Aby byl uživatel schopen prohlédnout scénu z jiných směrů než je ten na Obrázku 6.1, je mu umožněna pohyba kamery za pomocí myši. Kamera se dá otáčet a posouvat v prostoru.

Obě části jsou v programu aktualizovány ve smyčce a jediným elementem, který omezuje počet snímků aplikace je prohazování kreslícího a zadního bufferu knihovnou GLFW.

6.2 Systém uzlů

Protože počet uzlů je kvůli požadavkům v desítkách, je na systém kladen nárok snadného rozšíření. Toto se odráží i v implementaci. Pro jeden uzel zdrojový kód pokrývá nejčastěji 60-70 řádků a část z toho je navíc generována makrem. Seznam uzlů odpovídajících matematickému obsahu včetně popisu jejich vstupů a výstupů lze nalézt v přílohách: A.2, A.3, A.4, A.5, A.6, A.7 a A.8.

Ve formě uzlů byly implementovány i další požadavky jako je Booleova logika, konverze mezi matematickými objekty, kamera a modely externí či vnořené. Každý uzel má příslušné vstupy pro jejich ovládání a/nebo výstupy z funkcí.

Uzel kamery obsahuje dva vstupy. První pro pohledovou a druhou pro projekční matici.

Speciálně animace jsou zajištěny ovladatelným časem od spuštění aplikace. Do tvorby scény lze přidat uzel s časem a dobou od poslední aktualizace aplikace. Čas je dále možné zastavovat, vynulovat a spouštět tlačítky v hlavní liště aplikace.

Každý uzel je kreslen zvlášť ImGui přes volání jednoduchých geometrických obrazců jako jsou čtverec nebo kruh.

6.3 Ukládání scény

Jazyk C++ není jazykem s reflexí. Vytvořené progamy jsou přeloženy do konkrétních instrukcí architektury počítače. Z toho důvodu nelze instance objektů jednoduše deserializovat a poté zpět serializovat jako je tomu třeba u jazyka Java. Proto byl navrhnut a implementován vlastní textový formát souborů se scénami pojmenovaný jako „avc3t“. Na následujícím příkladě uložené matice je vysvětlena jeho struktura:

```
WorkspaceNodeMat4Value:
Id: 1
Position: 516;437
Outputs:
    Output:
        Value: -0.5;1.5;-1.5;0.5;1;-2.5;2;-0.5;-0.5;0;0.5;0;0;1;0;0
        Connections:
            Count: 1
            Connection:
                Id: 2
                Input index: 1
```

V prvním řádku je název třídy představující uzel. Na dalších řádcích následuje číselný identifikátor, pozice, hodnota výstupu a na něho napojené další uzly. V tomto případě má výstup připojen pouze jeden uzel s identifikátorem jedna na první vstup.

Číselné hodnoty vlastností uzlu (pozice či výstupy) jsou deserializovány do řádku oddělené středníkem.

6.4 Modely

Modely je možné načítat ve formátu .obj přes knihovnu Assimp. Je zde bohužel jedno omezení. Aby je aplikace byla schopná načíst, musí obsahovat texturu. Realizace načítání je provedena za pomocí uzlu s řetězcovým vstupem pro zadání cesty k souboru. V případě, že uživatel chce uložit scénu včetně modelu, má možnost přes tlačítko vedle cesty zapéct model, tím se tak zapeče do tvorby scény a při uložení se zapíše do souboru.

Kromě externích modelů aplikace poskytuje některé základní jako jsou: kostka, kužel, válec, koule a objekt ve tvaru jednotokového vektoru.

7 Testování

Tato kapitola je zaměřena na testování vytvořené. Testoval se její výkon pouze ve dvou odlišných scénářích na osobním počítači s operačním systémem Windows 10, grafickou kartou NVIDIA RTX 3070 a procesorem AMD Ryzen 5 5600X. V každém scénáři byl vždy vybrán jen jeden 3D model, který se poté duplikoval společně s počtem uzlů.

Všechny testy byly vytvořené automatickým skriptem pro návrh scény a dalším pro měření průměrných snímků za sekundu. Scény se generovaly náhodně a měření snímků se průměrovalo ze třicetisekundového intervalu.

7.1 Scénář s malým modelem

V prvním scénáři byl zvolen 3D model s 2000 vertexy. Mělo by tak dojít k tomu, že volání OpenGL API sloužící k jeho vykreslování bude při větší zátěži aplikace probíhat častěji než u modelu většího.

V Tabulce 7.1 je vidět výsledek testu prvního scénáře, kdy byly vykreslovány všechny uzly. V levém sloupci jsou vyčteny počty modelů a v prním řádku počty vygenerovaných uzlů. Hodnoty v tělu tabulky pak odpovídají počtům průměrných snímků.

Snímky	**10**	**512**	**1024**	**2048**	**4096**
10	144	105	51	24	12
128	144	98	50	24	12
1024	120	55	37	20	11
2048	60	38	28	17	10
4096	30	22	19	13	8

Tabulka 7.1: Počty snímků v prvním scénáři

Z tabulky lze vyčíst, že při deseti uzlech a exponenciálně roztoucím počtem modelů (vyjma testu s 10 a 128 modely) exponenciálně klesá i počet snímků. Toto je dobrý výsledek, protože to vyznačuje rychlost běhu aplikace při nezávislosti počtu modelů na stálý počet uzlů. Obdobně tomu tak je při deseti modelech a roztoucím počtu uzlů.

Při zahlcení aplikace nadměrným počtem uzlů nemá přidávání dalších modelů velký vliv. To je ostatně vidět na hodnotách v posledním sloupci, kde se snímky drží velmi blízko sobě v porovnání s prvním sloupcem.

Následující Tabulka 7.2 odpovídá výsledkům testů stejného scénáře, ale tentokrát bez vykreslování uzlů, byly posunuty mimo oblast „Workspace“.

Snímky	10	512	1024	2048	4096
10	144	144	98	49	23
128	144	144	97	47	23
1024	144	109	70	37	21
2048	118	73	55	32	20
4096	57	41	35	25	15

Tabulka 7.2: Počty snímků v prním scénáři

V porovnání s předešlou tabulkou si lze všimnout skoro dvojnásobných hodnot. Za tento rozdíl může způsob vykreslování uzlů. Dear ImGui, které je vykresluje, používá pro grafické zobrazení procesor namísto grafické karty.

7.2 Scénář s velkým modelem

Ve druhém scénáři byl zvolen 3D model o 1 000 000 vertexech. Oproti prvnímu scénáři by mělo dojít k většímu zatížení grafické karty. Výsledky testů druhého scénáře jsou vyzobrazeny v Tabulce 7.3. I když model má o tisíc více vertexů, hodnoty nejsou oproti Tabulce 7.1 třetinové. Zřejmě grafická karta optimalizuje vykreslování.

Snímky	10	512	1024	2048	4096
10	48	35	17	9	4
128	48	33	17	8	3
1024	41	19	12	7	3
2048	18	12	9	5	3
4096	9	7	5	4	2

Tabulka 7.3: Počty snímků v prním scénáři

Stejně jako u prvního scénáře se jedná o test, kdy byly vykreslovány všechny uzly. Když byl proveden stejný test bez vykreslování uzlů, počty snímků opět skoro dosáhly dvojnásobku, stejně jako u předchozího scénáře.

8 Shrnutí

Tato bakalářská práce měla jako hlavní cíl provést rešerši vhodných technologií pro vizuální návrh scény pomocí skládání transformací do stromu scény a rešerši pro vhodnou podmnožinu grafických transformací pro realizaci. Dále měla implementovat aplikace a otestovat ji z hlediska výkonu.

Nejdříve na základě konzultačních hodin s vedoucím práce Ing. Jiřím Jeníčkem, Ph.D. byly sesbírány požadavky na aplikaci. Některé se týkaly uživatelského rozhraní, některé matematického obsahu a zbylé obsahu pro ulehčení práce s takovou aplikací. Požadavků zahrnujících matematický obsah bylo výrazně více, což bylo očekáváno, protože transformace (hlavní téma rozsahu bakalářské práce) patří do oblasti lineární algebry.

Po sběru požadavků následovala rešerše existujících aplikací a nástrojů pro skládání transformací 3D scény. U většiny nástrojů byly shledány velké mezery v implementaci. Největší pokrytí požadavků měl Unreal Engine, vývojářské herní studio. Avšak jeho zdrojový kód není otevřený a z toho důvodu ho také nejde považovat za hledaný nástroj.

Práce pokračovala návrhem aplikace včetně ilustračních obrázků. Po konzultaci byl tento návrh schválen a mohla probíhat další, tentokrát kratší, rešerše technologií vhodných pro implementaci. Během rešerše byly nalezeny knihovny pro tvorbu aplikace a pro manipulaci s matematickými objekty.

Předposlední část bakalářské práce se věnovala samotnému vývoji aplikace. Podařilo se naplnit všechny požadavky z konzultačních hodin a dotáhnout aplikaci do použitelného stavu.

Na závěr byla aplikace podrobena testům, které se skládaly z několika druhů. Některé testy byly zaměřené na velké množství uzlů, jiné na vykreslování 3D modelů s velkým množstvím vertexů. Z výsledků testů vyplynulo, že při zátěži způsobené nadměrným množstvím uzlů, přidávání dalších 3D modelů do scény nemá význačný vliv na chod aplikace.

8.1 Další vývoj

Další významnou oblastí počítačové grafiky je stínování. Používá se k osvětlování modelů, aby nevypadaly plasticky. Jedním známým stínováním je například techniku Phongovo stínování používající interpolaci normál povrchu a Phongův způsob odrážení světla k osvětlení modelů [19].

Aplikace by mohla v dalším vývoji být schopna definovat techniky stínování a ty

poté následně vyzobrazovat do scény. Také by se hodilo ulehčení skládání transformací. Aktuálně je potřeba ke složení n transformací $n - 1$ operátorů. Dal by se navrhnout objekt, do kterého by se za sebe srovnaly matice s transformacemi bez nutnosti vytvářet uzly s operacemi pro násobení a zjednodušit tvorbu scény.

Použitá literatura

[1] *Desktop Operating System Market Share Worldwide: Apr 2022 - Apr 2023* [online]. Ireland: Statcounter, 2023 [cit. 2023-05-02]. Dostupné z: https://gs.statcounter.com/os-market-share/desktop/worldwide.

[2] *Blender* [online]. Amsterdam: Blender Foundation, 2023 [cit. 2023-05-20]. Dostupné z: https://www.blender.org/.

[3] *SteamDB* [online]. Djundik, 2023 [cit. 2023-05-20]. Dostupné z: https://steamdb.info/tech/.

[4] *REAL-TIME SOLUTIONS, ENDLESS OPPORTUNITIES* [online]. San Francisco: Unity Technologies, 2023 [cit. 2023-05-20]. Dostupné z: https://unity.com/pricing#plans-student-and-hobbyist.

[5] *Unity* [online]. San Francisco: Unity Technologies, 2023 [cit. 2023-05-20]. Dostupné z: https://unity.com/solutions.

[6] *Unreal Engine* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2023-05-21]. Dostupné z: https://en.wikipedia.org/wiki/Unreal_Engine.

[7] *Blueprints Visual Scripting* [online]. Cary (North Carolina): Epic Games, Inc., 2023 [cit. 2023-05-20]. Dostupné z: https://docs.unrealengine.com/5.2/en-US/blueprints-visual-scripting-in-unreal-engine/.

[8] *Autodesk Maya* [online]. San Francisco: Autodesk Inc., 2023 [cit. 2023-05-20]. Dostupné z: https://www.autodesk.com/products/maya/overview.

[9] *Autodesk 3ds Max* [online]. San Francisco: Autodesk Inc., 2023 [cit. 2023-05-20]. Dostupné z: https://www.autodesk.com/products/3ds-max/overview.

[10] *List of films made with Autodesk 3ds Max* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2023-05-20]. Dostupné z: https://en.wikipedia.org/wiki/List_of_films_made_with_Autodesk_3ds_Max.

[11] *Autodesk Maya* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2023-05-20]. Dostupné z: https://en.wikipedia.org/wiki/Autodesk_Maya.

[12] *OpenGL SuperBible*. 4th. Michigan: Pearson Education, Inc, 2007. ISBN 978--0-321-49882-3.

[13] *OpenGL Programming Guide*. 5th. Courier in Stoughton (Massachusetts): Shreiner, 2005. ISBN 0-321-33573-2.

[14] *C++* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2023-05-21]. Dostupné z: https://en.wikipedia.org/wiki/C%5C%2B%5C%2B.

[15] *The OpenGL Extension Wrangler Library* [online]. Beaverton: The Khronos Group Inc, 2016 [cit. 2023-05-21]. Dostupné z: https://www.opengl.org/sdk/libs/GLM/.

[16] *GLFW* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2023-05-22]. Dostupné z: https://en.wikipedia.org/wiki/GLFW.

[17] *OpenGL Mathematics* [online]. Beaverton: The Khronos Group Inc, 2016 [cit. 2023-05-21]. Dostupné z: https://www.opengl.org/sdk/libs/GLM/.

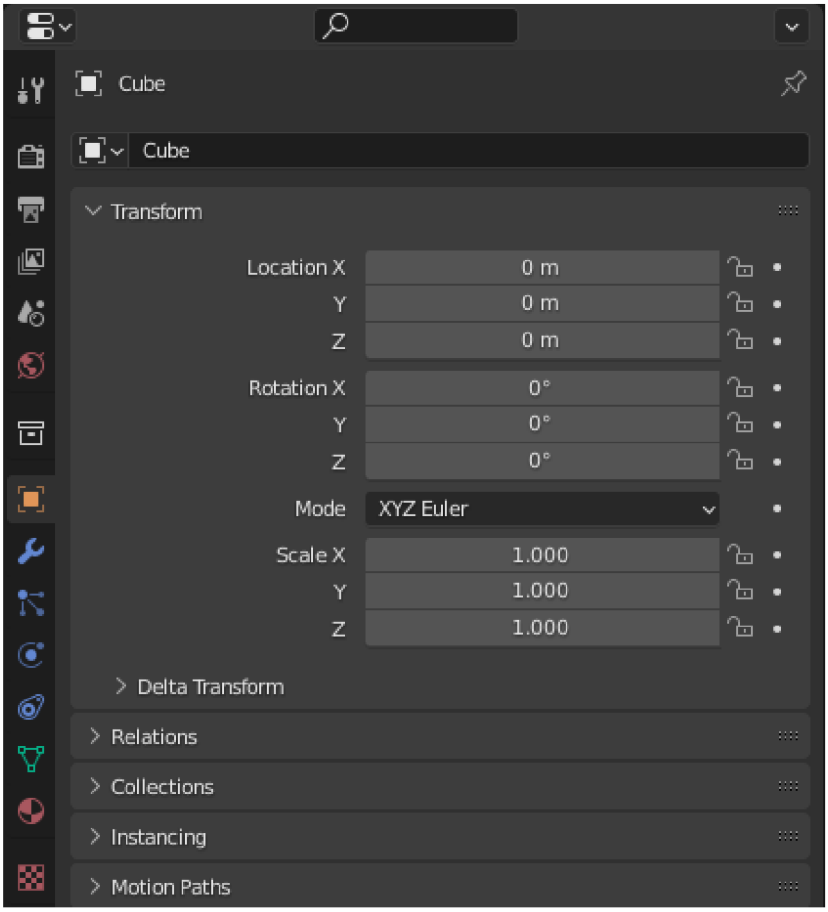
[18] *THE ASSET IMPORTER LIBRARY* [online]. Německo: Vilmring, 2023 [cit. 2023-05-21]. Dostupné z: https://www.assimp.org/.

[19] *Phong shading* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2023-05-22]. Dostupné z: https://en.wikipedia.org/wiki/Phong_shading.

[20] *OpenGL* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2023-05-22]. Dostupné z: https://en.wikipedia.org/wiki/OpenGL.

A Přílohy

A.1 Možnosti modelu v aplikaci Blender



Obrázek A.1: Blender - možnosti modelu

A.2 Definice uzlů

Mat. objekt	Název uzlu	Typy vstupů	Typy výstupů
Skalár	Float	–	Skalár
Třísložkový vektor	Vec3	–	Třísložkový vektor
Čtyřsložkový vektor	Vec4	–	Čtyřsložkový vektor
Matice	Mat4	–	Matice
Kvaternion	Quat	–	Kvaternion

Tabulka A.1: Definice uzlů s matematickými objekty

Funkce	Název uzlu	Typy vstupů	Typy výstupů
Součet	Float + Float	Skalár, skalár	Skalár
Rozdíl	Float - Float	Skalár, skalár	Skalár
Součin	Float * Float	Skalár, skalár	Skalár
Podíl	Float / Float	Skalár, skalár	Skalár
Sinus	Sin	Skalár	Skalár
Kosinus	Cos	Skalár	Skalár
Arkus sinus	Asin	Skalár	Skalár
Arkus kosinus	Acos	Skalár	Skalár
Absolutní hodnota	Abs	Skalár	Skalár
Horní celá část	Ceil	Skalár	Skalár
Dolní celá část	Floor	Skalár	Skalár
Clamp	Clamp	Skalár	Skalár
Mocnina	Float ∧ Float	Skalár, skalár	Skalár
Přirozený logaritmus	Ln	Skalár	Skalár
Modulo	Mod	Skalár, skalár	Skalár
Sign	Sign	Skalár	Skalár
Lineární interpolace	Mix float	Skalár, skalár, skalár	Skalár

Tabulka A.2: Definice uzlů se skalárními funkcemi

Funkce	Název uzlu	Typy vstupů	Typy výstupů
Součet	Vec3 + Vec3	Třísložkový vektor, třísložkový vektor	Třísložkový vektor
Rozdíl	Vec3 - Vec3	Třísložkový vektor, třísložkový vektor	Třísložkový vektor
Součin	Vec3 * Vec3	Třísložkový vektor, třísložkový vektor	Třísložkový vektor
Skalární součin	Vec3 . Vec3	Třísložkový vektor, třísložkový vektor	Skalár
Násobení skalárem	Float * Vec3	Třísložkový vektor, skalár	Třísložkový vektor
Normalizace	Normalize Vec3	Třísložkový vektor	Třísložkový vektor
Délka vektoru	Length Vec3	Třísložkový vektor	Třísložkový vektor
Lineární interpolace	Mix Vec3	Třísložkový vektor, třísložkový vektor, skalár	Třísložkový vektor

Tabulka A.3: Definice uzlů s funkcemi upravující třísložkové vektory

Funkce	Název uzlu	Typy vstupů	Typy výstupů
Součet	Vec4 + Vec4	Čtyřsložkový vektor, čtyřsložkový vektor	Čtyřsl. vektor
Rozdíl	Vec4 - Vec4	Čtyřsložkový vektor, čtyřsložkový vektor	Čtyřsl. vektor
Součin	Vec4 * Vec4	Čtyřsložkový vektor, čtyřsložkový vektor	Čtyřsl. vektor
Skalární součin	Vec4 . Vec4	Čtyřsložkový vektor, čtyřsložkový vektor	Skalár
Násobení skalárem	Float * Vec4	Čtyřsložkový vektor, skalár	Čtyřsl. vektor
Normalizace	Normalize Vec4	Čtyřsložkový vektor	Čtyřsl. vektor
Lineární interpolace	Mix Vec4	Čtyřsložkový vektor, čtyřsložkový vektor, skalár	Čtyřsl. vektor

Tabulka A.4: Definice uzlů s funkcemi upravující čtyřsložkové vektory

Funkce	Název uzlu	Typy vstupů	Typy výstupů
Součet	Mat4 + Mat4	Matice, matice	Matice
Rozdíl	Mat4 - Mat4	Matice, matice	Matice
Součin	Mat4 * Mat4	Matice, matice	Matice
Násobení skalárem	Float * Mat4	Skalár, matice	Matice
Transpozice	Transpose Mat4	Matice	Matice
Inverze	Inverse Mat4	Matice	Matice
Determinant	Determinant Mat4	Matice	Skalár
Dekompozice	Decompose Mat4	Matice	Třísložkový vektor, třísložkový vektor, kvaternion, třísložkový vektor, třísložkový vektor

Tabulka A.5: Definice uzlů s funkcemi upravující matice

Funkce	Název uzlu	Typy vstupů	Typy výstupů
Translace	Translation	Třísložkový vektor	Matice
Škálování	Scale	Třísložkový vektor	Matice
Rotace podle osy X	Rotation euler angle X	Skalár	Matice
Rotace podle osy Y	Rotation euler angle Y	Skalár	Matice
Rotace podle osy Z	Rotation euler angle Z	Skalár	Matice
Rotace podle osy a úhlu	Rotation - vector + angle	Třísložkový vektor, skalár	Matice
LookAt	Look at	Třísložkový vektor, třísložkový vektor, třísložkový vektor	Matice

Tabulka A.6: Definice uzlů s transformacemi

Funkce	Název uzlu	Typy vstupů	Typy výstupů
Ortografické	Ortographic	6x skalár	Matice
Perspektivní	Perspective	4x skalár	Matice
Frustum	Frustum	6x skalár	Matice

Tabulka A.7: Definice uzlů s projekcemi

Funkce	**Název uzlu**	**Typy vstupů**	**Typy výstupů**
Součin	Quat * Quat	Kvaternion, kvaternion	Kvaternion
Násobení skalárem	Float * Quat	Skalár, kvaternion	Kvaternion
Konjugát	Quat conjugate	Kvaternion	Kvaternion
Inverze	Inverse Quat	Kvaternion	Kvaternion
Normalizace	Normalize Quat	Kvaternion	Kvaternion
Délka kvaternionu	Length Quat	Kvaternion	Skalár
Lineární sférická interpolace	Slerp Quat	Kvaternion, kvaternion, skalár	Kvaternion

Tabulka A.8: Definice uzlů s funkcemi upravující kvaterniony