VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PARAMETRICKÉ 2D/3D MODELY

PARAMETRIC 2D/3D MODELS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PATRIK ŠOBER**
AUTHOR

VEDOUCÍ PRÁCE **prof. Dr. Ing. PAVEL ZEMČÍK,**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Šober Patrik**
Program: Informační technologie
Název: **Parametrické 2D/3D modely**
Parametric 2D/3D Models
Kategorie: Počítačová grafika

Zadání:

1. Prostudujte dostupnou literaturu a software pro 2D a/nebo 3D modelování s parametrickým vstupem. Cílem je mít možnost vytvořit model scény, který bude možné měnit a zobrazovat na základě zadaného parametru, například typu "výška budovy, délka vozidla, šířka sedačky" a podobně.
2. Navrhněte možnosti "geometrického provázání" objektů v parametrickém modelu a možnosti animace takového modelu v závislosti na nastavených parametrech. Prostudujte též možnost "interface" s uživatelským software.
3. Navrhněte způsob implementace (může být i nadstavbou nad vhodným CAD systémem) a diskutujte vlastnosti takového způsobu implementace.
4. Implementujte vybraný parametrický model a demonstrujte jeho vlastnosti na vhodném příkladě.
5. Zhodnoťte výsledky a diskutujte možnosti dalšího rozvoje práce.

Literatura:

- Dle pokynů veducího

Pro udělení zápočtu za první semestr je požadováno:

- -

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Zemčík Pavel, prof. Dr. Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 22. července 2020

Abstrakt

Cílem této práce bylo vytvoření aplikace pro tvorbu a manipulaci 2D a 3D modelů skrze parametrickou konstrukcí. Práce postupně rozebíra teoretický základ okolo tvorby a reprezentace 3D modelů v počítačové grafice, pojmy parametrické konstrukce a představení podobných aplikací včetně jejich rozdílů. Dále je popsán způsob vytvoření aplikace na základě vytvořeného návrhu ze získaných znalostí. Na závěr bylo provedeno testování aplikace, zaměřující se na uživatelské rozhraní a zátěže renderovací systému.

Abstract

The aim of this thesis was to create an application for the creation and manipulation of 2D and 3d models trough parametric construction. The work gradually describe the theoretical basis around the creation and representation of 3D models in computer graphics, the concepts of parametric construction and the introduction of similar applications, including their differences. Next is described method of creating an application based on the create design from the accuired knowledge. Finally, the application was tested, focusing on the testing user interface and the load on the rendering system.

Klíčová slova

Parametrická kosntrukce, konstrukční geometrie, dynamická geometrie, 2D/3D zobrazeni, Qt, Ogre

Keywords

Parametric construction, geometry construction, dynamic geometry, 2D/3D view, Qt, Ogre

Citace

ŠOBER, Patrik. *Parametrické 2D/3D modely.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Dr. Ing. Pavel Zemčík,

Parametrické 2D/3D modely

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Prof. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Patrik Šober
28. července 2020

Poděkování

Rád bych poděkoval vedoucímu práce, Prof. Dr. Ing. Pavlu Zemčíkovi za rady, návrhy, přípomínky a odborné rady, kterými mi pomohl při vypracování a dokončení této práce.

Obsah

Kapitola 1

Úvod

Slovo parametrická konstrukce nebo dynamická geometrie se v oblasti vývoje programů vyskytuje již kolem 80. let 20. století. Tento přístup dal vzniku nových kreslících nástrojů. Programy založené na tomto principu nabízí možnosti vytvořit základní geometrické primitiva jako jsou bod, úsečka, přímka, kružnice a nahrazuje tak dlouho používanou metodu ručního vytváření náčrtů, výkresů a také velkou hromadu papíru. Výhodou takových nástrojů je jejich přesnost a rychlost, nezáleží tedy tolik na chybě lidského oka. Díky těmto programům, můžeme v několika krocích vytvořit komplexnejší geometrické konstrukce. např: najít průsečík dvou přímek, spočítat úhel mezi dvěma tělesy v prostoru nebo vypočítat bez složitých operací obsah. Ovšem hlavní výhodou zůstává fakt, že tyto vytvořené objekty nejsou jenom obrázek na obrazovce. Scéna, ve které se tyto objekty vyskytují se skládá z grafických objektů, které mají mezi sebou logické vazby, odpovídající normálním pravidlům v geometrii, např: úsečka určena dvěma body, tři úsečky tvořící trojúhelník nebo v prostoru množina bodů, hran a ploch tvořící komplexní objekty. S těmito objekty nebo částmi lze manipulovat, měnit, uchopit a posunout nebo otočit v prostoru o nějaký stupeň. Uživatel může tímto způsobem vidět jak konstrukce dynamicky mění a které jsou statické. Z tohoto přístupu vzniká nový pohled na geometrie nazývající se dynamická geometrie, někdy také nazývána jako parametrická konstrukce objektů. Její základní myšlenka spočívá v tom, že jednotlivé objekty mohou být dynamicky měněný pomocí vytvořených parametrů připnutých k nim. I když tato myšlenka zní velmi dobře, přináší sebou i nemalé problémy, které se normálně v geometrii nevyskytují a jedná často a prosté implementační problémy při vývoji aplikací.

Tyto aplikace byli dříve často využívané pro studijní účely, ale jak šel vývoj kupředu, začala se tato myšlenka využívat v také v průmyslu, především pro návrh výkresů, vytváření průřezů přes modely, které by se uživatel musel většinou představit nebo vyrobit ručně a tak zničit výrobek. Další aplikací v průmyslu mohou být i elektronické obvody, ale také i stavebnictví, kde se vytváří plány budov a poté jejich promítnutí do 3D prostoru a vytváří tak náhled jak by mohla daná budova vypadat ve skutečnosti. Jednom z posledních možností využití lze najít také ve vytváření simulací a animací reálných objektů, které jsou vytvořeny v těchto nástrojích a jsou zde přidány parametry, které umožňují testovat tyto objekty v různých podmínkách. Z pohledu výuky je použití těchto programů jednoduší. Studenta se většinou snaží naučit základy geometrie nebo představení jak vypadá nějaká složité funkce v podobě grafů

Při výběru tohoto tématu jsem hledal hlavně problém, který by mi umožnil něco vytvořit a také mám v oblibě počítačovou grafiku, která mi přijde zajímavější než ostatní obory jako jsou síťové technologie nebo teorie jazyků nebo automatizace. Vždy jsem chtěl

vytvořit aplikaci, která je více než skript který pracuje přes příkazový řádek, ale má nějaké uživatelské grafické rozhraní.

Cílem této práce bylo vytvořit systém, který je schopen vytvořit model scény, kde je možné objekty v něm nebo samotnou scénu měnit na základě zadaných parametrů. Dále navrhnout způsob provázání těchto parametrů s vytvořenými objekty. K němu má být vytvořená také renderovací část, kterou může uživatel spustit, když se bude chtít podívat na svůj výtvor. Tato část čistě slouží jako vizuální pomůcka, to tedy znamená, že budeme moc projít se ve scéně a zobrazit, ale ne zpětně modifikovat co jsme udělali v první části programu.

Úvodem této práci je představení historie programů zabývající se stejnou nebo podobnou problematikou, které představující počátky první parametrické konstrukce a současný přehled softwarů na dnešním trhu jak 2D a 3D, včetně popisu použití v dnešní době. V kapitole Geometrie jsou popsány základní principy, na kterých pracuje právě tento projekt nebo i podobné nástroje. Další kapitoly obsahují návrh projektu, popis použití knihoven pro zhotovení, implementování jednotlivých problému a testování samotné funkce aplikace. V závěru je uvedeno zhodnocení práce, způsob splnění zadání a plány pro další možnosti vývoje aplikace.

Kapitola 2

Historie a přehled dnešních konstrukčních nástrojů

Kapitola se zaměřuje zejména na představení několik softwarů používané nejen v oblasti parametrické konstrukce. Je zde popis i počátků prvních softwarů a jejich postupný vývoj. Tyto programy byly vybrány pro představení různých přístupů při vytváření nějaké scény, objektů a jejich případná manipulace. Jsou zde blíže popsány jejich specifické vlastnosti, funkcionality a možnosti v oblasti konstruování objektů v počítačové grafice. Obsah této kapitoly není encyklopedický ani neobsahuje úplný popis jednotlivých softwarů, ale pouze představení důležitých částí, neboli ukázky co se vlastně nachází na dnešním trhu.

2.1 Historie prvních konstručních nástrojů

Celá historie parametrické konstrukce a přístupu z pohledu vývoje programů začíná kolem 60. let. 20. století [7]. Cílem vývojářů té doby byla myšlenka vytvořit kreslící nástroje schopné vytvořit základní geometrické útvary jako jsou úsečky, kružnice a nahradit tak tehdejší ruční práci pomocí pravítka a kreslení velkých výkresů. Vznikl tak pojem známý jako dynamická geometrie, jejímž účelem bylo vytvořené objekty uchopit, přemístit, modifikovat nebo upravovat vytvořenými parametry. Tato myšlenka s poté vyvíjela dál a vznikl termín známý jako CAD (Computer-aided design) [17].

Důležitým programem stojící za zmínění je program SKETCHPAD[1] [18] vytvořený Ivanem Sutherland, jenž dovoloval uživatelům manipulovat se svými počítači graficky, jinými slovy šlo o jednu z prvních technologií, která využívala myšlenku dotykového displeje, technologie byla založena na CRT monitorech a dotykového pera na podobném principu. SKETCHPAD[1] je považován za předchůdce moderních CAD kreslících nástrojů [18].

Kolem 80. let v Srpnu firma AutoDesk vydala první verzi své hlavní aplikace AutoCAD [17], jenž v té době byla jedna z mála aplikací využívající principu CAD systému. Poté se rozvoj podobných programů hodně zvedl a začala nová éra vývoje kreslících nástrojů založené právě na tomto systému, například: SOLIDWORKS[1], Creo, Pro/ENGINEER, FreeCAD, NX, atd.

Dnešní době už existuje velký počet těchto kreslících nástrojů pracujících jak 2D oblasti tak 3D. Vyskytují se od základní funkcionality s jednoduchým uživatelským rozhraním až po komplexní parametrické konstrukční systémy. Dalším moderním trendem, v oblasti

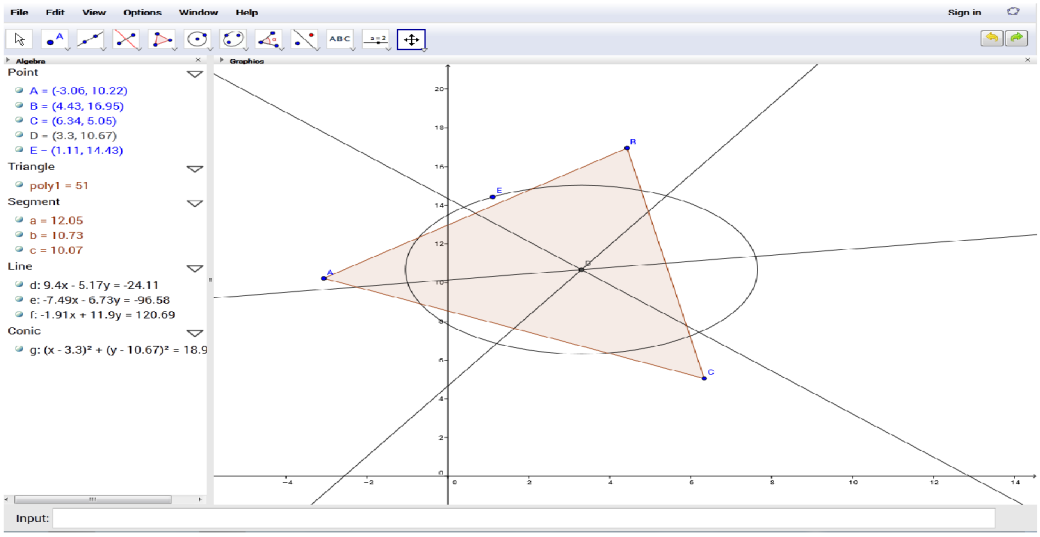
[1] více o SKETCHPAD: `https://en.wikipedia.org/wiki/Sketchpad`
[2] více o Texas Instruments: `https://www.ti.com/`

vývoje je rozšíření těchto druhů aplikaci i na mobilní zařízení. Právě díky enormnímu vývoji nových technologií, je dnes možnost vykonávat a provádět dříve nemožné výpočty geometrie a vykreslování, dokonce i na kalkulačkách. Jeden z těchto programů lze najít v zařízeních od Texas Instruments[2].

2.2 Nástroj GeoGebra

Jedním ze zástupců 2D/3D kreslících nástrojů pro dynamickou geometrii je program GeoGebra [9]. Aplikace se převážně zabývá v oblasti 2D geometrie, ale zhruba od verze 5.0 aplikace začala podporovat i 3D kreslení a konstrukci. Samotný program GeoGebra je realizován pomocí jazyka Java a jeho první verze byla vydána okolo roku 2002, později byla přidána podpora i html 5 a díky použité technologii lze nyní spustit a používat aplikace skrze webové prohlížeče.

Samotný program je interaktivní, uživatel může přímo vytvořit scénu, přidávat nebo měnit objekty a jejich vlastnosti pomocí pohybu pracovní myší nebo dotykového displeje. Jedním z důvodů o zmínění tohoto programu je jeho funkce sledování pohybu objektů ve scéně skrze operování s parametry. Tyto hodnoty lze libovolně měnit a jejich změna se poté promítne do scény. Díky této funkci je možnost sledovat samotné objekty scény, vracet zpátky změny na místo zpětného kroku v celém programu. GeoGebra obsahuje přehledný seznam objektů ve scéně, přičemž některé z nich mohou být reprezentaci v podobě rovnic. Ty lze libovolně změnit a dosáhnout tak úpravy části scény. K tomuto lze použít zabudovanou kalkulačku, ve které lze vytvořit rovnice, derivace nebo integrály.



Obrázek 2.1: Ukázka rozhraní nástroje GeoGebra[1]

GeoGebra také nabízí podporu skriptování. Jedno z možných propojení může být sledování pohybu a animací. Výsledkem je tedy průběh funkce v závislosti na čase a pomočí výstupu lze sestavit animaci. Samotné objekty ve scéně lze také provázat s tabulkami a tato vzniklá vazba je obousměrná. To znamená, že při zásahu to tabulek a zpátky se mohou změnit různé výsledky rovnic nebo podobu objektů. GeoGebra lze využívat pod licencí

[1]obr. převzat z: `https://en.wikipedia.org/wiki/GeoGebra`

GNU Genereal Public License (GPL). Lze ho tedy stáhnout a používat zdarma, pokud by se nejednalo o jiné produkty od stejné společnosti.

2.3 Nástroj AutoCAD

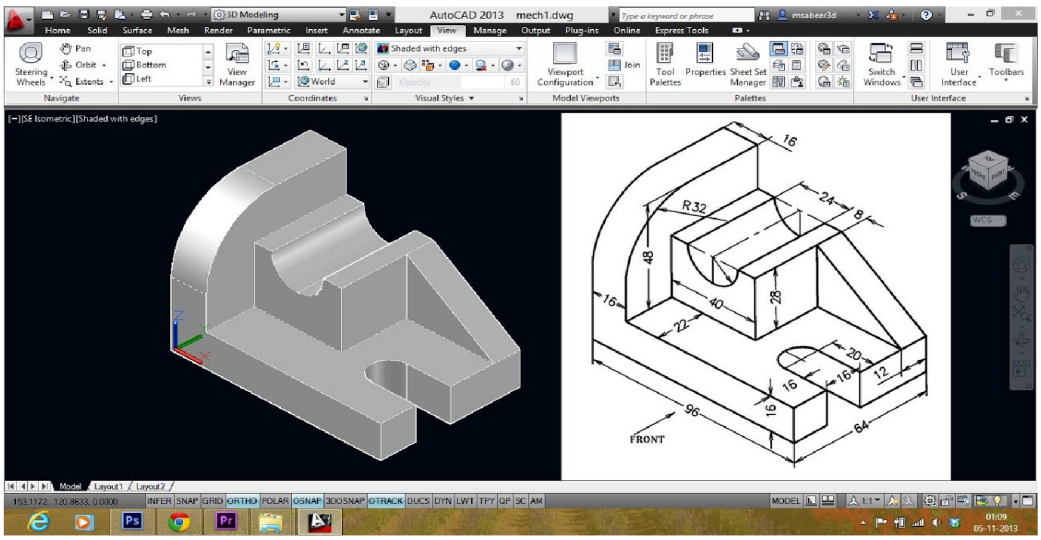
AutoCAD má jednu z nejdelších historií jako kreslící nástroj, který se hojně využívá v oblastech počítačové grafiky, modelování, ale i vytváření nákresů, stavebnictví a elektronických obvodů. Nástroj patří do skupiny CAD systémů [17]. Tento program byl vytvořen společností Autodesk [2] v roce 1982 a byl to jejich hlavní produkt, předtím než se společnost rozšířila a později vytvořila specializované aplikace v různých oborech.

Program lze nejen využít k vytvoření 3D modelů ale i taky pro vytvoření náčrtů, dokumentů a 2D nákresů. Všechny vytvořené křivky a objekty jsou tvořeny vektorovou grafikou. Objekty v AutoCADu obvykle začínají ve formě náčrtů nebo spíše 2D drátových modelů, kde pomocí funkcí lze vytvořit pevný model. Tyto drátové modely jsou vytvořeny ze základních geometrických prvků kde je možnost měnit jednotlivé hodnoty, přidat pro přehlednost kóty nebo geometrické pravidla jejíž modifikací lze zpětně tyto části měnit.

Při vytvoření vykreslení 3D modelů program dovoluje přidat materiály a také i světlo do scén a vytvořit až fotorealistické produkty. Je zde také možnost řídit vzhled a osvětlení přidaných materiálů a stínů modelů pro lepší výslednou vizualizaci.

Dalším nástrojem pro 3D modelování v AutoCADu je schopnost vytvářet roviny řezů a logické spojení vícero objektů v jedné komplexní. Úpravy těchto řezů umožňují vidět vnitřní části 3D objektů a umožňují krájet povrchy, tělesa, oblasti a sítě.

Nákresy v AutoCADU je poslední krok ve většině projektů vytvořené tímto programem. Je zde možnost vložit jednotlivé části objektů nebo průřezy objektem, přidávat kótování jednotlivých částí a přidávat šablony, které jsou v těchto oblastech standardní.



Obrázek 2.2: Ukázka rozhraní aplikace AutoCAD[2]

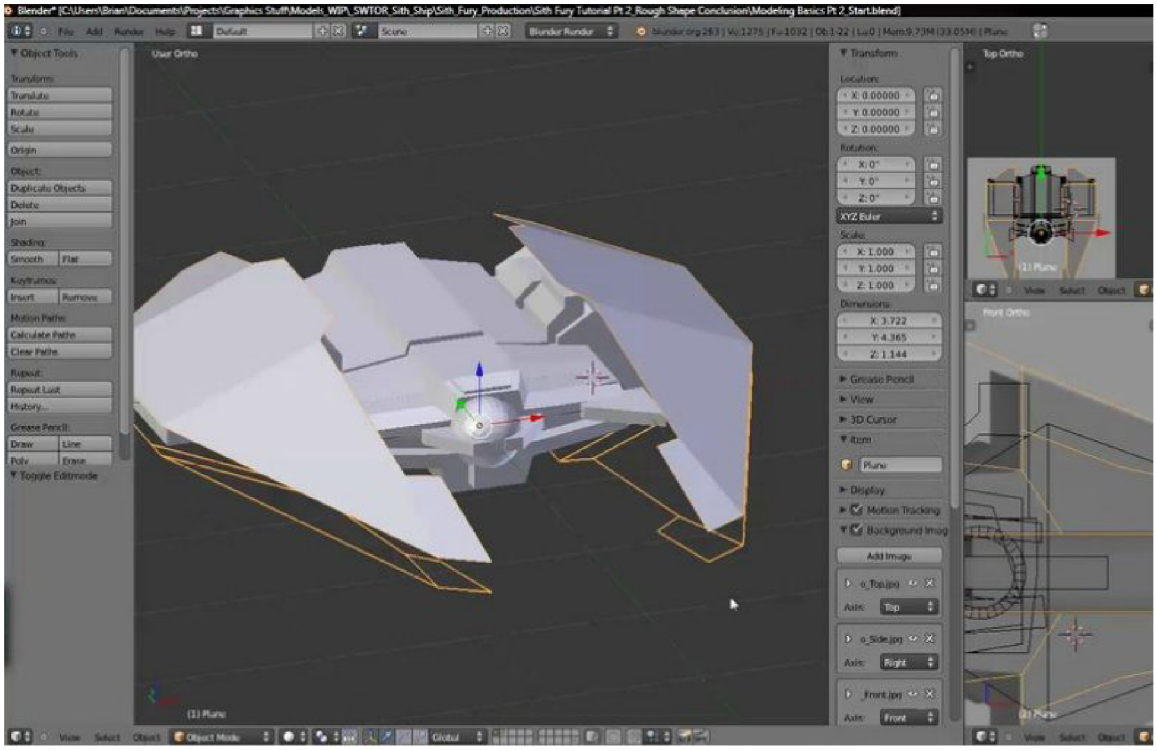
Jedním z dalších důvodů proč se AutoCAD využívá v takové míře je technologie, která data ukládá do DWG formátu[8], jenž je nejvíce rozšířen a standardizován pro převod mezi různými firmami. Také podporuje ostatní známé formáty jako DXF, MAX a výsledné

[2]obr. převzat z: `https://i.ytimg.com/vi/fHqolQwz93U/maxresdefault.jpg`

vyrenderované objekty mohou být poté uloženy ve standardních formátech JPG, PNG apod. AutoCAD je velmi dobrý nástroj, bohužel se, ale za něj musí zaplatit, pokud by ho chtěl uživatel používat pravidelně.

2.4 Nástroj Blender

Tento program není tak kreslící nástroj, ale spíše editor. Blender[3] byl vytvořen kolem roku 1994 autorem Ton Roosendaal. Později byla založena Blender Foundation, která jak skupina nezávislých programátorů podporovala jeho rozvoj. Díky této podpoře, se stal blender velmi oblíbený pro různé uživatele, jelikož mohou modifikovat skoro každou část tohoto nástroje a vytvářet vlastní postupy a metody konstrukcí objektů a vytvořit tak moduly, které si mohou nainstalovat další uživatelé a vytvořit tak jeden mocný nástroj, který v budoucnu může mít doplněné vlastnosti, které mají jiné kreslící nástroj, jako například AutoCAD[17].



Obrázek 2.3: Ukázka rozhraní nástroje Blender[4]

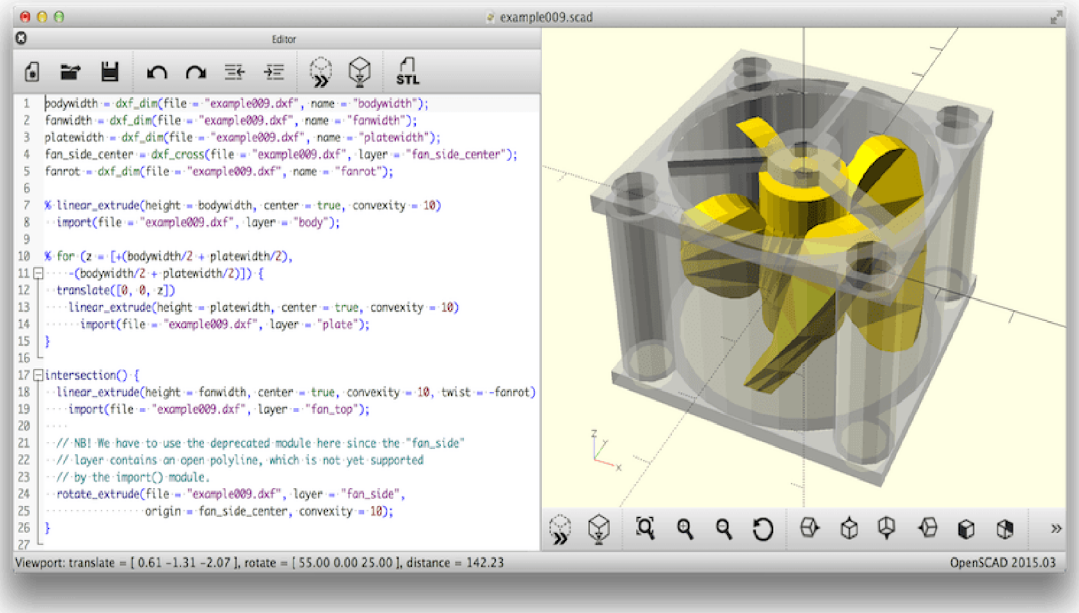
Narozdíl od ostatních nástrojů, Blender[3] je čistě program zaměřující se na přímou konstrukci objektů[3], pracuje ve 3D prostoru. Pomocí různých implementovaných nástrojů, lze zde vytvářet až abstraktní umění či animace. Ale i tento program má určitý stupeň parametrického přístupu, ale prozatím pouze v oblasti zabývající se vizualizací, kde můžeme opravdu měnit každý aspekt materiálu. Zároveň do jisté míry podporuje i skriptování skrze programovací jazyk Python.

[3] více o Blender: `https://www.blender.org/`

2.5 Nástroj OpenSCAD

Tento kreslící nástroj jeden z mála, který byl vytvořen nezávislým programátorem Marius Kintel v roce 2010 [14]. Základ tohoto programu tvoří CAD systémy, zejména za účelem tvorby pevných 3D CAD modelů. OpenSCAD je vytvořen pod technologií Qt knihovny pro vytvoření uživatelského rozhraní, CGAL pro vyhodnocení CSG (Constructive solid geometry)[3][11] neboli logických operací které propojují objekty to jednoho komplexního. Pro vykreslování používá starší renderovací systém OpenCSG.

Hlavním důvodem zmínky o tomto nástroji je fakt, že nejde o interaktivní nástroj, jde více o 3D-kompilátor, který čte skriptovací program, který popisuje model, postup jeho konstrukce a poté vykresluje. Díky této vlastnosti dává uživateli plnou kontrolu nad procesem modelování [13]. Jelikož jde o skript skoro až zdrojový soubor, lze měnit každou část modelu, přidávat vlastní nové proměnné, cykly nebo parametry které mohou ovlivnit celý proces tvorby.



Obrázek 2.4: Ukázka rozhraní OpenSCAD[5]

Program podporuje AutoCAD DXF soubory [8] a formáty ale pouze jeho 2D nákresy. Mezi další podporované formáty patří STL and OFF. Jelikož jde o výtvor nezávislého programátora, je tento program volně dostupný a nemusí se za něj platit. Program OpenSCAD je vydán pod licencí GPLv2.

[4]obr. převzat z: `https://www.iamag.co/wp-content/uploads/2012/12/modbasics-part1.jpg`
[5]obr. převzat z: `https://www.openscad.org/assets/img/screenshot.png`

2.6 Použití nástrojů v dnešní době

Využití parametrických kreslících nástrojů, zejména v dnešní době, je celá řada. Od standardních použití v oblasti vytváření modelů, výkresů až po vytváření architektury, složitých elektrických obvodů nebo sloužit jako podpora pro výuku geometrie a algrebry [7][9].

Pokud se zaměříme na výuku, jsou tyto programy značně zjednodušené, spíše se jedná o takové náčrtníky jako například aplikace GeoGebra [9].

Pro komplexní vytváření a možnosti vytvoření reálných simulací je vhodné pro většinu uživatelů použít aplikace zvanou AutoCAD [2], jenž vzala myšlenku parametrické konstrukce velmi vážně a můžeme zde ovlivnit skoro každou část modelu, včetně materiálu ze kterého by mohl být vytvořen.

Nástupem moderní technologie a vývojem menších zařízení najdou uplatnění tyto nástroje i v kalkulačkách. Programy v nich dokáží na základě zadané funkce vykreslit graf, vytvořit tabulku hodnot jenž vede k vykreslení nějakého objektu nebo, předvést i jednoduchý pohyb objektů.

Kapitola 3

Úvod do problematiky parametrické konstrukce

V této kapitole jsou popsány základní pojmy a geometrie týkající se parametrické konstrukce a modelování, rozdíly mezi přímou a parametrickou konstrukcí a na závěr použití parametrické konstrukce v praxi. Dále jsou zde popsány reprezentace 2D a 3D objektů v počítačové grafice.

3.1 Geometrie a transformace objektů

Geometrie a matematika [10] se používá v počítačové grafice v mnoha směrech, většinou pro zobrazování dat [3], které byly vytvořeny, nebo grafické uživatelské rozhraní. Také pro tvorbu nástrojů sloužící pro sestavování objektů a komplexních modelů, ale i simulací a animací. Důvodem pro zmínění této části, je určité nastínění s jakou částí geometrie se při vytváření podobných aplikací setkáváme.

V parametrických konstrukčních nástrojích je typické ovládat konstrukci a vytvářet základní geometrické objekty. Jedná se například o obyčejné body, přímky nebo i roviny. V rámci 2D je možná tvorba rovinných geometrických útvarů jakou jsou např. kuželosečky [10]. Ve 3D prostoru jsou běžné koule, válce, kužely.

Všechny tyto části se musí držet pravidel Eukleidovské a Neeuklidovské geometrie [11][16]. Eukleidovská geometrie je založena na axiómech (pravidlech) jenž nabízí jakýsi popis tvorby základních geometrických prvků, např:

- Každými dvěma body lze vést úsečku
- Tato úsečka může být prodloužena oběma směry (vytvoření přímky)
- Kruh je rovinný útvar ohraničený jednou čarou (nazývanou kružnice), a to tak, že všechny úsečky, které jsou ní vedeny z jednoho bodu, se navzájem rovnají
- Uvedený bod se nazývá střed kruhu

A tak by pokračovalo dále v podobě až 13 knih. Mimo jiné tato část geometrie dala vzniknout i tvz. neeuklediovské Geometrii [16]. Hlavním důvodem pro vytvoření této části byl možnost pro popisu křivek, přesněji hyperbolické , eliptické nebo sférické geometrie. Díky těmto principům lze vytvořit složité objekty, které spojením základních prvků mohou vytvořit kouli, hyperbolicky prohnutý objekt. Další funkcí obnáší možnost popisu jednodušších geometrických prvků na povrchu těchto objektů.

Při tvorbě objektů, mohou být v programech jednoduché funkce [11] pro výpočet obsahů, nebo objemů těles, výpočet vzdáleností těles, jejich úhel mezi nimi nebo také možnost vytvoření vlastních parametrů, kterými lze modifikovat celé nebo části objektů, jsou vlastně vázány s elementy scény a při jejich změně dochází k jejich přepočtu, např: velikost koule na jejím poloměru, vytvoření parametrů délky židle a změnit tak její zadní část a změnit jak bude dlouhá.

Transformace objektů

Jedním z posledních konstrukcí v oblasti 2D/3D prostoru, jsou tzv. transformační operace [3][11][16] Tyto operace zejména pracují s řídícími nebo spíše hlavními body ze kterých se konstrukce skládají. Transformace se aplikuje na všechny tyto body. V prostoru se většinou vyskytuje tvz. počáteční bod $[0, 0]$, dále podle typu transformace se zde vyskytuje bod $P[X, Y]$ a bod *P'[X', Y']*, který vznikl po aplikaci nějaké transformace T.
Mezi elementární transformace patří:

1. Rotace - jde o otočení bodu kolem středu vztažné soustavy o daný úhel α

$$P' = [X\cos(\alpha) - Y\sin(\alpha), X\sin(\alpha) + Y\cos(\alpha)]^T \tag{3.1}$$

$$R = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

2. Scaling (změna měřítka) - mění velikost výsledného objektu, ta je určena změnou velikosti podle souřadnicových os $[S_x, S_y]$

$$P' = [X * S_x, Y * S_y]^T \tag{3.3}$$

$$S = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix}, S = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & S_x \end{pmatrix} \tag{3.4}$$

3. Translace (posunutí) - transformace posunu. Posun je zde určen vektorem posunutí $\overrightarrow{p} = [p_x, p_y]$, popisuje také ja daleko bude posunut.

$$P' = P + \overrightarrow{p} \tag{3.5}$$

$$T = \begin{pmatrix} 1 & 0 & P_x \\ 0 & 1 & P_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.6}$$

Pro jednoduchost zde byli předvedeny transformace na rovině (2D), pro vetší zájem si lze o základech transformací dočíst zde [16]. Tyto jednotlivé transformace lze dále skládat do jedné matice což mnohdy vede rychlejšímu vypočtu a vykreslování z pohledu počítačové grafiky.

3.2 Reprezentace 2D objektů

Jelikož se jedná o 2D objekty [11], jejich tvorba je limitovaná pouze osou x a osou y - uspořádané do dvojice (x, y). Kombinace těchto dvou os dava 2 rozměrný prostor (tedy 2D). Reprezentace těchto objektů začína od jejich základních prvků (body, přímky, úsečky, atd.). Jejich spojováním určitými pravidly (některé znich byly v kapitole 3.1) a úhly jse možne vytvořit komplexní struktury. Důvodem proč tyto prvky můžeme nazvat jako 2D je, že nemají část která by pro popsání vyžadovala třetí rozměr.

Bod

Jak bylo zmíněno výše, bod je elementárním prvkem [11]. Je reprezentován dvojicí čísel určující jeho pozici v rovině - např: bod $A = (5, -4)$. Pokud bychom tento prvek chtěli, vytvořit také v prostoru (3D), je potřeba k jeho souřadnici přidat další rozměr (neboli třetí číslo do skupiny), např: bod $B = (0, 2, 3)$.

Sám o sobě není moc významný, ale je to základní stavební kámen od kterého můžeme definovat další prvky geometrie a složitější prvky, známé jako mnohoúhelníky. Dále jako prvek pro určení některých vlastností vytvořených objektů nebo grafů v matematice.

Úsečka

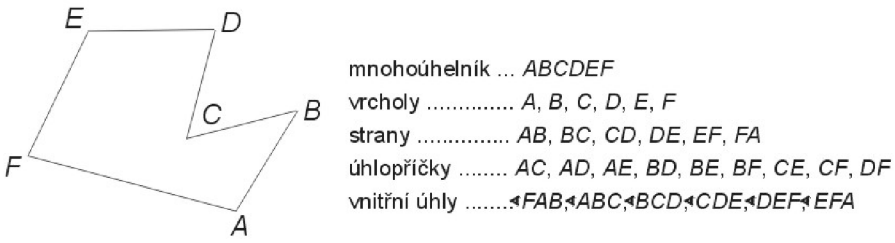
Úsečka [11] také jeden z elementární prvků. Jeho struktura je složena ze 2 bodů - počáteční a koncový bod. Tyto dva body udávají pozici úsečky, jeho vektor a délku, kterou může maximálně dosáhnout. Matematický zápis pro takovýto prvek je AB, to znamená, že je úsečka reprezentována body A a B.

Polygon

Polygon v matematice [11] spíše známý jako mnohoúhlehík, je část roviny vymezená úsečkami, které spojují určítý počet bodů, nejméně tří, které tvoří základní geometrický útvar zvaný trojúhelník. Musí platit, že tyto body nesmí ležet na jedné přímce.

Body, které mnohoúhelník určují se nazývají vrcholy, úsečky spojující sousední vrcholy se nazývají strany [11]. Objekt má také úhlopříčky vytvořené nesousedními vrcholy a neposlední řadě také vnitřní úhly vytvořené sousedními stranami. Máme různé druhy mnohoúhelníků:

- **Pravidelné** - všechny strany i vnitření úhly jsou shodné, při **nepravidelných** nikoliv
- **Konvexní** - všechny vnitřní úhly jsou menší nez 180° a **nekovexní** jež splňují tuto podmínku opačně
- **Pravoúhleníky** - všechny vnitřní úhly jsou pravé, přip. 270°
- **Jednoduché a degenerované** - u degenerovaných se alespoň 2 strany protínají



Obrázek 3.1: Představení pologyno v 2D prostoru

3.3 Reprezentace 3D objektů

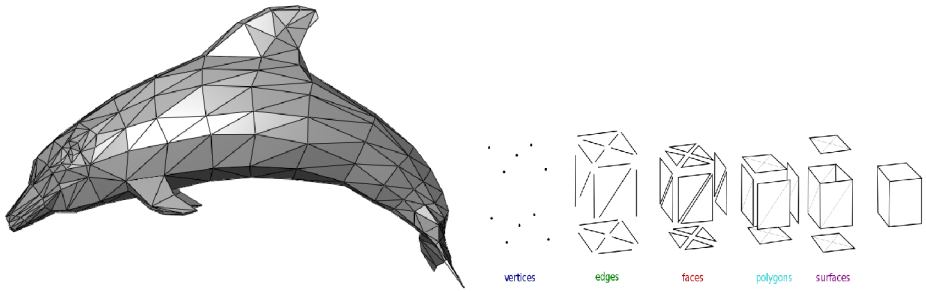
Stejně jako v 2D (rovina)[11], můžeme zmíněné objekty výše, reprezentovat i v prostoru a to přidáním čísla a vytvořením trojice čísel (body v prostoru, atd). Pokud se ovšem bavíme, jak se tyto objekty reprezentují v počítačové grafice, nazývaly bychom tyto objekty tkz. **Polygonálními modely** [11][3][6].

Polygonální modely

Tyto objekty jsou tvořeny tkz. polygonálními sítěmi [6][3], ty představují nějaký souhrn vrcholů, hran a ploch. Typickým představitelem jsou Polyhedrony [3], což jsou objekty vytvořené právě ze skupiny těchto základní prvků, kde mezi hlavní vlasnosti patří přímé hrany a ostré rohy.

Pokud bychom popsali tyto modely pouze v podobě oblasti 3D grafiky, pak se části polygonálního modelu se v tomto přídadě nazývají následujícím způsobem: vertexy, hrany, plochy, povrchy, polygony. Každý z těchto prvků [3][6] udržuje nějakou informaci potřebnou pro sestavení objektů v prostoru nebo manipulaci, či pro jeho vyuzalizaci pomocí nějakého renderovacího enginu. Nyní bližší popis těchto částí:

- **Vertexy (vertex)** - Představuje bod v modelu, ale zároveň udržuje další informace jako je barva, normálový vektor nebo souřadnice textury použité na povrchu objektu, tedy informace, které reprezentují model v počítačové grafice, zejména pro jeho vizualizace
- **Rohy (edges)** - Představuje propojení mezi dvěma vertexy
- **Plochy (face)** - Jsou uzavřené množiny rohů. Mezi typické typy v počítačové grafice, patří tri-face/quad-face (tři-rohová plocha, čtyř-rohová plocha)
- **Polygony (polygon)** - jsou uzavřené množiny ploch spojené dohromady. (žádné duplicitní body, pouze průnik společných bodů)
- **Povrchy (surface)** - tvořena skupinou polygonu, většinou pro důvod ulehčení orientace v modelu.

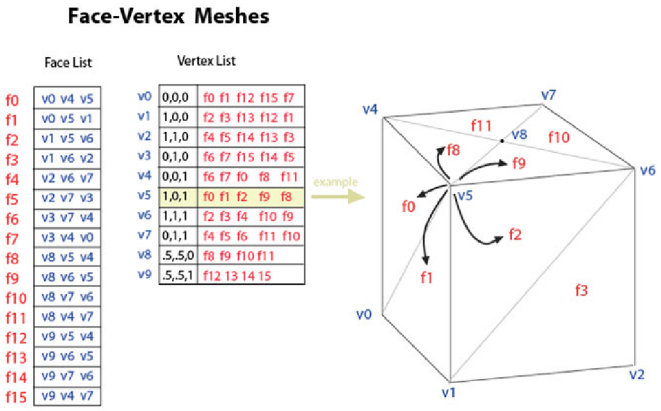


Obrázek 3.2: Reprezentace polygonálního modelu[1]

Nyní byli popsány základní prvky, ze kterých se skládají tyto polygonální modely, ale nastává problém jakým vhodným způsobem je lze prezentovat jako datové struktury [3], které by šly naprogramovat. Skrze historii byli vytvořené různé způsoby.

Face-vertex reprezentace

Tento přístup reprezentuje mesh (3D polygonální model) [3][6] jako objekt vytvořený z množin vertexů a ploch. Díky tomuto přístupu můžeme snadněji procházet jednotlivé plochy nebo vertexy. Bohužel neuchovává informaci o hranách modelu, tudíž je nutnost je explicitně určit. Další nevýhodou může být také vyhledání a určení nějaké plochy v této struktuře, protože je zde nutnost vyhledat všechny sousední plochy té hledané. Na obrázku 3.3, lze vidět jak lze z jednoho bodu popsat dokola jednotlivé přilehlé plochy s kterými bod V5 sousedí.



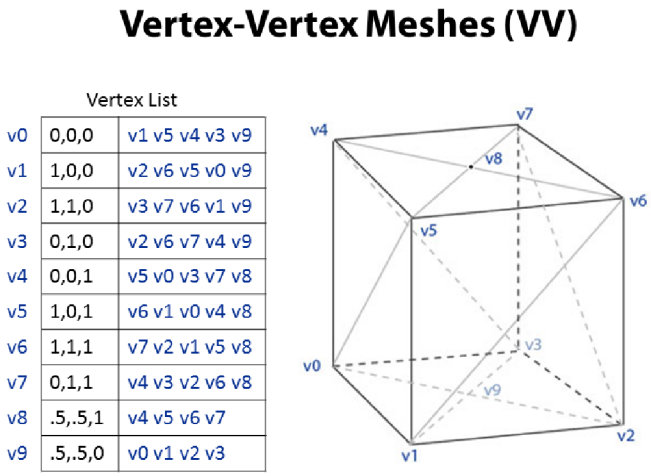
Obrázek 3.3: Reprezentace modelu pomocí Face-Vertex struktury[1]

Jde o jeden z nejvíce používaných reprezentací v počítačové grafice, zejména se využívá u výpočtů v moderních grafických hardwarech.

Vertex-vertex reprezentace

Tato reprezentace [3] je popsána pouze jako sada vertexů k dalším vertexům. Jde o nejlechčí reprezentaci, jelikož hrany a plochy se musí vytvořit samostatně, což vytváří problémy při jejich manipulaci. Výhodou je, ale malá velikost jako datová struktura. Stejně jako v předchozí reprezentaci, má jeden vertex informace o okolí, v tomto případě se jedná o další

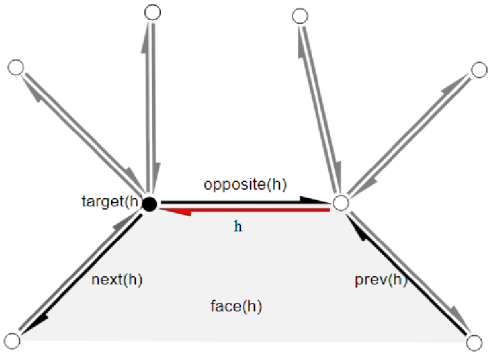
vertexy (počet sousedních vertexů záleží na komplexnosti modelu) jak lze vidět na obrázku 3.4.



Obrázek 3.4: Reprezentace modelu pomocí V-V struktury[1]

Half-edge reprezentace

Jedná se o datovou strukturu [3][5], která je schopna udržet informaci jak o hrany, vertexu tak i ploše. Hlavním nositelem informace je zde hrana, která je rozložena na půlhranu s určitým směrem (vždy každá půlhrana směřuje naopak od té druhé). K této půlhraně je vždy přiřazen jeden vedlejší vertex a přilehlá plocha. Pro vertexy a plochy je vždy jedna půlhrana. Tato struktura je pamětově náročná ale nabízí velkou kontrolu při vytváření jednotlivých objektů a jejich zpětnou manipulaci. Právě tato struktura je používána v projektu.



Obrázek 3.5: Reprezentace modelu pomocí Half-Edge struktury[2]

[1] obr. převzaty z: `https://en.wikipedia.org/wiki/Polygon_mesh`
[2] viz. o halfedge: `https://doc.cgal.org/latest/HalfedgeDS/index.html#Chapter_Halfedge_Data_Structures`

3.4 Rozdíl mezi přímou a parametrickou konstrukcí

Při postupném rozvoji a vytváření konstrukčních nástrojů [7] v oblasti 2D nebo 3D, se začaly vytvářet různé přístupy [1] jakým by bylo možné dané modely, objekty nebo jednoduché geometrické útvary konstruovat. Jakým způsobem by bylo vhodné je dále upravovat nebo přidávat jedinečné vlastnosti. Z těchto různých přístupů se uchytily hlavně přímý a parametrický přístup tvorby. Každý z nich se využíval dříve pro stejný účel, ale jak šel vývoj dopředu, jejich cíle se změnily.

I když mají tyto přístupy svoje vlastní výhody a nevýhody, často lze najít konstrukční nástroje používající oba, zejména aby uživatel dosáhl příjemnějšího přístupu a svobody při tvorbě modelů a scén nebo případných animací, simulací.

Příma konstrukce objektů

V posledních letech je tento přístup více využívaný nežli parametrický i když je tu většinou snaha oba přístupy kombinovat [1]. Jedním z důvodů je, že uživatel který chce vytvořit objekt má svobodu na jeho konstrukcí, od bodu po celý objekt nebo deformaci celého objektu, bez velké nutnosti dbát na geometrické pravidla, které původně daný model vytvořily. Díky tomu, že nás pravidla tolik neomezují můžeme snadno opravit nějaké nesrovnalosti, které by v případě parametrického přístupu nebylo možné bez rozsáhlého zásahu do konstrukce modelu. Záleží na použití modelu, ale tento přístup mají v oblibě hlavně uživatelé, kteří vytváří modely pro vizuální účely, jako například animace, reklamní modely. Na závěr by bylo vhodné shrnout výhody a nevýhody tohoto přístupu.
Výhody přímého přístupu konstruování objektů:

- Máme volnost nad všemi geometrickými částmi, které tvoří objekt (drátový model)
- Posunovat tyto části volně bez omezení nebo být limitovány geometrickými pravidly
- Rychlost tvorby modelu

Ovšem jsou zde také jisté nevýhody:

- Většinou v nástrojích není žádná asociativita, nelze tedy vytvořit parametr který by mohl s objekty manipulovat
- Vlastnosti, které můžeme dát objekty jsou limitovány nebo nelze vůbec vytvořit
- Při chybné úpravě modelu jsou uživatelé závislý na jejich uložených kopiích a schopností vybraného nástroje, protože je zde možnost, že by daný objekt mohl být nenávratně poškozen

Parametrická konstrukce objektů

Parametrický přístup konstrukce objektů je často představován jako “history-base modelling“ (strom modelu) [3][5]. To znamená, že kreslící nástroje si udržují vlastnosti modelu [17] a kroky které vedli k jeho vytvoření. Všechny tyto informace jsou v modelech uloženy v podobě stromu nebo katalogizovány v seznamech, jenž umožňuje jejich snadnou modifikaci. Při přidání nového kroku se tato nová informace uloží a případně projde celý strom, zdali je bezpečné tento krok vykonat bez ohrožení poškození modelu. Aktualizování nějakého stávajícího kroku vede k přepočítání celého stromu a je zde riziko, že část modelu může být

změněna způsobem, jakým si to uživatel nepřál. Dále je možnost vytvořit parametry, které budou představovat součást nebo součásti objektu a jejich změnou se změní značná část modelu. Tyto parametry mohou mít podobu kót části modelu, jeho tloušťku nebo hloubku nějakých vyříznutých částí. Tento přístup je využíván zejména v CAD systémech [17]. Na závěr by bylo vhodné shrnout výhody a nevýhody tohoto přístupu.
Výhody parametrického přístupu konstruování objektů:

- Automatizované změny - možnost okamžité změny celých součástí objektu jediným parametrem
- Snadná tvorba různých variant součástí.
- Asociativita - díky parametrům je zde snadná změna geometrie, která se může projevit i v jiných částí konstrukčních nástrojů (tento princip využívají například CAD systémy [17]).

I tento přístup se neobejde bez jistých nevýhod:

- V případě velkých modelů se zde může vyskytnout časová náročnost při provádění změn (konstrukce se většinou musí přepočítat celá)
- Možnost havárie modelu při provedené změně. Geometrie se nevytvoří nebo bude poškozena jejímž výsledkem je vznik chybové hlášení nebo rozpad modelu, dokud se chyba neopraví
- Je nutné uvažovat, v jakém pořadí prvky modelovat, zvlášť pokud je model součástí několika kombinovaných objektů. Toto pravidlo platí zejména pro vztah mezi objekty (rozměrové a geometrické parametry)

Kapitola 4

Analýza současného stavu

Tato kapitola pojednává o návrhu aplikace, jeho klíčových vlastnosti. Nejdříve jsou zde popsány podobné existující řešení a jejich shrnuté vlastnosti, dále požadavky na samotnou aplikaci, výběr vhodných technologiích při návrhu jednotlivých částí aplikace, včetně návrh grafického uživatelského rozhraní a zobrazený navržených datových struktur.

4.1 Existující řešení

Jednotlivé softwary (programy) popsané v kapitole 2, představují široké spektrum konstrukčních nástrojů ať se přímou nebo parametrickou konstrukci. Existuje i další řada podobých aplikacích, které mohou být nalezené na ruzných stránkách, které si o nich udržují přehled[1].

Následující tabulka shrnuje vlastnost jednotivých programů, jako je přístup konstruování a manipulace objektů, možnost vytvořit animace, možné skriptování a další významné prvky.

	Přístup konstrukce	Animace	Skriptováni	Další vlastnosti
Geobra	Pomocí GUI, skripty	Ano	Ano	Možnost ovlivňovat části pomocí parametrů, CAS kalkulačka, 3D Grafy
AutoCAD	Pomocí GUI	Ano	Ano	Vytváření nákresů, přidání materiálů a vlastností k modelu, možnost simulací
OpenSCAD	Skript, program	Ano	Ano	Možnost ovlivnit každou část konstrukce scény
Blender	Pomocí GUI, skripty	Ano	Ano	Možnost tvorby filmů, úpravu fotek, tvorba simulací

Tabulka 4.1: Přehled vlastností konstrukčních nástrojů

Konstrukce objektů v nástrojích je velmi podobná, v dnešní době spíše přetrvává vytváření prvků skrze přímý zásah [1] uživatele do objektu a možnost měnit a přidat k němu další prvky, ovšem to už se spíše jedná o přímou konstrukci objektů a proto je většinou přidána možnost vytváření parametrů a ovlivnit některé části modelu, scený jedinou změnou čísla.

[1] Přehled konstručních nástrojů: en.wikipedia.org/wiki/List_of_3D_modeling_software

To je většinou uchováno někde v GUI, odkud je možnost zpětně s nimi manipulovat. Čím více je to k této částí zaměřené, tím více upadá schopnost do modelu zasahovat přímo, což by v dnešní době mohlo nějakým uživatelům vadit, zejména zdůvodu pomalé konstrukce a nebo složitosti. Nejlepší je možnost, uchovat si z obou přístupů některé hlavní vlastnosti. Jedním nástrojů, který toto dokonale splňuje jsou CAD systémy [17], zmíněné v sekci 2.3, kde uživatel může přímo manipulovat s modelem a scénou, ale také jsou zde vytvořeny geometrické závislosti, které jsou většinou přeneseny do nějaké vytvořené tabulky nebo výkresu a zpětně do těchto vlastností zasahovat.

Na druhou stranu, čistá parametrické konstrukce nabízí možnost manipulovat s modely od nejmenších detailů až ovlivňovat celou scénu, pokud je vytvořené rozhraní jednoduché může to mít i své výhody. Příkladem by mohla být aplikace OpenSCAD [14].

4.2 Analýza a specifikace požadavků aplikace

Cílem moji práce bylo vytvoření a implementování aplikace, která by právě dokázala vytvořit nějaký model, nebo více modelů ve scéně, možnost vytvořený model dále upravovat, vlastně měnit jeho konstrukci. Dále možnost k modelu vytvořit parametr, který by ovlivňoval konstrukci modelu nebo jeho část při další manipulaci (např: animace modelu). Mezi další funkce programu by měla být možnost vytvoření animací s těmito modely, případně i skrze vytvořené parametry.

K vytvoření těchto modelu, by mělo být navrhnutí vhodného uživatelské rozhraní a v neposlední řadě vytvoření způsobu viuzalizace vytvořených modelů, které by se mohl uživatel podle potřeby zobrazit. Výsledná aplikace by měla splňovat následující funkce:

- Vytváření a odstranění prvků ve scéně
- Modifikování a manipulace vytvořených prvků
- Vytvoření parametrů a jejich přiřazení k existujícim prvkům ve scéně
- Automatické reakce na změny konstrukce zavislých prvků
- Možnost vytvoření animace
- Vizualizace vytvořených prvků

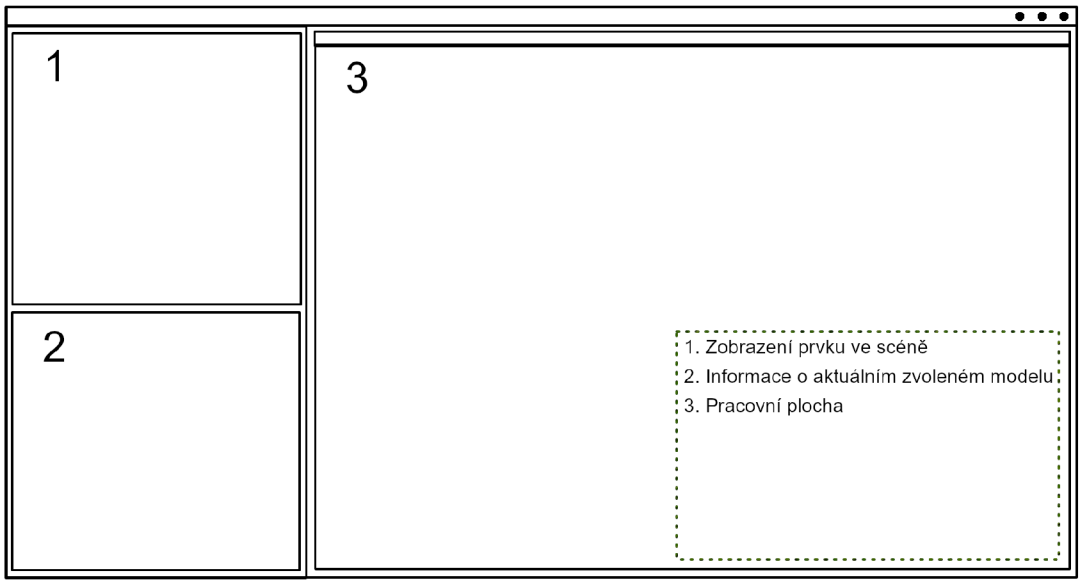
4.3 Návrh struktury programu

Pro funkčnost celé aplikace byla nutnost se zamyslet jakým způsobem se budou objekty vytvářet, jak s nimi později manipulovat nebo jakým způsobem tyto funkce vytvořit, aby byli pohodlné pro uživatele a mít přehlednost o tom co děláme. Dále bylo potřeba vymyslet jak vytvořit vizualizaci vytvořených dat.

Po zamyšlení o požadavcích aplikace by jasné, že manipulaci a vytváření objektů by bylo nevhodné využívat konzolí a psaných příkazů, jelikož by to strašně zkomplikovalo celý proces a celkově by to bylo nepřehledné, nebyly by vidět možnosti a funkce které by uživatel mohl dělat bez zdlouhového výpisu funkcí a jejich popisu, navíc to nenabízí velkou možnost dalšího vývoje a rozšíření a další funkce. Z těchto důvodů byl vybrán další a více používaný přístup a to je vytvoření nějakého grafického uživatelského rozhraní, které by komunikovalo s uživatelem a umožnily nám tak vytvářet a manipulovat s objekty.

Uživatelské rozhraní

Při návrhu uživatelského rozhraní jsem se zamyslel co je třeba ukázát, s čím by bylo možné manipulovat, kde mohly být umístěny prvky pro tvorbu a manipulaci. Návrh se skláda ze tří hlavní částí, jejižch umístění lze vidět v obrázku 4.1. Někdy uživatel potřebuje zobrazit prvky, jejich jména, možnost je třeba nechat i skrýt ve scéně a k tomu slouží první část v návrhu, kde jednotlivé modely jsou skládány v seznamu pod sebou, kde jsou zobrazeny jejich jména a tlačítko pro určení jejich viditelnosti.



Obrázek 4.1: Návrh uživatelského prostředí[2]

Další důležitou informací je samotný model, jsou zde informace, které někdy může uživatel vyžadovat, případně i změnit. Mezi tyto informace patří, základní poloha ve scéně, počet prvků ze kterých se model skládá, jeho barva. Tuto část jsem umístil do spodní části UI. Poslední nevyužitou částí, je umístění funkcí aplikace, kterou by manipulovala se scénou, umožnila vytvářet modely, animace, atd. Tato část je v návrh pojménována jako pracovní plocha. Ostatních funkce, které by se nevlezly do těchto částí, jsem umíštil do příslušných meny do horní lišty hlavního okna.

Tvorba a manipulace objektů

Aplikace je založena na parametrickém přístupu konstruování objektů, proto jsem se rozhodl navrhnou systém, který by opomenul přímý zásah do scény nebo objektu, skrze vykreslení objekt v okně. Umístění funkcí bude v pracovní ploše UI, obr. 4.1. Zde bylo potřeba vytvořit jednotlivé sekce - konstrukce a manipulace modelu a animace. Pro úpravu modelu je nutné aby v konstrukci mohl uživatel, kdykoliv měnit parametry vložených částí, které model tvoří a mít o nich přehled. Proto by bylo jednotlivé operace uložit jako příkazy a zobrazit jako seznam instrukcí pod sebou. Přidávat nové jednotlivé instrukce vlastnoručním psaním by bylo obtížné a zabralo by čas, proto byl navržen systém, který vytvoří po kliknutí pravého tlačítka myši na menu, kde jsou tyto operace v přehledném seznamu. Po vytvoření

by uživatel prostě zadal potřebné parametry pro úspěšné vykonání instrukce a potvrdil nějakou klávesou, že skončil. Poté program tuto instrukci postupně zpracuje, zkontrolujte jeho spravnou syntaxy, kontrolu správností jednotlivých vstupních prvků instrukce a při úspěchu přidá nebo změní určitou část vytvořeného modelu.

Zobrazeni vytvořených dat

Nakonec bylo potřeba vytvořit nějakou postup pro vizualizaci jednotlivých modelů. Nejdříve je potřeba vytvořit jakým způsobem vytvořit požadavek. K tomu použito tlačítko v horní liště, ta vyvolá vytvoření okna, kde se postupně načítají data z databáze modelů a vykreslují.

4.4 Výběr vhodné technologie

Každý návrh a implementace nějaké aplikace v IT (informačních technologií) zahrnuje i výběr vhodných technologií, kde zkoumá nejvíce v vhodnost pro požadavky aplikace, dále mohou snižovat časové náklady spojené s implementací a pokud by šlo i komerční aplikaci tak i finanční náklady.

Programovací jazyk

Jeden z prvních rozhodnutí okolo projektu bylo nutné vybrat vhodný programovací jazyk. Mezi kritéria, které jsem měl pro jazyk, byl jeho schopnost vytvářet objekty, jelikož pro tuto práci byla nutnost vytvořit datové struktury, další byli možnosti využití jazyka v oblasti implementování uživatelských rozhraní a práci s grafikou.

Jedním z jazyků by byla rozhodně Java, ovšem tento jazyk je čistě objektový, což vytváří mnohdy zdlouhavé zdrojové kódy a výpočty, i když na druhou stranu má vhodné prostředí pro vytváření aplikací s GUI. Mezi další kandidáty patřil i objektový jazyk Python, jenž nabízí podporu mnoha knihoven a mohl by tak ulehčit práci při vytváření aplikace a potřebných datových struktur. Tento jazyk je taky velmi přehledný co se týče syntaxe (zápisu v kódu), ale problém při práci v tomto jazyku je zejména jeho paměťová náročnost, nízký výkon v určitých oblastech IT.

Kvůli těmto nedostatkům jsem se rozhodl práci implementovat v programovacím jazyce C++[1]. V tomto jazyku jsem již párkrát pracoval a jsem obeznámen s možnými knihovnami a technologiemi, které splňují požadované kritéria. Jeho možnost pro využití v různých projektech se může využít jak v vysokých, tak nízkých aplikacích (tnz. např: možnost práce s hardware nebo i na úrovni webových technologií). Navíc jazyk C++ není čistě objektový, což dovoluje vytvářet jednodušší zápisy v programech, a taky vytvářet menší zátěž na paměť počítače. Má také silnou typovou kontrolu což znemožňuje udělat nějakou chybu nebo mylné přepsání dat při vývoji nebo i v samotné činnosti aplikace a tak vytvářet neočekávané chování.

Datové struktury

První z problému, které bylo třeba vyřešit byl nějaký návrh datových struktur, jakým způsobem prezentovat modely v počítačové grafice. Je jásné, že třeba vytvořit určité základní elemntarní geometrické prvky jako jsou bod, usečka, přímka a polygon, což jsou základní prvky pro vytvoření složitejších objektů - tkz. polygonálních modelů vysvětlených v sekci

[1] více informací o jazyku C++: http://www.cplusplus.com/info/description

3.3. I zde byla použita určitá sada knihoven. Při tvorbě padla myšlenka si vše postupně vytvořit skrze jazyk C++, ale bylo to časove náročné a muselo by se vše programovat o začátku, což by mohlo poždeji vést k mnoha úpravám. Taky by to vedlo k vyběrům ruzných algortimů a existujích datových struktur, používaných v těchto oblastech a museli by se všechny porovnávat a zjišťovat, který přístupy jsou nejlepší nebo znovu je vytvořit, což by vedlo (jelikož knihovny pro tyto problematiky existují) vytvoření nedostatků nebo chyb.

Z těchto důvodů jsem se tedy vybral sadu knihoven zaměřených na geometrii a matematiku - CGAL (Computational Geometry Algorithms Library)[4]. Tato sada knihoven poskytuje základní a pokročile datové struktury, které jsem použil při tvorbě aplikace (základní geometrické prvky, reprezentace polygonů, polygonových modelů, logické operace mezi nimi, atd). Velmi ulehčila práci, jelikož jsem se mohl soutředit převážně na funkce uživatelského rozhraní aplikace a vizualizace dat.

Uživatelské rozhraní

K sestavení byl potřeba nějaký toolkit, jelikož jazyk c++ sám o sobě není staveň pro tvorbu uživatelského rozhraní, pokud by si to chtěl někdo postavit v dnešní sobě sám, docela by to dlouho trvalo, a vytvářel by mnoho prvků, které již jsou za tu dobu vytvořeny. Mezi známými toolkity je například GTK, jenž podporuje Linux, Windows, MacOS a umožnuje vytoření základních oken a prvků, které jsou základy UI v programech, takže pracuje na nízké urovni, což by nemuselo stačit pro některé funkce aplikace.

Nakonec jsem se rozhodl použít druhý c++ toolkit, který umožnuje velmi mocnou práci při tvorbě UI a dynamické změny UI. Jedním z důvodů proč byl využit Qt toolkit[2], při tvorbě uživatelského rozhraní a komunikace s logikou je jeho technologie **Signály a Sloty**.

Tento mechanismus nahrazuje standardní funkci *callback*, která je normálně využívána při komunikaci s objekty. Jeho nevýhodou je, že pri volání této metody, musí poznat ukazatele na metodu, která *zavolala*. Důvodem proč mechanizmus Signaly a sloty tak používaný, je právě důvod že on neví kdo ho volá, ani to nepotřebuje, neřeší který typ objektu to byl. Řeší pouze kdo příjemce a kdo odesílatel. Také zde musí platit i stených typ a počet argumentů, který se těmito signály může předat. Fungování tohoto principu je velmi jednoduché. Stačí nám někde ve zdrojových souborech vytvořit třídu z knihovny Qt a zavolat metodu *connect(Object1, signal1, Object 2, slot1)*. První objekt představuje odesilatele a jeho metoda představující signál. Druhý objekt je příjemce a jeho slot metoda co se vykoná po přijetí signálu.

Zobrazení dat

Jedním z posledních problému, které bylo potřeba navrhnou a implementovat ja nějaká vizualizace vytvořených dat v aplikaci. Potřeba k této části je jednoduché zobrazení s datových struktur v podobe nějaké kostry modelu (vyznačené body, hrany, plochy, barevně rozlišitelné). Možnost vytvořit funkce pro ovládání a tvorbu kamer, které by se připnuly k jednolivým objektům a tak být využity pro sledovaní vytvořených animací, atd.

Problém nastává jak data vizualizovat. Jsou zde jisté jazyky, které zprostředkují komunikaci s hardwareme na základě programu a data vykreslí, jako je OpenGL[3] nebo rozhraní pro programování aplikácí se stejnou funkcí jako jsou DirectX nebo Vulkan.

[2] více informací o toolkitu Qt: https://wiki.qt.io/Qt_for_Beginners
[3] více informací o OpenGL: https://www.opengl.org

Mezi první volby patřila sada knihoven - GLFW, GLM, FLEW, která přímo pracuje s OpenGL, bohužel po pár operací jsem zjistil, že bylo lepší využít renderovacích enginů. Důvodem byla velká složitost pro vytvoření jednoduchých věcí. V projektu je potřeba pouze data vizualizovat a není nutnost vytvářet rozhraní pro jejich ovládaní a podobné části. Renderovací enginy vetšinou mají určinou abstrakci a pomocí svých method a funkcí se spojují s OpenGl nebo jinými renderovacími systémy a načítají data nebo přehravají nachystané animace 3D objektů. Vybral jsem Ogre 3D [12], který má schopnost pracovat s velkým množství dat a je schopen pracovat s jakýmkoliv renderovacím systémem (GL, Vulkan, DX).

Kapitola 5

Návrh a implementace

V této kapitole je popsána implementace klíčových částí aplikace, nutných datových struktur pro činnost aplikace a reprezentace modelů s kterými pracuje. Dále návrh grafického uživatelského rozhraní pro uživatele a výběr vhodného přístupu, aby aplikace byla nenáročná ale zároveň nabízela možnost pro další možné rozšíření. Druhá čast aplikace spočívá vytvořením systému, který propojí vytvořené objekty z navržených datových struktur do systému, který by pomocí příkazů utvářel animace nebo jenom sněmi mohl poté jsou so vytvořeny manipulovat, např. prostřednictvím vytvořených parametrů. V závěru je popsán způsob zobrazení těchto modelů.

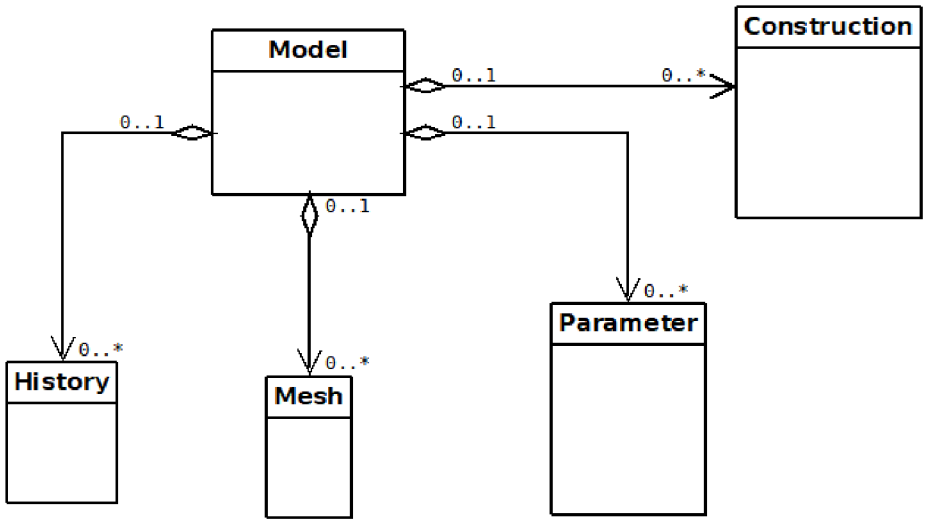
5.1 Reprezentace objektů, modelů

Při uchovávání nejen geometrických dat, ať už se jedná o různá prostorová data [3], či databáze a podobně, je zpravidla nutné vytvořit cestu, která by umožňovala provádět jejich výběr a změnu dat. Toto platí v parametrických konstrukčních nástrojů, kde se musí dávat pozor i na vytvořené vazby, které při konstrukci modelu vznikají. Při návrhu této první časti, bylo využito principu objektového orientovaného programování, jelikož nám umožňuje právě na tyto vazby dávat pozor, reagovat na ně nebo přidávat identifikaci, která je nutná abychom mohli vytvořeným modelem zpětně modelovat i poté co je vytvořen a případně vykreslen.

Následné jednoduché schéma popisuje blokově strukturu základní třídy **Model**, kde jsou uloženy skoro veškeré metody a vlastnosti pro ovládání takových vytvořených objektů.

Databáze objektů

Všechny modely jsou dále uchovány v jedné větší třídě nazývané **SceneData**, která v sobě drží databázi všechy modelů včetně jejich počtu. Tuto databázi lze ovládat pomocí method *addNewObject*, která slouží k přidání nových modelů a *deleteObject* jenž slouží jako způsob jejich odstranění. Mimo jiné je také možné tyto modely přejmenovat pomocí metody *renameObject*. Pro získání modelu z databáze je třeba volat pouze metodu zvanou *getObject*.



Obrázek 5.1: Blokové složení třídy Model

Model

Jak bylo zmíněno v úvodu, **Model** je jedna ze základních tříd v programu. Hlavní rolí této třídy je uložení veškerých informací o modelu a přístup k těmto datům pokud jsou potřeba, ať se jedná o jeho logický popis meshe sestavený z různých geometrických základních prvků, parametrů podle kterých lze upravovat jeho části nebo jako celek, či uchování podpůrných geometrických entit, např: přímka, kružnice, rovina v konstrukci.

Dále uchovává hodnoty **c__visibility**, **me __visibility**. Obě dvě jsou uloženy jako logický datový typ bool. Tyto hodnoty slouží jako pomůcka pro renderovací část programu a představuje zda se tato část modelů vykreslí do scény nebu zůstane schovaná.

Nezbytnou součástí modelu je taky schopnost uchovávat vector (uspořádaný seznam pomocí indexů $0 : N$) tříd zvaných **Operation__transaction**, kde se uchovávají veškeré provedené konstrukce, které vedli k vytvoření modelu.

Mesh

Tato třída představuje reprezentaci jak se vytvořený model reprezentuje pomocí dat. Tato část vychází z myšlenky od CGAL [4] a využívá část jeho knihovny, hlavně v oblasti vytváření mesh a základních geometrických prvků [5]. Tato třída, popisuje logickou reprezentaci objektu, dále má svůj vlastní název a určité rozhraní v podobě metod třídy, které dovolují tyto části modifikovat.

Hlavní část **Mesh** je zděděna od knihovny CGAL, přesněji jde o třídu **Surface__Mesh**[3][6], kde tato třída používá popis objektu jako množinu 3 základních geometrických útvarů v 3D prostoru: bod, úsečka/hrana, plocha. Princip, který tato třída používá pro sestavení meshe z těchto základní prvků se nazývá **halfedge** (půl-hrana) datová struktura. Každá hrana v meshi je rozložena právě na tyto dvě, které zároveň leží proti sobě. Každá z nich uchovává svou stranu, v ní se dá najít reference na body z nichž je sestavena, předchozí a následující half-edge v určité menší uzavřené skupině a nakonec plochu které náleží. Nachází se zde vertices (body) a také faces (plochy) a celé hrany [6]. Třída tyto všechny prvky uchovává pomocí indexů a ne reference mezi jednotlivými objekty. To znamená, že všechny části jsou uspořádané, nejsou závislé na jiných částech Meshe a hlavně mají menší náročnost na vý-

počet. Nevýhodou tohoto přístupu je obtížná zpětná modifikace dat. Proto bylo potřeba vytvořit metody, které tyto indexy, když se nová elementární část přidává, uchová v nějakém seřazeném seznamu a to pro všechny prvky ze kterých se objekt skládá (body, hrany, plochy). K tomu účelu posloužila standardni C++ datová struktura **vector**. Tyto metody se nazývají: *addNewPointIndex*, *addNewEdgeIndex*, *addNewFaceIndex*.

Zbytek je třídy je vytvořen tak, aby změnil data jako je jméno, ulehčil přidání základních geometrických prvků: *setName*, *makeTriFace*, *makeQuadFace*. Poslední dvě metody slouží k rychlejšímu vytváření trojúhelníkových nebo čtvercových ploch, které jsou standardem při vytváření složitějších objektů v 3D prostoru.

Konstrukce

Jelikož 3D objekty jsou popsány množinou bodů, hran a ploch a navíc jsou některé z nich složeny z velkého počtu těchto prvků, musí se uvažovat při konstrukci o abstraktnější způsobech vytváření. Proto byl vytvořena pro tento účel třída **Construction**. Je to taková další menší databáze pro ukládání geometrických prvků jako jsou přímky, kružnice, vektory, atd, které mohou sloužit jako podpora při konstrukci a také to nabízí nové možnosti konstruování.

Druhým důvodem je uchování těchto prvků, když vytváříme objekty skrze geometrická pravidla, např: vytvoření bodu, který prochází touto přímkou, vytvoření kolmých úhlů nebo vytvoření části pod nějakým úhlem, atd. Tyto vazby nejdou v modelech normálně vidět a po sestrojení modelu už nejsou ani moc důležité pokud ovšem nechce uživatel určitou část změnit později. Jsou zde standardní metody pro přidání, a odebrání těchto prvků, vice lze nalézt v hlavičkovém souboru construction.h.

Parametr

Jeden posledních elementů, které by mohli manipulovat s vytvořeným modelem se nazývá **Parameter**. Jeho část je důležitá pro ovládání prvků objektu dynamicky. Pokud ovšem nepočítáme samotnou konstrukci, kde můžeme využít i ostatní geometrické prvky nacházející se v konstrukci 5.1. Samotná struktura třídy je velmi jednoduchá. Jedná se pouze o množinu referencí v podobě klíčových strings (řetězců): p1, p2, e2, m2, f5, atd (bod, hrana, model, plocha), uložených v seznamu. Ve třídě se také nachází řetězec představující jméno parametru, který později slouží pro hledání parametrů v modelu. Zpracování parametrů se děje a přes samotnou konstrukci objektů nebo když se využívají k manipulaci či vytváření animací. Případě konstrukce, si program přečte jednotlivé reference, vytáhne potřebná data z objektu Model a aplikuje na tuto množinu transformační operaci 3.1. Záleží na typu parametru.

Historie konstrukce objektu

Sice jsou vytvořené datové struktury, které reprezentují nějaký model nebo skupinu objektů složeny v jeden model, ale všechno to je jenom jedna vykonaná nenávratná akce, jelikož není nikde uložen popis kroku, který k sestrojení modelu vedl. Proto byl projektu vytvořen seřazený seznam nazývaný **History**.

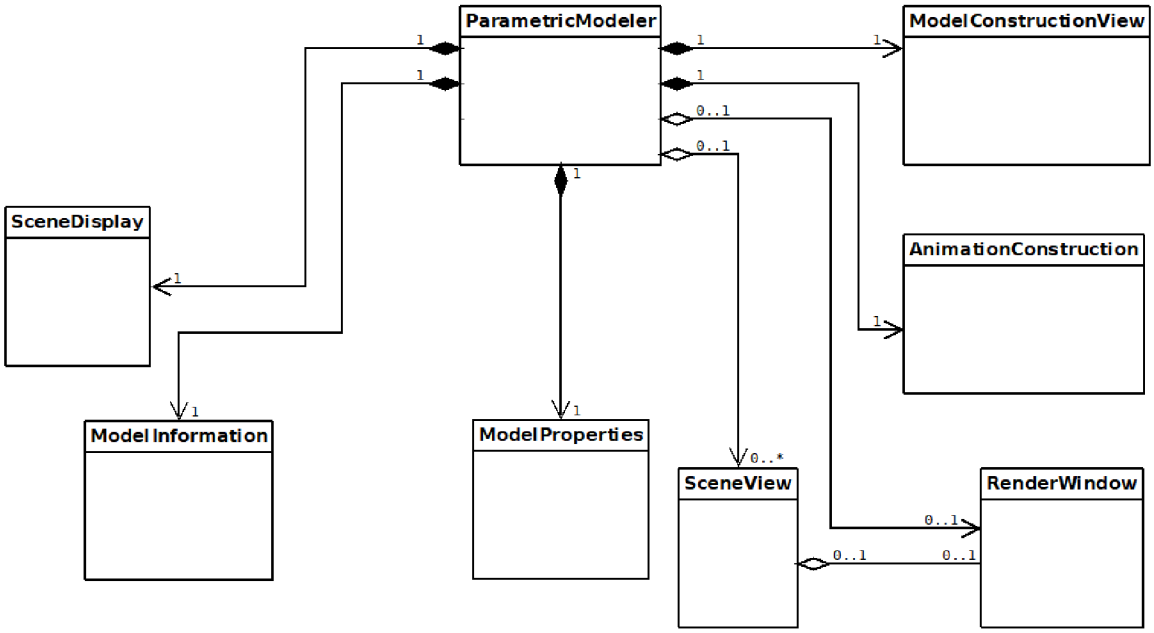
V tomto seznamu se nalézají datové struktury jménem *Operation_transaction*. Tato struktura ukládá potřebné informace o poslední úspěšně provedené konstrukční operaci a dovoluje aby se později tyto kroky mohl modifikovat. Pokud takové modifikaci dojde, může se přepočítat až celý model.

V samotné struktuře se nalézá:

- item - Představuje jméno prvku modelu (bod, hrana), neboli jeho referenci, diky níž se ví, která část modelu byla upravena, nebo přidána.
- operation - Popisuje typ operace, která se nad modelem vykonala. Viz o typech operací lze nalézt v sekci 5.3
- args - Je uspořádaný seznam ve kterou jsou potřebné hodnoty, které byli použity při provádění operace (parametry jedné operace)

5.2 Uživatelské rozhraní

K vytvoření teté částí bylo využito Qt frameworku [15] jak bylo zmíněno v sekci 4.4. Celé rozhraní je tvořeno kolem několika bloků, hlavní částí programu je třída **ParametricModeler**. Kolem této třídy jsou vytvořeny další bloky jenž slouží buď k zobrazení logických dat nebo také i ke jejich modifikace přes interakci od uživatele. Je zde navržena celé logika programu, jelikož spojuje jak grafické rozhraní tak i logickou část, kterou představuje databáze a ostatní datové struktury zmíněné v sekci 5.1. Podle potřeby posílá informace do ostatních bloků nebo zpracovává jejich požadavky. V následujícím obrázku lze vidět blokovou strukturu z jakých částí se tato třída skládá.



Obrázek 5.2: Struktura Grafického Rozhraní

SceneDisplay

Jednou z prvních částí, je třída zvaná **SceneDisplay**. Její hlavní úlohou je zobrazovat obsah databázi z hlavní části programu, při načítání, postupně zobrazí jednotlivé prvky, včetně odkazů na jejich mesh a konstrukci zmíněné v sekci 5.1. Jednotlivé prvky v tomto vytvořeném seznamu může uživatel vybírat. Jeho výběrem se vytvoří požadavek, který upozorní

hlavní část programu a poté další součásti, že došlo výměně a podle toho aktualizuje tyto ostatní části. Dále toto rozhraní nabízí možnost přejmenování těchto vytvořených modelů, za pomocí dvojkliku pomocí myši nad jejich jmény. V pravé části těchto buněk se nachází graficky vytvořená ikona představující viditelnost objektu (logická hodnota bool), jsou zde dva možné stavy: viditelný a schovaný.

ModelConstructionView

Další části je třída zvaná **ModelConstructionView**. Je složená ze dvou prvků, jeden pro konstrukci modelu a případně jeho modifikace a druhá pro aktivací renderovací okna, který lze schovat podle potřeby klavesnicovou zkratkou F5. V té první části je zde vykreslen seznam načítající historii konstrukce objektu. Jednotlivé buňky jsou vytvořeny jako obyčejné řetězce dokud uživatel na ně neklikne, čímž je změnit na řadu vstupních polí do kterých lze zadat příslušné hodnoty. Tato problematika byla blíže popsána v sekci 5.3.

AnimationConstruction

Tato část pro tvorbu animací nebo alespoň základních transformací nad modely. Hlavní část je tvořena podobně jako předchozí část ve **SceneDisplay**. Je zde seznam, který podobně ukládá a zobrazuje jednotlivé kroky v podobě řetězce. Poté se tyto úkony když se zavolá požadavek na aktualizaci nebo vznik nějaké animace postupně přečte a vykoná. I tato část rozhraní má, vedle logické části také renderovací okno pro vykreslení výsledků, které uživatel provedl s danými modely.

ParameterConstruction

Toto rozhraní slouží převážně k tvorbe parametrů v modelu. Tvoří ji dva seznamy, kde první část slouží k zobrazení vytvořených parametrů, s možností měnit jejich jména a pod touto buňku je zde seznam části modelu, kterých se tento parametr týká, pokud tedy nepředstavuje obyčejnou číselnou hodnotu, která by šla použít pro animační část. Druhý seznam představuje popis objektu v podobě množiny bodů, hran a ploch. Jejich jména odpovídá jejich pozici v meshi. Stejně jako ostatní části, které mají společnou nějakou manipulaci s modelem mají v sobě schované renderovací okno pro zobrazení.
Ostatní součásti **SceneView** a **RenderView** slouží jako rozhraní pro vykreslování našich vytvořených dat, popsaných v sekci 5.4.

5.3 Vytváření animací a manipulace s objekty

Manipulace s objekty nebo pouze jeho konstrukce je propojená s uživatelským rozhraním. Rozděluje se na akce uživatele a vykonání zadané operace od něj. Jména typu operací se nachází v souboru **enum.h**. Konstrukce modelu se provádí přes 2 částí. Z pohledu uživatele, je zde třída **ModelConstructionView**, která vytváří grafické rozhraní v podobě seznamu, do kterého můžeme postupně přidávat nové operace a tak konstruovat daný model. Po zobrazení této části UI (uživatelského rozhraní), klikneme na levou část pravým tlačítkem myši a objeví se meny s možnými příkazy. Po vybrání se zobrazí v seznamu a dvojím klikem začneme editovat. Po potvrzení se daná operace odešle ke zpracování, ještě předním se ovšem zkontroluje syntax dat, zda-li jsou zde pouze povolené symboly. Mezi základní operace, které program podporuje patří přidání bodu, vytvoření hrany, troj-plochy nebo čtyr-plochy, atd.

Druhá část se zaměřuje na logickou část, kde jsou informace zadané uživatelem převedeny a zpracovány. Tuto operaci vykonáva metoda *executeOperation* v hlavní třídě celého programu **ParametricModeler**. Před vykonáním program zkontroluje zda se jedná o vykonání nové operace nebo aktualizaci. Jednotlivé data předané předchozí třídou **ModelConstructionView** se přečtou a zjistí o jaký typ operace jde. Model není potřeba zjišťovat protože program je schopen rozlišit, které model je aktivní.

Animace modelu

Animace objektu, představuje jenom další typ instrukcí, které dále manipulují s objektem, případně ho i modifikují, např: pohyb objektu, změna velikosti, rotace, atd. Tato část se vykonává především v třídě **AnimationConstruction**. V této části programu je zde jednoduchý vytvořený seznam, který funguje na dávkovacím principu. Naplníme zásobník operacemi a postupně koukáme jak se zpracovávají a tento výsledek se promítne do renderovací části. Každá operace zmíněná v sekci 5.3 podobu funkce/operace, která má určité parametry, které přijímá. Mezi hlavní operace patří MOVE (posun), ROTATE (rotace), SCALE (změna měřítka). Pokud předtím byli použity parametry, je možné modifikovat objekt i bez těchto vlastností. Druhá část spočívá jaké parametry do těchto základních operací vloží. Struktura jde následujícím způsobem: Jméno modelu, číslo představující o kolik operace upraví daný objekt a v poslední řadě funkce např: sin() jenž nám určuje plynulost animace a také jeho rychlost. Veškeré operace se opět nachází ve souboru operations.

Narozdíl od první části se animace a transformace, které k němu vedou provádí pouze AnimationConstruction a tyto změny se neukládají jako změny v konstrukci. Všechny kroky, které model modifikují se zároveň převádí do přijatelných datových struktur, které podporuje Ogre 3D [12].

5.4 Zobrazení modelů

I když v projektu jsou všechny prostředky jak sestrojit, jediné co by jsme dostali je hromada údajů v nějaké struktuře. K tomuto účelu jsou často tyto data vykreslovány abychom měli nějakou grafikou reprezentaci a mohli jsme si tyto data lépe představit.

Pro tuto problematiku slouží dvě třídy: **CameraManager**, **RenderWindow**. Obě tyto části byli převzaty z [**?**]jenž umožňuje integrování technologie Ogre [12], Qt frameworku [15], dále upraveny podle potřeb v aplikaci. Cílem je aby se data v určité reprezentaci vykreslily v nějakém okně.

První část RenderWindow se nachází renderovací okno **Ogre:RenderWindow**, které představuje naše okno v kterém se budou naše objekty vykreslovat. Dále **SceneManager**, kde úkolem je zpracovat a uchovat veškeré data potřebné k vyrenderování, včetně objektů které se před vykreslením museli transformovat na datové struktury, které tyto enginy používají. V této třídě se nachází metody *render* a textitrenderNow. První slouží spíše jako vnitřní metoda pro renderování jednoho snímku do okna, ale jelikož chceme aby toto vykreslování bylo automatické a nereagovalo pouze na změny a mohli se doslova procházet ve scéně když se provádí nějaká animace, tak se raději používá ona druhá metoda. V prvním spuštěním okna se tato druhá metoda nemusí ani volat jelikož se při inicializaci okna zavolá sama.

Druhá část (třída) CameraManager se zaměruje pouze na možnosti průchodů ve scéně. Tato třída nám umožňuje využívat kameru ve scéne dle vaší libosti, jsou zde naprogramovány metody, které dovolují touto kamerou pohybovat ve scéně pomocí myší a klávesnico-

vých zkratek nebo vytvořit přes SceneManager přinutí kamery na objekt a měnit rychlosti jeho pozorování.

5.5 Popis řešení implementačních problémů

Reprezentace objektů

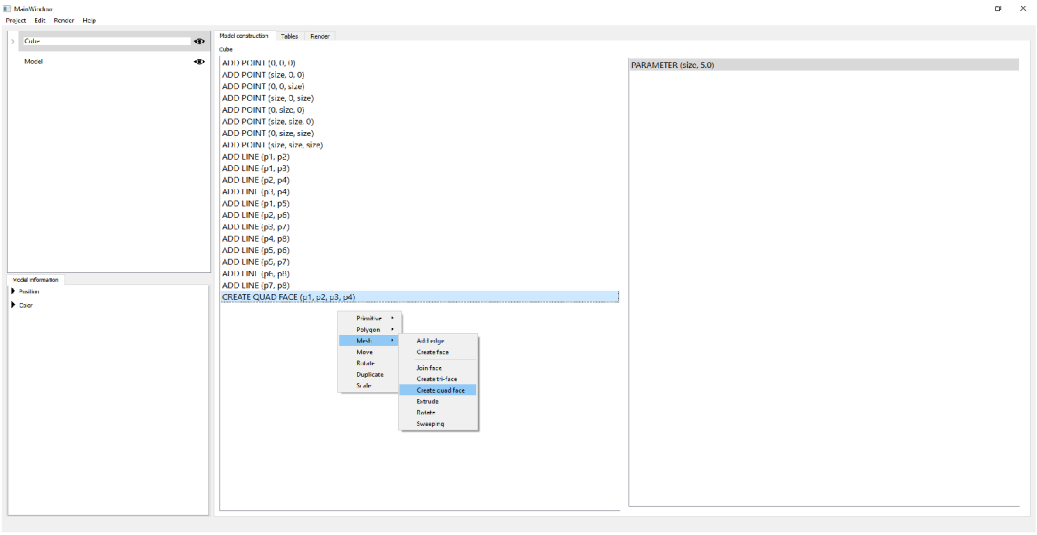
Při první tvorbě reprezentace objektů v projektu padla myšlenka, zda by bylo vhodné se reprezentaci geometrických útvarů a komplexních mesh udělat jenom pomocí jazyka C++. Takový přístup je většinou používán při sestavování nějakých programů, ale s vzrůstajícího počtu open source je lepší využít matematicky zaměřené knihovny, které mají většinu problémů, se kterými se můžeme setkat v takových tématech vyřešené. Z tohoto důvodu byl pro projekt nakonec využit projekt CGAL[5], který obsahuje velkou hromadu knihovně se zaměřením právě na problematiku reprezentace geometrických prvků, komplexních objektů, sestavování modelů z map a podobných témat.

I když CGAL dokáze logicky reprezentovat tyto geometrické prvky, má i své nevýhody, je destruktivní. To znamená, že skoro každá modifikace dat ať se jedná o souřadnice bodu nebo změnu v meshy, jiných geometrických entit, většinou končí vytvořením duplikátu s novými daty a přepisem těchto dat. Jinými slovy to chybí docela dost metod a funkcí, které by dovolili nastavování nových hodnot.

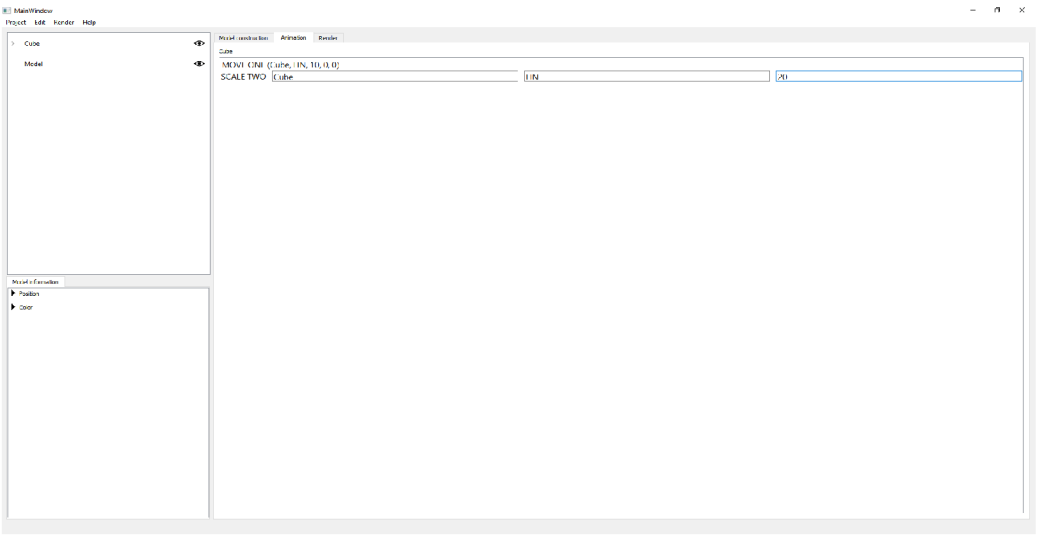
Další z nevýhod je fakt, že většina entit, které popisují struktury jako je mesh, polygon, atd., jsou uloženy indexově. Tedy není způsob jak poslední elementy, které přidáme odkazovat, z tohoto důvodu je nutnost tyto vytvořené indexy někde počas manipulace těchto entit uložit. Toto řešení bylo aplikováno na část projektu, popsány v sekci 5.1.

5.6 Testování uživatelské rozhraní

Při testování uživatelského rozhraní bylo zjišťováno jaký by byl nejlepší přístup pro vytváření úkonů od uživatele, jak zobrazit data, které jsou z důvodu zaměření hlavně na parametrickou konstrukci docela rozsáhlé. Testy také zjišťovaly rychlost tvorby a manipulace jednotlivých objektů ve scéně skrze příkazy. Testy bylo dokázáno, že aktuální verze není sice dokonalá, ale uživatel se nemusí zdržovat s pop-up (vyskakujícími okny) nebo vypsat příkazy ručne, takže se celá rychlost konstrukce podstatně zvyšuje. Kdyby zde byly jenom pop-up okna, uživatelé by museli do každých části psát udaje, vybírat typ metody, atd. Tento přístup by sice mě své výhody, ale pro aktuální funkci programu je to nepodstatné. Řešení pomocí výběrových menu je pro aktuální přístup nejlepší.



Obrázek 5.3: Ukázka testování tvorby objektu



Obrázek 5.4: Ukázka testování animace objektu

Druhým problém, který se při testování uživatelského rozhraní řešilo, byl jaké rozložení pro zobrazení dat použít. V mnohých podobných kreslících nástrojích, jsou zde dvě cesty. První komplexní přístup, kde se každá část, která lze modifikovat, má nějaké vlastní uživatelské rozhraní a druhá je většinou jednoduchý seznam příkazů, které je mohou modifikovat a vedle je většinou vykreslování. Nakonec testováním bylo zjištěno, že by byl vhodnější využít přístup, který se blíží k první formě a i když se zdá, že by uživatel mohl při spuštění aplikace být zmatený, nabízí tento přístup možnost snadného rozšíření o další funkce a využít celé pracovní prostředí aplikace, zatímco vykreslování objektů zůstává oddělené.

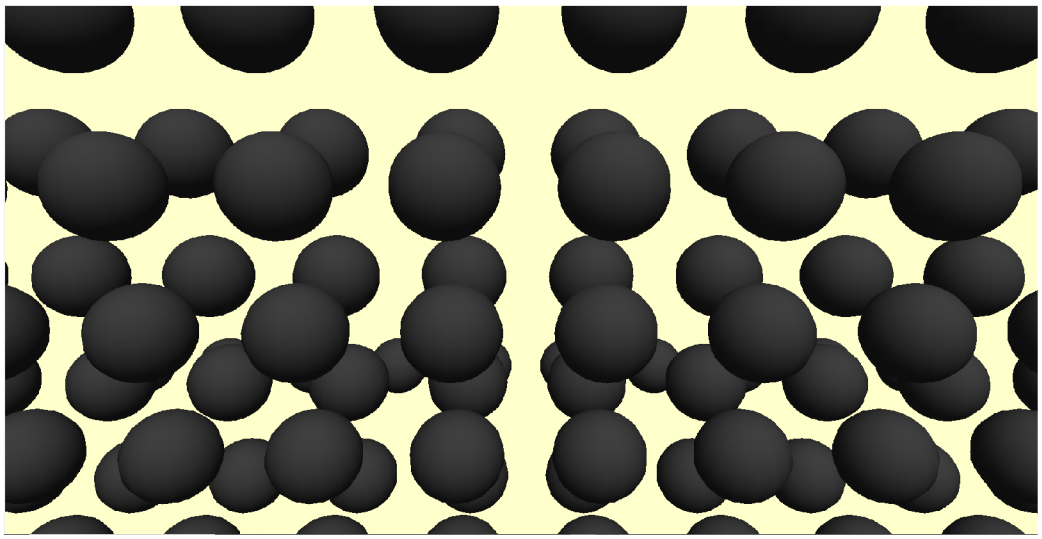
5.7 Vykreslování a testování zátěže

Samotný kreslící nástroj a renderovací část po engine Ogre 3D [?] je schopen vykreslovat poměrně rychle ať už hlavní výpočetní jednotkou je CPU nebo GPU. Jedním z problémů, ale při implementaci zůstává množství dat a čtení, které se dějí předím. Tyto akce zpomalují jednotlivé snímky a aktualizování změn na vykreslených modelech. Cílem testů bylo zjistit jak moc se změní výkon a stabilita renderování při větším objemu 3D objektů a případně i spuštěných animací. Výsledky lze vidět v následující tabulce:

Počet vykreslených objektů	Počet snímků za sekundu
1	60fps
5	60fps
10	55-60fps
50	47-60fps
100	30-40fps

Tabulka 5.1: Přehled vlastností konstrukčních nástrojů

Jedno z testování obsahovalo následující vytvoření prvků.



Obrázek 5.5: Ukázka testovaní výkonosti renderu

Kapitola 6

Závěr

Cílem této práce bylo navrhnout a realizovat parametrický konstrukční nástroj pro podporu zejména 3D geometrie a modelů. Tento úkol byl splněn a výsledkem je aplikace, která je popsána v ostatních kapitolách tohoto projektu.

Obsahem druhé kapitoly byl úvod do parametrických konstrukčních nástrojů a jejich historie. Dále zde byly předvedeny vybrané konstrukční nástroje s popisem jejich přístupů a tvorbě objektů skrze parametrickou konstrukci nebo jejich kombinací s přímou konstrukcí. Třetí kapitole byly popsány jednotlivé základní pojmy v oblasti geometrie a počítačové grafiky využívané pro aplikaci, včetně představení jejich možných datových struktur. Konec kapitoly nám popsal rozdíly parametrické a přímé konstrukce objektů. Čtvrtá kapitola popsala existující řešení problematiky v této práci, byla provedena analýza zadání a sestavení požadavků pro aplikaci, její návrh a výběr vhodné technologie podle sestavené specifikace. Pátá kapitola popsala způsob implementace jednotlivých datových struktur s využitím použitých technologií, grafického uživatelského rozhraní. Druhá část kapitoly popsala jakým způsobem byla implementována problematika pro manipulaci s vytvořenými objekty z navržených datových struktur a jejich způsob zobrazení. V šesté kapitole byli předvedeny jednotlivé testy aplikace, zjišťující vhodnost a přehlednost implementované uživatelského rozhraní pomocí instrukcí a parametrů pro manipulaci scény. Výsledek testu byl ucházející i když by bylo pro další verzi aplikace asi vhodnější více interaktivní rozhraní kde by parametry mohli být nastavovány více dynamicky. Druhou částí testů bylo zejména testování renderování objektu, přesněji zátěž při vykreslení modelů nebo provádění animaci včetně pohybu kamery ve scéně. Dosažení výsledků a jejich vyhodnocení lze najít v následujících odstavcích, včetně pár vět o možných postupech dalšího vývoje a rozšíření aplikace.

Výstupem této práce je 3D nástroj pro konstrukci objektů a jejich manipulace, umožňující tvorbu těchto prvků pomocí parametrů nebo jednotlivých konstrukcí. Předpoklad pro použití tohoto projektu je zejména znalosti základů konstrukční geometrie a znalost ovládání s počítačem. Na implementaci této práce bylo zejména obtížné zkombinovat více knihoven dohromady, který jsou samostatně komplexní.

Další vývoj této aplikace by mohl směřovat k většímu rozšíření konstrukcí, které by mohl generovat složitější modely (geometrické prvky) za využití parametrů. Například vytvořit tímto přístupem abstraktnější tvorbu scén než kontrolovat a manipulovat každou část modelů detailně. Pokud by mělo jít o rozšíření uživatelské rozhraní, mělo být více zobrazení komplexnější informace o scéně a modelech, čí přidání nových dat, které přímo nesouvisí s objektem, ale mohli by mu dát podobu jako například: popis materiálu, hmotnosti a podobných prvků. Poslední rozšíření by mohlo být zaměřeno na zobrazování scény a prvků v ní. Mít přehlednější zobrazení, více možností průchody scénou. Pokud by na takové rozšíření

mělo dojít, bylo možné také uvažovat o jiných přístupech implementací a technologií, které jsou v dnešní době používanější. Příkladem mohou být modernější renderovací nebo herní enginy.

Literatura

[1] Alba, M. *What's the Difference Between Parametric and Direct Modeling?* [online]. Červen 2018 [cit. 2020-05-24]. Dostupné z: https://www.engineering.com/DesignSoftware/DesignSoftwareArticles/ArticleID/16587/Whats-the-Difference-Between-Parametric-and-Direct-Modeling.aspx.

[2] *AutoCAD* [online]. 2018 [cit. 2020-05-24]. Dostupné z: https://www.autodesk.cz/products/autocad.

[3] Botsch, M., Pauly, M., Kobbelt, L., Alliez, P., Lévy, B. et al. *Geometric Modeling Based on Polygonal Meshes* [online]. Computer Graphics Laboratory - ETHzürich, 2008 [cit. 2020-07-14]. Dostupné z: https://cgl.ethz.ch/Downloads/Publications/Tutorials/2008/Bot08a/eg08-tutorial.pdf.

[4] *The Computational Geometry Algorithms Library* [online]. 1995-2020 [cit. 2020-05-24]. Dostupné z: https://www.cgal.org/.

[5] *CGAL Documentation* [online]. 1995-2020 [cit. 2020-05-24]. Dostupné z: https://www.cgal.org/documentation.html.

[6] *Surface Mesh* [online]. 1995-2020. Dostupné z: https://doc.cgal.org/latest/Surface_mesh/index.html''cited=.

[7] Davis, D. *A History of Parametric* [online]. Červen 2013 [cit. 2020-05-24]. Dostupné z: https://www.danieldavis.com/a-history-of-parametric/.

[8] Chakravorty, D. *The Most Common 3D File Formats* [online]. ALL3DP, srpen 2019 [cit. 2020-05-24]. Dostupné z: https://all3dp.com/3d-file-format-3d-files-3d-printer-3d-cad-vrml-stl-obj/.

[9] *GeoGebra software* [online]. 2020 [cit. 2020-05-24]. Dostupné z: https://www.geogebra.org/.

[10] Hildenbrand, D. *Foundations of Geometric Algebra Computing.* 2013th. Springer, 2013. ISBN 978-3-642-44572-9.

[11] Mortenson, M. *Mathematics for Computer Graphics Applications.* Second Edition. Industrial Press, 1999. ISBN 9-780831-131111.

[12] *Object-Oriented Graphics Rendering Engine* [online]. 2018 [cit. 2020-05-24]. Dostupné z: https://www.ogre3d.org/.

[13] Kintel, M. *OpenSCAD Documentation* [online]. 2018 [cit. 2020-05-24]. Dostupné z: https://www.openscad.org/documentation.html.

[14] Kintel, M. *OpenSCAD* [online]. 2018 [cit. 2020-05-24]. Dostupné z: https://www.openscad.org/.

[15] *Qt - cross-platform application development framework* [online]. 2006 [cit. 2020-05-24]. Dostupné z: https://www.qt.io/.

[16] Shene, D. C.-K. *Geometric Transformations* [online]. 1997-2014 [cit. 2020-04-24]. Dostupné z: https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/geometry/geo-tran.html.

[17] Wikipedia. *Computer-aided design* [online]. Wikipedia, prosinec 2003 [cit. 2020-05-24]. Dostupné z: https://en.wikipedia.org/wiki/Computer-aided_design.

[18] Wikipedia. *History of Computer-aided design* [online]. Wikipedia, únor 2019 [cit. 2020-05-24]. Dostupné z: https://en.wikipedia.org/wiki/History_of_CAD_software.

Příloha A

Obsah přiloženého paměťového média

Paměťové médium obsahuje následující složky a soubory.

Parametric_Modeler	Složka obsahuje zdrojové kódy aplikace, včetně knihoven potřebné k sestavení
build	Složka obsahuje přeloženou aplikaci k testování
tex	Složka obsahuje zdrojové soubory pro sestavení tohoto dokumentu
xsober00-Parametricke-2D-3D-modely.pdf	Soubor obsahuje tento dokument v elektronické podobě
Testing.mp4	Ukázka práce s aplikací
README.txt	Soubor obsahuje nápovědu ohledně překladu a spuštení apliakce a obsah pamětového media