



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 3D MODELY
PARAMETRIC 3D MODELS

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE
AUTHOR

Bc. MICHAL ONDREJÓ

VEDOUCÍ PRÁCE
SUPERVISOR

prof. Dr. Ing PAVEL ZEMČÍK

BRNO 2020

Zadání diplomové práce



22905

Student: **Ondrejó Michal, Bc.**

Program: Informační technologie Obor: Počítačová grafika a multimédia

Název: **Parametrické 3D modely**

Parametric 3D Models

Kategorie: Počítačová grafika

Zadání:

1. Prostudujte dostupnou literaturu, dostupná řešení i software pro 3D modelování s parametrickým vstupem.
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 systému včetně návaznosti na aplikační programy (může být i nadstavbou nad vhodným CAD systémem).
4. Implementujte a demonstrujte na vhodném příkladě, například na výsledcích simulace.
5. Zhodnoťte výsledky a diskutujte možnost další práce.

Literatura:

- Dle pokynů veducího

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: 20. května 2020

Datum schválení: 6. listopadu 2019

Abstrakt

Cieľom tejto práce je navrhnúť možnosti previazania objektov v parametrickom modele. Jednotlivé možnosti sú implementované v knižnici na tvorbu parametrických trojrozmerných modelov, ktorá umožňuje tvorbu modelov pomocou rôznych geometrických operácií, zmenu parametrov v ľubovoľnom čase, animovanie vytvoreného modelu a uloženie parametrického modelu v špeciálnom formáte, ktorý je čitateľný pre človeka.

Abstract

The aim of this work is to propose possibilities of interconnection of objects in parametric model. Individual options are implemented in the parametric three-dimensional modeling library, which allows the creation of models using various geometric operations, change parameters at any time, animate the created model, and save the parametric model in a human-readable format.

Klúčové slová

parametrické modely, 3D, grafika, parametre

Keywords

parametric models, 3D, graphics, parameters

Citácia

ONDREJÓ, Michal. *Parametrické 3D modely*. Brno, 2020. Semestrální projekt. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Dr. Ing Pavel Zemčík

Parametrické 3D modely

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána prof. Dr. Ing. Pavla Zemčíka. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Michal Ondrejó
12. apríla 2020

Poděkovanie

Týmto by som sa chcel poděkovat vedúcemu práce prof. Dr. Ing. Pavlovi Zemčíkovi za odbornú pomoc a nápady pri tvorbě této práce.

Obsah

1	Úvod	2
2	Parametrické trojrozmerné modely	3
2.1	História parametrických modelov	3
2.2	Systémy pre tvorbu parametrických modelov	5
3	Geometrické objekty a operácie	9
3.1	Bod	9
3.2	Úsečky	15
3.3	Plošné objekty	18
3.4	Objemové objekty	20
4	Zhrnutie stavu práce	23
4.1	Koncept práce	23
4.2	Skladanie operácií	24
4.3	Pridávanie a odoberanie objektov	25
4.4	Výrazy	27
4.5	Grafické rozhranie	30
4.6	Príklady použitia knižnice	33
4.7	Metrika	33
5	Záver	35
	Literatúra	36
	A Zoznam geometrických operácií	38

Kapitola 1

Úvod

Objekty vo svete sú rôzne, líšia sa tvarom aj veľkosťou. Každý si pod názvom nejakého objektu môže prestaviť trochu iný tvar a mohol by ho trochu inak vytvoriť. Na to slúži návrh objektu, ktorý obsahuje základné informácie o objekte. Na zdieľanie a uchovanie tohto návrhu je potrebné ho zaznamenať, zvyčajne zakresliť na papier pomocou ceruzky. Kedysi museli návrhári stráviť aj hodiny práce s kreslením na papier a pri každej zmene tento návrh pracne prekreslovať. Počítače sa ale stále zdokonaľujú a preto nie je prekvapením, že v dnešnej dobe nahradili ceruzky a papier a stále viac pomáhajú s návrhom modelov.

Úlohou tejto práce je vytvoriť knižnicu na tvorbu trojrozmerných parametrických modelov a prácu s nimi. Parametrické modelovanie využíva rôzne geometrické operácie na tvorbu modelov a umožňuje zmenu ich parametrov, v ľubovoľnom čase.

Aplikácia by mala umožňovať prácu programátorského a grafického rozhrania. Programátorské rozhranie by malo slúžiť na využitie parametrických modelov v aplikáciach a na nastavovanie hodnôt parametrov modelu. Grafické rozhranie by malo umožniť návrhárovi jednoduchú tvorbu modelov a tiež zobrazenie modelu a jeho animáciu v čase. Hodnota u parametrov v parametrickom modeli sa musí dať ľubovoľnom čase meniť.

Následujúca kapitola sa zaobráva parametrickými modelmi, ich tvorbou a ich históriou. V tejto kapitole sú popísané existujúce programy, pre tvorbu parametrických modelov.

V ďalšej, tretej kapitole sú opísané geometrické objekty a geometrické operácie. Tieto operácie sú rozdelené na štyri druhy, podľa toho aký typ objektu vytvárajú. Sú to bodové geometrické operácie, úsečkové, plošné a objemové.

V štvrtnej kapitole je opísaný koncept práce a súčasný stav práce. Popis a postup vytvárania geometrických modelov pomocou skladania geometrických operácií a použitie rôznych možností pri zápisе výrazu v parametroch geometrických operácií. Táto kapitola tiež obsahuje príklady použitia navrhnutej knižnice.

Posledná kapitola obsahuje zhrnutie práce a rôzne možnosti pre ďalšie rozšírenia a vylepšenia.

Kapitola 2

Parametrické trojrozmerné modely

Táto časť práce sa zaobera parametrickými modelmi. Ich históriaou a existujúcimi systémami určenými na tvorbu parametrických modelov.

Typickým návrhovým médiom je ceruzka a papier. Presnejšie je to ceruzka, guma a papier. Ceruzka pridáva a guma odoberá. Po pridaní niekolkých nástrojov, ako pravítko, kružidlo a uhlomer sa kresby stanú presnejšími a precíznejšími modelmi navrhovanej idei. Dizajnéri tieto značky pridávajú, odoberajú a spájajú ich [23].

Konvenčné návrhové systémy fungujú práve na tomto princípe. Parametrické modelovanie predstavuje zásadnú zmenu. Značky, ktoré sú základom návrhu spolu súvisia a vzájomne menia svoju pozíciu. Dizajnéri už nemusia iba pridávať a odoberať, ale môžu vytvárať medzi bodmi vzťahy a upravovať model, aby aj po zmazaní, časti závisiace od zmazaných častí mohli závisieť od častí, ktoré zostali [23].

Modelovanie sa rozdeľuje na tri druhy [18]:

Povrchové modelovanie (Surface) Tento typ modelovania je založený na systéme NURBS (Non-uniform rational basis spline). Je to technika, ktorá umožňuje prirodzenejšie tvary z kriviek. Je to vynikajúca metóda pre hladké tvary, ako sú karosérie automobilov alebo lopatky turbín.

Parametrické modelovanie (Parametric) Najpoužívanejší typ modelovania u profesionálnych dizajnérov, kvôli potrebe vysokej presnosti mechanických častí. Princípom je definovať parametre komponentov a rôznych funkcií. Komponenty sú potom parametricky riadené a ľahko upravované vďaka stromu histórie. Umožňujú matematicky definovať vzťahy pre parametre, čo umožňuje zmenu konštrukcie objektu pomocou zmeny niekoľkých hodnôt.

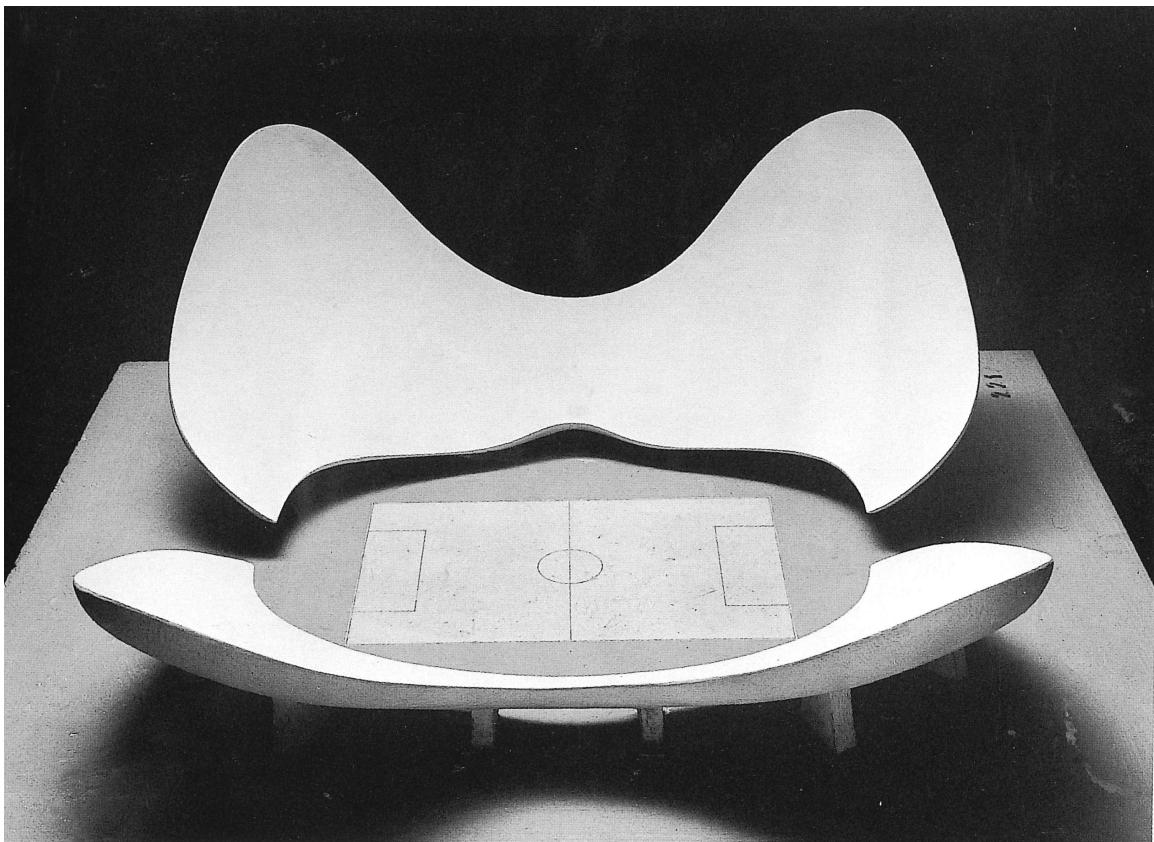
Priame modelovanie (Direct) Je jednoduchšie ako parametrické modelovanie. Odstránenie alebo presunutie niektoréj časti modelu je jednoduché, stačí s ňou pohnúť. Narozenie od parametrického modelovania, nie je reprezentovaný stromom operácií a teda sa netreba obávať, že sa celý model rozbitie.

2.1 História parametrických modelov

Pôvod slova *parameter* pochádza z matematiky. Popisuje matematickú metódu, ktorá používa nezávislé premenné, zvané parametre. Debatuje sa ale ohľadom toho, kedy začali

dizajnéri používať toto slovo [7]. David Gerber v svojej doktorskej práci *Parametric Practice* (2007) pripisoval Maurice Reiter prvé použitie tohto slova na papieri s názvom *Parametric Design* z roku 1988. V roku 1988 bol tiež vydaný prvý komerčne úspešný softvér pre parametrické modelovanie **Pro/ENGINEER** firmou Parametric Technology Corporation založenou matematikom Samuelom Geisbergom v roku 1985.

Robert Stiles ale dokázal, že skutočný pôvod tohto slova bol už o niekoľko dekád skôr pomocou zápisov architekta Luigi Morettiho z rokov 1940. Moretti písal o parametrickej architektúre, ktorú definoval ako architektonické systémy s cieľom definovať vzťahy medzi rozmermi závislými od rôznych parametrov. Moretti v roku 1960 použil ako príklad dizajn štadióna 2.1, kde vysvetlil, ako sa môže štadión meniť pomocou devätnásťich parametrov, ktoré zahŕňali pozorovacie uhli a cenové náklady na betón. Tento model vytvoril už za pomoci počítača 610 od spoločnosti IBM[9]. O pár rokov neskôr Moretti navrhol Watergate Complex 2.2, ktorý je prou výstavou stavebnej práce, na ktorú boli využité počítače [7].



Obr. 2.1: Model štadióna N od Luigi Moretti. Tento model bol vystavený na výstave parametrickej architektúry v Twelfth Milan Triennial v roku 1960. Parametrický model pozostávajúci z devätnásťich parametrov. Zdroj: [7]



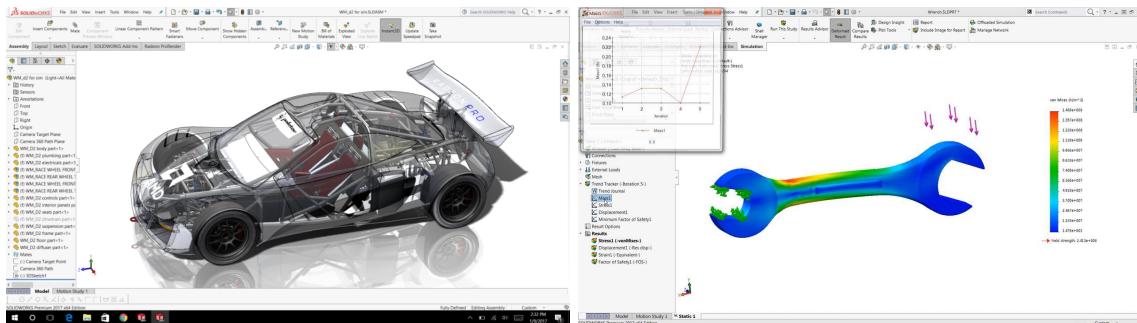
Obr. 2.2: Watergate Complex navrhnutý Luigi Morettim. Prvý veľký stavebný projekt, na ktorý boli využité parametrické modely. Zdroj: [15]

2.2 Systémy pre tvorbu parametrických modelov

Existuje niekoľko systémov pre tvorbu parametrických modelov. Zväčša sú platené, ale sú aj také, ktoré umožňujú vytváranie modelov zadarmo. Väčšinou tieto parametrické modelovacie systémy nie sú navrhnuté pre domáceho používateľa ale skôr pre profesionálnych 3D dizajnérsov. Táto časť je podľa blogu [11].

Solidworks

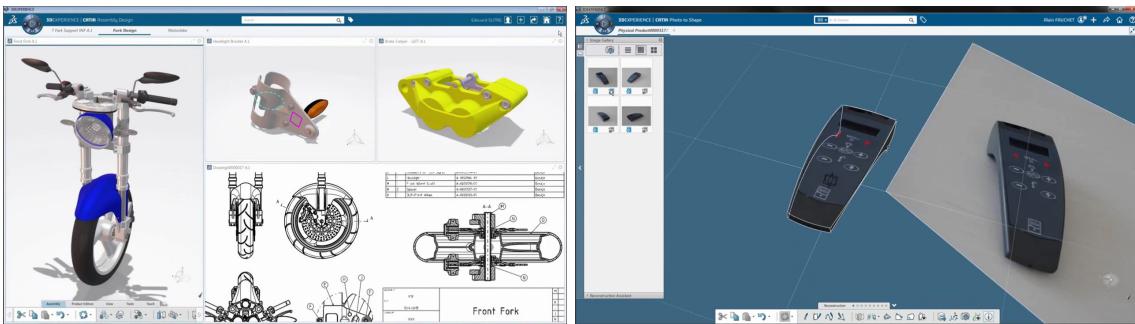
Solidworks je jedným z najlepších softvérov na tvorbu mechanických častí. Tento parametrický softvér je profesionálny nástroj pre inžinierov a dizajnérov. Umožňuje vykonať nad vytvoreným modelom aj rôzne simulácie, napríklad zátažové alebo tepelné ako je vidieť na obrázku 2.3.



Obr. 2.3: Prostredie programu Solidworks. Vpravo je zátažová simulácia. Zdroj: [20] [2]

Catia

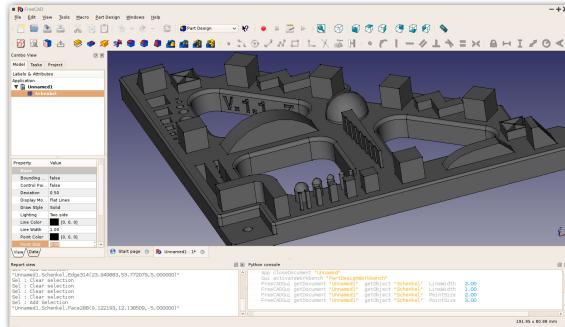
Tento softvér je náročnejší pre menej zdatných užívateľov. Je to komplexný nástroj určený hlavne pre profesionálov a umožňuje množstvo pokročilejších nástrojov, ako je prevedenie dvojrozmerného obrázku do trojrozmerného modelu alebo generovanie organických tvarov, ktoré spĺňajú pevnostné požiadavky, ale sú od pôvodného modelu značne odlahčené [6].



Obr. 2.4: Prostredie programu Catia. Vpravo je ukážka prevodu z 2D obrázku do 3D modelu. Zdroj: [6]

FreeCAD

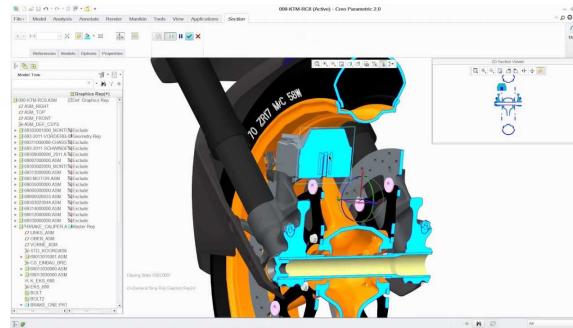
FreeCad je voľne dostupný softvér s intuitívnym grafickým rozhraním. Umožňuje množstvo podobných nástrojov ako Catia a SolidWorks. Je zameraný na strojárstvo a návrh výrobku [10].



Obr. 2.5: Prostredie programu FreeCAD. Zdroj: [10]

Creo Parametric

Jedná sa o softvér pre priemyselný dizajn. Umožňuje vytvárať zložité trojrozmerné modely. Poskytuje veľa účinných nástrojov prispôsobených priemyselnému výrobnému prostrediu. Tento softvér sa často používa napríklad v automobilovom priemysle.

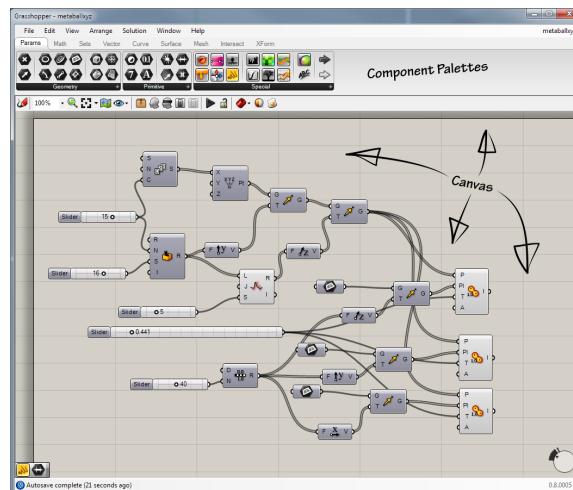


Obr. 2.6: Prostredie programu Creo Parametric Zdroj: [17]

Rhino s pluginom Grasshopper

Rhino je profesionálny 3D CAD softvér používaný v množstve spoločností. Pre prácu s parametrickými modelmi je potrebný plugin Grasshopper.

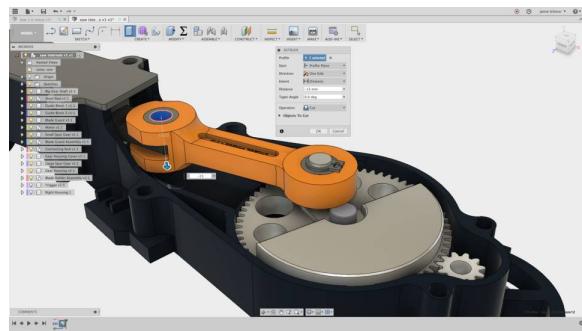
Grasshopper je vizuálny programovací jazyk, znázornený na obrázku 2.7. Umožňuje tvorbu parametrických modelov pre konštrukčné inžinierstvo, architektúru a výrobu.



Obr. 2.7: Prostredie programu Grasshopper Zdroj: [19]

Fusion 360

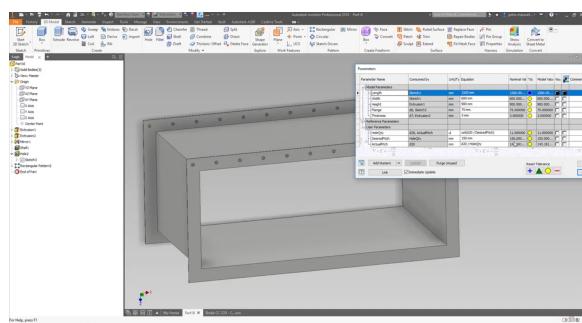
Fusion 360 nie je iba parametrický modelovací systém. Podporuje aj priame modelovanie a umožňuje prechod medzi týmito typmi modelovania. Kedže pri priamom modelovaní nie je model vytváraný pomocou stromu operácií, je tento strom pri prenose odstránený. Má dobré simulačné a modelovacie nástroje, ktoré pomáhajú pri návrhu.



Obr. 2.8: Fusion 360 Zdroj: [11]

Inventor

Inventor je rovnako ako Fusion 360 vytvorený spoločnosťou Autodesk. Tento program je často používaný v strojárstve. Inventor ponúka množstvo parametrických možností, ktoré pomáhajú vytvárať 3D modely a hlavne mechanické konštrukcie.



Obr. 2.9: Fusion 360 Zdroj: [5]

Kapitola 3

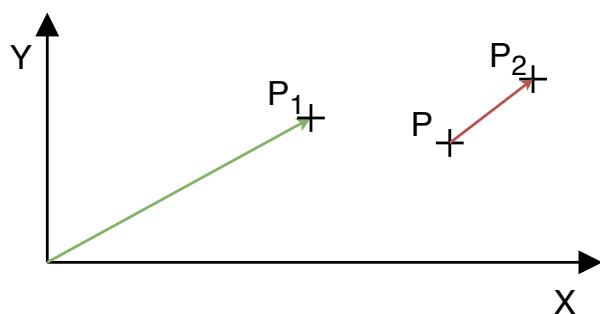
Geometrické objekty a operácie

Geometrické objekty môžeme rozdeliť do 4 kategórií podľa dimenzií. Body, úsečky, plochy a objemové objekty. Spájaním objektov z nižších dimenzií, vznikajú zložitejšie objekty z vyšších dimenzií.

Táto kapitola opisuje jednotlivé geometrické objekty a operácie, ktoré ich vytvárajú, a to od jednoduchších až k zložitejším.

3.1 Bod

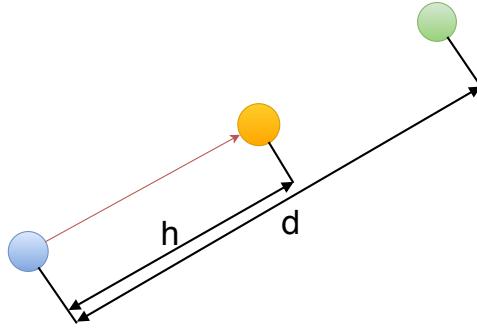
Bod je základnou stavebnou jednotkou všetkých objektov. Všetky geometrické útvary sa dajú definovať ako množina bodov. Je to bezrozmerný geometrický útvar, teda nemá šírku, výšku ani hrúbku. Jeho úlohou je označiť pozíciu v priestore. Pozíciu bodu v trojrozmernom priestore udáva vzdialenosť na jednotlivých osiach ortogonálneho súradnicového systému X, Y a Z. Táto pozícia môže byť v absolútnej tvare, teda od stredu súradnicového systému alebo v relatívnom tvare, kedy je závislá na pozícii iného bodu.



Obr. 3.1: Bod P_1 s absolútou pozíciou a bod P_2 s relatívnu pozíciou od bodu P

Lineárna interpolácia

Lineárna interpolácia umožňuje získať bod, ktorý je na rovnakej priamke ako dva zadané body. Na obrázku 3.2 sú tieto body zobrazené modrou a zelenou farbou, výsledný bod je zobrazený žltou farbou. Pozícia bodu závisí od zadanej vzdialenosť od počiatočného bodu. Táto vzdialenosť môže byť zadaná dĺžkou alebo percentuálne, kde 50% vytvorí bod uprostred počiatočného a koncového bodu.



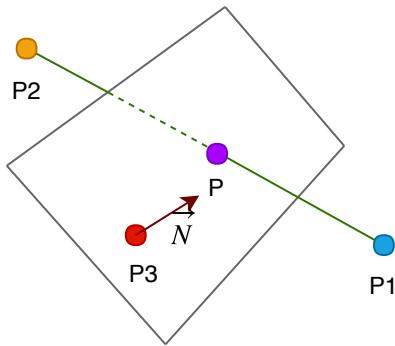
Obr. 3.2: Lineárna interpolácia medzi bodmi. d označuje vzdialenosť medzi zadanými bodmi a h označuje vzdialenosť v ktorú sa má vykonať interpolácia

Ak je zadaná vzdialenosť pomocou dĺžky, použije sa vzorec 3.1. Pri percentuálnej vzdialnosti sa použije obdobný vzorec, ale vektor medzi bodmi bod1 a bod2 sa nenormalizuje.

$$bod = bod1 + \text{norm}(bod2 - bod1) * vzdialenosť; \quad (3.1)$$

Priesečník plochy a úsečky

Pri tejto operácii sa používa ľubovoľný plošný objekt ako rovina a úsečka ako priamka. Táto geometrická operácia vytvorí bod v mieste, kde sa priamka pretína s rovinou.



Obr. 3.3: Pretnutie plochy priamkou

Rovnica pre rovinu, ktorá je tvorená bodom P_3 nachádzajúcim sa na rovine a normálou N , sa dá zapísat ako 3.2 [4].

$$N \cdot (\mathbf{P} - \mathbf{P}_3) = 0 \quad (3.2)$$

Rovnica priamky 3.3, ktorá je určená bodmi P_1 a P_2

$$\mathbf{P} = \mathbf{P}_1 + u(\mathbf{P}_2 - \mathbf{P}_1) \quad (3.3)$$

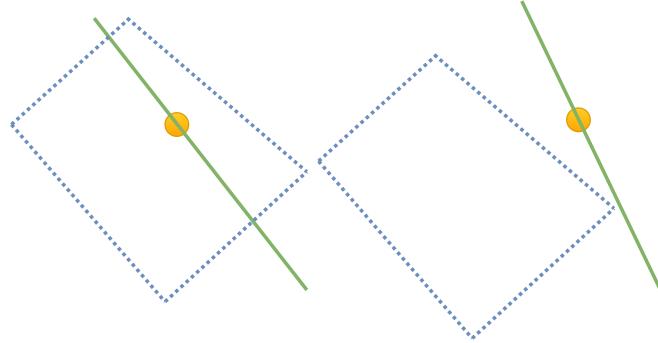
Bod P označuje priesečník medzi rovinou a priamkou. Pomocou substitúcie získame rovnicu 3.4.

$$N \cdot (\mathbf{P}_1 + u(\mathbf{P}_2 - \mathbf{P}_1)) = N \cdot \mathbf{P}_3 \quad (3.4)$$

Po vyriešení tejto rovnice dostaneme rovnicu 3.5. Výslednú pozíciu bodu dostaneme dosadením u do rovnice pre priamku 3.3.

$$u = \frac{N \cdot (\mathbf{P}_3 - \mathbf{P}_1)}{N \cdot (\mathbf{P}_2 - \mathbf{P}_1)} \quad (3.5)$$

Ako je vidieť na obrázku 3.4, pomocou tejto operácie sa vytvorí bod aj mimo zadaných objektov. Problém nastáva, ak je zadaná úsečka paralelná s plochou a teda je kolmá na normálou plochy N . Skalárny súčin v menovateli je potom rovný 0. V tomto prípade priesecník buď neexistuje, alebo je priesecníkov nekonečne veľa, ak úsečka leží na rovine.



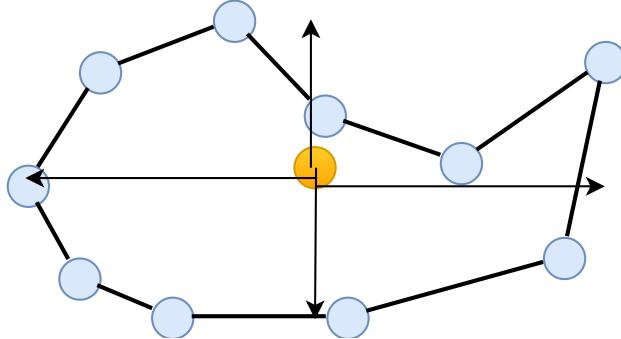
Obr. 3.4: Priesecník plochy a priamky. Vpravo je zobrazený prípad, ak sa priesecník nachádza na rovine, ale mimo plochy objektu

Stred plochy

Existuje množstvo možností ako získať stred objektu. Do tejto práce som vybral dve metódy a to metódu minimálneho štvorca a priemer všetkých bodov.

Minimálny štvorec

Pri tejto operácii sa prejdú všetky body a zistí sa maximálna a minimálna hodnota v jednotlivých osiach a výsledný bod sa nachádza uprostred nich.



Obr. 3.5: Stred plochy pomocou minimálneho štvorca

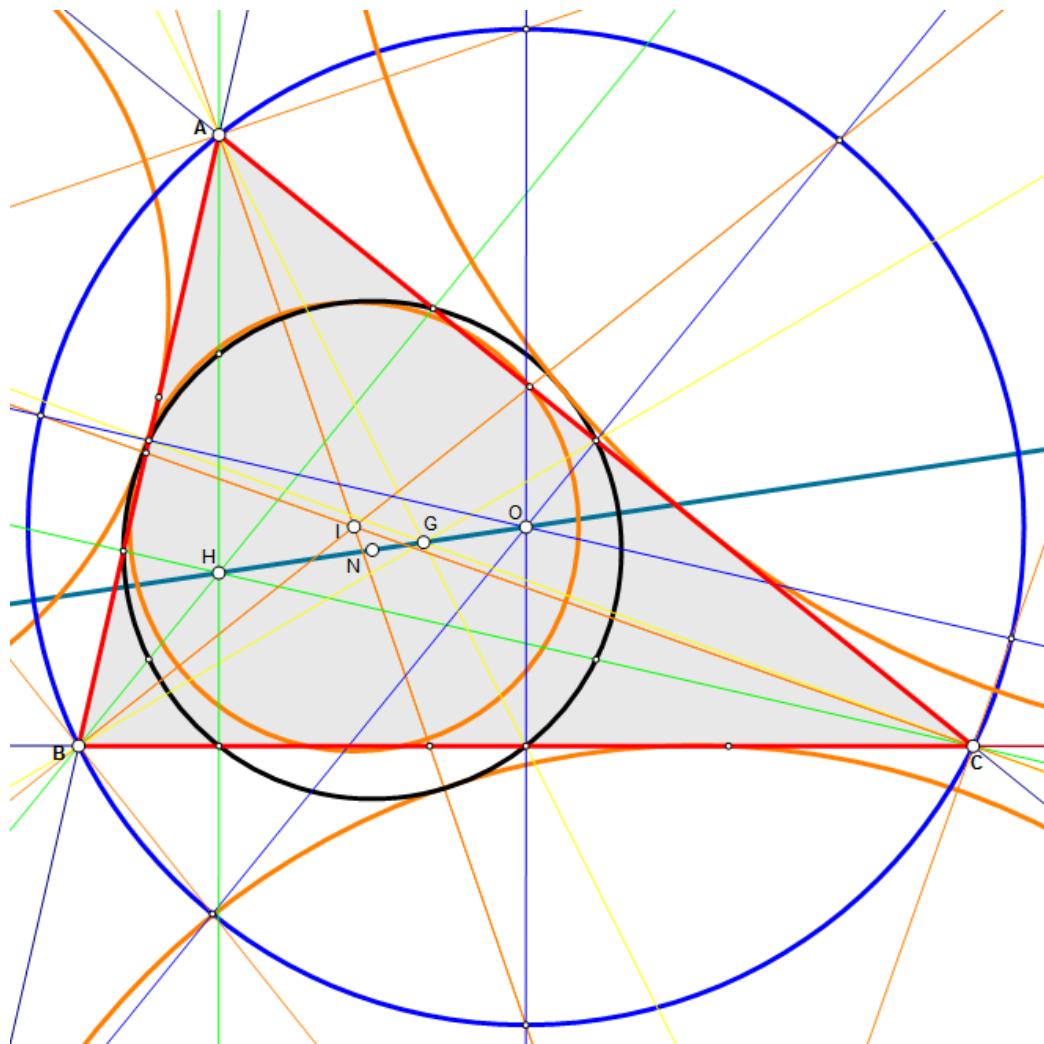
Priemer všetkých bodov

Nájdenie aritmetického priemeru všetkých bodov polygónu.

$$\frac{1}{n} \sum_{i=0}^n p_i \quad (3.6)$$

Stred trojuholníka

Aj pre trojuholník existuje množstvo typov stredu. V súčasnosti je podľa encyklopédie stredov trojuholníkov známych až 30 714 trojuholníkových centier [14]. Tento počet každým dňom narastá. Pre porovnanie, v roku 1994 bolo známych 101, v roku 1998 bolo 360 a v decembri 2004 bolo známych 3053 [22]. Medzi najznámejšie patria tažisko (G), ortocentrum (H), stred vpísanej kružnice (I), opísanej kružnice (O) aj stred kružnice deviatich bodov (N). Tieto body sú zaznačené na obrázku 3.6.



Obr. 3.6: Najznámejšie stredy trojuholníka sú tažisko (G), ortocentrum (H), stred vpísanej kružnice (I), opísanej kružnice (O) aj stred kružnice deviatich bodov (N) [24]

Tažisko (Centroid)

Tažisko trojuholníka sa nachádza v priesecníku troch mediánov trojuholníka. Na nájdenie jeho pozície stačí vypočítať aritmetický priemer vrcholov trojuholníka v jednotlivých osiach 3.7 [3].

$$\frac{A + B + C}{3} \quad (3.7)$$

Stred vpísanej kružnice (Incenter)

Z každého bodu trojuholníka urobíme priamku tak, aby uhly na oboch stranach priamy boli rovnaké. Táto priamka sa nazýva tiež bisektor [21]. Stred vpísanej kružnice je na priesčníku týchto priamok.

Stred vpísanej kružnice sa dá vypočítať aj pomocou vzorca 3.8, kde A, B, C sú vrcholy trojuholníka a a, b, c sú dĺžky strán protiľahlých k vrcholom A, B, C . [16].

$$O = \frac{a * A + b * B + c * C}{a + b + c} \quad (3.8)$$

Stred opísanej kružnice (Circumcenter)

U jednotlivých strán trojuholníka zistíme stred a z tohto bodu urobíme kolmice. Tam kde sa tieto kolmice stretňú, vznikne stred vpísanej kružnice. Veľkosť kružnice je vzdialenosť od stredu k ľubovoľnému vrcholu trojuholníka. Táto vzdialenosť je pre všetky vrcholy rovnaká.

Ortocentrum (Orthocenter)

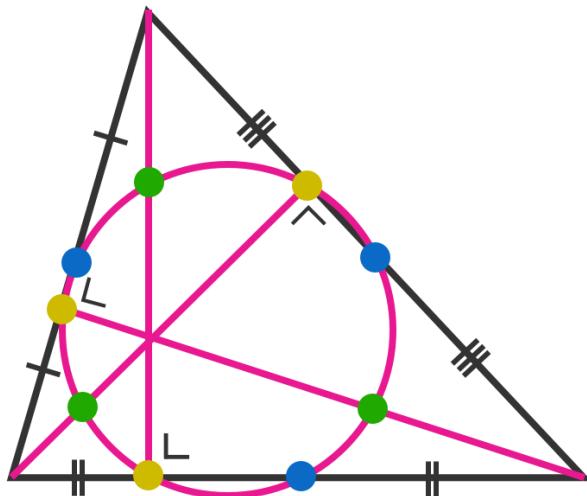
Ortocentrum sa nachádza na priesčníku kolmíc, ktoré prechádzajú cez protiľahlý vrchol. Tieto kolmice sa tiež nazývajú výškou trojuholníka. Ak je trojuholník tupý, ortocentrum sa nachádza mimo trojuholníka, ak je trojuholník v niektorom vrchole kolmý, nachádza sa v takomto vrchole aj ortocentrum trojuholníka.

Pre získanie pozície ortocentra zistíme aspoň dve kolmice pomocou operácie 3.2. Pozícia ortocentra sa nachádza v mieste, kde sa tieto kolmice pretínajú.

Stred kružnice deviatich bodov (NinePointCenter)

Kružnica deviatich bodov, tiež známa ako Feuerbachova kružnica, po nemeckom matematikovi Karl Wilhelm Feuerbach, ktorý ako prvý dokázal, že sa kružnica deviatich bodov dotýka vpísanej a pripísaných kružník [8].

Teorém 1 ([1] Teorém kružnice deviatich bodov) Nech ABC je všeobecný trojuholník, P, Q, R nech sú päty jeho výšok, K, L, M nech sú stredy jeho strán, O nech je priesčník výšok a T, U, V nech sú postupne stredy úsečiek AO, BO, CO . Potom 9 bodov $P, Q, R, K, L, M, T, U, V$ leží na jednej (tzv. Feuerbachovej) kružnici.



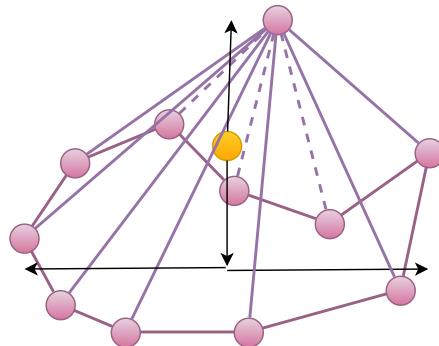
Obr. 3.7: Kružnica deviatich bodov. Modré body označujú stredy strán, žlté body označujú päty výšok a zelené sú stredy medzi vrcholmi a priesečníkom výšok[13]

Stred objektu

Rovnako ako pri dvojrozmerných objektoch, je aj u trojrozmerných objektoch viacero variant získania stredu objektu. Zvolil som dve metódy a to metódu ohraničujúceho kvádra a metódu priemerného stredu všetkých bodov.

Stred pomocou ohraničujúceho kvádra

Pri tejto metóde sa prejdú všetky body a zoberie sa maximálna a minimálna hodnota v osiach X, Y a Z. Takto dostaneme ohraničujúci kváder (Bounding box) a ako výsledný bod sa zoberie stred tohto kvádra.



Obr. 3.8: Stred ohraničujúceho kvádra

Priemer všetkých bodov

Pri tejto operácii sa scítajú koordináty všetkých bodov. To vytvorí tri veľké čísla, ktoré sa následne vydelia počtom bodov.

3.2 Úsečky

Úsečka je časť priamky medzi dvoma koncovými bodmi. Aby bolo možné používať v geometrických operáciach smerový vektor z úsečky, môžeme tieto body označiť ako počiatočný a koncový, ako je zobrazené na obrázku 3.9.

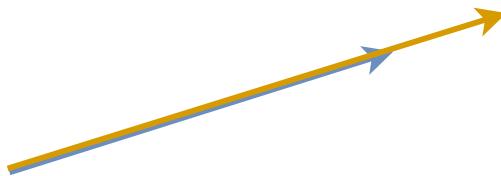
$$P = P_1 + u(P_2 - P_1) \quad (3.9)$$



Obr. 3.9: Zobrazenie úsečky s počiatočným a koncovým bodom

Zmena dĺžky úsečky

Vytvorenie úsečky so zadanou dĺžkou. Dĺžka môže byť zadaná vzdialenosťou alebo percentuálne od veľkosti zadanej úsečky. Výsledná úsečka je v rovnakom smere a začína v rovnakom bode ako zadaná úsečka. V prípade ak veľkosť výslednej úsečky rovná jedna, táto geometrická operácia sa nazýva aj normalizácia.



Obr. 3.10: Zmena veľkosti úsečky

Najkratšia úsečka medzi bodom a priamkou

Priamka je definovaná dvoma bodmi, bodom $P_1(x_1, y_1)$ a bodom $P_2(x_2, y_2)$. Rovnica priamky je 3.10.

$$P = P_1 + u(P_2 - P_1) \quad (3.10)$$

Na získanie najkratšej vzdialenosťi je potrebné nájsť kolmicu od bodu $P_3(x_3, y_3)$ na priamku. Z toho vyplýva, že skalárny súčin medzi nimi musí byť rovný 0, tento vzťah je vidieť v rovnici 3.11.

$$(P_3 - P) \cdot (P_2 - P_1) = 0 \quad (3.11)$$

Bod P označuje najbližší bod na priamke k bodu P_3 .

Substitúciou rovnice 3.10 do 3.11 získame rovnicu 3.12.

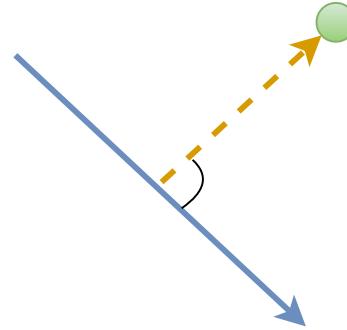
$$[P_3 - P_1 - u(P_2 - P_1)] \cdot (P_2 - P_1) = 0 \quad (3.12)$$

Vyliešením tejto rovnice získame rovnicu 3.13, ktorú následne môžeme dosadiť do rovnice pre priamku 3.10 a získať pozíciu bodu P . Ak by sme chceli, aby sa bod vytváral iba na úsečke a nie mimo nej, bolo by potrebné testovať vzdialenosť u na rozmedzie $(0,1)$. Ak je

hodnota u záporná alebo väčšia ako jedna, najbližší bod na priamke k bodu P_3 sa nachádza mimo zadanej úsečky.

$$u = \frac{(x_3 - x_1)(x_2 - x_1) + (y_3 - y_1)(y_2 - y_1) + (z_3 - z_1)(z_2 - z_1)}{\|P_2 - P_1\|^2} \quad (3.13)$$

Výsledná úsečka je tvorená bodmi P a P_3 [4].



Obr. 3.11: Najkratšia vzdialenosť medzi úsečkou a bodom

Najkratšia úsečka medzi dvomi priamkami

Kedže v trojrozmernom priestore často nenastáva pretnutie dvoch úsečiek v jednom bode, je práve najkratšia úsečka medzi dvoma priamkami používaná ako priesčník priamok v trojrozmernom priestore.

Hľadáme najkratšiu úsečku s bodmi P_a a P_b , kde P_a leží na priamke definovanou bodmi P_1 a P_2 , a P_b leží na priamke, ktorá je definovaná bodmi P_3 a P_4 .

Pre bod P_a môžeme napísť rovnicu 3.14.

$$P_a = P_1 + u_a(P_2 - P_1) \quad (3.14)$$

Podobne pre bod P_b rovnicu 3.15.

$$P_b = P_3 + u_b(P_4 - P_3) \quad (3.15)$$

Pri hľadaní najkratšej úsečky medzi týmito priamkami si stačí uvedomiť, že najkratšia úsečka bude na tieto priamky kolmá, čo nám umožňuje zapísť nasledovné rovnice 3.16.

$$\begin{aligned} (P_a - P_b) \cdot (P_2 - P_1) &= 0 \\ (P_a - P_b) \cdot (P_4 - P_3) &= 0 \end{aligned} \quad (3.16)$$

Po doplnení P_a a P_b do týchto rovníc získame rovnice 3.17.

$$\begin{aligned} ((P_1 + u_a(P_2 - P_1)) - (P_3 + u_b(P_4 - P_3))) \cdot (P_2 - P_1) &= 0 \\ ((P_1 + u_a(P_2 - P_1)) - (P_3 + u_b(P_4 - P_3))) \cdot (P_4 - P_3) &= 0 \end{aligned} \quad (3.17)$$

Kedže by tieto rovnice boli veľmi rozsiahle, je vhodné si pre riešenie tejto rovnice definovať nasledovnú substitúciu 3.18.

$$d_{mnop} = (x_m - x_n)(x_o - x_p) + (y_m - y_n)(y_o - y_p) + (z_m - z_n)(z_o - z_p) \quad (3.18)$$

Pomocou tejto substitúcie sa dajú rovnice 3.17 zapísť v nasledovne 3.19.

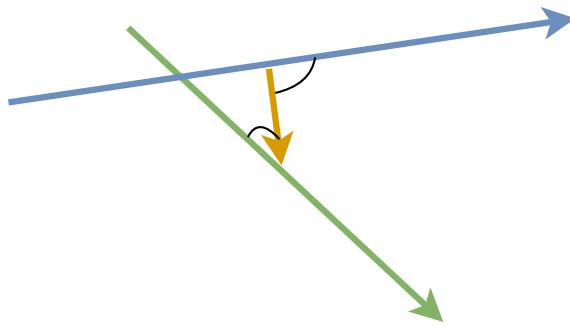
$$\begin{aligned} d_{1321} + u_a d_{2121} - u_b d_{4321} &= 0 \\ d_{1343} + u_a d_{2143} - u_b d_{4343} &= 0 \end{aligned} \quad (3.19)$$

Vyriešením týchto rovníc pre u_a získame rovnicu 3.20.

$$u_a = \frac{d_{1343}d_{4321} - d_{1321}d_{4343}}{d_{2121}d_{4343} - d_{2143}d_{2143}} \quad (3.20)$$

Kedž už poznáme u_a , pomocou dosadenia do rovnice 3.21 získame u_b [4].

$$u_b = \frac{d_{1321} + u_a d_{2121}}{d_{4321}} \quad (3.21)$$



Obr. 3.12: Najkratšia vzdialenosť medzi dvomi úsečkami

Najkratšia úsečka medzi plochou a bodom

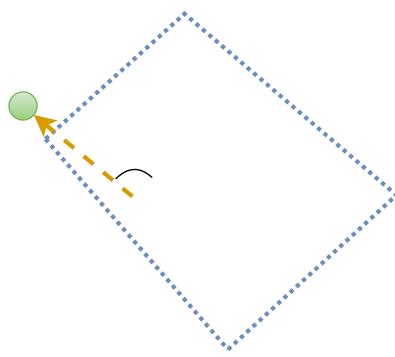
Rovina je definovaná pomocou normáli $\vec{N} = (A, B, C)$ a bodom na nej ležiacom $P_a = (x_a, y_a, z_a)$. Hľadaná úsečka má rovnaký smer ako normálka plochy. Vzdialenosť medzi touto rovinou a bodom $P_b = (x_b, y_b, z_b)$ dostaneme pomocou vzorca 3.22, kde premietneme vektor medzi bodom P_b a bodom P_a na normálku roviny \vec{N} pomocou skalárneho súčinu.

$$distance = (P_b - P_a) \cdot \vec{N} \quad (3.22)$$

Bod P získame vynásobením vektora \vec{N} touto vzdialenosťou a následným odčítaním od bodu P_b 3.23.

$$P = P_b - (\vec{N} * distance) \quad (3.23)$$

Výsledná úsečka má počiatočný bod na ploche a koncový bod P_b [4].

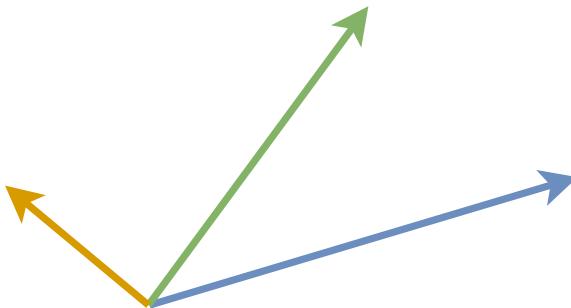


Obr. 3.13: Najkratšia vzdialenosť medzi plochou a bodom

Vektorový súčin

Vektorový súčin (cross product) vytvára úsečku, ktorá je kolmá na dané úsečky (priamky). Zadané úsečky sa najprv prevedú na vektory (*koncový_bod – začiatočný_bod*). Veľkosť úsečky závisí od veľkosti zadaných úsečiek a od uhla, ktorý zvierajú.

$$L_1 \times L_2 = \{y_{l1}z_{l2} + y_{l2}z_{l1}, z_{l1}x_{l2} + z_{l2}x_{l1}, x_{l1}y_{l2} + x_{l2}y_{l1}\} \quad (3.24)$$



Obr. 3.14: Vektorový súčin

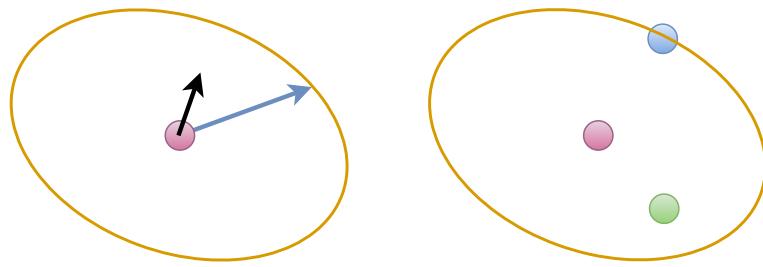
Táto operácia sa často používa aj v počítačovej grafike, kde sa používa pre výpočet normály plošných útvarov. Výsledný vektor sa musí normalizovať, teda vydeliť jeho dĺžkou.

3.3 Plošné objekty

Kruh

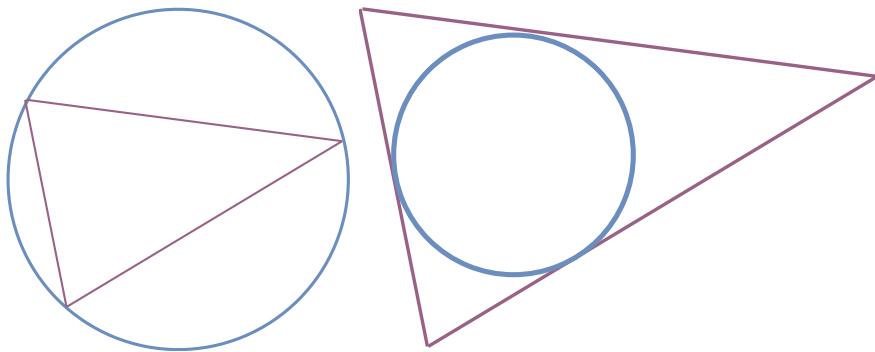
Je viac metód ako zadávať kruh. Základným spôsobom pre vytvorenie kruhu je zadanie stredového bodu a priemeru. Keďže chceme aby bol kruh v trojrozmernom priestore, je potrebné zadať aj normálu.

Ďalším spôsobom je zadanie troch bodov, kde jeden z bodov označuje stred kruhu, druhý bod leží na obvode kruhu a tretí bod ležiaci na ploche kruhu tak, aby neboli s ostatnými bodmi na jednej priamke, označuje natočenie kruhu.



Obr. 3.15: Vľavo je kruh pomocou priemeru a normály a v pravo je kruh tvorený troma bodmi, kde jeden bod udáva stred a pomocou ďalších dvoch bodov sa určí priemer a smer normály

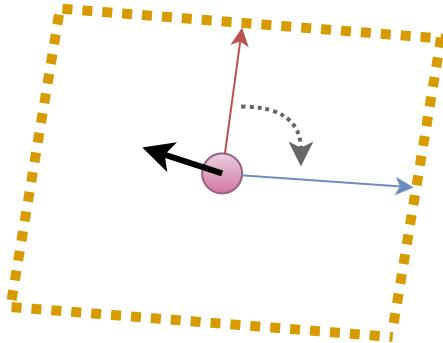
Alternatívne môžeme vyjadriť kruh pomocou opísanej (Circumscribed) alebo vpísanej (Inscribed) kružnice trojuholníka (obrázok 3.16). Postup pre nájdenie stredu opísanej a vpísanej kružnice trojuholníka je v časti 3.1.



Obr. 3.16: Opisaná a vpisaná kružnica trojuholníka

Obdĺžnik

Často používaným geometrickým objektom v grafike je aj obdĺžnik. Základnými parametrami obdĺžnikov je výška a šírka. Ďalším parametrom je často aj natočenie okolo normály. Kedže sa zaoberáme objektmi v trojrozmernom priestore, obdĺžnik musí definovať aj normálu, ako je vidieť na obrázku 3.17.



Obr. 3.17: Obdĺžnik

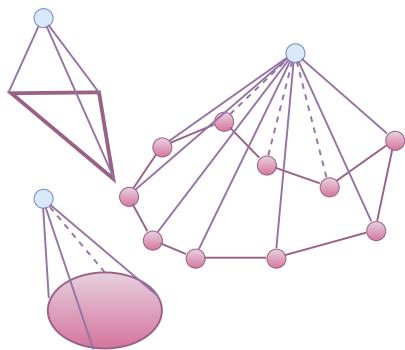
Polygón

Kedže v reálnom svete nie sú základom objektov len dokonalé objekty ako kruh, trojuholník alebo štvorec, zvolil som pre túto prácu aj tvorbu polygónov. Polygóny pozostávajú z množiny bodov a množinou hrán, ktoré tieto body spájajú a vytvárajú obvod polygónu. Keďže polygón je plošný útvar, musí byť tvorený minimálne 3 bodmi. Táto operácia má ako jediná premenlivý počet bodov.

Všetky primitívne objekty, ako štvorec, kruh alebo trojuholník, sú konvexné, teda všetky vnútorné uhly sú menšie alebo rovné 90 stupňov. Proces triangulácie je v tomto prípade veľmi jednoduchý. Stačí spojiť jeden ľubovoľný bod so všetkými ostatnými bodmi. Pri zadávaní polygónov pomocou bodov sa ale môže stať, že niektorý vnútorný uhol polygónu bude tupý, teda väčší ako 90 stupňov. Klasická metóda triangulácie teda nepostačuje a je potrebné vybrať niektorú inú metódu, napríklad *Ear clipping method*.

3.4 Objemové objekty

Medzi základné geometrické objemové telesá patrí guľa, kváder, valec, a ihlan. Nad týmito geometrickými telesami je možné vykonávať rôzne geometrické operácie.

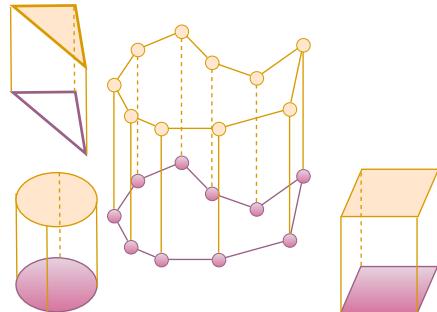


Obr. 3.18: Ihlany s rôznymi základmi.

Extrudovanie

Natiahne plochu do priestoru v smere normály, čím získa na objeme. Pomocou tejto operácie sa dá vytvoriť aj napríklad kváder alebo valec, ak je základ štvorcového, respektívne

kruhového tvaru. Táto operácia je zobrazená na obrázku 3.19, kde je fialovým zobrazená počiatočná plocha a žltým je zobrazená extrudácia. Extrudáciu je možné spraviť tak, že sa vytvorí kópia zadanej plochy a posunie sa o zadaný offset. Tieto dve plochy sa následne prepoja. Častou modifikáciou je extrudovanie po krivke a/alebo postupná zmena veľkosti.



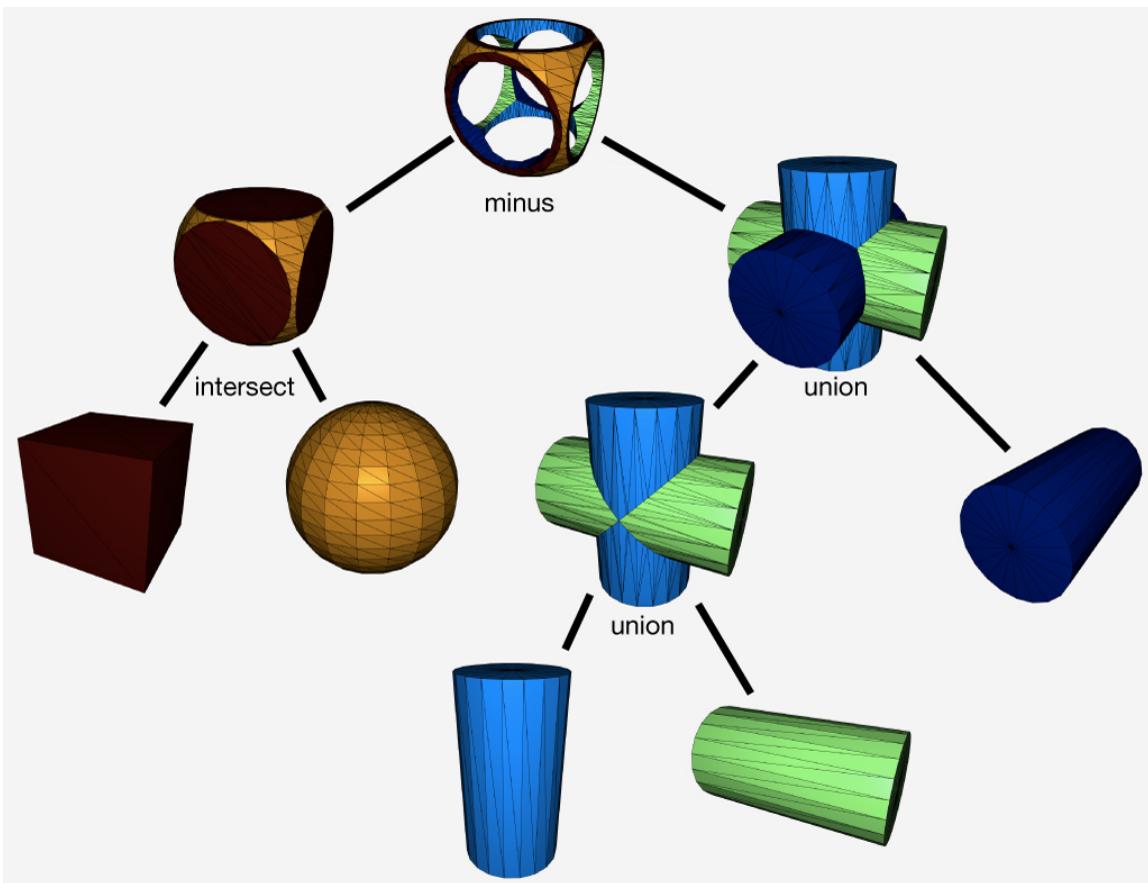
Obr. 3.19: Extrudovanie

Booleovské operácie nad objemovými telesami

Pre vytvorenie zložitejších objektov je potrebné jednoduché objekty sklaňať pomocou booleovských operácií ako zjednotenie (union), prienik (intersection) a rozdiel medzi množinami (minus). Tieto operácie sú zobrazené aj na obrázku 3.20, kde tvoria CSG (constructive solid geometry) strom. Kedže sú tieto operácie náročné na spracovanie, rozhodol som sa využiť niektorú z voľne dostupných knižníc. Našiel som niekoľko dostupných knižníc, ktoré by sa dali použiť. Jednou z nich je [libigl](#)[12].

Knižnica libigl je používaná aj veľkými spoločnosťami ako je Activision, EA, Adobe, Google, Epic Games, Microsoft, Pixar, UBISOFT a ďalšie. Táto knižnica je distribuovaná pod licenciou Mozilla Public License (MPL). Oblubnená je hlavne pre svoju jednoduchosť. Pre tvorbu booleovských operácií nad mesh stačí zavolať funkciu `mesh_boolean`. Táto funkcia pracuje s bodmi (V) a trojuholníkmi (F). Na výpočet zjednotenia (typ `MESH_BOOLEAN_TYPE_UNION`) pre mesh $A = (VA, FA)$ a mesh $B = (VB, FB)$ je nasledovná funkcia. Výsledok tejto funkcie sa uloží do mesh $C = (VC, FC)$,

```
igl::copyleft::cgal::mesh_boolean(VA,FA,VB,FB,MESH_BOOLEAN_TYPE_UNION,VC,FC);
```



Obr. 3.20: Skladanie objektov pomocou booleovských operácií

[[Pridat aj výpočet obsahu, obvodu, povrchu a objemu?]]

Kapitola 4

Zhrnutie stavu práce

V tejto časti je opísaný aktuálny stav práce, návrh možností previazania objektov v parametrickom modely a implementácia knižnice určenej na tvorbu trojrozmerných parametrických modelov, kde sú tieto možnosti previazania implementované.

4.1 Koncept práce

Koncept práce pre implementáciu knižnice a vlastnosti, ktoré by mala by mala táto knižnica obsahovať:

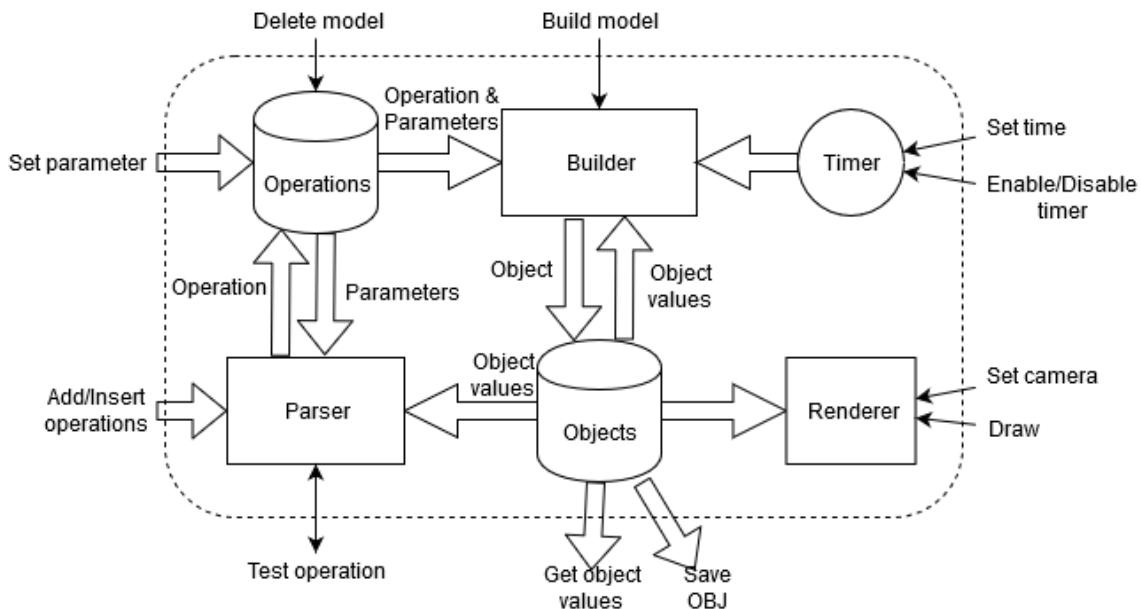
- Parametrické modelovanie
- Previazanie objektov v modeli
- Skladanie geometrických operácií
- Dynamický model
- Grafický výstup
- Animácie modelu v čase
- Vlastný parametrický formát súboru
- Uloženie objektu v polygonálnom formáte - OBJ
- Táto knižnica musí byť tiež intuitívna a ľahko použiteľná pre používateľa.

Operácie ukladané v zozname operácií

[[pouzitie kniznice na vypocty (vytvorit lubovolny model a ziskat z neho nejake parametre)]]

[[ukazka vytvorenia niekolkych model a obrazok]]

[[vykreslenie modelu pomocou kniznice, ze je potrebne najprv vytvorit okno a graficky kontext, napisat ze sa pre vykreslenie pouziva kniznica glew a napisat aké funkcie treba volat pri init a draw. Pouzitie kniznice s roznimy typmi na tvorenie kontextu(aspon opisat glfw a QT (možno aj kniznicu SDL)), napisat aké su minimalne poziadavky pre spustenie grafickej casti (OpenGL >=3.3 kvôli shaderom)]]



Obr. 4.1: Blokové schéma knižnice

4.2 Skladanie operácií

Všetky geometrické operácie sa dajú zapisovať pomocou textovej podoby. Textová podoba geometrickej operácie sa začína názvom použitej geometrickej operácie a parametrami uvedenými v zátvorkách (a), oddelenými čiarkou. U jednotlivých parametrov sa medzery na začiatku a konci ignorujú.

Kedže pri niektorých objektoch existuje viac možností, ako objekt vytvoriť, inšpiroval som sa pretažovaním funkcií v jazyku c++ a umožnil som, aby mali niektoré operácie mali rovnaký názov, ale rozdielny typ argumentov. Parametre môžu byť dvoch typov a to bud názov objektu, alebo v tvare výrazu. Viac o možnostiach výrazov je v časti 4.4. Operácie, ktoré táto knižnica umožňuje využiť, sú uvedené v prílohe A, s ich textovým zápisom a s požadovanými parametrami, potrebnými na zostavenie týchto operácií.

Parametrom operácie, ktoré sú typu *Expression*, je možné vytvoriť názov, pomocou ktorého sa bude dať na daný parameter neskôr pristupovať. Tento názov sa zadáva za hodnotou parametra oddelený dvojbodkou.

Jednotlivé operácie sa oddeľujú bodkočiarkou (;). Táto textová podoba geometrických operácií sa používa pri vytváraní modelu a aj pre uloženie do súboru a tiež jeho následné načítanie.

Pre predstavu ako sa jednotlivé objekty vytvárajú uvádzam príklad pre vytvorenie ihlana s kruhovou podstavou s polomerom 5 a výškou 10:

- Na vytvorenie ihlana je potrebné najprv vytvoriť bod na pozícii XYZ, napr. [1, 2, 3], ktorý bude slúžiť ako stred kruhovej podstavy.

```
Point(Stred podstavy, 1, 2, 3, 00000000);
```

- Na vytvorenie kruhu (podstavy ihlana) je potrebný polomer a normála. Ako normálu je možné využiť smer ľubovoľnej úsečky, ale keďže zatiaľ žiadna úsečka vytvorená nie je, je potrebné ju vytvoriť. Úsečka sa skladá z dvoch bodov a to počiatočného a

koncového. Ako normála sa použije normalizovaný vektor medzi bodom počiatočným a koncovým 4.1.

$$\overrightarrow{normal_vector} = \frac{end_point - begin_point}{\|end_point - begin_point\|} \quad (4.1)$$

```
Point(Počiatočný bod, 0, 0, 0, 00000000);
Point(Koncový bod, 1, 1, 1, 00000000);
Line(Normála, Počiatočný bod, Koncový bod, 00000000);
```

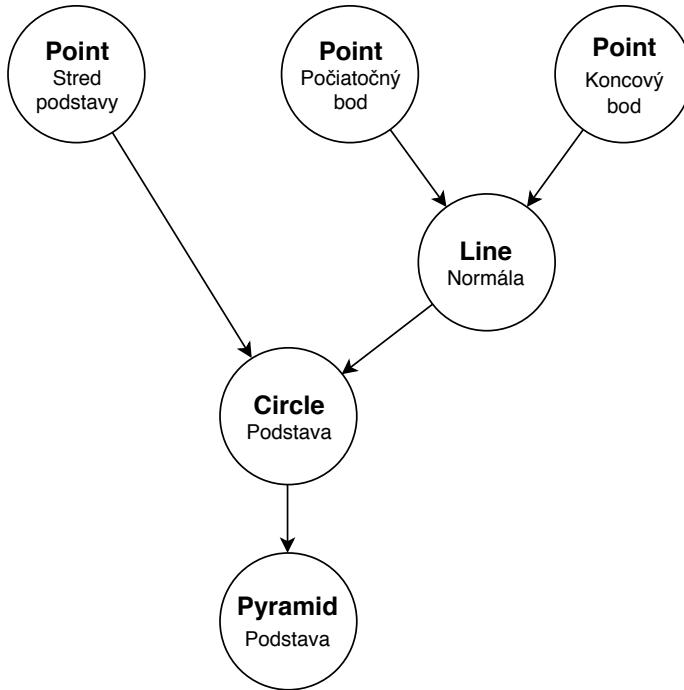
- Keď už je normála vytvorená, je možné vytvoriť kruh s polomerom 5.

```
Circle(Podstava, Stred podstavy, 5, Normála, 00000000)
```

- Ostáva vytvoriť samotný ihlan s výškou 10.

```
Pyramid(Ihlan, Podstava, 10, 00FF00FF)
```

Tento sled operácií je možné zobraziť pomocou orientovaného grafu 4.2. Operácie používajú iba už vytvorené objekty a ich hodnoty, teda žiadny objekt nemôže byť závislý od iného objektu, ktorý je závislý na tomto objekte. Takýto graf sa tiež nazýva aj acyklický.



Obr. 4.2: Sled operácií pomocou orientovaného grafu

4.3 Pridávanie a odoberanie objektov

Jedným z mojich cieľov bolo, aby sa dal model dynamicky upravovať. Na to vo vytvorennej knižnici slúžia následovné metódy:

```

bool AddOperations(std::string s)
bool AddOperation(std::string s)
void AddOperation(Operation *c)
void InsertOperation(size_t index, Operation *c)
void ReplaceOperation(size_t index, Operation *c)
void RemoveOperation(size_t index)
void DeleteModel()

```

Pre pridávanie operácií do modelu slúžia metódy `AddOperation` a `AddOperations` ktoré umožňujú pridávanie operácií v `String`-ovej forme v tvare ako je napísané v časti 4.2. Metóda `AddOperations` umožňuje pridanie množiny operácií, kde jednotlivé zápisy operácií sú oddelené znakom bodkočiarky (`;`). Operácie je tiež môžne zapísat aj v tvare štruktúry `Operation`, kde sa zapisujú jednotlivé zložky operácie, ako sú typ operácie, názov vytváraného objektu a parametre operácie. Pomocou metódy `InsertOperation` je možné vkladať operácie do už vytvoreného modelu na ľubovoľnú pozíciu, prípadne pomocou metódy `ReplaceOperation` nahradíť ľubovoľnú operáciu. Metóda `RemoveOperation` umožňuje odstránenie jednej operácie na zadanej pozícii. Pre vymazanie celého modelu slúži metóda `DeleteModel`.

Knižnica tiež umožňuje uloženie parametrického modelu do súboru a jeho následné načítanie. Aby bolo možné použiť vytvorený objekt aj v iných aplikáciach, je možné tento objekt exportovať do súborového formátu `OBJ`, ktorý kvôli svojej jednoduchosti patrí medzi najviac podporované geometrické súborové formáty. Pre ukladanie a načítanie modelu slúžia následovné metódy:

```

void Save(std::string filePath);
void Load(std::string filePath);
void SaveOBJ(std::string filePath);

```

Pre otestovanie, či v zapísanom reťazci nieje chyba, či už syntaktická alebo sémantická, slúži testovanie. Na začiatku každého testovania je potrebné zavolať metódu `void resetTest()` a následne je možné pridať operácie pomocou metódy

```
bool TestOperation(std::string s)
```

Takto pridané operácie sa nezapisujú do samotného modelu a budú vymazané pri ďalšom zavolaní metódy `resetTest()`. Metóda `TestOperation` vracia hodnotu `True`, ak sa nevyskytla žiadna chyba, a je teda možné operáciu do modelu bezchybne vložiť.

Aby sa urýchliť výpočet modelu, spracovanie sa nezačína hned pri každej úprave modelu, ale je potrebné, aby bola zavolaná metóda

```
void BuildModel()
```

Pri skladaní iba jednoduchých operácií stačilo celý model pred generovaním zahodiť a vytvoriť znova celý model od začiatku. Problém nastal, keď som sa rozhodol implementovať konštruktívnu geometriu. Keďže model je vytváraný v polygonálnej forme, použil som už spomenutú knižnicu libigl (viac o konštruktívnej geometrii je v časti 3.4), ktorá umožňuje jednoduchú prácu s boolevkými operáciami nad polygonálnymi objektami. Výpočet boolevskej operácie trvá nejaký čas a teda nemôžeme hovoriť o real-timeovej operácii. Preto som algoritmus prerobil tak, aby sa znova generovali iba objekty ktoré sa nejakým spôsobom modifikovali, či už pridaním novej operácie alebo zmenou hodnoty niektorého parametru operácie, alebo ak sa zmenila niektorá operácia na ktorej je táto operácia tvoriaca daný objekt závislá. Taktiež je potrebné kontrolovať, či niektorá premenná v zadanom výraze

nezmenila svoju hodnotu (napríklad časová premenná alebo parameter niektorého objektu - viac v časti 4.4) a je teda potrebné znova vypočítať zadaný výraz a podľa nových hodnôt vytvoriť objekt. **[[pridať graf na ukazku tejto optimalizácie]]**

4.4 Výrazy

Zo začiatku umožňovala knižnica v parametroch operácií iba číselnú hodnotu, s presnosťou dátového typu *double*. To ale často nebolo dostatočné, preto som sa rozhodol prerobiť túto hodnotu na výrazy so základnou aritmetikou a s použitím aj iných hodnôt z objektov. Neskôr som pridal aj trigonometrické operácie a zaokrúhlovanie. Pridal som aj časové premenné, aby sa dal model animovať v čase. Parametre operácií môžu byť zapísané v tvare výrazu. Tieto výrazy sa počas zápisu overia a spracúvajú. Vyhodnocovanie prebieha až v čase vytvárania modelu.

Výrazy môžu obsahovať:

- čísla (0; 1.5; -5.5)
- parametre **[[TODO]]**
- hodnotu z objektov **[[TODO]]**
- časové premenné
 - time - časová hodnota v milisekundách
 - time_seconds - časová hodnota v sekundách
- unárne operácie
 - round - zaokrúhlenie na celé číslo
 - ceil - zaokrúhlenie na vyššie celé číslo
 - floor - zaokrúhlenie na nižšie celé číslo
 - trunc - odstráni desatinnú časť čísla
 - sin - síanus
 - cos - kosínus
 - tan - tangens
 - asin - arkus síanus
 - acos - arkus kosínus
 - atan - arkus tangens
 - sqrt - odmocnina
- binárne operácie
 - + scítanie
 - - odčítanie
 - * násobenie
 - / delenie
 - % modulo

- \wedge exponent
- zátvorky
 - (
 -)

Príklady zápisu výrazov:

```
Point(P1,1:param1,-5:param2,1.5:param3,FF0000FF);
Point(P2,param1+param2, sin(param3*time),P1.X,00FF00FF);
```

Precedenčná tabuľka

Na vyhodnocovanie výrazov je potrebná precedenčná tabuľka. Tá označuje, ktorá operácia má vyššiu prioritu pri výpočte. Zvislé popisky udávajú, aký znak je na vrchole zásobníka a vodorovné popisky udávajú, aký znak práve prišiel (vstupný token). Znak \$ na zásobníku určuje dno zásobníku a \$ na vstupe určuje koniec. Ak je výsledná hodnota z precedenčnej tabuľky znak X, znamená to, že zadaný výraz je zapísaný v chybej forme a teda nie je možné takýto výraz ďalej riešiť.

Pre vyhodnocovanie výrazov, sa v knižnici využíva nasledovná precedenčná tabuľka:

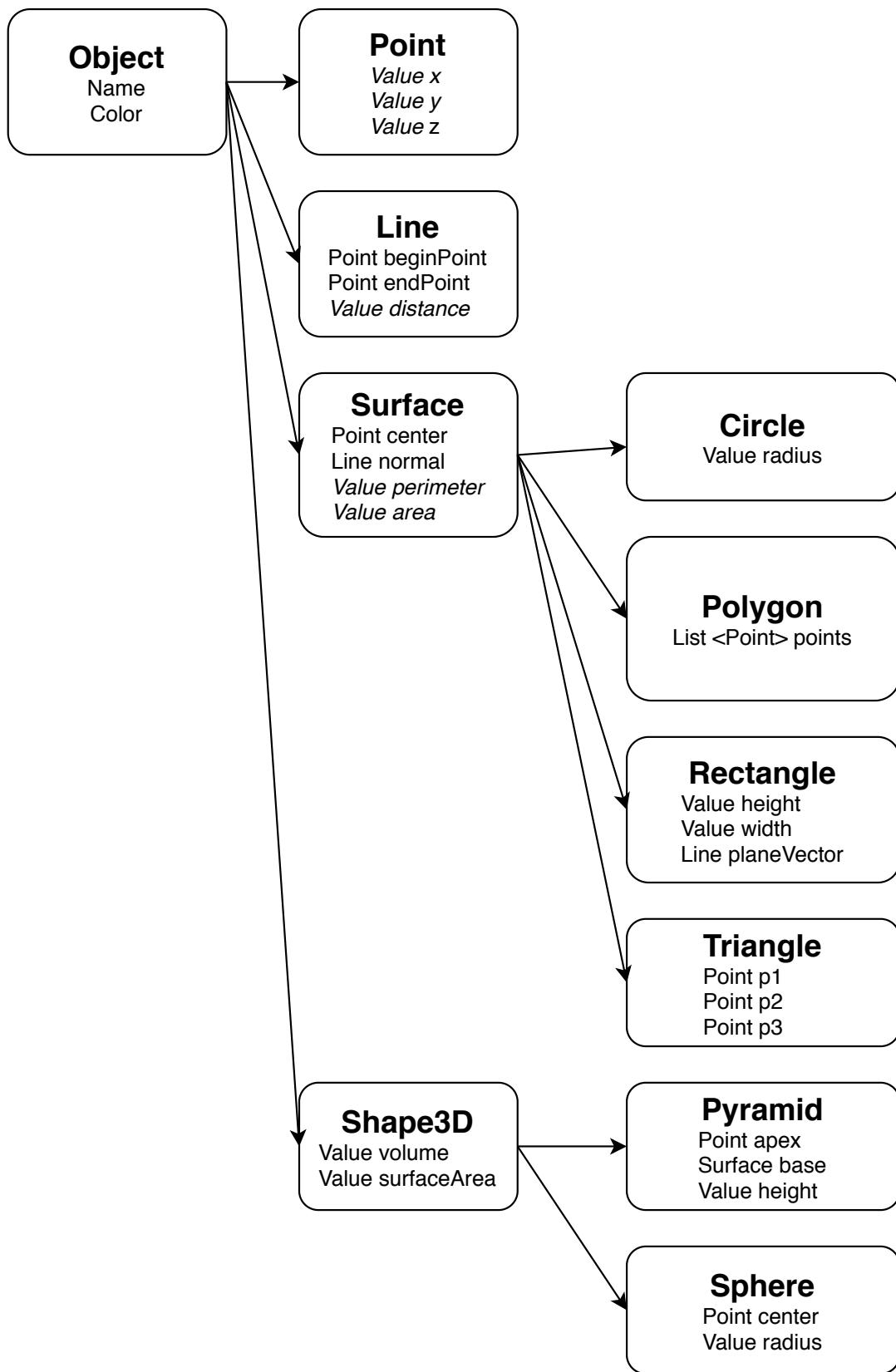
	+	-	*	/	%	i	()	\wedge	unary	\$
+	>	>	<	<	<	<	<	>	<	<	>
-	>	>	<	<	<	<	<	>	<	<	>
*	>	>	>	>	<	<	>	<	<	<	>
/	>	>	>	>	>	<	<	>	<	<	>
%	>	>	>	>	>	<	<	>	<	<	>
i	>	>	>	>	>	X	X	>	>	>	>
(<	<	<	<	<	<	<	=	<	<	X
)	>	>	>	>	>	X	X	>	>	X	>
\wedge	>	>	>	>	>	<	<	>	<	<	X
unary	>	>	>	>	>	<	<	>	>	>	>
\$	<	<	<	<	<	<	<	X	<	<	O

Tabuľka 4.1: Precedenčná tabuľka.

Ak nastane situácia, kedy nie je možné niektorý výraz vyriešiť, napríklad delenie nulou, geometrická operácia nevytvorí takýto objekt a taktiež operácie, ktoré sú na tomto objekte závislé nevytvoria svoje objekty.

Parametre objektov

Niekteré geometrické objekty majú podobné parametre. Všetky plošné útvary majú normálu, stred a aj plochu a obvod. Tieto parametre pre geometrické objekty sa dajú zobrazit pomocou stromu dedičnosti. Ten je zobrazený na obrázku 4.3. Jednotlivé geometrické objekty obsahujú priame hodnoty, teda parametre na ktorých je objekt závislý, ako výška alebo šírka objektu a hodnoty ktoré je potrebné vypočítať, ako sú napríklad obvod alebo objem objektu. Napríklad ihlan s kruhovou podstavou obsahuje stred a polomer základne



Obr. 4.3: Štruktúry objektov vytvárajú strom dedičnosti. Pri každom objekte je uvedené, aké dátá uchováva.

a výšku ihlanu. Vypočítanými hodnotami tohto objektu sú obvod a obsah základne alebo objem a plocha ihlanu.

Vo vytvorennej knižnici je možné pristupovať k jednotlivým hodnotám objektom pomocou: **[[TODOTODOTODO + Pridanie ukazky ako vybrat hodnotu z objektu]]**.

```
double GetObjectValue(std::string s, bool* Err).
```

Alternatívne sa dá pristupovať k štruktúram objektov priamo, pomocou funkcie:

```
Object::GeometricObject* GetObject(std::string objectName).
```

Časové premenné

Pre možnosti animácie modelu som vytvoril časové premenné time a time_seconds pre časovú hodnotu v milisekundách, prípadne v sekundách. Táto časová hodnota je po zapnutí aplikácie nastavená na 0 a behom aplikácie sa zvyšuje. V prípade, ak je potrebné nastaviť túto hodnotu na inú hodnotu je možné nastaviť túto hodnotu manuálne pomocou metódy

```
void setTime(unsigned long milliseconds).
```

V prípade ak užívateľ chce čas pozastaviť alebo preferuje nastavovanie času manuálne, je možné pozastaviť časovač pomocou následujúcej metódy

```
void useManualTimer(bool enable).
```

4.5 Grafické rozhranie

Na vytvorenie grafického rozhrania je potrebné vytvoriť okno a k nemu grafický kontext. Na ukážku som vytvoril aplikácie, ktoré používajú túto knižnicu s grafickým rozhraním a pre vytvorenie grafického kontextu používa jedna aplikácia knižnicu GLFW¹ a druhá aplikácia používa knižnicu QT² s použitím widgetu QOpenGLWidget.

Pre použitie grafického rozhrania je potrebné, aby používateľ po vytvorení kontextu zavolał metódu

```
void InitRenderer().
```

Táto metóda inicializuje GLEW a vytvorí program so shaderami. Minimálna požiadavka pre spustenie grafického rozhrania je podpora OpenGL vo verzii 3.3 v ktorom sú napísané shadery. Následne je potrebné volať vo vykresľovacej slučke metódu

```
void Draw(int x, int y, float fov, int width, int height),
```

ktorá ako parameter berie pozíciu pre vykreslovanie, veľkosť zorného poľa a pomer strán okna, v ktorom sa má vykreslovať.

Následne je pre prácu s obrazom možné používať tieto metódy:

- pre nastavenie pozície kamery:

```
void SetRendererCameraPosition(float X, float Y, float Z),
```

- pre natočenie kamery:

¹<https://www.glfw.org/>

²<https://www.qt.io/>

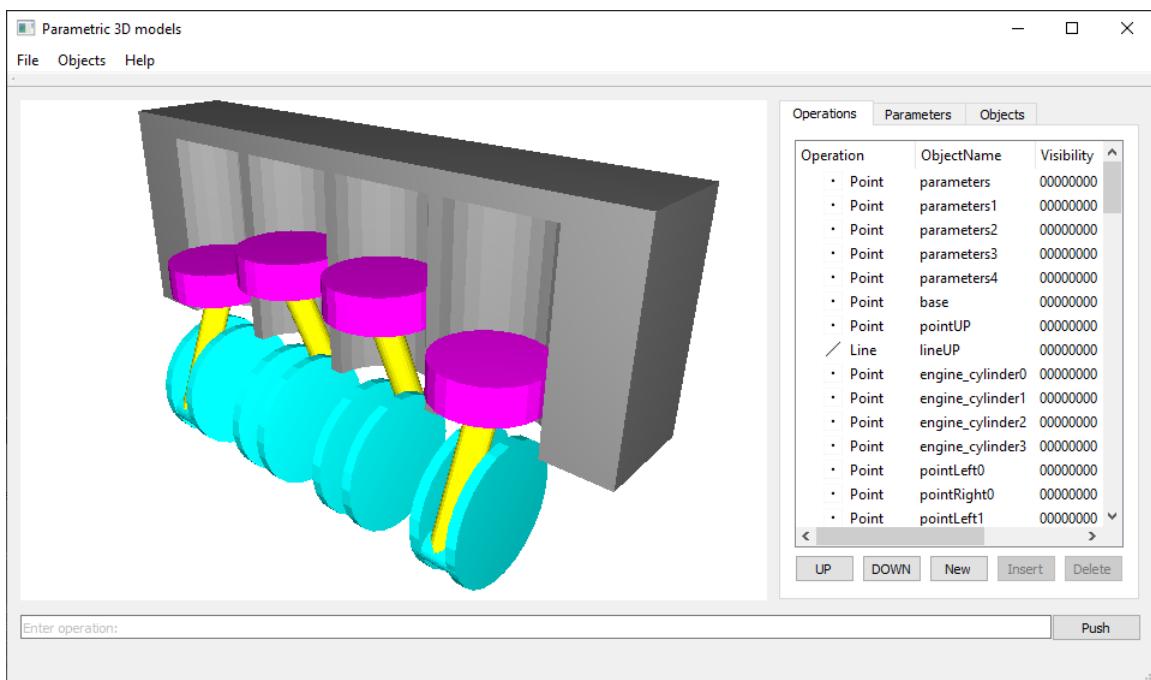
```
void SetRendererCameraRotation(float Pitch, float Yaw, float Roll),
```

- nastavenie sýtosti ambientného osvetlenia v rozmedzí <0;1> :

```
void setRendererAmbientStrength(float ambientStrength).
```

Grafický editor

Práca obsahuje aj jednoduché grafické rozhranie, ktoré návrhárovi pomáha, pri vytváraní parametrického modelu a uľahčuje prácu s geometrickými operáciami, a zároveň pri tom zobrazuje aj internú štruktúru modelu, teda jednotlivé operácie, objekty a parametre a zároveň tento model aj vizualizuje. Táto aplikácia je vytvorená pomocou knižnice QT a jej grafické rozhranie je zobrazené na obrázku 4.4.



Obr. 4.4: Grafický editor na tvorbu parametrických trojrozmerných modelov.

Zoznam geometrických operácií

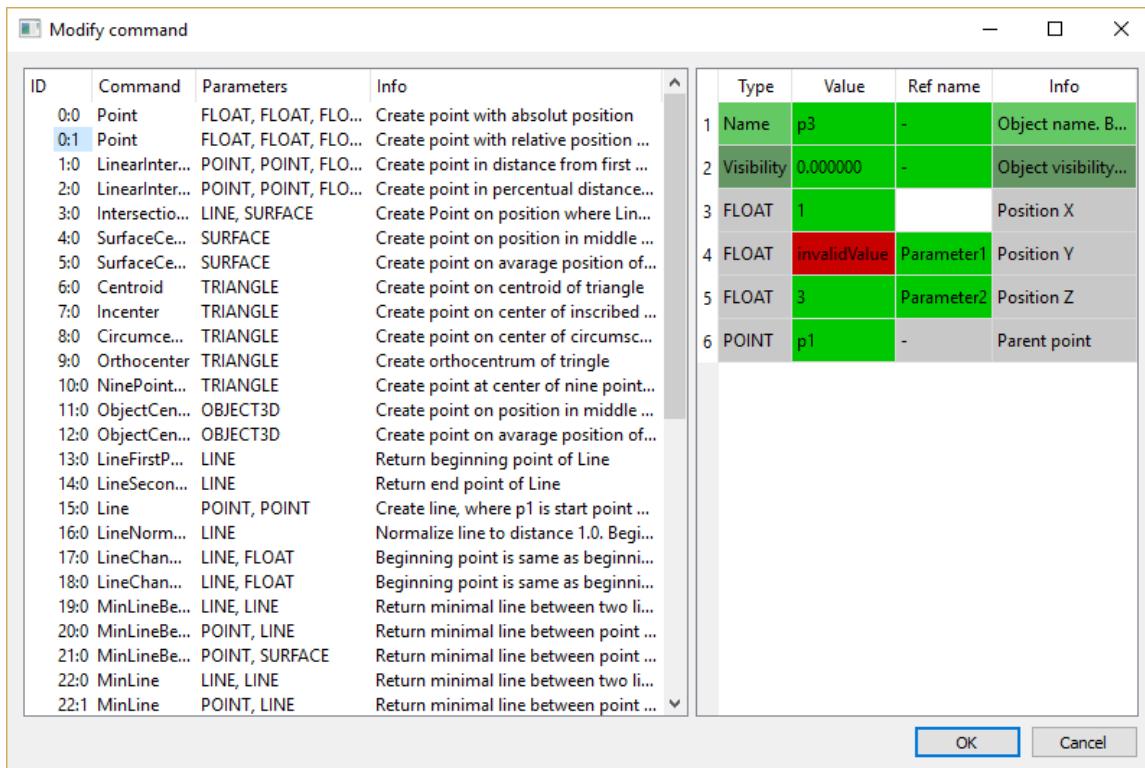
Aby bolo jednoduchšie vytvárať objekty, grafické rozhranie umožňuje geometrické operácie pridať, upraviť, mazať a aj vložiť na ľubovoľné miesto. Tieto operácie je tiež možné ľubovoľne presúvať. Operácie sú zoradené zhora dole v poradí, v akom sa vyhodnocujú. Parametre je možné zapisovať rovnako, ako v programátorskej časti, teda aj vo forme výrazu s odkazmi na hodnoty iných objektov a parametrov modelu. Kedže parametre operácií môžu odkazovať iba na skôr vytvorené objekty, je nutné pri upravovaní, presúvaní a mazaní tieto parametre kontrolovať. Grafické rozhranie následne zobrazí, u ktorých operácií sa nachádzajú nevyhovujúce parametre a tieto objekty (a objekty na nich závislé) sa nebudú vykreslovať.

Pridanie, vloženie a úprava geometrických operácií

Dialógové okno umožňuje výber operácie, ktorá sa má použiť, zo zoznamu geometrických operácií. V zozname sa nachádza názov geometrickej operácie, parametre, ktoré operácia potrebuje a nápoveda, ktorá informuje užívateľa čo daná operácia robí. Po vybraní operácie zo zoznamu, sa zobrazia v pravej časti dialógového okna parametre vybranej operácie. Ak je dialógové okno v režime úpravy (okno sa zobrazilo po dvojkliku na zozname geometrických operácií), sú pri vybraní rovnakého typu operácie tieto parametre už predvyplnené hodnotami zvolenej operácie na úpravu.

Ako prvý parameter je názov takto vytvoreného objektu, ktorý je pri novo vytváraných objektoch (režim okna pridanie alebo vloženie) predvyplnený, ale je možné ho upraviť na ľubovolnú hodnotu. Po názve objektu nasleduje viditeľnosť objektu, ktorá určuje, ako bude takto vytvorený objekt viditeľný. Pri hodnote viditeľnosti ≤ 0 je objekt neviditeľný a pri hodnote ≥ 1 je objekt plne viditeľný. Ak je táto hodnota v rozmedzí (0-1) je tento objekt priehľadný. Ďalej nasledujú samotné parametre operácie typu *Expression*, *Point*, *Line*, *Surface* a ďalšie. Parametrom typu *Expression* sa dá nastaviť názov, podľa ktorého sa na ne bude odkazovať. Tento názov ale nie je povinný a pre nereferencované parametre stačí nechať políčko prázdné. Parametre typu *Expression*, ktoré sú týmto názvom pomenované, sa následne zobrazia aj v zozname referencovaných parametrov. V poslednom stĺpci sa nachádza nápoveda k danému parametru.

Po zadaní hodnoty sa overuje, či je táto hodnota valídna pre daný parameter. To zahŕňa testovanie, či hodnota zadaná parametru *Expression* je platný výraz a pri ostatných parametroch sa testuje, či existuje objekt s rovnakým názvom ako bol zadaný a či je tento objekt správnym typom prípadne podtypom (viac o typoch objektov v kapitole 3). Tiež sa testuje aj názov objektu a názov referencovaného parametra na unikátnosť. Ak hodnota parametra nesplňuje niektorú požiadavku, je toto políčko označené červenou farbou pozadia, čo upozorňuje užívateľa na chybu. Ak je hodnota valídna, políčko sa označí zeleným pozadím. Prázdné políčko je označené bielym pozadím.



Obr. 4.5: Dialógové okno na pridanie, vloženie a úpravu operácií. Ukážka úpravy bodu s názvom *p3* s jednou chybne zadanou hodnotou

V hlavnom okne aplikácie je možné zadať operácie v textovej podobe. Tento text sa pri zadaní testuje a zafarbuje podľa validity. Ak je zadaný text valídný, zafarbí sa na zeleno. Naopak ak je v niektornej operácii chyba, či už syntaktická alebo sémantická, zafarbí sa na červeno. Viac o možnostiach pridávania, úpravy a odoberanie objektov je v časti 4.3.

4.6 Príklady použitia knižnice

Okrem grafického editora, ktorý je popísaný v časti 4.5, som vytvoril aj ďalšie dve aplikácie znázorňujúce použitie tejto knižnice na tvorbu parametrických trojrozmerných modelov. V tejto časti sú tieto aplikácie opísané. **[Opisanie vytvorených ukazok]**

Aplikácia s použitím glfw

Aplikácia s použitím Qt

4.7 Metrika

Počet riadkov zdrojového kódu: 7000 riadkov

Počet geometrických operácií: 44

- 16 bodových
- 10 čiarových
- 10 plošných

- 8 objemových

Použité externé knižnice:

- GLEW - Vykreslovanie modelu do okna pomocou OpenGL
- GLM - Geometrické výpočty
- Libigl - Operácie konštruktívnej geometrie nad trojrozmernými objektami

Kapitola 5

Záver

[[čo bolo cieľom práce - asi 10 slov. Ako sa mi to podarilo splniť,]]"Ciel práce sa podarilo splniť."

[[Odpovedať na jednotlivé časti zo zadania, a zakomponovať to tak, aby to nebolo z toho zjavné,že odpovedám na dané body.]]

[[Opísat jednotlivo kapitoly/o čom sú dané kapitoly. - približne 15 riadkov]]

V druhej kapitole práce som sa venoval parametrickým modelom, ich históriaou a systémami, ktoré umožňujú v súčasnom svete tvoriť parametrické modely.

Tretia kapitola sa zaoberá geometrickými objektami a geometrickými operáciami, ktoré ich tvoria.

[[Počet riadkov a súborov práce?]]

Do budúcnosti by som chcel zamerať na dorobenie priehľadnosti objektov. Táto možnosť je už z časti implementovaná. Je možné jednotlivým objektom nastaviť farbu aj s priehľadnosťou, ale pre zameranie práce na rôzne iné časti, zatial nie sú tieto objekty vykreslované ako priehľadné. Tiež je vhodné pridanie textúr na vytvorené objekty Ďalej by som chcel pridať nahrávanie animácie, kde by si užívateľ mohol nastaviť rôzne nastavenia, ako je rozmer, kódelek alebo dĺžka nahrávania a tiež by mohol medzi jednotlivými snímkami nastavovať parametre modelu alebo model úplne prerobiť, pridaním alebo odobraním operácií.

Literatúra

- [1] Významné prvky trojuholníka. Technická správa, Univerzita Mateja Bela v Banskej Bystrici.
URL <https://lms.umb.sk/mod/book/view.php?id=76930&chapterid=1746>
- [2] Ames, J.: *SOLIDWORKS Simulation: Which Package Is Right for Me?* 2017.
URL <https://hawkridgesys.com/blog/solidworks-simulation-package-right>
- [3] Boi; Katz, A.; Liu, D.: Centroid of a Triangle | Brilliant Math & Science Wiki.
URL <https://brilliant.org/wiki/triangles-centroid/>
- [4] Bourke, P.: Point, Line, Plane. 1988–1998.
URL <http://paulbourke.net/geometry/pointlineplane/>
- [5] Cadline: *Autodesk Inventor: Rounding Parameters.* 2017.
URL <https://www.youtube.com/watch?v=KAeaonxJ5Es>
- [6] CZ, T.: *CATIA R2017x.* 2017.
URL <https://www.youtube.com/watch?v=1NYHYrbqPZI>
- [7] Davis, D.: *Modelled on Software Engineering: Flexible Parametric Models in the Practice of Architecture.* Dizertačná práca, RMIT University, 2013.
- [8] Feuerbach, K. W.; Buzengeiger, C. H. I.: Eigenschaften einiger merkwürdigen Punkte des geradlinigen Dreiecks und mehrerer durch sie bestimmten Linien und Figuren.
<http://resolver.sub.uni-goettingen.de/purl?PPN512512426>, 1822.
- [9] Frazer, J.: Parametric Computation: History and Future. *Architectural Design*, ročník 86, č. 2, 2016: s. 18–23, doi:10.1002/ad.2019,
<https://onlinelibrary.wiley.com/doi/pdf/10.1002/ad.2019>.
URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/ad.2019>
- [10] FreeCAD: *Freecad default.* 2018.
URL https://www.freecadweb.org/wiki/About_FreeCAD
- [11] Gaget, L.: Top 8 of the best parametric modeling software. 2018.
URL <https://www.sculpteo.com/blog/2018/03/07/top-8-of-the-best-parametric-modeling-software/>
- [12] Jacobson, A.; Panozzo, D.; aj.: libigl: A simple C++ geometry processing library. 2018.
URL <http://libigl.github.io/libigl/>

- [13] Katz, A.; Prakash, S.; Khim, J.: *Nine Point Circle*.
URL <https://brilliant.org/wiki/nine-point-circle/>
- [14] Kimberling, C.: ENCYCLOPEDIA OF TRIANGLE CENTERS. 2019.
URL <http://faculty.evansville.edu/ck6/encyclopedia/ETCPart16.html>
- [15] Munger, S.: *Earth: The Watergate complex, famous for more than just burglars and tapes*. 2015.
URL <https://seanmunger.com/2015/06/17/earth-the-watergate-complex-famous-for-more-than-just-burglars-and-tapes/>
- [16] Page, J.: Incenter coordinates of a triangle given the vertex coordinates (Coordinate Geometry) - Math Open Reference. 2011.
URL <https://www.mathopenref.com/coordincenter.html>
- [17] PTC: *PTC Creo Parametric in Action - PTC*. 2013.
URL <https://www.youtube.com/watch?v=kQYCmq8EmNk>
- [18] Recrosio, E.: Save Time with Parametric Design for 3D Printing. 2017.
URL <https://www.sculpteo.com/blog/2017/05/10/save-time-with-parametric-design-for-3d-printing/>
- [19] Rutten, D.: *A screen shot of the Grasshopper main window*. 2011.
URL https://en.wikipedia.org/wiki/Grasshopper_3D
- [20] SOLIDWORKS: *Radeon Technologies Group*. 2017.
URL <https://blogs.solidworks.com/solidworksblog/2017/02/solidworks-world-partner-pavilion-preview-part-v.html>
- [21] Weisstein, E. W.: Perpendicular Bisector – from Wolfram MathWorld.
URL <http://mathworld.wolfram.com/PerpendicularBisector.html>
- [22] Weisstein, E. W.: Kimberling Center – from Wolfram MathWorld. 2004.
URL <http://mathworld.wolfram.com/KimberlingCenter.html>
- [23] Woodbury, R.: *Elements of Parametric Design*. Routledge, 2010, ISBN 9780415779876.
URL <https://books.google.sk/books?id=Zho5QAAACAAJ>
- [24] Zhiyanm: Triangle center. 2012.
URL https://en.wikipedia.org/wiki/Triangle_center

Príloha A

Zoznam geometrických operácií

Geometrické operácie z ktorých sa skladajú parametrické 3D modely. Tieto operácie sa začínajú názvom operácie, nasledované parametrami v zátvorkách, ktoré sú oddelené čiarkou.

Bodové operácie

- **Point(string name, expression X, expression Y, expression Z, RRGGBA color)** - Create point with absolut position
- **Point(string name, expression X, expression Y, expression Z, Point parent, RRGGBA color)** - Create point with relative position of entered point
- **LinearInterpolationDist(string name, Point from, Point to, expression distance, RRGGBA color)** - Create point in distance from first point to second
- **LinearInterpolationPerc(string name, Point from, Point to, expression percentage, RRGGBA color)** - Create point in percentual distance from first point to second
- **Intersection_Plane_Line(string name, Line l, Surface s, RRGGBA color)** - Create Point on position where line l intersect surface s
- **SurfaceCenterBoundingSquare(string name, Surface s, RRGGBA color)** - Create point on position in middle of bounding box of entered surface
- **SurfaceCenterAverage(string name, Surface s, RRGGBA color)** - Create point on avarage position of all points of entered surface
- **Centroid(string name, Triangle t, RRGGBA color)** - Create point on centroid of triangle
- **Incenter(string name, Triangle t, RRGGBA color)** - Create point on center of inscribed circle in triangle
- **Circumcenter(string name, Triangle t, RRGGBA color)** - Create point on center of circumscribed circle over triangle
- **Orthocenter(string name, Triangle t, RRGGBA color)** - Create orthocentrum of triangle

- **NinePointCenter(string name, Triangle t, RRGGBBAA color)** - Create point at center of nine points circle of triangle
- **ObjectCenterBoundingBox(string name, Object3D o, RRGGBBAA color)**
- Create point on position in middle of bounding box of entered object
- **ObjectCenterAverage(string name, Object3D o, RRGGBBAA color)** - Create point on average position of all points of entered object
- **LineFirstPoint(string name, Line l, RRGGBBAA color)** - Begin point of line
- **LineSecondPoint(string name, Line l, RRGGBBAA color)** - End point of line

Úsečkové operácie

- **Line(string name, Point p1, Point p2, RRGGBBAA color)** - Create line, where p1 is start point and p2 is end point
- **LineNormalize(string name, Line l, RRGGBBAA color)** - Normalize line to distance 1.0. Beginning point is same as beginning point of entered line, end point is in distance 1.0 in direction to end point of line
- **LineChangeLengthDist(string name, Line l, expression distance, RRGGBBAA color)** - Beginning point is same as beginning point of entered line, end point is in entered distance in direction to end point of line
- **LineChangeLengthPerc(string name, Line l, expression percent, RRGGBBAA color)** - Beginning point is same as beginning point of entered line, end point is in entered percentual distance in direction to end point of line
- **MinLineBetweenLineAndLine(string name, Line l1, Line l2, RRGGBBAA color)** - Minimal line between two lines
- **MinLineBetweenPointAndLine(string name, Point p, Line l, RRGGBBAA color)**
- **MinLineBetweenPointAndSurface(string name, Point p, Surface s, RRGGBBAA color)** - Minimal line between point and surface
- **SurfaceNormal(string name, Surface s, RRGGBBAA color)** - Normal vector of surface
- **LineRelocationByPoint(string name, Line l, Point p, RRGGBBAA color)**
- Line is moved to new location. Beginning point is entered point and end point is in distance of entered line with same direction.
- **CrossProduct(string name, Line l1, Line l2, RRGGBBAA color)** - Create cross-product of entered lines

Plošné operácie

- **RectangleFromLine(string name, Line l, expression width, Point pointOn-Surface, type, RRGGBBAA color)** - Create rectangle surface from line
- **RectangleFromLine(string name, Line l, expression width, Line normal, type, RRGGBBAA color)** - Create rectangle surface from line by normal
- **Circle(string name, Point center, expression radius, Line lineNormal, RRGGBBAA color)** - Create circle by center, radius and normal vector
- **Circle(string name, Point center, Point outlinePoint, Point planePoint, RRGGBBAA color)** - Create circle by 3 points
- **Triangle(string name, Line l, Point p, RRGGBBAA color)** - Create triangle by line and point
- **Triangle(string name, Point p1, Point p2, Point p3, RRGGBBAA color)** - Create triangle by 3 points
- **Rectangle(string name, Point center, expression X, expression Y, expression Roll , Line normal, RRGGBBAA color)** - Create rectangle
- **Polygon(string name, Point p1, Point p2, Point p3, ..., RRGGBBAA color)**
- Create Polygon by connecting points, minimum 3 points. Multiple points are divided by ;
- **Circumscribed(string name, Triangle t, RRGGBBAA color)** - Create circumscribed circle around triangle
- **Inscribed(string name, Triangle t, RRGGBBAA color)** - Create inscribed circle in triangle

Objemové operácie

- **Pyramid(string name, Surface s, expression distance, RRGGBBAA color)**
- Create pyramid from surface and distance from center
- **Pyramid(string name, Surface s, Point p, RRGGBBAA color)** - Create pyramid from surface and specific point
- **Extrude(string name, Surface s, expression distance, RRGGBBAA color)**
- Enlarge surface to 3D with distance
- **Sphere(string name, Point center, expression radius, RRGGBBAA color)**
- Sphere
- **BooleanUnion(string name, Object o1, Object o2, RRGGBBAA color)** - Union objects
- **BooleanIntersection(string name, Object o1, Object o2, RRGGBBAA color)** - Intersection between objects

- **BooleanMinus(string name, Object o1, Object o2, RRGGBBAA color)** -
First object minus second object
- **BooleanXOR(string name, Object o1, Object o2, RRGGBBAA color)** -
Symmetric difference between objects