SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Stavebná fakulta

Evidenčné číslo: SvF-5342-81238

IMPLEMENTÁCIA ALGORITMOV PRE TVORBU POLYGONÁLNYCH SIETÍ V PROSTREDÍ SOFTVÉRU RHINOCEROS

Bakalárska práca

Študijný program:	Matematicko-počítačové modelovanie
Študijný odbor:	9.1.9 Aplikovaná matematika
Školiace pracovisko:	Katedra matematiky a deskriptívnej geometrie
Vedúci záverečnej práce:	doc. Mgr. Mariana Remešíková, PhD.
Konzultant:	Mgr. Marián Šagát

Bratislava 2018 **Aneta Alexandra Ožvat**

Slovenská technická univerzita v Bratislave
Katedra matematiky a deskriptívnej geometrie

Stavebná fakulta
Akademický rok: 2017/2018
Evidenčné číslo: SvF-5342-81238

STU SvF

ZADANIE BAKALÁRSKEJ PRÁCE

Študentka:	**Aneta Alexandra Ožvat**
ID študenta:	81238
Študijný program:	matematicko-počítačové modelovanie
Študijný odbor:	9.1.9. aplikovaná matematika
Vedúca práce:	doc. Mgr. Mariana Remešíková, PhD.
Konzultant:	Mgr. Marián Šagát
Miesto vypracovania:	KMDG

Názov práce: **Implementácia algoritmov pre tvorbu polygonálnych sietí v prostredí softvéru Rhinoceros**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Cieľom práce je implementovať algoritmy pre tvorbu optimálnych polygonálnych sietí – diskretizovaných povrchov. Výstupom by mal byť aspoň jeden použiteľný plug-in pre softvér Rhinoceros, resp. jeho nadstavbu Grasshopper. Algoritmy by mali byť implementované v jazyku C#.

Rozsah práce: 20-30 strán

Riešenie zadania práce od:	17. 04. 2018
Dátum odovzdania práce:	03. 05. 2018

Aneta Alexandra Ožvat
študentka

prof. RNDr. Radko Mesiar, DrSc.
vedúci pracoviska

prof. RNDr. Karol Mikula, DrSc.
garant študijného programu

POKYNY
na vypracovanie bakalárskej práce

Úvodné ustanovenie

V zmysle zákona č. 131/2002 Z. z. o vysokých školách a o zmene a doplnení niektorých zákonov v znení neskorších predpisov je súčasťou štúdia podľa každého študijného programu aj záverečná práca. Jej obhajoba patrí medzi štátne skúšky. Záverečnou prácou pri štúdiu podľa bakalárskeho študijného programu je bakalárska práca. Podkladom na vypracovanie bakalárskej práce je zadanie bakalárskej práce

Štruktúra záverečnej práce

- titulný list,
- zadanie záverečnej práce,
- pokyny na vypracovanie,
- vyhlásenie autora,
- názov a abstrakt v slovenskom a v anglickom jazyku (spolu v rozsahu jednej strany),
- obsah s očíslovaním kapitol,
- zoznam príloh,
- zoznam skratiek a značiek,
- text samotnej práce (odporúčané členenie),
  - úvod,
  - súčasný stav problematiky,
  - ciele záverečnej práce,
  - vlastné riešenie členené na kapitoly podľa charakteru práce,
  - zhodnotenie dosiahnutých výsledkov resp. navrhnutých riešení,
  - záver,
- resumé v slovenskom jazyku v rozsahu spravidla 10 % rozsahu ZP (len pre práce vypracované v cudzom jazyku),
- zoznam použitej literatúry,
- prílohy (výkresy, tabuľky, mapy, náčrty) vrátane postera s rozmermi 1000x700 mm.

Rozsah a forma

1. Obsah a forma záverečnej práce musí byť spracovaná v zmysle vyhlášky MŠVVaŠ SR č. 233/2011 Z. z., ktorou sa vykonávajú niektoré ustanovenia zákona č. 131/2002 Z. z. a v zmysle Metodického usmernenia č. 56/2011 o náležitostiach záverečných prác.
2. Vyžadovaný rozsah bakalárskej práce je 20 až 30 strán. Odovzdáva sa v dvoch vyhotoveniach. Jedno vyhotovenie musí byť viazané v pevnej väzbe (nie hrebeňovej) tak, aby sa jednotlivé listy nedali vyberať. Rozsiahle grafické prílohy možno v prípade súhlasu vedúceho práce odovzdať v jednom vyhotovení.
3. Autor práce je povinný vložiť prácu v elektronickej forme do akademického informačného systému. Autor zodpovedá za zhodu listinného aj elektronického vyhotovenia.

4. Po vložení záverečnej práce do informačného systému, predloží autor fakulte ním podpísaný návrh licenčnej zmluvy. Návrh licenčnej zmluvy je vytvorený akademickým informačným systémom.
5. Odporúčaný typ písma je Times New Roman, veľkosť 12 a je jednotný v celej práci. Odporúčané nastavenie strany - riadkovanie 1,5, okraj vnútorný 3,5 cm, vonkajší 2 cm, zhora a zdola 2,5 cm, orientácia na výšku, formát A4.
6. Obrázky a vzorce sa číslujú v rámci jednotlivých kapitol (napr. obr. 3.1 je obrázok č. 1 v kapitole 3). Vzorce sa číslujú na pravom okraji riadku v okrúhlych zátvorkách - napr. (3.1).
7. Všetky výpočty musia byť usporiadané tak, aby bolo možné preveriť ich správnosť.
8. Pri všetkých prevzatých vzorcoch, tabuľkách, citovaných častiach textu musí byť uvedený prameň.
9. Citovanie literatúry vrátane elektronických materiálov sa uvádza podľa STN ISO 690 (01 0197): 2012. *Informácie a dokumentácia. Návod na tvorbu bibliografických odkazov na informačné pramene a ich citovanie.*
10. Príklad zoznamu bibliografických odkazov:
    ABELOVIČ, J. a kol.: *Meranie v geodetických sieťach.* Bratislava: Alfa 1990. 104 s. ISBN 0-1554-9173.
    MICHALČÁK, O. – ADLER, E.: Výskum stability dunajských hrádzí. In: *Zborník vedeckých prác Stavebnej fakulty SVŠT.* Bratislava: Edičné stredisko SVŠT 1976, s. 17-28. ISBN 0-3552-5214.
    ŠÜTTI, J.: Určovanie priestorových posunov stavebných objektov. *Geodetický kartografický obzor*. 2000, roč. 2, č. 3, s. 8-16. ISSN 0811-6900.
    Article 18. Technical Cooperation. http://www.lac.uk/iso/tc456 (2013-09-28)
11. Za jazykovú a terminologickú správnosť záverečnej práce zodpovedá študent.
12. Formu postera (elektronická alebo aj tlačená) určí garant študijného programu.
13. Vzor pre poster je uvedený na dokumentovom serveri v akademickom informačnom systéme univerzity.

.....
podpis garanta študijného programu

Ustanovenia týchto pokynov som vzal na vedomie. Som si vedomý(á), že ak nebude moja bakalárska práca vypracovaná v súlade s týmito pokynmi, nebude prijatá na obhajobu.

V Bratislave ....
podpis študenta

Čestné prehlásenie

Vyhlasujem, že som bakalársku prácu „Implementácia algoritmov pre tvorbu polygonálnych sietí v prostredí softvéru Rhinoceros“ vypracovala samostatne s použitím citovanej literatúry a s odbornou pomocou vedúcej práce.

4. mája 2018 vlastnoručný podpis

.....

Poďakovanie

Týmto spôsobom by som sa chcela poďakovať vedúcej doc. Mgr. Mariane Remešíkovej, PhD. za cenné rady, usmerňovanie pri realizácii tejto bakalárskej práce a predovšetkým za trpezlivosť a ochotu. Moja vďaka patrí aj Mgr. Mariánovi Šagátovi za cenné rady, ktoré mi poskytol pri implementácii algoritmov. Zároveň sa chcem poďakovať aj rodine a blízkym za neustálu podporu a pochopenie.

Súhrn

V bakalárskej práci sa zaoberáme implementáciou matematického modelu vývoja plochy podľa strednej krivosti v prostredí populárneho komerčného softvéru Rhinoceros. Naším cieľom je vytvoriť funkčný komponent pre Rhinoceros, respektíve pre jeho nadstavbu Grasshopper, ktorý by bol schopný generovať diskrétne minimálne plochy pre zadanú okrajovú krivku. Práca je rozdelená na dve súvislé časti. V prvej časti rozoberáme matematický model, ktorý iba stručne popíšeme a následne sa o čosi podrobnejšie venujeme numerickej aproximácii tohto modelu. V druhej časti popíšeme prostredie, v ktorom sme pracovali, čiže softvér Rhinoceros a Grasshopper. Tiež sa podrobne venujeme spôsobu, akým sme postupovali pri implementácii algoritmu. Na záver uvedieme ukážky, ako náš vytvorený komponent funguje.

Kľúčové slová

Minimálna plocha, stredná krivosť, vývoj plôch, metóda konečných objemov, Rhinoceros, Grasshopper

Abstract

In this bachelor thesis we deal with the implementation of the mathematical model of evolution of the surface employing the mean curvature flow in the popular commercial software Rhinoceros. Our goal is to create the functional component for Rhinoceros, or more precisely for its plug-in Grasshopper, which has the capacity to generate discrete minimum surface of the given boundary curve. The thesis is divided into two continuos parts. In the first part we take the mathematical model which we briefly explain and then will expound more on the numeric approximation of this model. In the second part we describe the enviroment in which we worked on, that is the software Rhinoceros and Grasshopper. In detail we will also deal with the procedure by which we implemented the algorithm. In end of the thesis we will show examples of how our component works.

Keywords

Minimal surface, mean curvature flow, surface evolution, finite volume method, Rhinoceros, Grasshopper

Obsah

Kapitola 1

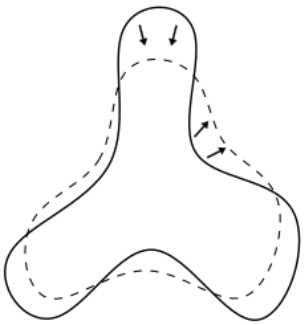
Motivácia

1.1 Minimalizácia plošného obsahu a minimálne plochy

Táto práca sa zaoberá tvorbou minimálnych plôch, ktoré sú definované ako plochy s nulovou strednou krivosťou. Preto si pre začiatok intuitívne načrtneme, ako otázka minimalizácie plochy vôbec s krivosťou súvisí. Pre lepšie predstavenie si celého problému uvedieme pár príkladov z reálneho života.

1.1.1 Uzavretá plocha

Na každý dej v prírode je potrebná energia, a keďže vieme, ako ťažko sa energia vyrába, nemali by sme s ňou plytvať. Ani príroda energiou neplytvá a snaží sa, aby všetko, čo vytvorí, bolo vytvorené s čo najmenšou spotrebou energie. Keď sa tak pozrieme napríklad na dažďovú kvapku, aký má tvar a akú plochu má jej povrch? Je to malá guľa, keďže teraz uvažujeme kvapku v pokoji, a to znamená, že jej povrch je sféra. Ale ako príroda vie, že plocha obaľujúca objem vodnej kvapky má byť práve sféra? Keďže sa riadi princípom šetrenia energie, tak vytvorí pre danú kvapku práve obal zaberajúci minimálnu plochu.



Obr. 1.1: Náčrt jedného variantu iného tvaru dažďovej kvapky v 2D, s naznačeným smerov vývoja do tvaru s minimálnym povrchom

Teoreticky si môžeme predstaviť, že by dažďová kvapka mala nejaký iný tvar (1.1). Takýto tvar zaberá nejakú plochu a s istotou vieme povedať, že to nie je minimálna plocha - hovorí o tom známa izoperimetrická nerovnosť. Príroda na jej vytvorenie spotrebuje viac energie ako minimum, čo vedie k porušeniu princípu šetrenia energie. Ako si môžeme všimnúť, pri tejto ploche sú určité úseky viac zakrivené, čiže majú väčšiu krivosť ako iné úseky. Aby sme dosiahli minimálny povrch dažďovej kvapky, musíme úseky s veľkou krivosťou vyhladzovať. Dažďová kvapka má určitý objem, ktorý sa musí zachovať, čiže keď sa krivé oblasti budú vyhladzovať, iné oblasti sa budú zakrivovať. Takto sa dosiahne, že v každom bode povrchu bude napokon konštantná krivosť. V tomto stave bude povrch kvapky zaberať minimálnu plochu a príroda minie čo najmenej energie. S tohto dôvodu dažďové kvapky nebudú mať žiaden iný tvar ako tvar sféry.

1.1.2 Plocha s okrajom

Poďme sa ešte pozrieť, ako je to pri bublifuku. Asi každý vie, ako bublifuk vyzerá, no pre úplnosť si ho popíšeme. Je to vlastne mydlová voda v nádobke, a obruč, alebo iný zaujímavejší tvar, na tyčke. Nás konkrétne zaujíma obruč, ktorá predstavuje okraj bubliny, ktorá sa vytvorí pri namočení obruče do vody. Táto obruč je rovinnou krivkou a teda aj bublina, ktorá sa vytvorí, je rovina, keď uvažujeme bublifuk v pokoji. Rovinná plocha nemá žiadnu krivosť, čiže zaberá najmenšiu možnú plochu. Teraz si ale predstavme, že obruč nie je obruč, ale má zaujímavejší tvar, napríklad tvar okraja konského sedla. Bublina, ktorá sa vytvorí pri zmočení bublifuku do mydlovej vody, môže mať v podstate akýkoľvek tvar, no plocha tejto bubliny bude mať určite minimálnu veľkosť.

Odpoveď na otázku, prečo blana bubliny bude mať minimálnu plochu je jednoduchá – princíp šetrenia energie. Príroda si šetrí energiu, a preto zariadi, aby v každom bode tejto plochy bola hodnota strednej krivosti nulová. Tým dosiahne, že aj keď plocha blany nie je rovinná, zaberá čo najmenšiu plochu. Takéto plochy potom nazývame minimálne plochy.

Vo všeobecnosti potom ak vezmeme hocaký okraj bublifuku, vytvorí sa nám bublina s minimálnou plochou. Samozrejme tento fakt sa začal využívať hneď aj na osožnejšie účely ako len na vytváranie zaujímavých tvarov pri bublifuku. Architekt Frei Otto a aj mnohí iní, sa inšpirovali presne týmto faktom o mydlových bublinách, keďže to má mnoho estetických kvalít (Obr. 1.2). Okrem toho, že takto navrhnuté strechy budú zaujímavé na pohľad, keďže sú to minimálne plochy, bude aj spotreba materiálu minimálna a z pohľadu statiky bude potrebné minimálne množstvo nosných prvkov. No otázka je, ako takúto minimálnu plochu pre zadaný okraj získame. Presne to bude témou tejto bakalárskej práce, popísať algoritmus na vytváranie diskrétnych minimálnych plôch pre zadaný okraj a implementovať ho v prostredí komerčného softvéru Rhinoceros.

Vychádzať budeme z prác [1] a [2], v ktorých bol nami zvolený algoritmus popísaný. Naším hlavným príspevkom je práve implementácia v prostredí Rhinoceros, ktorá má potenciál byť využívaná širokou komunitou používateľov tohto softvéru.

Obr. 1.2: Vľavo, model použitý na zistenie tvaru strechy pre, vpravo uvedený, nemecký pavilón v Montreale, v Kanade, postavený v roku 1967 a navrhnutý architektmi F. Ottom a R. Gutbrodom. [5]

Kapitola 2

Matematický model a jeho aproximácia

2.1 Tvorba matematického modelu

Definujme si zobrazenie $F^0 : X \to \mathbb{R}^3$, ktoré je vložením 2-rozmernej Reimannovej variety do 3-rozmerného Euklidovského priestoru. Vývoj plochy $X^0 = F^0(X)$ predstavuje jednoparametrickú triedu vložení $F : X \times [0, t_f] \to \mathbb{R}^3$. Keď je daný fixný bod $x \in X$, tak zobrazenie $x \mapsto F(x) = F(x, \cdot)$ je hladká krivka v $\mathbb{R}^3$. Označme si $v^t(x)$ dotykový vektor krivky v danom bode $F^t(x) = F(x, t)$ a majme zobrazenie $v : X \times [0, t_f] \to \mathbb{R}^3$. To predstavuje rýchlostné pole daného vývoja. Inými slovami, zobrazenie F je riešením rovnice

$$\partial_t F = v. \tag{2.1}$$

Rovnicu (2.1) uvažujeme s počiatočnou podmienkou $F(x, 0) = F^0(x)$ a pre plochu X s okrajom sú uvažované Dirichletove okrajové podmienky v nasledujúcej podobe

$$F(x, t) = F^0(x), \qquad x \in \partial X, \quad t \in [0, t_f]. \tag{2.2}$$

Voľbou Dirichletových okrajových podmienok sme dosiahli, aby okraj oblasti bol statický. Samozrejme, dá sa počítať aj s inými okrajovými podmienkami, no my sa v našej práci budeme zaoberať iba s okrajovými podmienkami (2.2).

Poďme si teraz rýchlosť v rozpísať ako súčet rýchlostí vývoja v normálovom smere

(v_N) a v tangenciálnom smere (v_T). Ako neskôr uvidíme, tento krok bude výhodný z praktického hľadiska. Platí teda

$$\partial_t F = v_N + v_T. \tag{2.3}$$

Každá z týchto zložiek má určitú úlohu vo vývoji plochy. Normálová zložka rýchlosti v_N vplýva na výslednú plochu tak, že posúva a tvaruje vloženú plochu X^t v okolitom priestore a tangenciálna zložka rýchlosti v_T iba posúva body po ploche X^t, čiže neposúva ich nijako v okolitom priestore. Popísaný jav sa bude dať pozorovať pri implementácii daných rovníc a na diskretizácii bude možné dobre vidieť, ako sa prejaví zahrnutie tangenciálnej zložky. Body siete sa budú posúvať pozdĺž plochy a vhodnou voľbou v_T sa potom dá dosiahnuť čo najvhodnejšie rozloženie bodov na sieti.

V našej práci budeme uvažovať vývoj plôch v $\mathbb{R}^3$ riadený strednou krivosťou. Polohový vektor $F(x,t)$, pri toku podľa strednej krivosti môžeme zapísať ako

$$\partial_t F = HN. \tag{2.4}$$

Ako si môžeme všimnúť v tomto modeli je prítomná iba normálová zložka rýchlosti vyjadrená ako $v_N = HN$, kde $H(x,t)$ predstavuje strednú krivosť vyvíjajúcej sa plochy v bode x v čase t a $N(x,t)$ predstavuje jednotkovú normálu plochy $X^t = F^t(X)$ v bode $x \in X$. Vektor $h = HN$ predstavuje vektor strednej krivosti (smerujúci v smere normály). Použitím vzorca $h = \triangle_{g_F} F$ môžeme prepísať model (2.4) ako

$$\partial_t F = \triangle_{g_F} F, \tag{2.5}$$

s uvažovanou počiatočnou podmienkou

$$F(x,0) = F^0(x), \qquad x \in X,$$

a okrajovou podmienkou

$$F(x,t) = F^0(x), \qquad x \in \partial X, \quad t \in [0, t_f].$$

Symbolom $\triangle_{g_F}$ značíme Laplace-Beltramiho operátor.

2.1.1 Tangenciálna redistribúcia bodov povrchu

Ako sme už spomínali, pri zadanom vývoji podľa strednej krivosti máme danú iba rýchlosť v smere normály, $v_N = HN$. Aby sme dosiahli lepšiu kontrolu nad rozmiestnením diskretizačných bodov vyvíjajúcej sa diskrétnej plochy, môžeme model (2.5)

rozšíriť o vhodne zvolenú tangenciálnu rýchlosť v_T. Teda budeme mať

$$\partial_t F = \triangle_{g_F} F + v_T. \tag{2.6}$$

Pri variete X máme metriku g_X a mieru ξ indukovanú touto metrikou. Metrický tenzor g_{F^t} indukovaný vložením F^t indukuje inú mieru χ_{F^t} na X. Vzťah medzi týmito dvoma mierami je nasledovný

$$d\chi_{F^t} = G^t d\xi. \tag{2.7}$$

Veličinu G^t nazývame hustotou plochy a symbol G bude označovať vyvíjajúcu sa hustotu plochy zodpovedajúcu vloženiu F. Pri vývoji diskretizovanej plochy bude naším cieľom postupne dosiahnuť čo najrovnomernejšie rozmiestnenie bodov diskretizačnej siete. V spojitej formulácii môžeme túto požiadavku vyjadriť tak, že chceme aby

$$G \xrightarrow[t\to\infty]{} C,$$

kde C je kladná konštanta.

Pri konštrukcii dotykového poľa budeme predpokladať, že ide o gradientné pole teda, že ho možno vyjadriť ako $\nabla_{g_F}\psi$, kde ψ je funkcia $\psi : X \to R$. Podľa práce [2] na základe uvedených podmienok pre funkciu ψ platí

$$\triangle_{g_F} \psi = v_N \cdot h - [v_N \cdot h]_\chi + \left(C\frac{A}{G} - 1\right)\omega, \tag{2.8}$$

kde $\omega > 0$ predstavuje rýchlosť tangenciálnej redistribúcie a $[v_N \cdot h]_\chi$ je priemerná hodnota $v_N \cdot h$ vzhľadom na mieru χ_{F^t}. S cieľom zabezpečiť jednoznačnosť ψ budeme rovnicu (2.8) uvažovať s vhodne zvolenou okrajovou podmienkou.

2.2 Numerická aproximácia

V tejto časti si predstavíme diskretizáciu nášho problému. Budeme pracovať s dvomi alternatívami plôch. Prvá bude plocha bez okraja, čiže ako sme už v motivácii avizovali, môžeme si predstaviť dažďovú kvapku. Druhá alternatíva bude plocha s okrajom, pri ktorej budeme uvažovať konkrétne Dirichletove okrajové podmienky, ako sme to uviedli pri tvorbe matematického modelu. Pre názornosť si môžeme znova za plochou s okrajom predstaviť opisovanú mydlovú blanu na bublifuku.

2.2.1 Časová diskretizácia

Pre diskretizáciu modelu (2.6) sme v rámci našej práce použili explicitnú a semi -implicitnú metódu. Časovú deriváciu budeme aproximovať konečnými diferenciami. Nech τ predstavuje časový krok a $M = t_f/\tau$ je počet časových krokov. V prípade explicitnej schémy dostaneme

$$\frac{F^n - F^{n-1}}{\tau} = \triangle_{F^{n-1}} F^{n-1} + v_T^{n-1}, \tag{2.9}$$

pre $n = 1, \ldots, M$, z čoho vieme priamo vyjadriť F^n. V prípade semi-implicitnej schémy použijeme tangenciálnu rýchlosť z predchádzajúceho časového kroku. Laplace-Beltramiho operátor vyčíslime z plochy F^n, avšak vzhľadom k metrike $g_{F^{n-1}}$ z predchádzajúceho časového kroku. Teda dostaneme

$$\frac{F^n - F^{n-1}}{\tau} = \triangle_{F^{n-1}} F^n + v_T^{n-1}, \tag{2.10}$$

pre $n = 1, \ldots, M$, kde symbol $\triangle_{F^{n-1}}$ predstavuje Laplace-Beltramiho operátor z minulého časového kroku.

2.2.2 Priestorová diskretizácia

Priestorová diskretizácia je uskutočnená aplikáciou metódy konečných objemov. Plochu X aproximujeme trojuholníkovou sieťou, ktorá bude obsahovať vrcholy x_i, $i = 1 \ldots, n_F$. Triangulácia plochy X bude určovať, ako sa trianguluje vložená plocha $X^n = F^n(X)$ s vrcholmi $F_i^n = F^n(x_i), i = 1, \ldots, n_F$. Pre naše výpočty si potrebujeme vytvoriť kontrolné objemy $V_i, i = 1, \ldots, n_F$. Skonštruujeme ich pomocou barycentrického delenia trojuholníkov triangulácie plochy X.

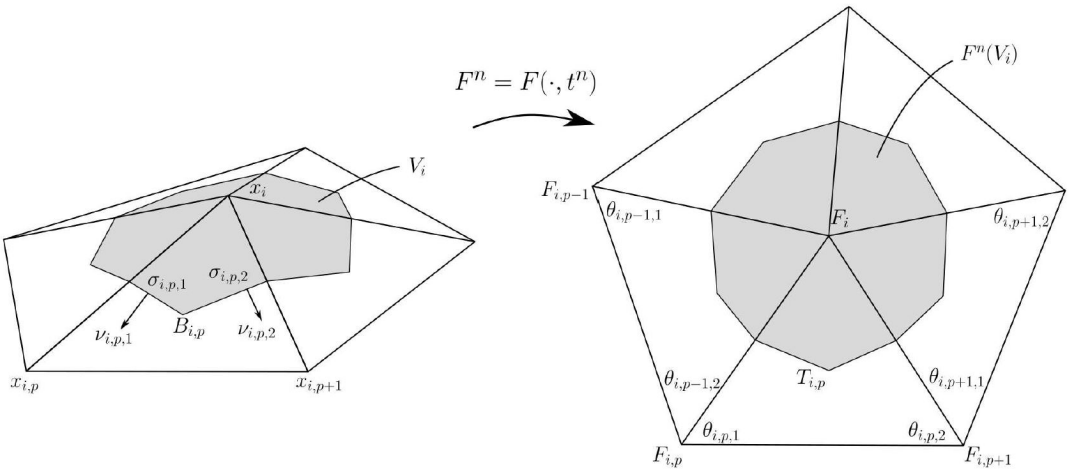
Poďme teraz rovnicu (2.10) zintegrovať cez konečný objem. Dostaneme

$$\int_{V^i} \frac{F^n - F^{n-1}}{\tau} d\chi_{F^{n-1}} = \int_{V^i} \triangle_{F^{n-1}} F^n d\chi_{F^{n-1}} + \int_{V^i} v_T^{n-1} d\chi_{F^{n-1}}. \tag{2.11}$$

Ďalším krokom bude aproximácia jednotlivých integrálov. Pre integrál na ľavej strane rovnice máme

$$\int_{V^i} \frac{F^n - F^{n-1}}{\tau} d\chi_{F^{n-1}} \approx A_i^{n-1} \frac{F_i^n - F_i^{n-1}}{\tau}, \tag{2.12}$$

kde $A_i^n = \chi_{F^{n-1}}(V_i)$ je plocha konečného objemu V_i pri vrchole F_i a vyznačená je na obrázku 2.1 šedou farbou.



Obr. 2.1: Vľavo, konečný objem V_i okolo vrchola x_i. Vpravo, označenie susedných vrcholov k vrcholu $F_i^n = F^n(x_i)$ a uhlov pri susedných vrcholoch. [1]

Ďalej budeme pokračovať s prvou časťou pravej strany rovnice (2.11), s integrálom Laplace-Beltramiho operátora, ktorý aproximujeme pomocou známej kotangensovej schémy [3]

$$\int_{V^i} \triangle_{F^{n-1}} F^n d\chi_{F^{n-1}} \approx \frac{1}{2} \sum_{p=1}^{m_i} (\mathrm{cotg}\theta_{i,p-1,1}^{n-1} + \mathrm{cotg}\theta_{i,p,2}^{n-1})(F_{i,p}^{n-1} - F_i^{n-1}). \tag{2.13}$$

V tomto vzťahu m_i označuje počet susedných vrcholov vrchola F_i^n a platí, že $\theta_{i,0,1}^{n-1} = \theta_{i,m_i,1}^{n-1}$. Vrcholy $\theta_{i,p,1}^{n-1}$ a $\theta_{i,p,2}^{n-1}$ sú označené na obrázku 2.1.

Pre tangenciálnu rýchlosť, čo je druhá časť pravej strany rovnice (2.11), použijeme aproximáciu

$$\int_{V^i} v_T^{n-1} d\chi_{F^{n-1}} \approx A_i^{n-1} v_{T,i}^{n-1},$$

kde $v_{T,i}^{n-1}$ je tangenciálna rýchlosť vo vrchole F_i^{n-1}.

Poďme si teraz tieto tri aproximácie zhrnúť do jedného systému, pomocou ktorého budeme schopní vypočítať nové pozície vrcholov. Systém má nasledovný tvar

$$a_i^{n-1} F_i^n + \sum_{p=1}^{m_i} b_{i,p}^{n-1} F_{i,p}^n = F_i^{n-1} + \tau v_{T,i}^{n-1}, \tag{2.14}$$

pre $n = 1, \ldots, M$ a pre každé i také, že $F_i^n \notin \partial X^n$, ak plocha, s ktorou počítame, je plocha s okrajom. V prípade, že plocha je bez okraja, tak tento tvar rovnice platí

pre všetky vrcholy. Koeficienty a_i^{n-1} a $b_{i,p}^{n-1}$ budú mať tvar

$$\begin{aligned} a_i^{n-1} &= 1 + \frac{\tau}{2A_i^{n-1}} \sum_{p=1}^{m_i} (\mathrm{cotg}\theta_{i,p-1,1}^{n-1} + \mathrm{cotg}\theta_{i,p,2}^{n-1}) \\ b_{i,p}^{n-1} &= -\frac{\tau}{2A_i^{n-1}} (\mathrm{cotg}\theta_{i,p-1,1}^{n-1} + \mathrm{cotg}\theta_{i,p,2}^{n-1}), \end{aligned}$$

pre $p = 1, \ldots, m_i$. Dirichletovu okrajovú podmienku (2.2) pre plochu s okrajom aplikujeme veľmi jednoducho,

$$F_i^n = F_i^{n-1} \tag{2.15}$$

pre vrcholy na hranici $F_i^n \in \partial X^n$.

Teda rovnica (2.14) v prípade, že máme plochu bez okraja, a rovnice (2.14) a (2.15), v prípade, že máme plochu s okrajom, formujú systém n_F lineárnych rovníc pre neznáme vrcholy $F_i^n, i = 1, \ldots, n_F$. Začiatočná pozícia vrcholov F_i^0 je daná začiatočnou podmienkou, teda $F_i^0 = F^0(x_i)$.

Pri vhodne zvolenom časovom kroku τ bude vytvorená matica systému výrazne diagonálne dominantná, čo je veľmi dobrá vlastnosť matíc, pretože zvyšuje rýchlosť konvergencie iteračných metód pri riešení lineárneho systému rovníc a tým pádom skracuje výpočtový čas pri implementácii v počítači.

Kotangensová schéma, uplatnená v rovnici (2.5), môže byť použitá aj na vyjadrenie aproximácie vektora strednej krivosti h_i^{n-1} vo vrchole F_i^{n-1}

$$h_i^{n-1} \approx \frac{1}{2A_i^{n-1}} \sum_{p=1}^{m_i} (\mathrm{cotg}\theta_{i,p-1,1}^{n-1} + \mathrm{cotg}\theta_{i,p,2}^{n-1})(F_{i,p}^{n-1} - F_i^{n-1}). \tag{2.16}$$

Potom aproximácia strednej krivosti H_i bude

$$H_i^{n-1} = \|h_i^{n-1}\|$$

2.2.3 Diskretizácia tangenciálnej rýchlosti

V tejto časti si predvedieme diskretizáciu rovnice (2.8) a začneme tým, že danú rovnicu zintegrujeme cez konečný objem V_i a opäť aproximujeme jednotlivé integrály. Podrobnejšie odvodenie každej jednotlivej časti môžeme nájsť v článku [1].

Aproximácia ľavej strany rovnice bude mať tvar

$$\int_{V^i} \triangle_{F^{n-1}} \psi^{n-1} d\chi_{F^{n-1}} \approx \frac{1}{2} \sum_{p=1}^{m_i} (\mathrm{cotg}\theta_{i,p-1,1}^{n-1} + \mathrm{cotg}\theta_{i,p,2}^{n-1})(\psi_{i,p}^{n-1} - \psi_i^{n-1}) \tag{2.17}$$

pre vnútorné vrcholy siete, čiže pre $F_i^n \notin \partial X^n$. Pre vrcholy nachádzajúce sa na hranici plochy (Obr. 2.2) bude integrál nadobúdať tvar [1]

$$\begin{aligned}\int_{V^i} \triangle_{F^{n-1}} \psi^{n-1} d\chi &\approx \frac{1}{2}\sum_{p=1}^{m_i}[\mathrm{cotg}\theta_{i,p,2}(\psi_{i,p}-\psi_i)+\mathrm{cotg}\theta_{i,p,+}(\psi_{i,p+1}-\psi_i)] \\ &= \frac{1}{2}[\mathrm{cotg}\theta_{i,1,2}(\psi_{i,1}-\psi_i)+\mathrm{cotg}\theta_{i,m_i-1,1}(\psi_{i,m_i}-\psi_i) \\ &+ \sum_{p=2}^{m_i-1}(\mathrm{cotg}\theta_{i,p,2}+\mathrm{cotg}\theta_{i,p-1,1})(\psi_{i,p}-\psi_i)].\end{aligned} \tag{2.18}$$

Ak naša sieť bude mať tvar dažďovej kvapky, čiže nebude mať vrcholy na okraji, budeme používať iba rovnicu (2.17).

Poďme si ďalej aproximovať aj pravú stranu rovnice (2.8). Prvá časť bude aproximovaná ako

$$\int_{V^i} v_N \cdot h d\chi_F \approx A_i v_{N,i} \cdot h_i = \begin{cases} A_i H_i^2 & , akF_i^n \notin \partial X^n \\ 0 & , \quad akF_i^n \in \partial X^n \end{cases}, \tag{2.19}$$

kde H_i je stredná krivosť plochy v bode F_i.

Aproximácia druhej časti pravej strany bude

$$\int_{V^i} [v_N \cdot h]_{\chi_F} d\chi_F \approx A_i [v_N \cdot h]_{\chi_F} \approx \frac{A_i}{A}\sum_{p=1}^{m_i} H_j^2 A_j, \tag{2.20}$$

kde $A = \sum_{i=1}^{n_F} A_i$. Dôležité je poznamenať, že suma v (2.20) prechádza iba vnútornými vrcholmi.

A nakoniec posledná, tretia časť pravej strany rovnice (2.8), bude aproximovaná ako [1]

$$\int_{V^i} \left(C\frac{A}{G}-1\right)\omega d\chi^{n-1} \approx A_i^{n-1}\left(\frac{A^{n-1}\mu_i}{n_F^* A_i^{n-1}}-1\right)\omega, \tag{2.21}$$

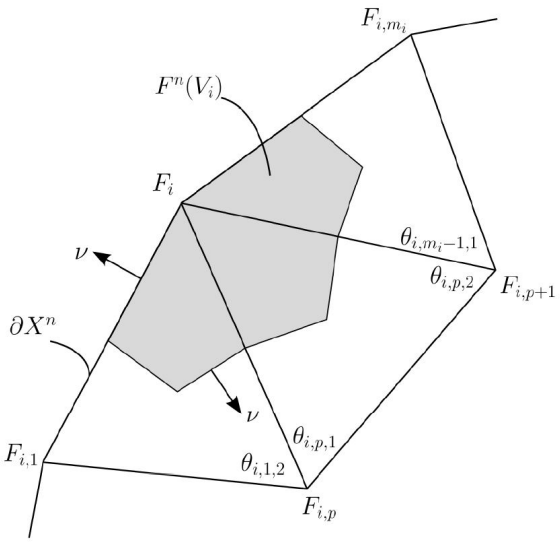
kde μ_i bude veľkosť uhla konečného objemu V_i pri vrchole F_i a n_F^* bude redukovaný počet vrcholov siete. Vypočítame ich nasledujúcim spôsobom. Pre hodnotu μ_i platí

$$\mu_i = \frac{\alpha_i}{2\pi},$$

kde pre vnútorné vrcholy platí $\alpha_i = 2\pi$, z čoho vlastne vyplýva, že potom $\mu_i = 1$ a pre vrcholy na hranici je α_i uhol medzi vektormi $\overrightarrow{F_iF_{i,1}}$ a $\overrightarrow{F_iF_{i,m_i}}$ (pozri Obr. 2.2). Redukovaný počet vrcholov sa bude počítať ako

$$n_F^* = \sum_{i=1}^{n_F}\mu_i.$$

Keď sme si už uviedli, ako budú vyzerať aproximácie každej časti rovnice (2.10), môžeme ich spojiť do jedného systému pre výpočet ψ_i^{n-1}.



Obr. 2.2: Konečný objem na hranici plochy s okrajom. [1]

V prípade, že naša plocha je plocha s okrajom, budeme osobitne počítať systém pre vnútorné a vonkajšie vrcholy.

Vnútorné vrcholy

$$\hat{a}_i\psi_i + \sum_{p=1}^{m_i} \hat{b}_{i,p}\psi_{i,p} = H_i^2 - \frac{1}{A}\sum_{j,F_j\notin\partial X} H_j^2 A_j + \left(\frac{A}{n_F^* A_i} - 1\right)\omega, \tag{2.22}$$

pre všetky i také, že $i \notin \partial X$, kde

$$\begin{aligned} \hat{a}_i &= -\frac{1}{2A_i}\sum_{p=1}^{m_i}(cotg\theta_{i,p-1,1} + cotg\theta_{i,p,2}) \\ \hat{b}_{i,p} &= \frac{1}{2A_i}(cotg\theta_{i,p-1,1} + cotg\theta_{i,p,2}), \end{aligned}$$

pre $p = 1, \ldots, m_i$, pričom platí, že $\theta_{i,0,1}^{n-1} = \theta_{i,m_i,1}^{n-1}$.
Vrcholy $\theta_{i,p,1}^{n-1}$ a $\theta_{i,p,2}^{n-1}$ sú označené na obrázku 2.1.

Vrcholy na hranici

$$\hat{a}_i\psi_i + \sum_{p=1}^{m_i} \hat{b}_{i,p}\psi_{i,p} = -\frac{1}{A}\sum_{j,F_j\notin\partial X} H_j^2 A_j + \left(\frac{A\mu_i}{n_F^* A_i} - 1\right)\omega, \tag{2.23}$$

pre všetky i také, že $i \in \partial X$, kde

$$\begin{aligned}
\hat{a}_i &= -\frac{1}{2A_i}\left(cotg\theta_{i,1,2} + \sum_{p=2}^{m_i-1}(cotg\theta_{i,p-1,1} + cotg\theta_{i,p,2}) + cotg\theta_{i,m_i-1,1}\right) \\
\hat{b}_{i,1} &= \frac{1}{2A_i}cotg\theta_{i,1,2} \\
\hat{b}_{i,p} &= \frac{1}{2A_i}(cotg\theta_{i,p-1,1} + cotg\theta_{i,p,2}), \qquad pre \quad p = 2, \ldots, m_i - 1 \\
\hat{b}_{i,m_i} &= \frac{1}{2A_i}cotg\theta_{i,m_i-1,1}.
\end{aligned}$$

Na obrázku 2.2 si môžeme pozrieť rozmiestnenie vrcholov $\theta_{i,p,1}^{n-1}$ a $\theta_{i,p,2}^{n-1}$.

V prípade, že daná plocha je plocha bez okraja, potom bude platiť, že $\mu_i = 1$, $i = 1, \ldots, n_F$ a $n_F^* = n_F$ a systém rovníc bude vytvorený iba drobnou modifikáciou systému rovníc (2.22), konkrétne

$$\hat{a}_i\psi_i + \sum_{p=1}^{m_i}\hat{b}_{i,p}\psi_{i,p} = H_i^2 - \frac{1}{A}\sum_{j=1}^{n_F}H_j^2A_j + \left(\frac{A}{n_FA_i} - 1\right)\omega \tag{2.24}$$

pre $i = 1, \ldots, n_F$.

Aby mal systém v prípade plochy bez okraja jednoznačné riešenie, predpíšeme si hodnotu $\psi^n, n = 1, \ldots, M$, v jednom zvolenom bode, napríklad $\psi_1^n = 0, n = 1, \ldots, M$.

Dôležité je poznamenať, že pri takto zostavenej matici lineárnych rovníc pre ψ_i^{n-1}, $i = 1, \ldots, n_F$, neexistuje žiadna záruka, že matica bude diagonálne dominantná. Z tohto dôvodu je systém pre $\psi_i^{n-1}, i = 1, \ldots, n_F$, oveľa ťažšie riešiteľný ako systém pre F_i^n, $i = 1, \ldots, n_F$. V praxi to znamená, že konvergencia iteračnej metódy môže byť oveľa pomalšia.

2.2.4 Počítanie tangenciálnej rýchlosti s ψ

Pre výpočet tangenciálnej rýchlosti

$$v_{T,i}^{n-1} \approx \frac{1}{A_i^{n-1}}\int_{V_i} v_T^{n-1}d\chi_{F^{n-1}}, \tag{2.25}$$

ktorú môžeme rozpísať podľa [4], použijeme nasledovnú aproximáciu

$$\int_{V_i} v_T^{n-1}d\chi_F \approx \sum_{p=1}^{m_i}(\|\sigma_{i,p,1}\|\psi_{i,p,1}^{n-1}\nu_{i,p,1}^{n-1} - \|\sigma_{i,p,2}\|\psi_{i,p,2}^{n-1}\nu_{i,p,2}^{n-1}) - \psi_i^{n-1}h_i^{n-1}A_i^{n-1}, \tag{2.26}$$

kde $\|\sigma_{i,p,s}^{n-1}\|$ označuje dĺžku strany $\sigma_{i,p,s}^{n-1}$, $\nu_{i,p,s}^{n-1}$ predstavuje normálu k príslušnej strane $\sigma_{i,p,s}^{n-1}$ a $\psi_{i,p,1}^{n-1}$ a $\psi_{i,p,2}^{n-1}$ označujú hodnotu ψ^{n-1} v strede strany $\sigma_{i,p,1}^{n-1}$, respektíve $\sigma_{i,p,2}^{n-1}$.

Graficky je to znázornené na obrázku 2.1 vľavo. Hodnoty $\psi_{i,p,1}^{n-1}$ a $\psi_{i,p,2}^{n-1}$ získame pomocou lineárnej interpolácie nasledovne

$$\begin{aligned} \psi_{i,p,1}^{n-1} &= \frac{5\psi_i^{n-1} + 5\psi_{i,p}^{n-1} + 2\psi_{i,p+1}^{n-1}}{12} \\ \psi_{i,p,2}^{n-1} &= \frac{5\psi_i^{n-1} + 2\psi_{i,p}^{n-1} + 5\psi_{i,p+1}^{n-1}}{12} \end{aligned} \tag{2.27}$$

kde hodnoty ψ_i^{n-1} a $\psi_{i,p}^{n-1}$ sú hodnoty ψ^{n-1} vo vrcholoch F_i^{n-1}, respektíve $F_{i,p}^{n-1}$.

Kapitola 3

Implementácia algoritmu v prostredí softvéru Rhinoceros

Pri tvorbe matematického modelu nepotrebujeme žiadne nástroje technického charakteru, stačí nám, preexponovane povedané, iba papier a ceruzka. V prípade, že si matematický model chceme experimentálne overiť, použijeme nejaký vhodný softvér. Ako uvádza samotný názov tejto bakalárskej práce, my budeme odvodený matematický model implementovať v prostredí softvéru Rhinoceros, konkrétne použijeme jeho nadstavbu Grasshopper.

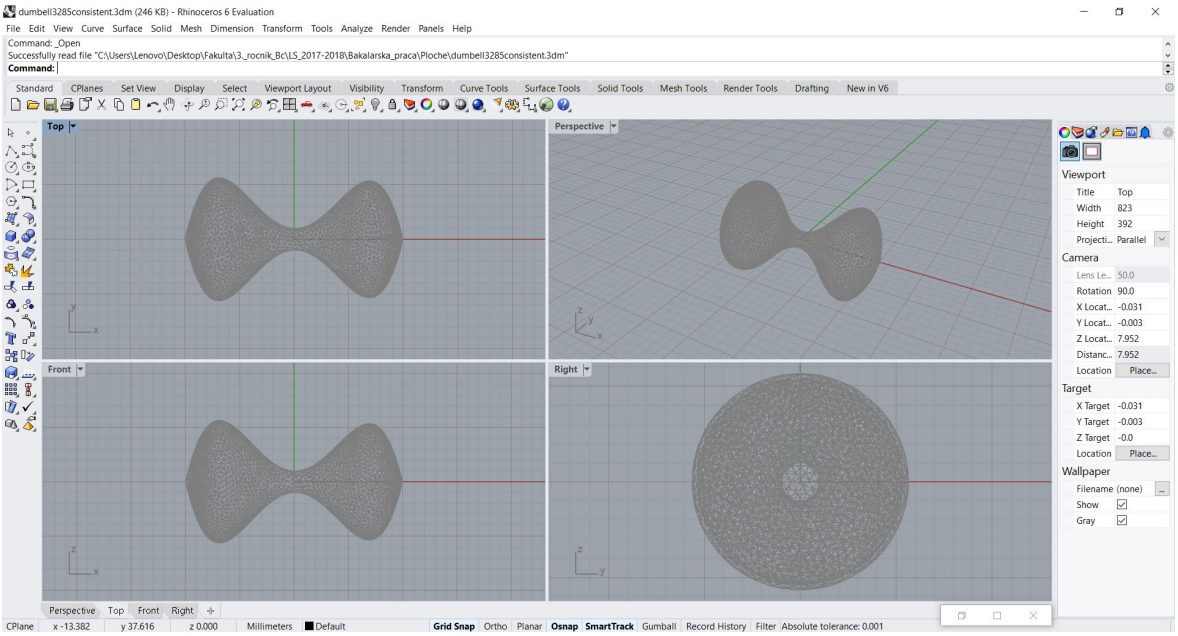
3.1 Prostredie softvéru

Pri implementácii a testovaní matematického modelu sme sa rozhodli použiť softvér Rhinoceros, v komunite používateľov známy pod prezývkou Rhino. Rhinoceros je komerčný 3D grafický softvér využívaný najmä pri precíznej tvorbe rôznych konštrukcií. Samozrejme, niekto by si mohol povedať, že pre potrebu precízneho modelovania nám postačuje AutoCAD alebo softvéry s ním súvisiace, a vlastne, ako sa dozvieme, Rhinoceros sa zrodil vďaka AutoCADu. Ibaže ak nepotrebujeme tvoriť iba pre výrobný proces alebo pre rôzne simulácie, ale chceme spraviť aj pútavé vizualizácie, je nám potrebný nejaký vhodný nástroj. Existuje množstvo iných 3D softvérov, ktoré majú krásne vizuálne efekty, ale na úrok ohúrenia klienta ich precíznosť utrpí. A preto Robert

McNeel & Associates prišli na nápad vytvoriť softvér precízny ako CAD (Computer Aided Design) softvéry a ohurujúci ako iné 3D softvéry pre tvoru 3D modelov. Nápad im vnukla ponuka od spoločnosti Applied Geometry, ktorá v roku 1992 prišla s požiadavkou na spoluprácu pri integrácii geometrických knižníc a NURBs knižníc do AutoCADu. Asi po roku spolupráci s Applied Geometry, McNeel preberá velenie nad celým projektom a pomenováva ho najprv Sculptura 2. Až asi o rok neskôr projekt dostal terajšie meno Rhinoceros.

Čo je to ten NURBs? Neuniformný racionálny B-splajn, po anglicky Non-Uniform Rational B-spline, je typ parametrickej krivky alebo plochy často využívaný práve v počítačovej grafike na reprezentovanie či už 2D alebo 3D geometrie. NURBs krivky a plochy sú obľúbené v praktických aplikáciách, pretože sú flexibilnejšie ako uniformné splajny, vďaka väčšiemu počtu parametrov, ktoré ovplyvňujú výsledný tvar objektu a umožňujú lepšie a reálnejšie nastaviť simulovanú geometriu. Preto je veľmi jednoduché presne modelovať nielen štandardné geometrické objekty, ako sú napríklad čiary, kruhy, elipsy, ale aj geometrické objekty s „voľným“ tvarom, ako sú karosérie áut alebo ľudské telo. Tvorcovia Rhinonocera sa rozhodli pre NURBs kvôli tomu, že takáto reprezentácia geometrie je štandardná, alebo aspoň existujú štandardizované spôsoby, ako ju zmeniť na iný formát. Chceli takto umožniť svojim klientom prenášať si svoj geometrický model vytvorený v Rhinocerose aj do iných programov, napríklad do softvéru na renderovanie, animáciu alebo inžiniersku analýzu, keďže zo začiatku Rhinoceros nedisponoval takýmito nástrojmi. Ďalšou príčinou, prečo si zvolili NURBs, bolo to, že NURBs má presnú a dobre známu definíciu, ktorej sa venuje väčšina matematických a informatických škôl, čo môžem potvrdiť z vlastnej skúsenosti, aj my sme sa jej venovali. To znamená, že dodávatelia softvérov, nie len Rhinocera, ale aj iných softvérov založených na NURBs reprezentácii, môžu zamestnať programátorov, ktorí budú schopní pracovať s touto technikou. [6]

Prvý komerčný projekt, ktorý bol vytvorený pomocou Rhinocera, bola rybárska loď, ktorá sa neskôr aj reálne vyrobila. Potom pomaly, ale isto, sa Rhinoceros vyvíjal, pridávali sa nové funkcionality, expandovalo sa do iných krajín a jazykov, a tak sa aj komunita používateľov rozrastala. Aj dnes Rhinoceros ide s dobou a existuje niekoľko plug-inov, ktoré umožňujú exportovať modely do formátov, ktoré sú podporované mnohými 3D tlačiarňami.



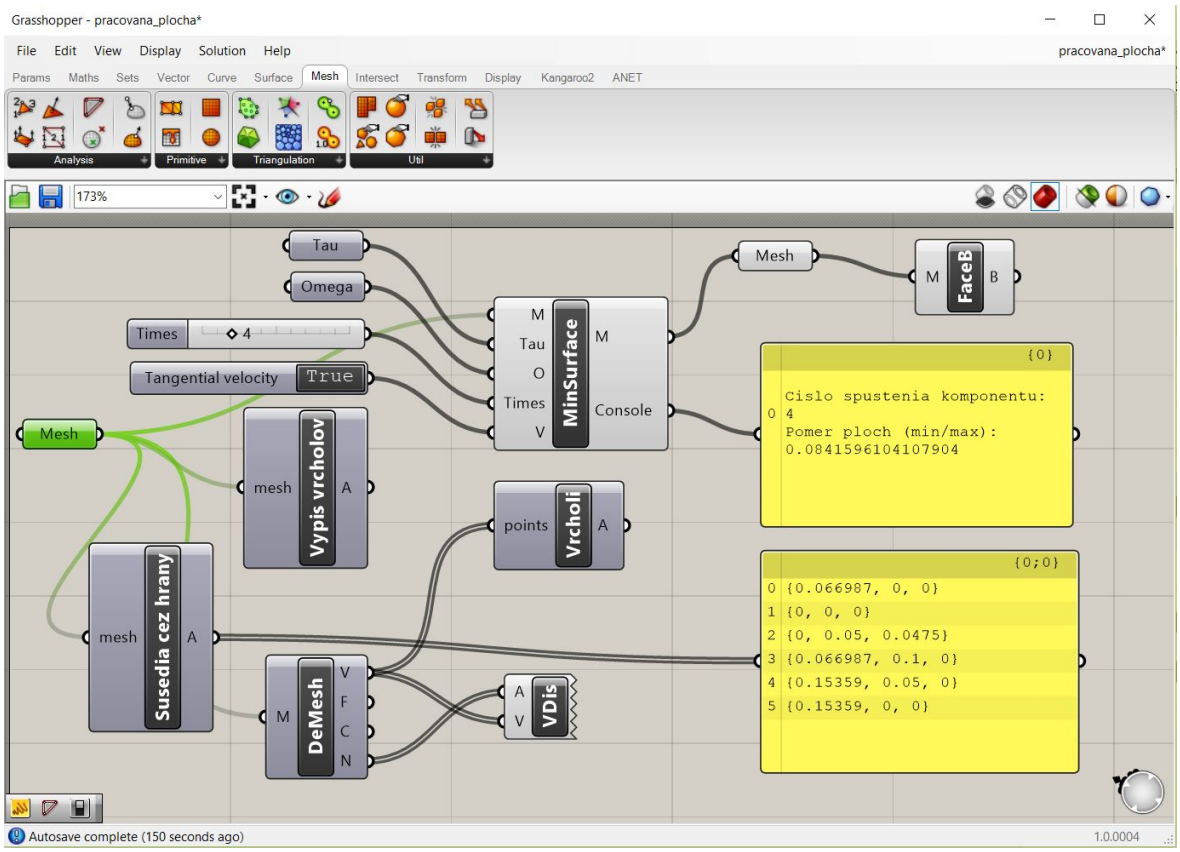
Obr. 3.1: Prostredie softvéru Rhinoceros

Princíp práce s Rhinocerosom sa trošku líši od iných 3D softvérov a viac sa podobá CAD programom. Pracovná plocha, ktorú môžeme vidieť na obrázku 3.1, pozostáva zo štyroch projekcií (v strede), z viacerých panelov s nástrojmi (naľavo), z príkazového riadku (hore) a z panela vlastností (napravo). Všetky objekty na pracovnej ploche je možné si prispôsobiť, či už premiestniť alebo dať do nejakej skupiny. Ovládanie Rhinocera z príkazového riadku je tiež veľmi ľahké, pretože pri vpísaní prvého slova príkazu sa zobrazí zoznam príkazov začínajúcich na zadané slovo. Ako každý seriózny program, aj Rhionoceros obsahuje makrá a možnosť skriptovania. Podporované sú dva skriptovacie jazyky: Rhinoscript, ktorý je založený na Visual Basic Script, a Python, no ani jednu z uvedených možností sme nevyužili pri našej práci. Namiesto toho sme využili SDK, Software Development Kit, čo je vlastne skupina nástrojov pre vývoj softvéru, ktorá je plne funkčná v Rhinocerose. Obsahuje tiež aj množstvo plug-inov, ktoré rozširujú jeho funkcionalitu.

3.2 Využitý plug-in

Jeden z najdôležitejších plug-inov pre nás je parameticky modelujúci/vizualizačný programovací nástroj, zvaný Grasshopper. „The long wait is over: Rhino 6 includes Grasshopper.“ [7], v preklade „Dlhé čakanie skončilo: Rhino 6 zahŕňa Grasshopper“,

boli slová, ktorými víťazoslávne oznámila spoločnosť Robert McNeel & Associates, že sa obľúbený plug-in vyvinutý v roku 2008 Davidom Ruttenom konečne stal, po 10 rokoch, súčasťou Rhinocerosu. Prvé pomenovanie tohto plug-inu bolo Explicit History a určitý čas bol pod týmto menom propagovaný, no neskôr ho premenovali na terajší názov. Grasshopper je zaujímavý pre užívateľov hlavne kvôli tomu, že sa v ňom veľmi ľahko dajú vytvárať komplexné algoritmické štruktúry a mnohé komponenty vytvárajú aj 3D geometriu. Jeho používanie je tiež veľmi intuitívne a názorné. Primárne využitie Grasshoppera je na tvorbu generatívnych parametrických modelov, kde je potrebné vedieť akým spôsobom získané údaje mohli byť vygenerované. Využíva sa pri generatívnom umení, čo je v súčastnosti rozvíjajúca sa oblasť a založená je na náhodných kódoch a rôznych algoritmoch. Pre konštrukčné inžinierstvo, architektúru a výrobu sa používajú nástroje na parametrické modelovanie. Tiež sa využívajú aj iné typy algoritmov ako napríklad numerické, textové, dokonca aj audio-vizuálne. Zaujímavosťou Grasshopperu je aj to, že sa zameriava aj na ekologickú stránku architektúry a existujú v ňom nástroje napríklad na analýzu výkonu osvetlenia alebo energetickej spotreby.



Obr. 3.2: Prostredie plug-inu Grasshopper

Na obrázku 3.2 je znázornené prostredie Grasshoppera a ako si môžeme všimnúť je značne iné ako prostredie Rhinocera. Najväčšiu časť okna zaberá pracovná plocha (v strede), na ktorej sa nenachádza žiadna geometria, ale iba komponenty, ktoré operujú s geometriou, ktorá je načítaná v Rhinocerose. Práve kvôli tejto skutočnosti Grasshopper nemôže existovať samostatne, ale iba ako nadstavba Rhinocera. Jednotlivé komponenty sú medzi sebou rôzne poprepájané a ako celok dávajú požadovaný výsledok. Takáto štruktúra grafického rozhrania by sa dala popísať ako orientovaný graf. Vrcholy grafu sú komponenty s rôznymi funkciami, ktoré v zásade možno rozdeliť na vstupné, algoritmické a výstupné komponenty. Vstupné komponenty sú charakteristické tým, že hrany, ktoré ich prepájajú z inými komponentami, sú orientované smerom von. Hranu smerujúcu von spoznáme podľa toho, že vystupuje z pravej časti vstupného komponentu a následne vstupuje do ľavej časti iného komponentu. Väčšinou vstupné komponenty obsahujú konštantné hodnoty. Výstupné komponenty zas naopak sú charakteristické tým, že hrany, ktoré ich spájajú s inými vrcholmi sú orientované smerom dnu. Teda spoznáme ich podľa toho, že vystupujú z pravej časti iného komponentu a následne vstupujú do ľavej časti výstupného komponentu. Komponenty, ktoré sa nachádzajú v strede takéhoto grafu, sú algoritmické. Údaje, ktoré prídu na vstup sa spracujú a následne sa ako výsledok pošlú na výstup. Náš komponent MinSurface patrí medzi algoritmické komponenty a jeho vzhľad a fungovanie si popíšeme trošku neskôr. Všetky komponenty sú roztriedené podľa funkcionality a nachádzajú sa v palete nástrojov (hore) a ako si môžeme všimnúť, je tam jedna neštandardná záložka, ktorú sme vytvorili my.

Aj keď práca v Grasshoppere nevyžaduje žiadne znalosti programovania, Grasshopper poskytuje možnosť užívateľom, čo programovať vedia, aby svoje znalosti využili. V ponuke sú aj komponenty, ktorých obsah a funkcionalitu užívateľ sám určí písaním objektovo-orientovaného kódu. Dokonca niektoré náležitosti, ako je definícia triedy, či hlavné hlavičkové súbory, Grasshopper sám vygeneruje. Užívateľ si má možnosť vybrať z dvoch jazykov na skriptovanie: Visual Basic Script alebo C#. Kvôli takejto možnosti je Grasshopper v neustálom vývoji nielen zo strany jeho tvorcov, ale aj zo strany svojich užívateľov. [8]

3.3 Vývoj komponentu

Ako sme už spomínali, my sme nevyužívali možnosť skriptovania v Rhinocerose, pretože naším cieľom bolo vytvoriť funkčný komponent práve pre Grasshopper. No tiež sme naplno nevyužili ani skriptovanie v Grasshopperi, pretože sme pracovali na vývoji vo Visual Studiu 2017, ktoré je odporúčané spoločnosťou Robert McNeel & Associates [9]. Existujú dve možnosti programovania vlastného komponentu, ku ktorým je poskytovaná aj dokumentácia - programovanie vo Visual Basic .NET alebo v C# .NET. Mnohí vývojári kombinujú obe možnosti, pretože presne takým spôsobom je vytvorený aj samotný Grasshopper, no my sme sa rozhodli pre čisté C#.

Pri vývoji nášho komponentu sme používali aj vstavané komponenty Grasshopperu, ako napríklad:

- Mesh – komponent, ktorý reprezentuje polygonálnu plochu. Obsahuje zoznam a súradnice bodov, informácie o plôškach a iných detailoch o objekte,
- Number – predstavuje celé alebo desatinné konštantné číslo, ktoré sa používa ako vstupný parameter pre iné komponenty,
- Slider – môže byť celočíselný alebo desatinný posuvník, ktorý sa používa ako vstupný parameter pre iné komponenty,
- Boolean toggle – je prepínač s hodnotami TRUE (áno) alebo FALSE (nie) a tiež sa používa ako vstupný parameter pre iné komponenty,
- Face Boundaries – konvertuje Mesh, ktorý dostane ako vstupný parameter, na krivku, ktorá ohraničuje sieť Meshu a ako výstupný parameter vráti práve túto krivku,
- Panel – predstavuje pomyselný terminál, ktorý zobrazuje dáta, ktoré dostáva na vstup.

Ale predsa boli aj situácie, kedy nám existujúce komponenty nestačili, a tak sme si vytvorili vlastný. Keďže sme potrebovali vytvoriť jednoduchý komponent a potrebovali sme ho vytvoriť rýchlo, nebolo potrebné vytvárať nový projekt, stačilo využiť spomínanú možnosť skriptovania. Krátka ukážka skriptu na hľadanie a označovanie susedov zadaného vrchola je na obrázku 3.3.

Script Editor

Script component: Susedia cez hrany

```
68  private void RunScript(Mesh mesh, ref object A)
69  {
70    List<Point3d> tmp_points = new List<Point3d>();
71    int topology_vertex;
72    int[] connected_edges;
73    int[] neighboorhood;
74
75    // prejdeme secke vrcholi
76    for (int i = 0; i < mesh.Vertices.Count; i++) {
77      topology_vertex = mesh.TopologyVertices.TopologyVertexIndex(i);
78
79      // vyznacime vrchol
80      tmp_points.Add(new Point3d (mesh.TopologyVertices[topology_vertex]));
81
82      // zosortujeme hrany pri vrchole, zisitime indexi hran pri vrchole
83      mesh.TopologyVertices.SortEdges(topology_vertex);
84      connected_edges = mesh.TopologyVertices.ConnectedEdges(topology_vertex);
85
86      neighboorhood = new int[connected_edges.Length];
87      for (int j = 0; j < connected_edges.Length; j++) {
88        // zistime vrcholi na hrane
89        int first_vertex_of_edge_topology = mesh.TopologyEdges.GetTopologyVertices(connected_edges[j]).I;
90        int second_vertex_of_edge_topology = mesh.TopologyEdges.GetTopologyVertices(connected_edges[j]).J;
91        int first_vertex_of_edge = mesh.TopologyVertices.MeshVertexIndices(first_vertex_of_edge_topology)[0];
92        int second_vertex_of_edge = mesh.TopologyVertices.MeshVertexIndices(second_vertex_of_edge_topology)[0];
93
94        // vykreslime si tie body
95        if (first_vertex_of_edge == i) {
96          neighboorhood[j] = second_vertex_of_edge;
97          tmp_points.Add(new Point3d (mesh.TopologyVertices[second_vertex_of_edge]));
98        }
99        else {
100         neighboorhood[j] = first_vertex_of_edge;
101         tmp_points.Add(new Point3d (mesh.TopologyVertices[first_vertex_of_edge]));
102       }
103     }
104
105     // priradime susedov do pomocnih susedov
106     neighboorhood_tmp = neighboorhood;
107   }
108
109   // priradime mesh do pomocniho meshu
110   mesh_tmp = mesh;
111
112   A = tmp_points;
113 }
114
```

Obr. 3.3: Časť C# skriptu napísaného v Grasshopperi

Geometria, s ktorou sme pracovali, nám bola vopred poskytnutá a bola vo formáte .vtk. Ibaže Rhinoceros nepodporuje formát súboru .vtk, a preto sme museli najprv vymyslieť, ako sa s týmto vysporiadať. Na oficiálnej webovej stránke Rhinocera [7] sú uvedené rôzne formáty, s ktorými softvér dokáže pracovať, či už ich iba načítať, alebo aj robiť v nich zmeny. Jeden z formátov, ktoré vie Rhinoceros čítať a aj modifikovať, je .ply. Rozhodli sme sa naprogramovať si vlastný vtkToPly konvertor, aplikáciu, ktorá si od nás vypýta súbor vo formáte .vtk a zmení ho na súbor vo formáte .ply. Tento konvertor sme naprogramovali v programovacom jazyku C++ a pritom sme naplno využili knižnice vtk. Následne sme v Rhinocerose iba načítali geometriu a pokračovali sme v práci už v Grasshoppere.

Dôležitým komponentom Grasshopperu, ktorý je treba nielen spomenúť, ale aj o ňom viacej povedať, je Mesh. Tento komponent je štandardným dátovým typom, ktorý sa nachádza v Grasshopper.Kernel.Types namespace. Je to komponent, ktorý nám značne uľahčil prácu preto, že sme sa nemuseli starať o osobitné načítávanie vrcholov, hrán a plôšok polygonálnej siete, ale stačilo využiť Mesh a on to všetko za nás

načítal. Mesh predstavuje 3D polygonálnu sieť zloženú buď zo štvoruholníkov alebo z trojuholníkov. Dokonca nebolo potrebné ani definovať vzťahy medzi bodmi a hranami, všetko toto vybavil komponent Mesh a ešte naviac spočítal za nás aj normály vo vrcholoch siete, normály plôšok a mnohé iné veci.

3.3.1 Implementácia matematického modelu

Implementáciu vytvoreného matematického modelu sme si rozdelili do troch fáz. V prvej fáze sme explicitnou metódou počítali vývoj plochy v čase. V druhej fáze implementácie sme matematický model diskretizovali semi-implicitnou schémou. V prvých dvoch fázach sme ešte neuvažovali tangenciálnu rýchlosť, pretože model bez nej je jednoduchší a výpočtovo menej náročný a v niektorých prípadoch postačuje. Ale už v poslednej, tretej fáze sme zahrnuli aj tangenciálnu rýchlosť.

Prvá fáza

V tejto časti implementácie sme si vypočítali hodnoty vrcholov F^n z vrcholov F^{n-1}, ktoré sme už poznali z predchádzajúceho časového kroku. Čiže aplikovali sme vzťah (2.9), bez uvažovania tangenciálnej rýchlosti v_T, kde $\triangle_{F^{n-1}} F^n$ je podľa vzťahu (2.4) a (2.5) vektor strednej krivosti. Strednú krivosť v každom vrchole siete sme vypočítali podľa vzorca (2.16).

Druhá fáza

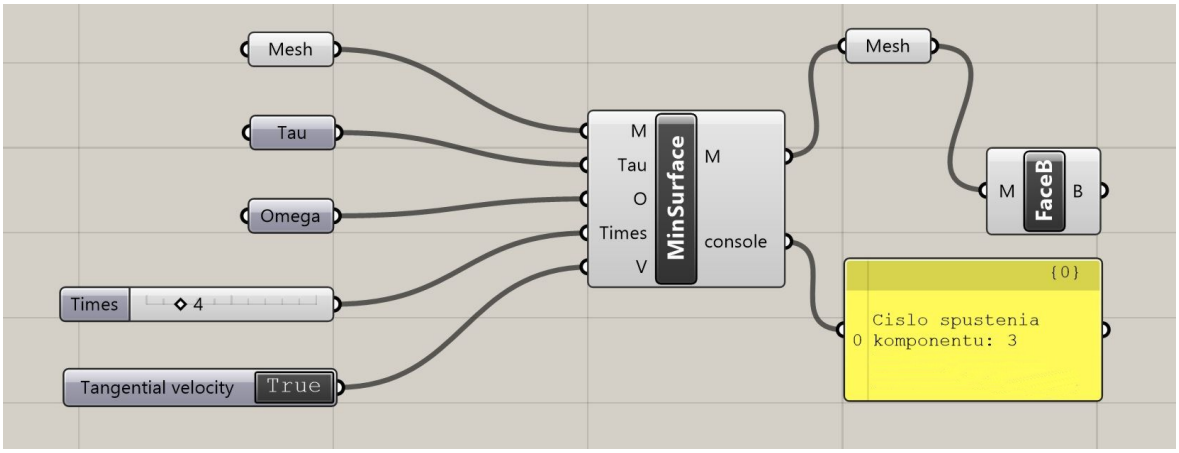
V druhej fáze sme pre výpočet nových vrcholov F^n riešili systém lineárnych rovníc (2.14), tiež bez uvažovania tangenciálnej rýchlosti v_T. Systém rovníc sme riešili aplikáciou iteračnej metódy BiCGStab (BiConjugate Gradient Stabilized method) pre riešenie nesymetrických lineárnych sústav rovníc.

Tretia fáza

Na konci tvorby komponentu sme sa popasovali s implementáciou tangenciálnej rýchlosti. Rovnako ako aj v druhej fáze, aj tu sme riešili systém lineárnych rovníc (2.14), ale už teraz aj s členom v_T, ktorý sme si vypočítali z rovnice (2.26). Tu sme museli ešte raz využiť iteračnú metódu BiCGStab na výpočet ψ, pre sústavu rovníc (2.22) a (2.23) v prípade plochy s okrajom alebo (2.24) prípade uzavretej plochy.

3.3.2 Vzhľad komponentu MinSurface

Grasshopper má vopred definovaný záväzný vzhľad svojich komponentov a preto aj komponent, ktorý sme vytvorili my, má rovnaký vzhľad. Ale aj napriek tomu sú určité parametre, ktoré je možné nastaviť. Štandardom je, že vstupné údaje sa nachádzajú na ľavej strane komponentu. My sme si mohli nastaviť, koľko vstupných parametrov bude a akého budú typu, tiež sme nastavili ich názvy a krátky popis pre užívateľa, čo daný vstup vyžaduje. Výstupy sa štandardne nachádzajú na pravej strane komponentu a my sme mohli tiež nastaviť, koľko výstupov chceme, aké budú mať názvy, krátky popis, čo predstavujú a akého typu budú. Názov celého komponentu akoby rozdeľoval vstupné a výstupné údaje, nachádza sa v strede komponentu. Okrem toho, že si môžeme tento názov zvoliť sami, tiež môžeme uviesť stručný popis nového komponentu.



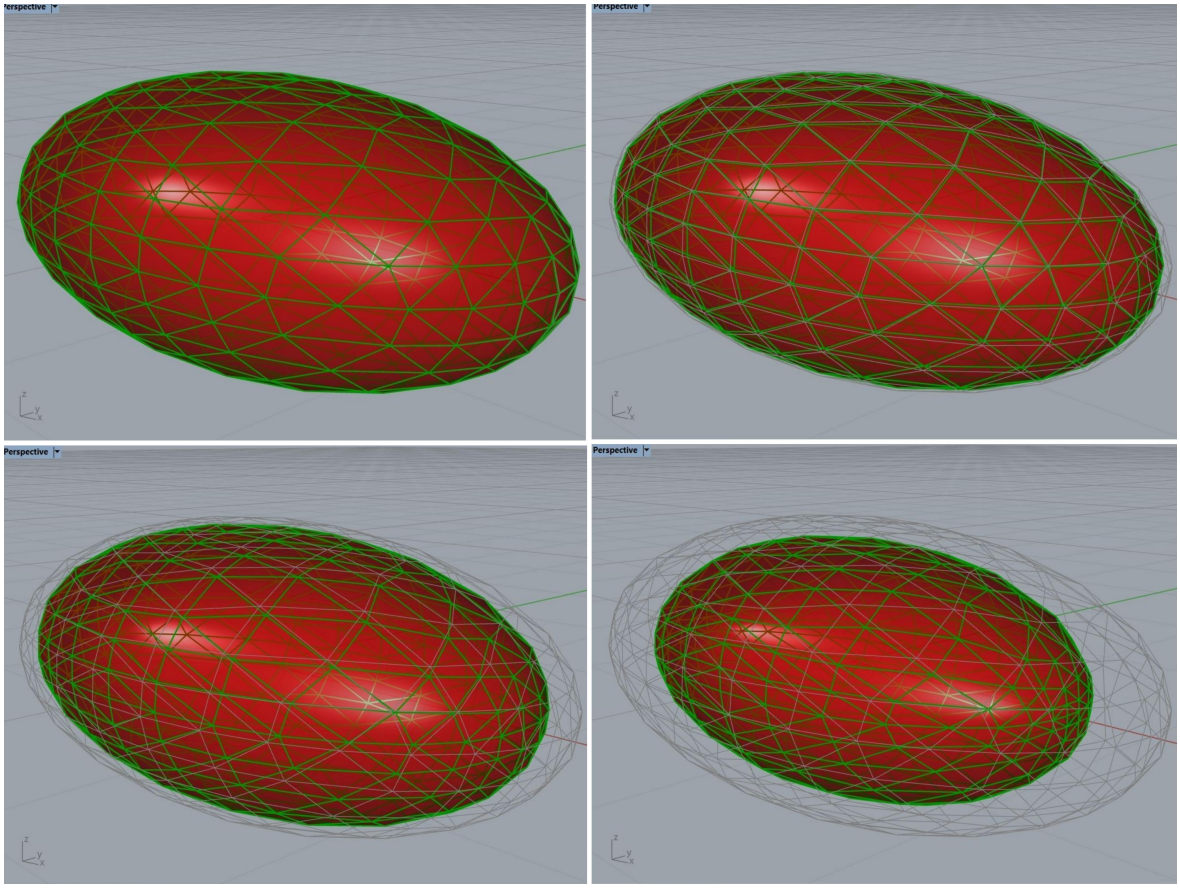
Obr. 3.4: Vytvorený komponent MinSurface s nastavenými parametrami

Na obrázku 3.4 je znázornený náš komponent. Môžeme si všimnúť, že má päť vstupných parametrov a dva výstupné. Ako sme spomenuli, používame aj vstavané komponenty Grasshoppera, ako je napríklad Number komponent, ktorým dávame na vstup konštanty τ, ω. Využívame aj Slider na vnášanie počtu opakovaní implementovaného algoritmu. Ak je Boolean toggle nastavený na TRUE, ako v prípade na obrázku, tak sa bude počítať s tangenciálnou rýchlosťou. Výstupom nášho komponentu je zmenený Mesh, ktorý následne vchádza do ďalšieho vstavaného komponentu Face Boundaries, ktorý zvýrazní zmenenú polygonálnu sieť. Druhým výstupným parametrom je iba reťazec textu zobrazujúci sa v Paneli.

3.4 Ukážky fungovania komponentu

Naše rozprávanie by sme mohli uzavrieť tým, že si ukážeme, ako náš komponent funguje. Budeme uvažovať rôzne geometrie a rôzne nastavenia, a tak bude možné porovnať, presne ako sme na začiatku avizovali, ako sa prejaví napríklad tangenciálna rýchlosť alebo rôzne nastavenia parametra rýchlosti redistribúcie ω.

Rovnakým štýlom, ako sme odvodzovali a implementovali model, si spravíme aj vizualizácie. Najprv budeme pracovať bez tangenciálnej rýchlosti. Ukážeme si len výsledky z druhej fázy implementácie, kedy sme riešili systém rovníc, kvôli tomu, že vizuálny rozdiel je minimálny. Nastavíme si časový krok $\tau = 0.001$ a postupne budeme sledovať vývoj plochy v rôznych časoch.

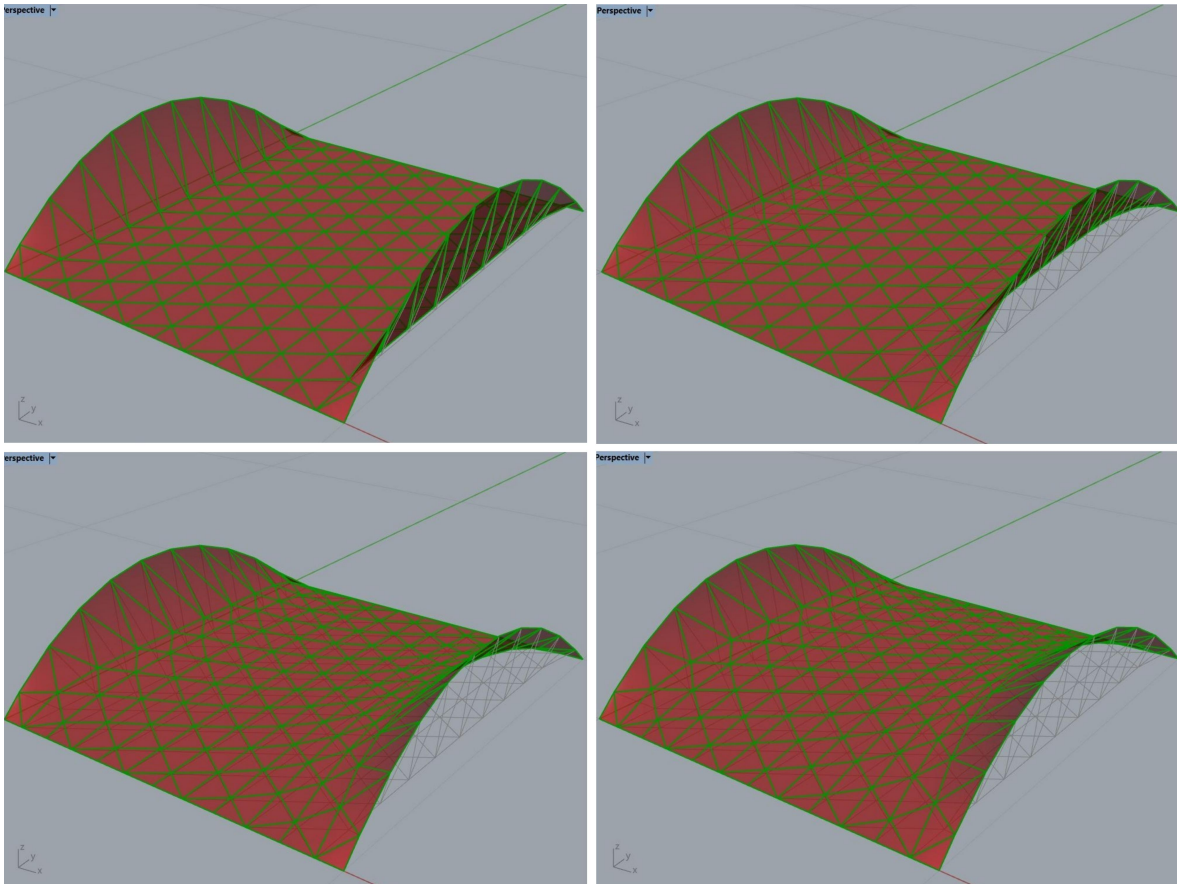


Obr. 3.5: Vývoj elipsoidu bez tangenciálnej zložky v časových krokoch $n = 1, 10, 40, 100$.

Na obrázku 3.5 si môžeme všimnúť, že sa uzavretá plocha, konkrétne elipsoid, zmenšuje. Ako sme hovorili, normálová zložka rýchlosti zabezpečuje posun bodov v okolitom priestore, čiže v tomto prípade dovnútra útvaru. Keď sa pozrieme na sieť nového

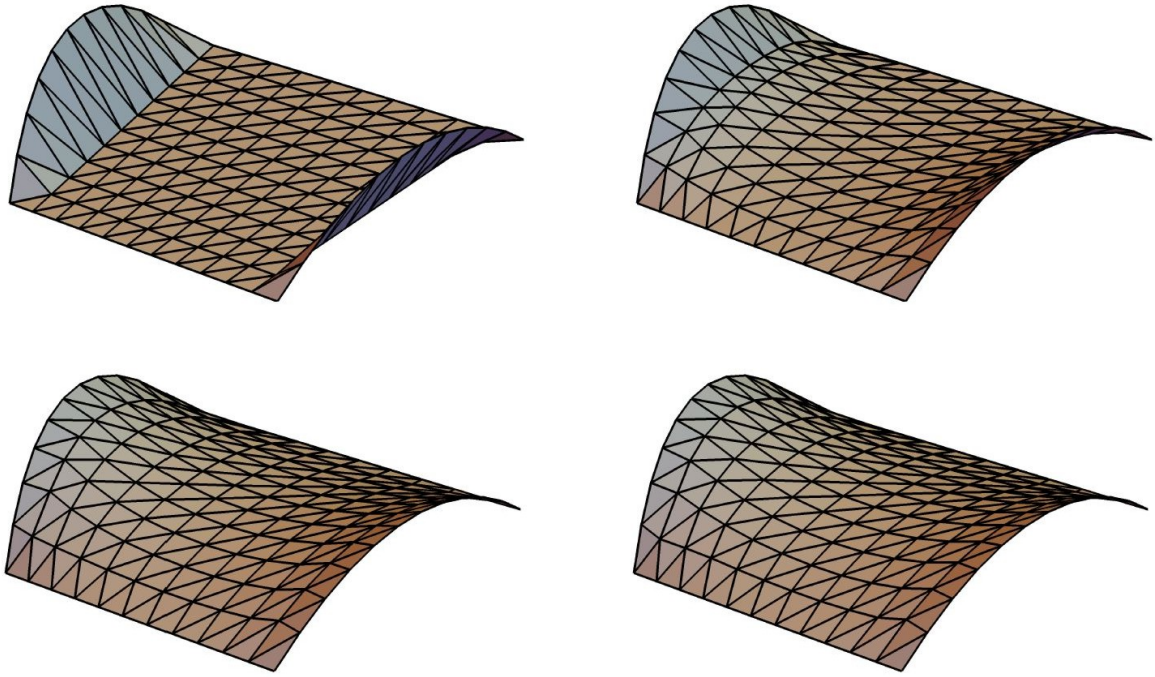
elipsoidu (na obrázku zelená sieť), vidíme, že sa trojuholníky iba zmenšujú a nekoná sa žiadne znovurozdelenie siete.

Tiež si urobíme názornú ukážku aj pre plochy s okrajom. Rovnako ako aj pri uzavretej ploche, aj tu si zvolíme časový krok $\tau = 0.001$. Na tejto ploche (Obr. 3.6) sa oveľa lepšie dá vidieť, že sa body nepohybujú po povrchu, ale iba v okolitom priestore. Všimnime si zvýraznenú sieť (na obrázku zelená sieť), ktorá je nerovnomerne rozdelená hlavne v oblasti, kde mala pôvodná plocha ostré rohy. Aby sme dosiahli rovnomerné rozdelenie siete bolo by vhodné do výpočtu zahrnúť aj tangenciálnu zložku rýchlosti.



Obr. 3.6: Vývoj plochy s okrajom bez tangenciálnej zložky v časových krokoch $n = 1, 10, 40, 100$.

Pri realizovaní tretej fázy implementácie matematického modelu, kde je už zahrnutá aj tangenciálna rýchlosť, sme narazili na komplikácie a nepodarilo sa nám ju úspešne dokončiť. Plná realizácia tejto fázy preto zostáva predmetom ďalšej práce.



Obr. 3.7: Vývoj plochy s okrajom s tangenciálnou redistribúciou v časových krokoch $n = 0, 5, 30, 100$. [1]

Aby sme ukázali ako by sa mala prejaviť tangenciálna zložka rýchlosti, uvedieme si ukážku z článku [1]. Zvolili sme si príklad plochy s okrajom, keďže sme aj my s podobnou plochou pracovali. Časový krok bol nastavený na $\tau = 0.01$ a do výpočtu bola zahrnutá aj tangenciálna rýchlosť s nastaveným parametrom rýchlosti redistibúcie $\omega = 50$. Vidíme, že sa vývoj realizuje podobne ako na našom príklade (Obr. 3.6), čiže plocha sa vyvíja v normálovom smere. Upriamiť pozornosť by sme mali na rozdelenie polygonálnej siete. Keďže v tomto prípade sa tangenciálna rýchlosť brala do úvahy, môžeme si všimnúť, že sa body pohybujú aj po samotnej sieti. Čiže dosiahneme znovurozdelenie siete a tým zabezpečíme, aby výsledná plocha okrem toho, že bude minimálna, bola aj pravidelne rozdelená.

Kapitola 4

Záver

Úspešne sa nám podarilo vytvoriť nový komponent MinSurface pre softvér Rhinoceros, respektíve pre jeho plug-in Grasshopper. Naprogramovaný komponent podľa popísaného matematického modelu dokáže zo zadanej počiatočnej plochy vytvoriť minimálnu plochu. Takáto plocha sa dosahuje pomocou vývoja podľa strednej krivosti. Komponent je navrhnutý tak, že sa vývoj môže konať iba v smere normál alebo aj v tangenciálnom smere, čo by zabezpečovalo vhodnejšie rozdelenie bodov polygonálnej siete. Nepodarilo sa nám dosiahnuť pravidelnú sieť po dosiahnutí minimálnej plochy, ale samozrejme budeme sa snažiť dokončiť začatú prácu a dosiahnuť to, aby aj pri novej ploche bola sieť rovnomerne rozdelená. Plánom do budúcna je doladiť komponent tak, aby pracoval presne, ako matematický model popisuje a následne ho poskytnúť užívateľom, ktorým by mohol pomôcť pri vytváraní nových vecí.

Literatúra

[1] Tomek L., Remešíková M., Mikula K., *Computing minimal sufraces by mena curvature flow with area-oriented tangential redistribution*, Acta Math. Univ. Comenianae, Vol. (submitted to AMUC).

[2] Mikula K., Remešíková M., Sarkoci P. a Ševčovič D., *Manifold evolution with tangential redistribution of points*, SIAM J. Scientic Computing, Vol. 36, No.4 (2014).

[3] Meyer M., Desbrun M., Schroeder P. a Barr A. H., *Discrete differential geometry operators for triangulated 2-manifolds*, in Visualization and Mathematics III, H.-C. Hege and K. Polthier, eds., Springer-Verlag, Berlin, (2003).

[4] Dziuk G. a Elliot C. M., *Finite elements on evolving surfaces*, IMA J. Numer. Anal. 27 (2007).

[5] Langdon D., *AD Classics: German Pavilion, Expo '67 / Frei Otto and Rolf Gutbrod*, https://www.archdaily.com/623689/ad-classics-german-pavilion-expo-67-frei-otto-and-rolf-gutbrod?ad_medium=gallery, ArchDaily (2015).

[6] McNeel, *McNeel Wiki*, https://wiki.mcneel.com/homepage.

[7] Robert McNeel & Associates, *Rhinoceros*, https://www.rhino3d.com/.

[8] Devetakovic M., Petrusevski Lj., Kijanovic J., Arsic P., Nikolic I. a Peric A., *Parametric modeling of urban complexes using graphic algorithm editor*, Univ. of Belgrade, Faculty of Architecture (2011).

[9] Robert McNeel & Associates, *Grasshopper SDK*, http://developer.rhino3d.com/api/grasshopper/html/723c01da-9986-4db2-8f53-6f3a7494df75.htm.

[10] Van der Vorst H. A., *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992).