Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Využití existujícího modelu písčitého terénu pro modifikace jiných typů povrchů ve virtuální realitě

Plzeň, 2009 Luboš Kolářík

Poděkování

Děkuji vedoucí této diplomové práce Doc. Dr. Ing. Ivaně Kolingerové za její vědecký dohled a čas strávený konzultacemi ohledně této práce. Dále děkuji svým rodičům, za jejich trpělivost a podporu během mého celého studia. Děkuji také Ing. Romanu Soukalovi za konzultace ohledně procházek po trojúhelníkové síti. V neposlední řadě děkuji Václavu Purchartovi za spolupráci i nad rámec jeho vlastní diplomové práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni, dne 2. května 2009 Luboš Kolářík

Abstrakt

Use of existing model of sandy terrain for modifications of other surface types in virtual reality

This work expands the project which simulates a sandy terrain using Delaunay triangulation for a triangle mesh generation. By the use of constrained edges it models virtual device imprints in the triangle mesh. We will try to modify an already existing incremental algorithm of planar triangulation for surfaces of some simple 3D objects such as a sphere or a cylinder. Then we will try to explore a possibility of expanding this algorithm to even more complicated triangle meshes.

In the last part of this work we will explore the possibility of connecting of haptic devices with this application. These devices would be used for control of a virtual device within an application.

Obsah

1. Úvod

Tato práce je částí projektu GAČR č 201/09/00977 „Triangularizované modely pro haptiku a virtuální realitu“ a navazuje na řešení, na kterém se podíleli J. Kadlec [Kad07], V.Purchart [Pur07] a J. Sedmihradský [Sed07]. V počátcích práce byl projekt ve stádiu, kdy bylo možné simulovat rytí do písečného povrchu. K tomu byl použit inkrementální algoritmus planární triangulace s možností vkládání vynucených hran. Aplikace byla již postavena tak, aby byla schopna simulovat písečný povrch (práce J. Sedmihradského [Sed07]). Část původního zadání diplomové práce předpokládala rozšíření tohoto modelu i o jiné typy povrchů. V průběhu plánování prací se ale ukázalo, že pro projekt by bylo přínosnější rozšíření aplikace směrem k přesunu algoritmu do třetího rozměru. Proto jsme se, po dohodě s vedoucí diplomové práce, rozhodli k mírnému odchýlení od zadaného tématu.

Tato práce se tedy zabývá možnostmi triangulace na povrchu 3D objektů. A to jak speciálních případů – koule a válec, tak i možností triangulace složitějších těles, u kterých není možné použít triangulaci na povrchu právě koule nebo válce. Pro speciální případy triangulace na kouli a válce byly vytvořeny metody, pomocí nichž je možné na povrchu těchto těles provádět Delaunayovu triangulace bez problémů. Bylo také navrženo a implementováno několik metod pracujících přímo v prostoru bez použití projekce na povrch jakéhokoliv tělesa. Všechny tyto poznatky byly implementovány a přidány do již zmíněného programu. Tato práce se také zabývá porovnáním urychlení vyhledávání trojúhelníků pomocí procházky ve 3D oproti procházce v transformovaných 2D souřadnicích.

Další částí je propojení aplikace s haptickým zařízením a následné použití tohoto zařízení pro ovládání virtuálního nástroje.

Stručný přehled kapitol:

Triangulace – Tato kapitola obsahuje popis důležitých myšlenek použitých jak pro triangulaci na ploše, která byla hotová již před započetím této práce, tak pro triangulaci na kouli, válci a i pro triangulaci obecných objektů.

Původní řešení – V této kapitole je popsána aplikace, která byla vytvořena před započetím této práce, která z ní vychází.

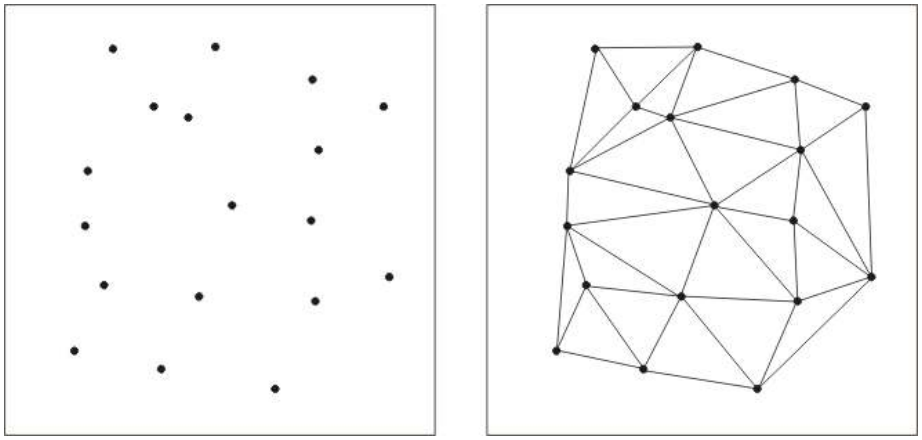
Úpravy – Tato kapitola popisuje úpravy provedené v aplikaci.

Výsledky experimentů – Zde jsou shrnuty výsledky implementace prezentovaných algoritmů.

2. Triangulace

2.1 Triangulace na ploše

Triangulace v rovině je proces, při kterém je z množiny N bodů vytvořena trojúhelníková síť (obr. 2.1.1). Žádné dvě hrany se nesmí protínat. Protože pro každou množinu bodů může existovat velké množství takovýchto triangulací, je nutné při konstrukci triangulace využít ještě další kritéria, která budou popsána dále. Tato práce se z velké části zabývá pouze tzv. Delaunayovou triangulací. Ta používá kritérium, které zaručuje, že v kružnici opsané každému trojúhelníku v triangulaci nebude žádný další bod. Tato triangulace je velmi často používaná pro její dobré výsledky. Pro zadanou množinu bodů je také trojúhelníková síť vytvořená pomocí Delaunayovy triangulace (až na singulární případy, kdy 4 a více bodů leží na kružnici) jednoznačně daná.

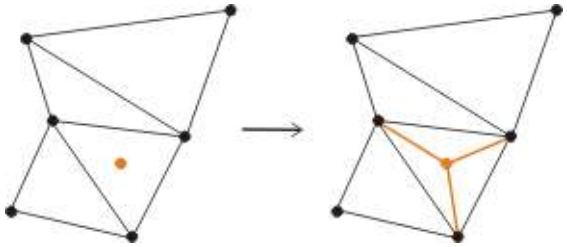


Obr. 2.1.1 – Příklad triangulace

Pro konstrukci triangulace je známo několik algoritmů. Jde například o zametací algoritmy, rozděl a panuj, nebo o algoritmus využívající Voronoiův diagram. Protože tato diplomová práce staví na již rozpracovaném systému využívajícím inkrementální algoritmus, budeme se dále zabývat pouze tímto algoritmem.

Inkrementální algoritmus funguje na principu postupného přidávání bodů do triangulace vytvořené již vloženými body. Přidání bodu je provedeno ve třech krocích. Prvním je nalezení trojúhelníku, který obsahuje nově přidávaný bod. Tento trojúhelník je potom rozdělen na tři menší s použitím nového bodu (obr. 2.1.2.). Posledním krokem je takzvaná legalizace nově vytvořených hran tak, aby tyto hrany odpovídaly triangulačnímu kritériu. Inkrementální algoritmus potřebuje pro každé vložení bodu již nějakou hotovou triangulaci. Proto se, ještě před vložením prvního bodu, uměle vytvoří trojúhelník (nebo

obecně jakákoliv trojúhelníková síť), který je dostatečně velký na to, aby obsahoval všechny vstupní body.



Obr 2.1.2 – Vložení nového bodu do triangulace

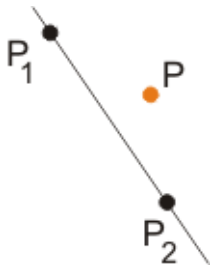
2.1.1 Určení trojúhelníku pro vkládaný bod

Nalezení trojúhelníku obsahujícího vkládaný bod je první krok pro vytvoření nové triangulace. Dá se využít toho, že u trojúhelníku známe jeho hrany. Trojúhelník je také vždy konvexní. Proto se dá rozhodnout o tom, zda je bod uvnitř, porovnáním polohy bodu oproti každé přímce procházející hranou trojúhelníku. Pokud je poloha bodu stejná vzhledem ke všem třem hranám, bod je uvnitř trojúhelníku. Na rozhodnutí o poloze bodu vůči hraně používáme tzv. znaménkové testy.

Znaménkový test vrátí jedno číslo. Pokud je toto číslo větší než nula, bod je „před“ hranou. Pokud je menší než nula, je bod „za“ hranou. Pokud je výsledek znaménkového testu nulový, bod leží přesně na hraně.

Znaménkový test přímky a bodu

Přímka je definována dvěma body, v našem případě dvěma vrcholy trojúhelníku. Abychom mohli tento test použít, je nutné mít vrcholy všech trojúhelníků v jednom směru (po/proti směru hodinových ručiček).



Obr. 2.1.3 – Test polohy bodu vůči přímce dané dvěma body

Pro provedení znaménkového testu (obr. 2.1.3) potřebujeme libovolný bod na přímce a její normálu. Body máme dva, takže stačí jeden vybrat. Normálu N zjistíme následovně.

$$N = \left[P_2{}^x - P_1{}^x, P_1{}^y - P_2{}^y \right]$$

Pro účely znaménkového testu není délka normály důležitá. Výslednou polohu bodu P vůči přímce pak spočítáme takto:

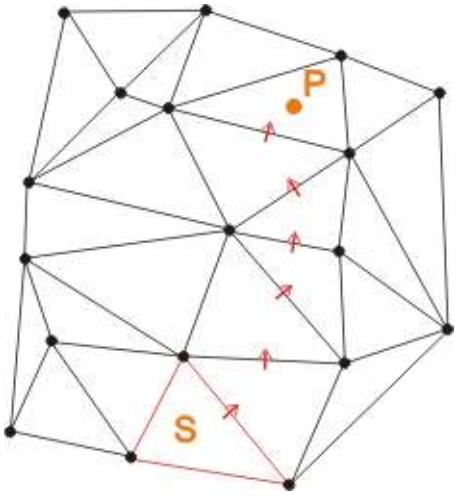
$$res = (P - P_1) \bullet N$$

Přímka (a normála) jsou v tomto případě orientovány tak, že výsledek *res* pro bod P z případu na obr. 2.1.3 by vyšel kladný. Všechny tyto operace se dají složit do jedné operace spočítání determinantu matice o velikosti 2x2.

$$res = \begin{vmatrix} P_2{}^x - P_1{}^x & P^x - P_1{}^x \\ P_2{}^y - P_1{}^y & P^y - P_1{}^x \end{vmatrix}$$

Vyhledávání trojúhelníku je určujícím faktorem rychlosti triangulace. Sekvenční prohledávání o složitosti O(n), které se navíc spouští pro každý vložený bod, je dostačující pouze pro jednoduché sítě. Původně použitý algoritmus ve „zděděném“ programu byl tzv. algoritmus procházky.

Tento algoritmus pracuje tak, že od startovního trojúhelníku (který může být například náhodně vybraný) postupuje vždy podle výsledku znaménkového testu na souseda tak, aby se přibližoval k vkládanému bodu (obr. 2.1.4).



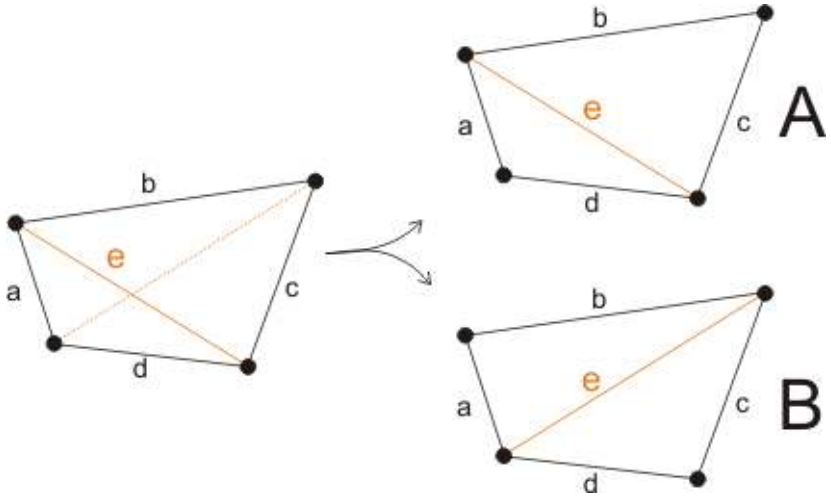
Obr. 2.1.4 – Naznačení principu algoritmu procházky. S- počáteční trojúhelník, P – hledaný bod

Je zde použit stejný znaménkový test jako u sekvenčního průchodu. Pokud se ale bod ukáže vně z trojúhelníku, přejde se na souseda. Soused je určen hranou, pro kterou bod neprošel při znaménkovém testu. Tímto způsobem se značně sníží počet testovaných trojúhelníků a tím i čas nutný pro hledání.

2.1.2 Legalizace hran

Proces legalizace hran má za úkol upravit síť s nově vloženým bodem tak, aby všechny trojúhelníky v síti odpovídaly triangulačním kriteriím. Vložení bodu je operace, která ovlivní jenom jeden trojúhelník (ten, který byl vložením bodu rozdělen). Proto není nutné po vložení každého bodu testovat celou síť, ale stačí projít jenom bezprostřední okolí vloženého bodu.

Při legalizaci hrany se vezmou dva trojúhelníky sousedící s touto hranou. Podle legalizačního kritéria se hrana v tomto čtyřúhelníku buď prohodí (tak, aby tvořila úsečku mezi druhou dvojicí bodů ve čtyřúhelníku), nebo zůstane (obr. 2.1.5).



Obr. 2.1.5 – Dva možné výsledky legalizace hrany

Pokud se hrana nemění, je legalizace u konce. Pokud je ovšem hrana prohozena, je nutné provést legalizaci i pro krajní hrany čtyřúhelníku. Změnou hrany **e** se totiž změnily trojúhelníky sousedící s hranami **a,b,c** a **d**. Proto je možné, že dvojice trojúhelníků sousedící s těmito hranami už nesplňuje triangulační kritérium.

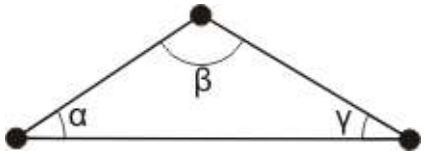
Pro některá kritéria (např. dále zmíněné kritérium zohledňující délky hran trojúhelníků) je nutné testovat konvexitu čtyřúhelníku. Při nekonvexním čtyřúhelníku totiž může prohozením hrany vzniknout situace, kdy se budou hrany v triangulaci křížit.

2.1.3 Kritéria používaná pří triangulaci

Na výběru triangulačního kritéria závisí vzhled výsledné sítě. Kritéria se dají rozdělit do dvou kategorií – úhlová a hranová.

Minimalizace maximálního úhlu/ maximalizace minimálního úhlu

Toto kritérium vybírá trojúhelníky tak, že porovnává úhly mezi hranami trojúhelníků (obr. 2.1.6).

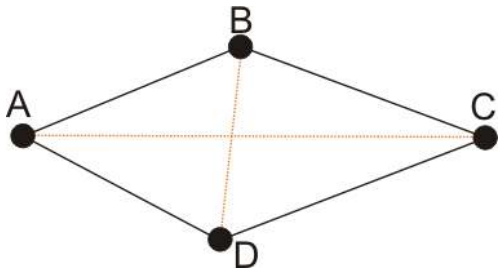


Obr 2.1.6 – Úhly v trojúhelníku

Z těchto tří úhlů vybere maximální/minimální hodnotu. Ta se potom používá pro porovnání s druhou možnostmi triangulace.

Délky hran

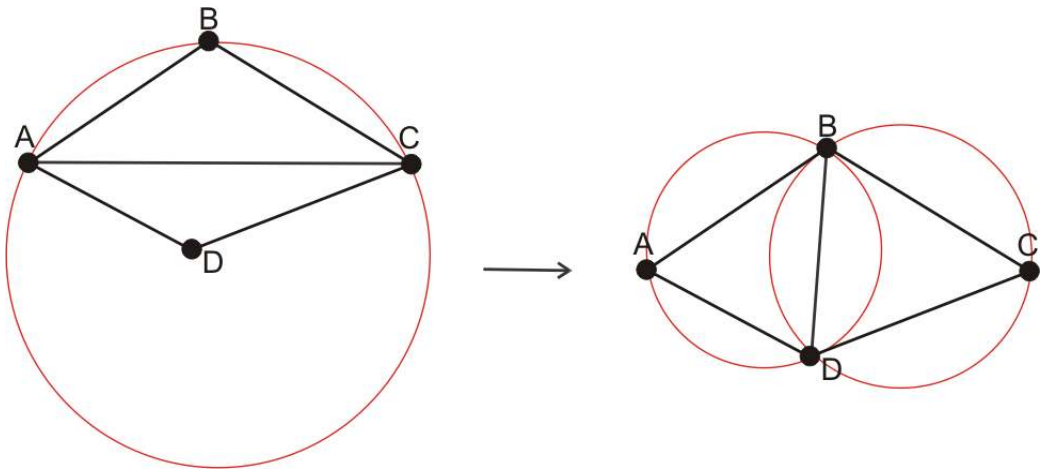
Toto kritérium se vybírá trojúhelníky tak, aby součet jejich hran byl co nejmenší. Podle obr. 2.1.3.2. máme dvě možnosti. Buď trojúhelníky ABC a ACD, nebo ABD a BCD. Při porovnávání délek se ovšem hrany AB, BC, CD a DA odečtou. Proto toto kritérium nakonec porovnává pouze délky legalizovaných hran. Takže vybere kratší hranu z hran AC a BD (obr. 2.1.7).



Obr. 2.1.7 – Dvě možnosti volby hrany

Delaunayovo kritérium

Také známé jako kritérium prázdné opsané kružnice. Toto kritérium zaručuje, že v kružnici opsané třem bodům trojúhelníku se nenachází žádný další bod. Na obrázku 2.1.3.3 je vidět, jak toto kritérium funguje. Kružnice opsaná bodům A, B a C obsahuje ještě další bod D. To je signálem k prohození hrany. Tím vzniknou dva trojúhelníky, jejichž opsaná kružnice neobsahuje žádné další body (obr. 2.1.8). Toto kritérium je používáno asi nejčastěji, protože výsledné trojúhelníky se hodně blíží rovnostranným. Toto kritérium je také použito v řešení upravovaném v rámci této práce.



Obr. 2.1.8 – Znázornění Delaunayova triangulačního kritéria

2.2 Vynucování hran na ploše

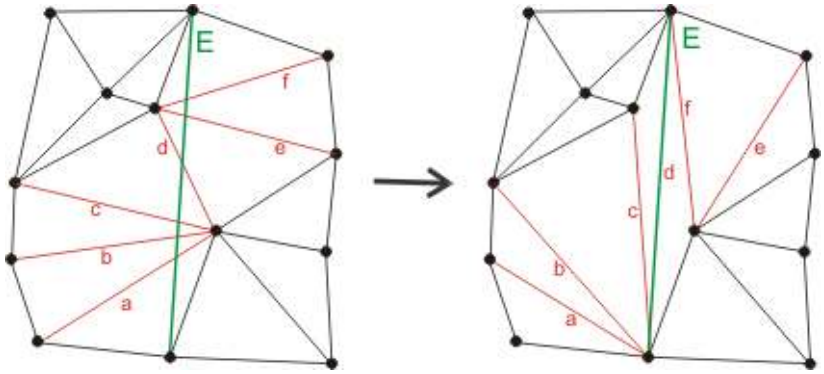
Původním záměrem projektu bylo vytvořit program, který by byl schopen simulovat rytí do povrchu virtuálním nástrojem. Virtuální nástroj je v podstatě model, který je při zamáčknutí do povrchu obtisknut do trojúhelníkové sítě. Při obtisku nástroje tedy v trojúhelníkové síti musí vzniknout nové body odpovídající tvaru nástroje. Při triangulaci ovšem může dojít k tomu, že hrany mezi těmito body se vytvoří v souladu s triangulačním kritériem, ale zcela jinak, než byly určeny nástrojem. Proto je využito takzvané *Constrained Delaunay Triangulation* (CDT). Při obtisku nástroje se vytvoří i hrany, které jsou označené jako *Constrained* – vynucené. Tyto hrany nejsou testovány na triangulační kritérium a vždy se nechávají tak, jak byly vloženy.

2.2.1 Vkládání vynucených hran

Algoritmus pro vynucování hran počítá s již vytvořenou triangulací, kde jsou vloženy koncové body hrany. Proto je nutné jako první věc při vkládání hrany vložit její dva koncové body.

Dalším krokem je nalezení těch hran v triangulaci, které tuto novou hranu protínají. Pro otestování křížení hrany s hranou lze znovu použít znaménkový test využitý pro vyhledávání trojúhelníků (kap. 2.1.1). Použitím znaménkového testu na body hrany A oproti hraně B získáme dva výsledky. Pokud jsou znaménka těchto výsledků stejná, hrana A je zcela mimo hranu B a tyto dvě hrany se nemohou protínat. Pokud jsou znaménka rozdílná, znamená to, že hrana A protíná přímku definovanou body hrany B. V tomto případě se pak provede stejný test s body hrany B oproti hraně A. Pokud budou znaménka výsledků opět rozdílná, hrany se protínají.

Hrany, protínající novou vynucenou hranu, je potom nutné upravit tak, aby novou hranu neprotínaly. To je možné zařídit pomocí prohazování hran. Ze seznamu protínajících hran vybereme první. Pokud dva trojúhelníky s touto hranou sousedící tvoří konvexní čtyřúhelník, tuto hranu prohodíme. Pokud stále kříží vynucenou hranu, vložíme ji na konec seznamu. Tuto operaci provádíme tak dlouho, dokud není seznam hran prázdný (obr. 2.2.1). Výsledek je takový, že hrany protínající vynucenou hranu jsou bud prohozeny tak, aby byly zcela mimo hranu, nebo aby jeden koncový bod protínající hrany byl stejný jako jeden z koncových bodů vynucené hrany.



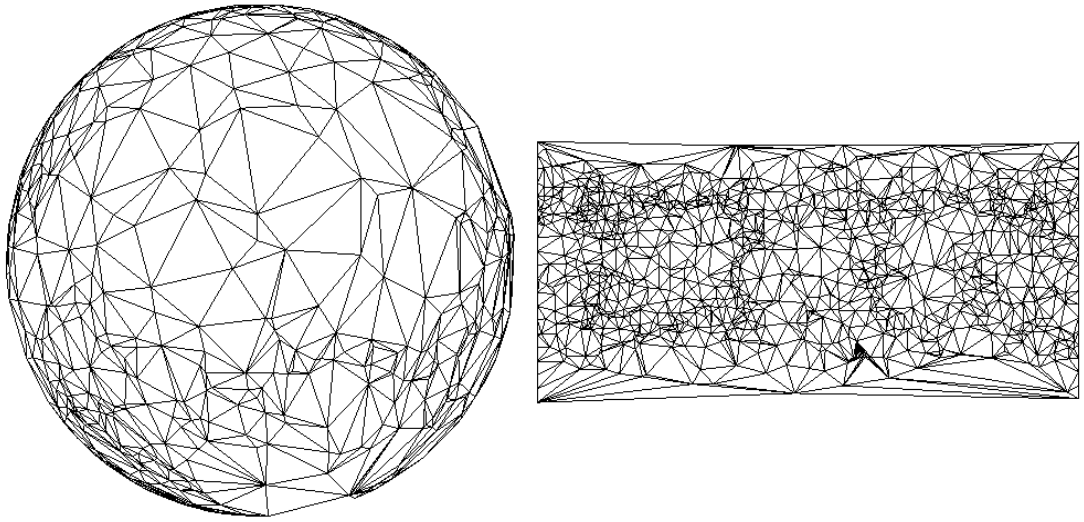
Obr. 2.2.1 - Vynucování hrany. E – nová hrana, a-e – hrany křížící novou hranu

Celý postup prohazování hran z obrázku 2.2.1 lze nalézt v příloze 8.1. Detailnější popis algoritmu je v [Kad07]

2.3 Triangulace na jednotkové kouli

2.3.1 Tranformace do sférických souřadnic

Jde o transformaci do jiné sady souřadnic. Každý bod má potom pozici v 3D prostoru, která je použita pro zobrazení, a původní pozici na 2D ploše, která je použita při triangulaci (obr. 2.3.1).



Obr. 2.3.1 – Triangulovaná koule a její reprezentace použitá pro triangulaci

Pro převod sférických souřadnic do kartézských a naopak jsou používány tyto vzorce.

Převod kartézských souřadnic do sférických

$$r = \sqrt{x^2 + y^2 + z^2}$$
$$\varphi = \text{atan2}(y, x)$$
$$\theta = \arccos\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right)$$

Kde x, y a z vyjadřují polohu bodu v kartézských souřadnicích. Funkce atan2 je potom vyjádřena takto:

$$a\tan 2(y,x) = \begin{cases} \tan^{-1}\left(\left|\frac{y}{x}\right|\right)\cdot \text{sgn}(y) & x > 0 \\ \frac{\pi}{2}\cdot \text{sgn}(y) & x = 0 \\ \left(\pi - \tan^{-1}\left(\left|\frac{y}{x}\right|\right)\right)\cdot \text{sgn}(y) & x < 0 \end{cases}$$

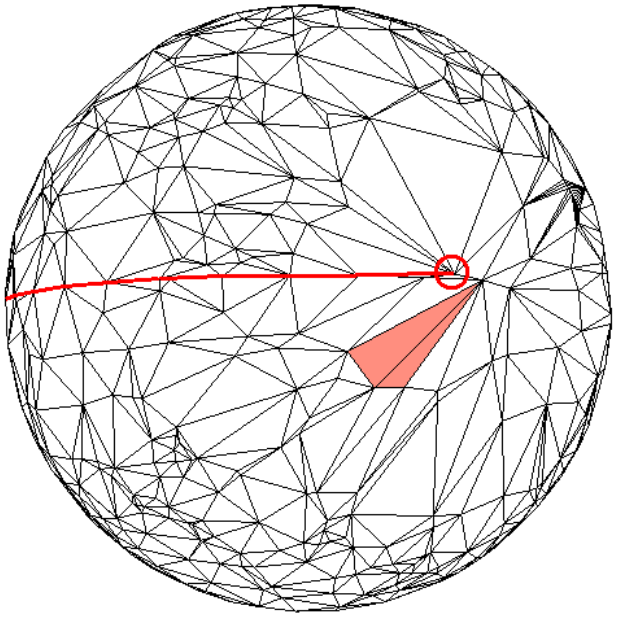
Převod sférických souřadnic do kartézských

$$x = r\cdot\sin\theta\cdot\cos\varphi$$
$$y = r\cdot\sin\theta\cdot\sin\varphi$$
$$z = r\cdot\cos\varphi$$

Výhodou použití této transformace je rozšíření triangulace na kouli, při minimálních úpravách triangulace. Ta funguje pořád na ploše. Toto řešení má ovšem nevýhody, které prakticky znemožňují jeho použití.

Prvním problémem je existence švu. Jak je vidět na obrázku 2.3.1, triangulace probíhá na ploše. Tato plocha je potom jenom „natažena“ na kouli. Proto v místě, kde se na kouli střetne levý a pravý okraj tohoto plátu, vznikne šev nepropojených bodů. Do sítě na obrázku 2.3.2 jsou uměle dodány body, které jsou přesně na tomto švu, takže výsledná koule vypadá zcela uzavřená. Protože triangulace na švu ale nemá informaci o bodech na druhé straně švu, může docházet k vytvoření nedelaunayovských trojúhelníků.

Další problém je, že transformace není rovnoměrná. V okolí rovníku koule je všechno v pořádku, ale jak se začneme dostávat směrem k pólům koule, nerovnoměrnost se zvětšuje. Jeden bod koule – pól se dokonce promítá jako přímka tvořící vrchní okraj plochy použité pro triangulaci. Proto, se zvyšující se vzdáleností od rovníku koule, opět dochází k vytvoření nedelaunayovských trojúhelníků.



Obr. 2.3.2 – Koule se zvýrazněným švem, pólem a dvojicí nedelaunayovských trojúhelníků

2.3.2 Triangulace na kouli bez použití transformace

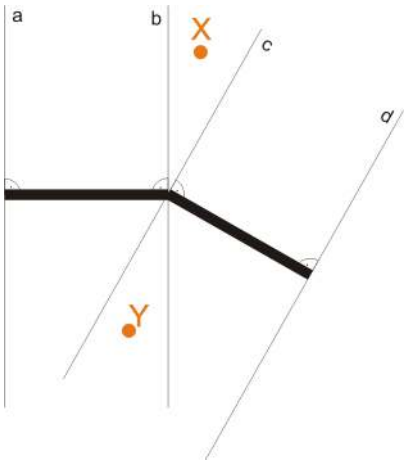
Vzhledem k problémům, vzniklým při použití transformace, bylo nutné přijít s jiným způsobem triangulace. Abychom se vyhnuli problémům se švy a okolo pólů, nebylo možné dále používat sférické souřadnice. To znamenalo začít provádět triangulace v pseudo 3D. Sice se budou využívat tři osy pro určení pozice, ale triangulace bude omezena na povrch koule. Bylo nutné upravit dvě části původně 2D algoritmu. Vyhledávání trojúhelníku pro nový bod a

rozhodnutí, zda je bod uvnitř kružnice opsané trojúhelníkem. Zbytek triangulace nijak nevyužívá pozici bodů a jenom upravuje topologii sítě.

2.3.3 Sekvenční prohledávání trojúhelníkové sítě

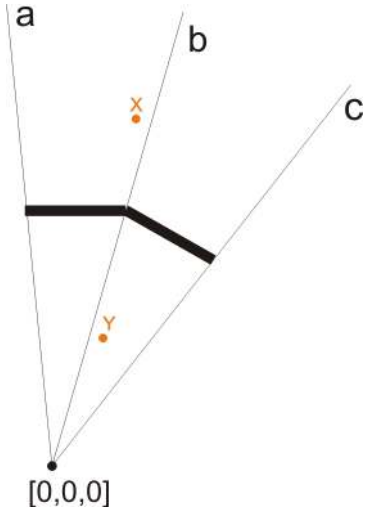
Sekvenční prohledávání je nejjednodušší způsob nalezení trojúhelníku a zároveň možnost otestovat úpravy znaménkového testu pro trojrozměrné vyjádření bodu.

Ve třech rozměrech je nutné ke dvěma bodům hrany ještě přidat další informaci tak, aby společně s touto hranou definovala plochu. Tato plocha nemůže být definována podobně jako ve 2D, kdy je kolmá na rovinu trojúhelníku. Při přidávání nového bodu do triangulace je totiž nutné, aby bylo možné pro každý bod v prostoru jednoznačně přiřadit trojúhelník v triangulaci. Roviny vytvořené tímto způsobem tuto podmínku nesplňují. Jak je vidět na obrázku 2.3.3, tímto způsobem mohou vzniknout body, pro které nebude nalezen žádný trojúhelník (bod X), nebo body, pro které jich bude nalezeno víc (bod Y).



Obr. 2.3.3 – Přiřazování bodů trojúhelníkům (a,b,c,d – plochy kolmé na rovinu trojúhelníku vytvořené z jeho hrany)

Tento problém je vyřešen tak, že plocha je vytvořena pomocí dvou bodů hrany a počátku souřadnic (předpokládá se, že triangulujeme na kouli se středem v bodu [0,0,0]). Tím je celý prostor jednoznačně rozdělen (obr. 2.3.4).



Obr. 2.3.4 – Přiřazování bodů trojúhelníkům – druhý způsob (a,b,c plochy vytvořené z hran trojúhelníku procházející počátkem)

Znaménkový test plochy a bodu

Tento test funguje na podobném principu jako ve 2D. Jen pracujeme s třísložkovými vektory. Plochu máme definovanou třemi body P_1,P_2 a P_3. Bod je označen P. Bod plochy P_1 je vždy nulový.

Získání normály plochy:

$$N = (P_2 - P_1) \times (P_3 - P_1)$$

Toto platí pro obecnou rovinu. Protože bod P_1 je ale nulový, dá se v tomto případě zjednodušit na $N = P_2 \times P_3$.

Získání „vzdálenosti“ od plochy (kdybychom chtěli vypočítat pravou vzdálenost bodu od roviny, bylo by nutné normálu znormalizovat, v našem případě to není nutné, protože nás zajímá jen znaménko) je potom stejné jako ve 2D: $res = (P - P_1) \bullet N$. To je opět možné zjednodušit na $res = P \bullet N$. *Res* ukazuje „vzdálenost“ (velikost této vzdálenosti je závislá na délce normály N) bodu P od plochy definované bodem P_1 a normálou N.

Opět je možné vyřešit tento test jako determinant matice, tentokrát o velikosti 3x3:

$$res = \begin{vmatrix} P_3^{\,x} - P_1^{\,x} & P_2^{\,x} - P_1^{\,x} & P^{x} - P_1^{\,x} \\ P_3^{\,y} - P_1^{\,y} & P_2^{\,y} - P_1^{\,y} & P^{y} - P_1^{\,y} \\ P_3^{\,z} - P_1^{\,z} & P_2^{\,z} - P_1^{\,z} & P^{z} - P_1^{\,z} \end{vmatrix}$$

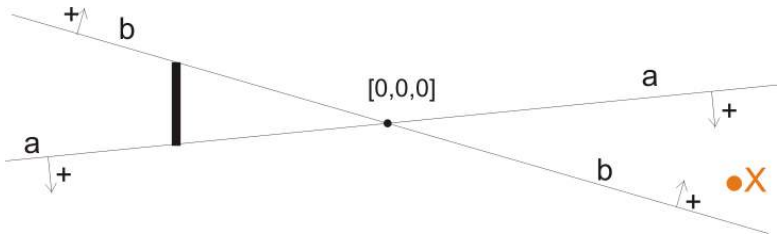
Toto znovu platí pro obecnou plochu (bude využito v dalších částech práce). Pro plochu procházející počátkem používáme jednodušší tvar:

$$\text{res} = \begin{vmatrix} P_3^{\,x} & P_2^{\,x} & P^x \\ P_3^{\,y} & P_2^{\,y} & P^y \\ P_3^{\,z} & P_2^{\,z} & P^z \end{vmatrix}$$

Po této úpravě znaménkového testu je již vyhledávání pomocí sekvenčního průchodu stejné jako v kapitole 2.1.1.

2.3.4 Prohledávání sítě pomocí procházky

Procházka na kouli funguje opět velmi podobně jako její 2D verze. Pokud pomineme úpravu znaménkového testu, je nutné ošetřit pouze jednu situaci, kdy může selhat. Tato situace nastane, pokud je vybrán počáteční trojúhelník, k němuž je bod přesně na druhé straně koule. Po zjištění znamének u všech ploch to potom vypadá, jakoby tento bod byl při testu polohy vůči všem hranám mimo trojúhelník. Pak je problém určit, kam by měla procházka dál postupovat (obr. 2.3.5). Pokud se vždy přechází na první hranu, která neprojde znaménkovým testem, je zde dokonce možnost chycení procházky v nekonečném cyklu. Naštěstí je řešení tohoto problému celkem jednoduché. Pokud u znaménkového testu všech tří hran trojúhelníka vyjde kladné číslo (to by znamenalo, že bod je mimo trojúhelník pro každou hranu), vybereme (náhodně) jiný trojúhelník ze sítě. Velmi pravděpodobně se tím dostaneme blíž k hledanému trojúhelníku – v rovnoměrné síti je cesta z tohoto problémového trojúhelníku k hledanému nejdelší možná.



Obr. 2.3.5 – Špatně vybraný trojúhelník při procházce na kouli

2.3.5 Test Delaunayova kritéria na povrchu koule

Zatímco změny u vyhledávání trojúhelníků byly oproti triangulaci na ploše minimální, test Delaunayova kritéria bude nutné zcela změnit.

Jednoduchou možností by mohlo být převedení všech použitých bodů do sférických souřadnic a použít stejný test jako na ploše. To ale kvůli výše zmíněné deformaci nedává správné výsledky v okolí pólů a švu koule.

Proto bylo nutné použít jiný postup. Prvním problémem bylo zjištění středu kružnice ležící na povrchu koule. Tento střed umíme bez problémů vypočítat u trojúhelníku v prostoru s využitím barycentrických souřadnic. Ten ale určitě nebude na povrchu naší koule. Proto jsme využili jednoduchý trik. Opět jsme k bodům trojúhelníku přidali ještě počátek. Ze čtyř bodů v prostoru je možné vypočítat opsanou kouli. Tato koule má tři body na jednotkové kouli a jeden v jejím středu. Průnik těchto dvou koulí je právě naše hledaná kružnice. Získáním středu koule získáme přímku, která prochází počátkem a středem této kružnice. Střed koule C získáme z bodů A,B,C a D**.**

$$C = A + \frac{|D-A|^2 \cdot [(B-A)\times(C-A)] + |C-A|^2 \cdot [(D-A)\times(B-A)] + |B-A|^2 [(C-A)\times(D-A)]}{2\cdot(B-A)\cdot[(C-A)\times(D-A)]}$$

zdroj:[4]

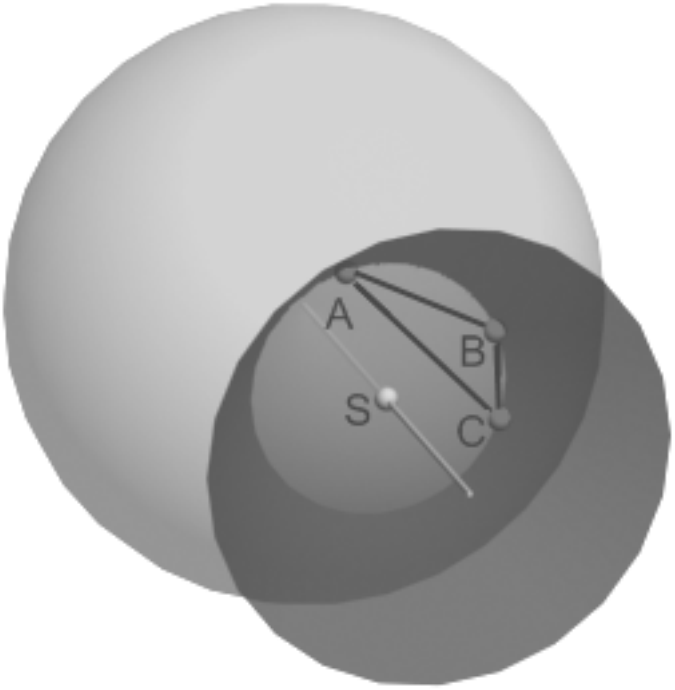
Protože jeden z bodů je vždy nulový, můžeme tento bod označit jako A a tím si znovu výpočet zjednodušit:

$$C = \frac{|D|^2 \cdot [B\times C] + |C|^2 \cdot [D\times B] + |B|^2 [C\times D]}{2\cdot B\cdot [C\times D]}$$

Tím jsme sice získali střed koule, ale ten jenom spolu s počátkem definuje přímku, na které se nachází střed kružnice. Protože ale provádíme triangulaci na jednotkové kouli, střed kružnice se musí také nacházet na této kouli. Takže stačí znormalizovat vektor C, čímž tento střed získáme.

Dále je nutné rozhodnout, zda je bod uvnitř této kružnice. Je možné vypočítat vzdálenost dvou bodů na povrchu koule jako délku kruhové výseče. Máme normalizované body C a X (testovaný bod). Úhel mezi vektory těchto dvou bodů je (při středu kružnice v počátku) $\alpha = \cos^{-1}(C \bullet X)$. Délka výseče je potom $l = \pi \cdot \alpha$. Z těchto vzorců je vidět, že pokud se zvětšuje úhel α, zvětšuje se i délka l. Podobně pokud se zmenšuje výsledek skalárního součinu vektoru C a X, zvětšuje se úhel alfa. Proto pro rozhodnutí, zda je bod uvnitř kružnice opsané trojúhelníku, stačí pouze porovnat výsledky skalárních součinů normalizovaného středu koule s testovaným bodem a jednom z bodů trojúhelníku.

Ve výsledku tato metoda vlastně testuje, zda je bod uvnitř kuželu s vrcholem v počátku a osou procházející středem koule opsané bodům trojúhelníku (obr 2.3.6).



Obr. 2.3.6 – Princip testování kritéria opsané kružnice na kouli. ABC – body trojúhelníku, S - střed opsané koule promítnutý na povrch jednotkové koule.

2.4 Triangulace na válci

2.4.1 Transformace do polárních souřadnic

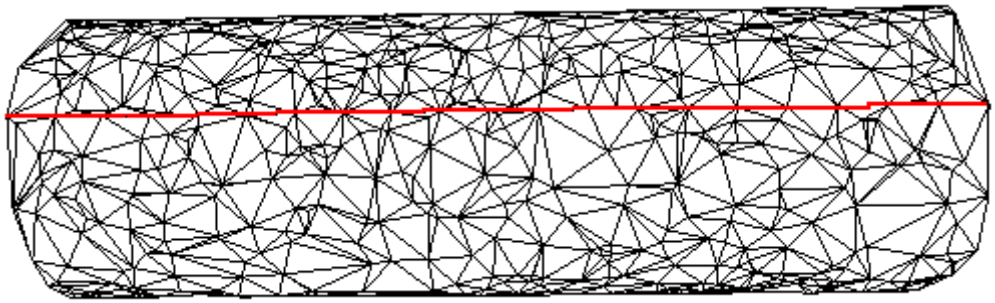
Transformace do polárních souřadnic je prakticky transformace do sférických souřadnic pouze ve dvou rozměrech.

$$r = \sqrt{x^2 + y^2}$$

$$\varphi = \operatorname{atan2}(y, x)$$

$$a\tan 2(y,x) = \begin{cases} \tan^{-1}\left(\left|\frac{y}{x}\right|\right)\cdot \operatorname{sgn}(y) & x > 0 \\ \frac{\pi}{2}\cdot \operatorname{sgn}(y) & x = 0 \\ \left(\pi - \tan^{-1}\left(\left|\frac{y}{x}\right|\right)\right)\cdot \operatorname{sgn}(y) & x < 0 \end{cases}$$

Třetí parametr z se potom nechává tak, jak je. Transformace plochy na válec je méně problémová, než transformace na kouli. Nedochází k žádným deformacím. Jediným zbývajícím problémem je opět šev v místě, kde úhel φ přechází z hodnoty 2πk do nuly (obr 2.4.1).



Obr. 2.4.1 – 2D triangulace promítnutá na válec se zvýrazněným švem

2.4.2 Triangulace na válci bez použití transformace

Kvůli tomuto švu jsme se opět, jako u koule, pokusili o triangulaci ve třech rozměrech. Jako u triangulace na kouli bylo nutné upravit dvě části – prohledávání sítě a testování kritéria prázdné opsané kružnice.

Prohledávání sítě je velmi podobné jako u koule. Jediným rozdílem je změna třetího bodu tvořícího plochu při znaménkovém testu. Tento bod byl u koule vždy v počátku ([0,0,0]). To u válce pořád platí, ale pouze pro složky **x** a **y**. Složka **z** tohoto bodu se musí pohybovat po ose válce tak, aby byla vždy v polovině hrany. Při bodech hrany E_1 a E_2 dostaneme tento třetí bod jako $P_0 = \left[0, \quad 0, \quad \frac{E_1{}^z + E_2{}^z}{2}\right]$.

V algoritmu procházky, po úpravě znaménkového testu, není nutné provádět žádné změny oproti kouli.

2.4.3 Test Delaunayova kritéria

Zato v testu Delaunayova kritéria je nutný zcela jiný postup. Průnik koule s válcem již není kružnice, ale elipsa. Proto není možné jednoduše převést postup použitý u koule na válec. Naštěstí můžeme využít toho, že transformace plochy na válec neprovází žádná deformace. Tudíž je možné provádět tento test v polárních souřadnicích.

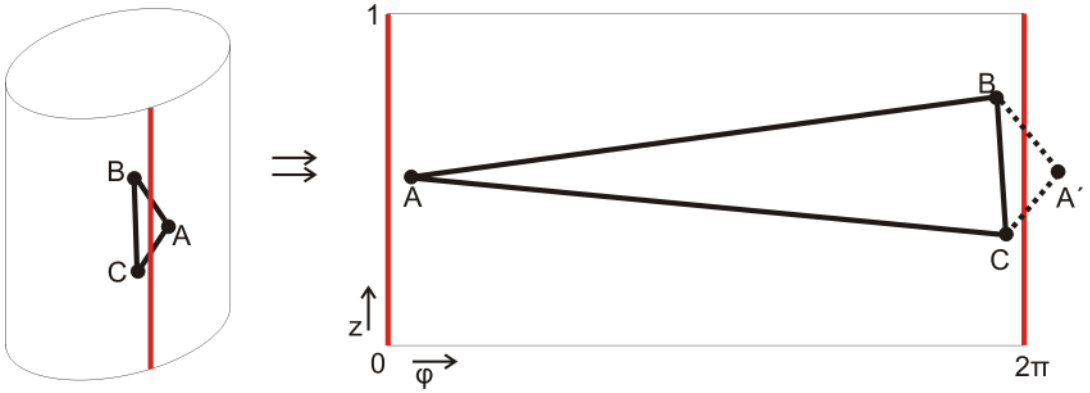
Test polohy bodu vůči kružnici

Máme tři body trojúhelníku P_1, P_2,P_3. Tyto tři body (pokud neleží na přímce) definují kružnici. Pokud jsou tyto tři body zároveň seřazeny po směru hodinových ručiček, můžeme rozhodnout, zda je čtvrtý bod P uvnitř této kružnice na základě následujícího determinantu:

$$d = \begin{vmatrix} P_1^x - P^x & P_1^y - P^y & \left(P_1^x - P^x\right)^2 + \left(P_1^y - P^y\right)^2 \\ P_2^x - P^x & P_2^y - P^y & \left(P_2^x - P^x\right)^2 + \left(P_2^y - P^y\right)^2 \\ P_3^x - P^x & P_3^y - P^y & \left(P_3^x - P^x\right)^2 + \left(P_3^y - P^y\right)^2 \end{vmatrix}$$

Pokud je tento determinant záporný, bod P je mimo kružnici. Pokud je kladný, je bod uvnitř.

Tento postup bez problémů funguje na většině povrchu válce. Výjimkou jsou trojúhelníky procházející přes šev. Na obrázku 2.4.2 je naznačeno promítnutí takového trojúhelníku do polárních souřadnic.



Obr. 2.4.2 – Promítnutí trojúhelníku procházejícího přes šev do polárních souřadnic.

Takovéto trojúhelníky je nutno detekovat. Nejjednodušší možností je porovnávat úhly φ. Pokud je úhel některého z bodů větší než daná hodnota φ_1, například $2\pi/3$, a zároveň má jiný z bodů trojúhelníku úhel φ menší než hodnotu φ_2 (třeba $\pi/3$), můžeme říct, že tento trojúhelník přechází přes šev. Volba hodnot φ_1 a φ_2 ovlivňuje, jak velké mohou být trojúhelníky v triangulaci. Přiblížením hodnot k 2π, resp. k 0 se snižuje možnost chybného označení trojúhelníku jako jdoucího přes šev, zatímco se zvyšuje možnost neoznačení trojúhelníku, který přes šev prochází. Vhodnou volbou počáteční trojúhelníkové sítě se dá ovšem takovýmto chybám zcela zamezit.

Pokud je tedy nalezen trojúhelník procházející přes šev, jednoduše se k úhlu φ přičte hodnota 2π, čímž se tvar trojúhelníku opraví (na obr. 2.4.3.1. je to trojúhelník BCA´). V tuto chvíli je také nutné porovnat φ testovaného bodu, a případně také přičíst 2π.

2.5 Triangulace složitějších objektů

Protože triangulace na kouli nebo válci je omezující, pokoušeli jsme se prozkoumat možnosti triangulace obecných trojúhelníkových sítí. Snažíme se tedy rekonstruovat tělesa, která již není možné triangulovat na povrchu koule nebo válce, ale stále jde o tělesa s jednoduchým povrchem bez množství detailů. Pořád jde o změny stávající aplikace, takže

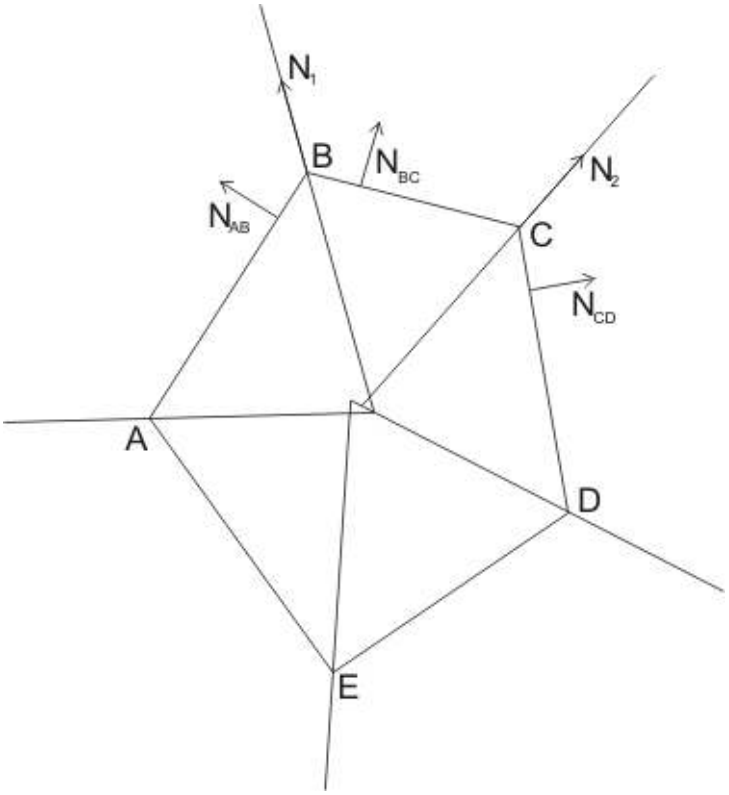
opět řešíme dva problémy. Výběr trojúhelníku, který bude rozdělen přidáním nového bodu, a rozhodování o přehození hrany ve čtyřúhelníku. Protože ani přibližně nevíme, jakého tvaru je těleso tvořené vstupními body, je nutné se ještě zamyslet nad tvarem počátečního tělesa. Protože algoritmus, který je použit pro vyhledávání trojúhelníků, je závislý na momentálně vytvořené trojúhelníkové síti, je nutné i zohlednit možnost seřazení bodů na vstupu podle nějakého kritéria.

Každý z těchto problémů má několik možností řešení. Zároveň je možné je zaměňovat nezávisle na ostatních. Tím vzniká mnoho možností triangulace těchto těles.

2.5.1 Výběr trojúhelníku pro vkládání

Pořád se pokoušíme docílit toho, že každý bod v prostoru bude jednoznačně přidělen jednomu trojúhelníku. Je jednoduché zajistit, aby každý bod v prostoru byl přidělen k alespoň jednomu trojúhelníku.

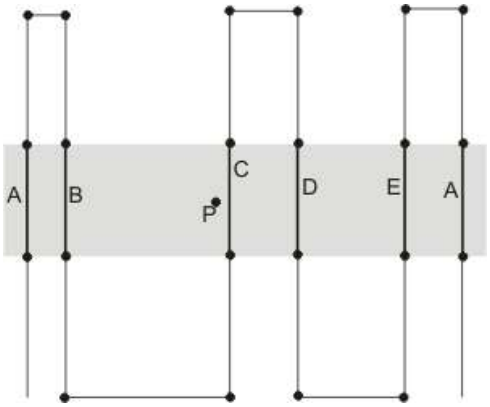
Problém je opět výběr plochy procházející hranou. U koule a válce jsme si jednoduše dodefinovali třetí bod, pomocí kterého se dala tato plocha určit. Teď použijeme podobný postup. Jenom není možné použít jeden bod jako u koule nebo jednu přímku jako u válce. Místo toho použijeme normály trojúhelníků, sousedících s hranou (obr. 2.5.1.1.). Průměrem z těchto normál (N_{AB}, N_{BC}, N_{CD} atd. na obrázku) získáme „normálu hrany“ (N_1 a N_2 na obr.). Přičtením hodnoty této normály k jednomu z bodů hrany dostaneme třetí bod pro definici plochy.



Obr. 2.5.1 – Rozdělení prostoru – zjednodušený 2D pohled

Na obr. 2.5.1 uprostřed a na obr. 2.5.2 je také vidět, že pomocí tohoto rozdělení nedosáhneme jednoznačného určení trojúhelníku. Proto jsme přidali ještě další kritérium pro výběr trojúhelníku. Tím je vzdálenost bodu, pro který je trojúhelník hledán, od plochy tvořené body trojúhelníku. Z vybraných trojúhelníků je potom vybrán ten, který má tuto vzdálenost v absolutní hodnotě minimální.

Tato vzdálenost se dá jednoduše spočítat upravením znaménkového testu pro rozhodnutí o vzájemné poloze plochy a bodu (detailněji popsán v kapitole 2.3.3). Plochu tentokrát tvoří tři body trojúhelníku. Ve znaménkovém testu stačí pouze zajistit, aby použitá normála plochy měla délku 1. Výsledkem testu potom bude vzdálenost bodu od nejbližšího bodu v rovině.



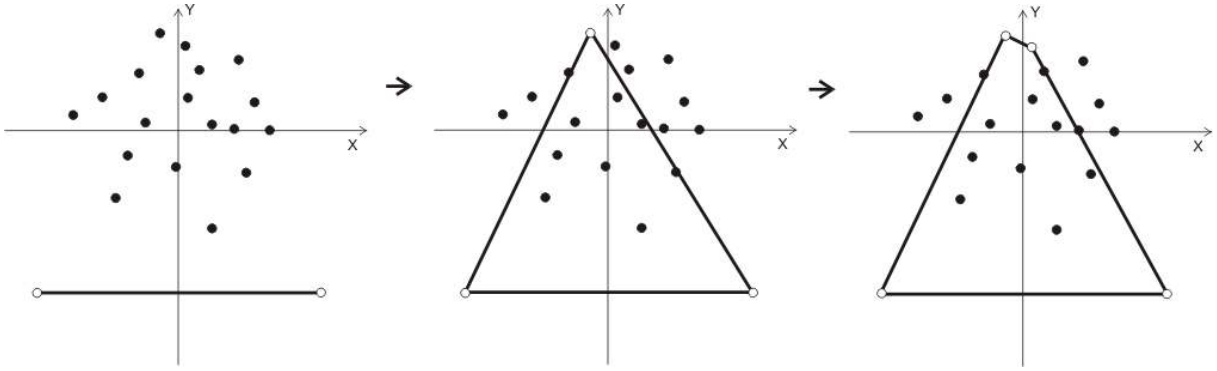
Obr. 2.5.2 – Případ, kdy je možné část prostoru přidělit více vyznačeným trojúhelníkům. V tomto případě vyhovují všechny vyznačené trojúhelníky, ale použit je pouze trojúhelník C, který je bodu P nejblíže.

2.5.2 Sestrojení počátečního tělesa

Máme dvě možnosti, jak sestavit těleso pro začátek triangulace. Buď můžeme vytvořit nějakou fixní trojúhelníkovou síť, jejíž body po triangulaci odstraníme, nebo vytvoříme těleso z bodů, které jsou na vstupu. Pro každou možnost jsme zkoumali jeden možný přístup

Těleso složené z bodů nezávisle na vstupu

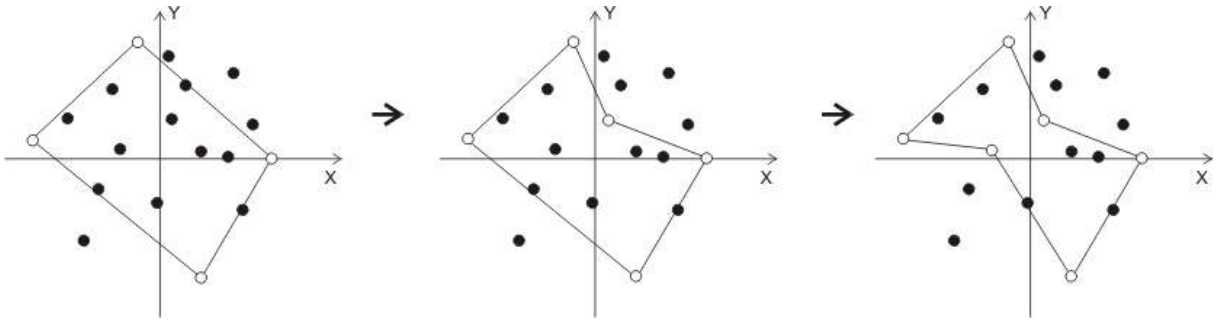
Tento přístup se snaží adaptovat původní 2D řešení i pro složitější případy. Těleso je složeno z jednoho trojúhelníku, který je zcela mimo vstupní body. Například posunuté po ose Y tak, aby byly všechny body nad ním. Zároveň je tento trojúhelník dostatečně velký, aby všechny body, promítnuté do roviny XZ, byly uvnitř tohoto trojúhelníku (obr 2.5.3).



Obr. 2.5.3 – Počáteční těleso a přidání prvních dvou bodů. Používá se seřazení bodů podle osy Y

Těleso vytvořené z bodů vstupu

Tato varianta se snaží vytvořit takový šestistěn, který je složen z krajních bodů vstupní množiny. Nalezne body s maximální a minimální složkou X,Y,Z a z těchto šesti bodů vytvoří počáteční těleso (obr. 2.5.4). Samozřejmě za předpokladu, že těchto šest bodů je každý jiný. V opačném případě je nutné vybrat jiné body (s druhou největší vzdáleností od nuly podle dané osy).



Obr. 2.5.4 – Naznačení tvorby počátečního tělesa a vložení prvních dvou bodů ve 2D. Vkládané body jsou vybírané náhodně.

2.5.3 Ohodnocování trojúhelníků v prostoru

Přesunem do tří rozměrů vznikají nové možnosti rozhodování o prohození hrany. Pořád je možné používat kritéria popsaná v kap. 2.1.2. Jedinou výjimkou je Delaunayovo kritérium opsané kružnice. To by bylo nutné promítnout na nějakou plochu, třeba definovanou body jednoho z trojúhelníků. To by ale způsobovalo, při větším úhlu mezi trojúhelníky problémy.

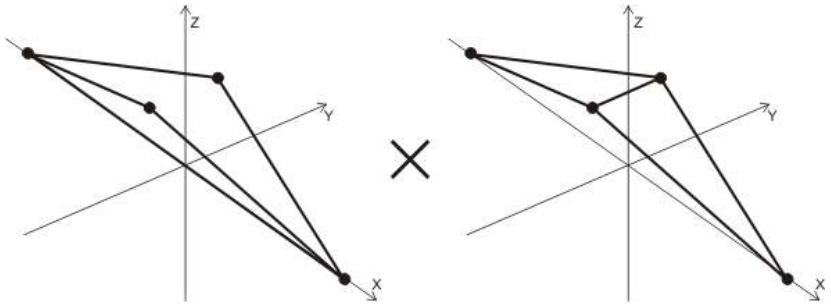
Oproti triangulaci na ploše je možné využít toho, že trojúhelníky nejsou na ploše.

Úhel mezi normálami trojúhelníka

Ze dvou možností umístění hrany vybere vždy tu, při níž svírají normály trojúhelníků menší úhel. Tím tvoříme těleso tak, aby se každé dva sousední trojúhelníky co nejvíce blížily k ploše (obr 2.5.5).

Obsah plochy trojúhelníků

Snaží se vybírat hrany tak, aby se tvořily trojúhelníky s co nejmenším obsahem plochy (obr 2.5.5).



Obr. 2.5.5 – Dvě možnosti vytvoření hrany ve čtveřici bodů. Je vidět, že trojúhelníky na pravé straně mají menší obsah plochy a zároveň jejich normály svírají menší úhel.

2.5.4 Výběr přidávaného bodu

Protože výsledky algoritmu použitého pro rozhodování, který trojúhelník bude rozdělen vložením nového bodu, se liší spolu s měnícím se pořadím vstupních bodů, je možné ovlivnit výslednou trojúhelníkovou síť seřazením bodů na vstupu. Zkoumali jsme tři možné postupy řazení bodů.

Poměrně jednoduchými možnostmi jsou náhodný výběr bodů a body seřazené podle jejich polohy. Druhá možnost by mohla být vhodná pro triangulace s jedním velkým trojúhelníkem. Těleso by pak bylo triangulací vytvářeno postupně od nejvzdálenějších bodů k nejbližším.

Další možnost vybírá body tak, že vždy ze všech ještě nepoužitých bodů vybere ten, který je od všech použitých bodů nejdále. Tím se snažíme simulovat postupné utváření objektu od nejdůležitějších tvarů k detailům.

3. Původní řešení

Původní řešení dokázalo triangulovat body v rovině XY. Dále bylo možné pomocí několika nastavitelných virtuálních nástrojů rýt do této plochy a tím měnit trojúhelníkovou síť v reálném čase. Dále bylo možné zapnout simulaci písčitého terénu. Detailní popis lze nalézt v [Kad07], [Pur07] a [Sed07]. Po spuštění je do startovního objektu vložena množina bodů. Výsledná trojúhelníková síť potom slouží jako prostor pro vlastní rytí nástrojem.

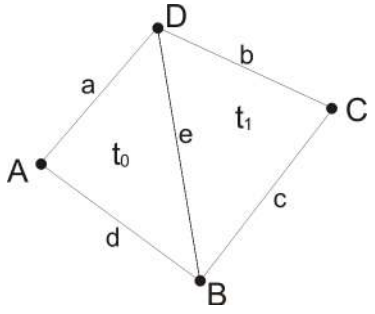
Program využívá multiplatformní knihovnu SDL – Simple DirectMedia Layer, která zajišťuje využívání systémových funkcí. Pro zobrazení je využito OpenGL. Tím by měla být zajištěna funkčnost i na jiných než win32 platformách.

Trojúhelníkovou síť zastupuje třída *Mesh*. Ta zastřešuje veškerou práci se sítí, jako je vkládání nových bodů a vynucených hran.

Další důležitou třídou je abstraktní třída *VirtualTool.* Ta má na starosti ovládání virtuálního nástroje.

3.1 Reprezentace trojúhelníkové sítě

Trojúhelníková síť je reprezentována strukturami trojúhelníků, bodů a hran. Body obsahují spolu s dalšími atributy jejich polohu. Trojúhelníky a hrany potom pracují s odkazy na struktury. Strukturu jejich propojení bude asi nejjednodušší vysvětlit na jednoduchém příkladu (obr. 3.1.1).



Obr. 3.1.1 – Jednoduchá trojúhelníková síť

Trojúhelníky

	Bod A	Bod B	Bod C	Hrana A	Hrana B	Hrana C	Soused A	Soused B	Soused C
t_0	A	B	D	e	a	d	t_1	-	-
t_1	B	C	D	b	e*	c	-	t_0	-

* u hrany znamená, že je v trojúhelníku označená jako invertovaná

Hrany

	a	b	c	d	e
Bod A	A	B	C	D	B
Bod B	B	C	D	A	D

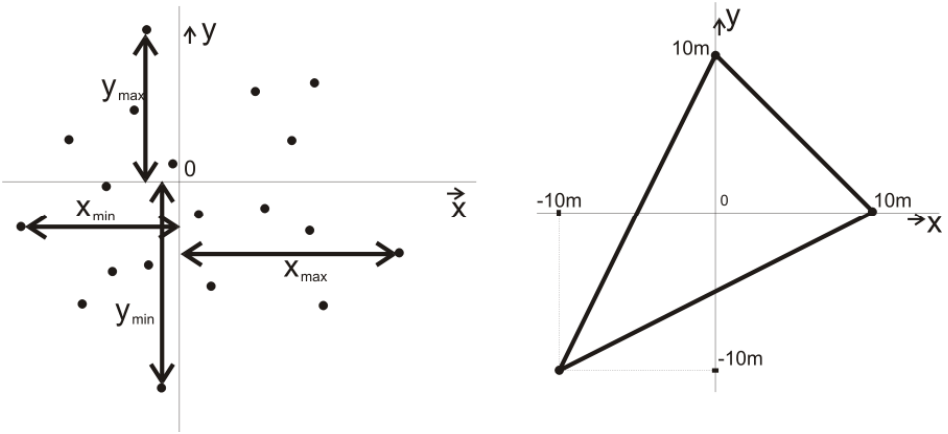
Z tohoto příkladu je vidět, že body trojúhelníku jsou vždy orientované proti směru hodinových ručiček. Pokud je hrana sdílená, je u jednoho z trojúhelníků vždy označená jako invertovaná. Hrany a sousedi jsou v trojúhelníku uloženy tak, že hrana A (soused A) jsou vždy naproti bodu A atd.

Ve všech strukturách jsou navíc ještě uloženy další atributy. Ve struktuře bodu jsou navíc informace o hranách začínajících nebo končících v bodu a seznam trojúhelníků, který tento bod obsahuje. Tyto informace ale nejsou používané v této práci.

Detailnější informace o stavbě a reprezentaci sítě lze nalézt v [Pur07].

3.2 Počáteční objekt pro triangulaci

Počáteční objekt je trojúhelníková síť vytvořená ještě před vložením prvního bodu. Inkrementální algoritmus totiž předpokládá vkládání bodů do trojúhelníkové sítě. Proto je nutné před započetím triangulace vytvořit jeden trojúhelník, do kterého vložíme vstupní body. Tyto body jsou při vytváření tohoto objektu načteny. Proto je možné najít maximální a minimální hodnoty podle jednotlivých os (obr. 3.1.1. vlevo). Z těchto čtyř hodnot je potom určena hodnota m takto: $m = \max(x_{min}, x_{max}, y_{min}, y_{max})$. Pomocí hodnoty m je potom vytvořen počáteční trojúhelník, jak je naznačeno na obrázku 3.2.1. vpravo.



Obr. 3.2.1. Vytvoření počátečního trojúhelníku.

3.3 Vkládání počátečních bodů

Jako počáteční body je označován seznam bodů vložených do startovního trojúhelníku hned při startu aplikace. Vložením bodů vzniká síť, na které je potom možné provádět operace s virtuálním nástrojem. Existují tři možnosti vzniku tohoto seznamu

a) Náhodně vygenerované body

Vygeneruje zadaný počet bodů v rozsahu <0, 1>x<0, 1> s nulovou souřadnicí Z. K tomu je použita implementace generátoru náhodných obsažená ve standardní knihovně C

b) Body načtené ze souboru

Načte určitý počet bodů ze souboru, jehož jméno je předáno aplikaci jako argument. Struktura souboru obsahuje na první řádce počet bodů v souboru. Zbytek souboru je potom jedna řádka pro každý bod. Bod je definován třemi čísly v plovoucí řádové čárce oddělenými mezerou.

c) Kompletní trojúhelníková síť načtená ze souboru

Tato možnost načte kompletní informace o trojúhelníkové síti. Na rozdíl od prvních dvou možností není nutné vkládat body a triangulovat síť.

Body se načítají stejně jako u možnosti b.

Podobně jako u bodů následuje na jedné řádce počet trojúhelníků. Poté následuje pro každý řádek trojice indexů do pole bodů.

Poslední část souboru obsahuje informace o sousednosti. Pro každý trojúhelník opět následuje trojice indexů do pole trojúhelníků určující sousední trojúhelník.

3.4 Ovládání nástroje pro rytí

Vzhledem k pouze 2D triangulaci bylo ovládání nástroje uděláno velmi jednoduše. Před vykreslením každého snímku byla pomocí funkce `SDL_GetMouseState` zjištěna pozice kurzoru myši vzhledem k oknu aplikace. Tyto souřadnice potom byly přemapovány na pozici nástroje následovně:

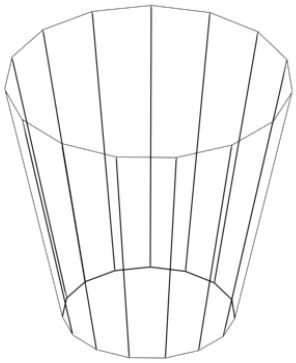
$$P^x = x/\text{ScreenWidth}$$
$$P^y = 1 - (y/\text{ScreenHeight})$$

Kde x a y jsou pozice kurzoru vzhledem k oknu aplikace v pixelech, ScreenWidth/ScreenHeight je šířka šířka/výška okna aplikace. P^x,P^y jsou potom souřadnice polohy nástroje.

3.5 Rytí nástrojem pomocí vynucování hran

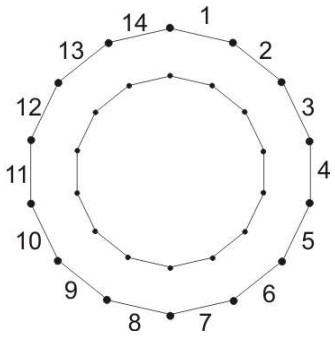
Stisknutím tlačítka myši se vytvoří obtisk nástroje v trojúhelníkové síti. Toho je docíleno pomocí vkládání vynucených hran (kap. 2.2). Nástroj je prezentován třídou implementující rozhraní *VirtualTool*. Proto se může tvar nástroje měnit dle potřeby.

V aplikaci byl naimplementován nástroj ve tvaru kulatého hrotu (obr. 3.5.1.) s nastavitelnou složitostí.



Obr. 3.5.1. – Náčrt tvaru použitého nástroje

Obtisk tohoto nástroje probíhá ve třech krocích. V prvních dvou krocích se do sítě obtiskne vnější a vnitřní kruh právě pomocí vkládání vynucených hran. Hrany jsou vkládány do kruhu za sebou tak, že počáteční bod hrany je vždy koncovým bodem hrany předcházející. (obr. 3.5.2.)



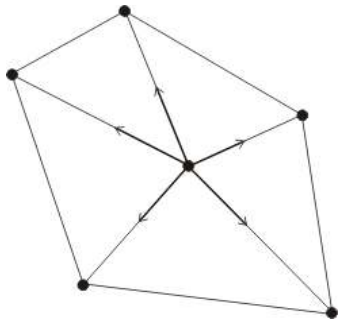
Obr. 3.5.2. – Pořadí vkládání hran při obtisku nástroje

Po vložení vnitřního a vnějšího kruhu je ještě nutné snížit výšku bodů, které již existovaly uvnitř kruhů před obtiskem nástroje. Tyto body by po vložení hran měly pořád starou výšku. To je zařízeno pomocí rekurzivního algoritmu, který postupuje po trojúhelnících a pozici všech bodů každého trojúhelníku sníží.

Podrobnosti o postupu použitém při rytí nástrojem je možné nalézt v [Pur07]

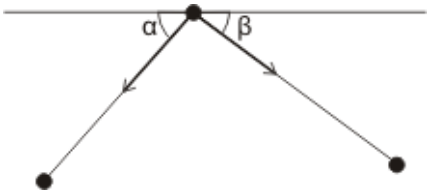
3.6 Fyzikální simulace rytí do pískového povrchu

Pro simulaci pískového povrchu je použit algoritmus simulující přesýpaní písku mezi jednotlivými vrcholy. Ten vyhledává vrcholy splňující kritérium pro přesýpání písku a změnou výšky (hodnoty z-ové souřadnice) potom provede vlastní přesun písku (obr. 3.6.1).



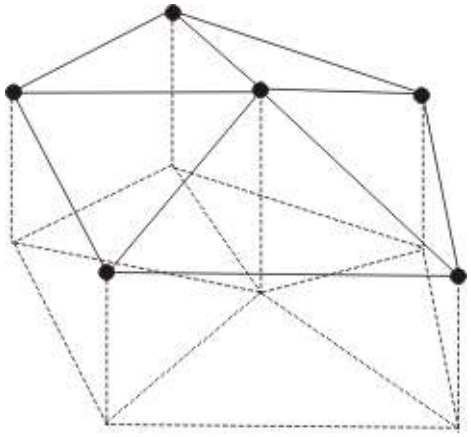
Obr. 3.6.1. - Přenos písku mezi jedním bodem ve větší výšce, než okolní body

Tímto kritériem je úhel mezi hranou a vodorovnou přímkou procházející vyšším bodem (obr. 3.6.2.). Podle [Sed07] je hraniční úhel pro přesýpání suchého písku asi 30°.



Obr. 3.6.2. – Úhly mezi hranami dvou bodů a vodorovnou přímkou

Velikost změny výšky bodů je vypočítána podle objemů těles, která vzniknou spuštěním stěn ze hran trojúhelníku a na spodní straně uzavřením vodorovnou plochou (obr. 3.6.3). Součet objemů těchto těles pro celou síť by se potom v průběhu simulace neměl měnit.



Obr. 3.6.3. – Tělesa použitá pro počítání objemu

Opět lze detailnější popis algoritmu nalézt v [Sed07]

4. Úpravy

4.1 Obecné úpravy

Aby bylo jednoduše možné měnit způsob triangulace, bylo nutné všechny funkce závislé na poloze bodů seskupit na jedno místo. Tak vznikla abstraktní třída *Projector*. Ve třídě *Mesh* přibyla jedna proměnná právě tohoto typu. Všechny tyto funkce jsou potom odstraněny z třídy *Mesh* a přesunuty/implementovány jako *Projector*. Konkrétně jde o tyto metody.:

getTriangleAtPoint – podle zadaného bodu určí, do kterého trojúhelníku v triangulaci má být nový bod přidán

pointInTriangleCircle – ze tří bodů trojúhelníku a jednoho testovaného bodu určí, zda je testovaný bod uvnitř kružnice opsané trojúhelníkem (případně může používat i jiné triangulační kritérium)

createInitialMesh – Vytvoří počáteční triangulaci, do které jsou potom přidávány body.

getRandomPoint – vygeneruje nový náhodný bod. Je použito při nastaveném generování náhodných bodů.

normalizePoint – zadaný bod posune na povrch tělesa (koule, válce…)

getNormalAtPoint – získá normálu na triangulačním tělese v místě bodu

getPositionAtPoint – funguje podobně jako *normalizePoint*, ale testuje okraje objektu (např. u válce hodnotu z povolí pouze v intervalu <0,1>.

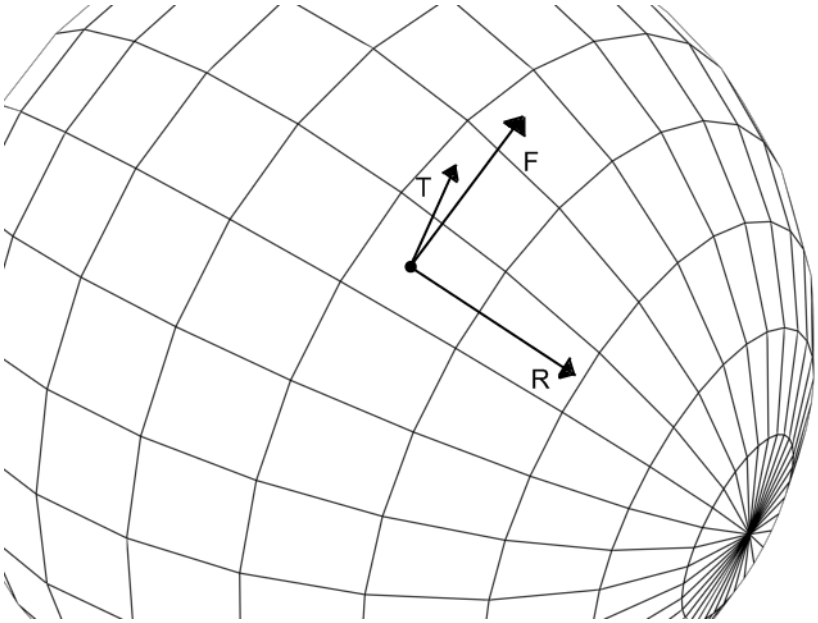
isPointInside – zjistí, pokud je bod, předaný jako parametr, uvnitř triangulačního tělesa.

Abstraktní metody této třídy implementuje několik tříd. Změnou obsahu proměnné ve třídě *Mesh* dojde zároveň ke změně tělesa, na kterém je triangulace prováděna.

4.2 Změna ovládání nástroje

4.2.1 Úprava pro použití v prostoru

Při použití myši (jako snímač pracující ve 2D) bylo nutné upravit nástroj tak, aby se posouval po povrchu triangulačního tělesa. Kvůli tomuto úkolu jsou v třídě *Projector* *getNormalAtPoint* a *getPositionAtPoint.* Díky nim můžeme vytvořit a udržovat souřadný systém odpovídající pohybu nástroje. Pomocí tohoto systému jsou potom i generovány body definující tvar nástroje.



Obr. 4.2.1 – Souřadný systém používaný pro pohyb nástroje po povrchu koule

Pohyb myši, snímaný v rovině XY, je takto přemapován na rovinu tvořenou vektory R a F (vektory R,F a T jsou na sebe kolmé). Následně je posunuta pozice nástroje. Tím se ovšem nástroj odpoutá od povrchu tělesa. Proto je pomocí funkce *getPositionAtPoint* tato pozice posunuta zpět na povrch tělesa (obr. 4.2.1).

Posunem se zároveň změní i normála povrchu tělesa (na obr. 4.2.1.1. označena jako T). Proto je nutné změnit i vektory R a F tak, aby byly stále kolmé. Protože počítáme pouze s malými změnami pozice nástroje za jeden snímek, bude se pouze málo měnit i normála. Proto můžeme pro vypočítání nových vektorů použít staré hodnoty.

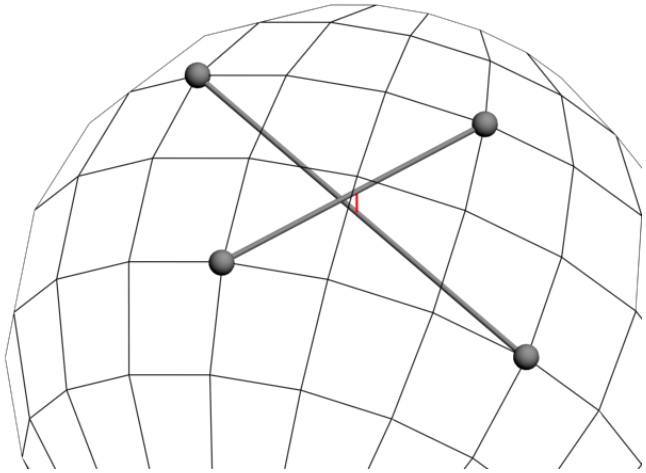
$$R' = T \times F$$
$$F' = R' \times T$$

R‘ a F‘ jsou nové vektory, R a F jsou vektory před posunem.

4.3 Upravený způsob testu na křížení hran

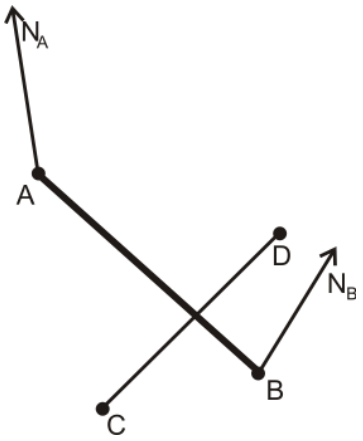
Díky přesunu do 3D bylo nutné změnit i tento test. Ve třech rozměrech je řešení průsečíku dvou přímek poměrně složité. Díky omezené přesnosti čísel v počítači je dokonce možné vyhodnotit dvě přímky, které se protínají, jako neprotínající se.

To nám ale nemusí vadit, protože hrany s body na zakřiveném povrchu se až na výjimky protínat nebudou. Tyto hrany se ale můžou pořád protínat vzhledem k povrchu tělesa, na kterém je prováděna triangulace (obr. 4.3.1.).



Obr. 4.3.1. – Dvě hrany, které se v trojrozměrném prostoru nekříží. Pokud by ovšem byly promítnuty na povrch koule (nebo transformovány do sférických souřadnic), tak by jejich společný bod existoval.

Nový univerzální test používá znaménkový test popsaný v kap. 2.3.3. Pomocí něho se dá zjistit poloha bodu vůči přímce. Pro zjištění, zda se protínají dvě úsečky v prostoru, se dá v tomto případě použít podobný postup, jako ve 2D (kap. 2.2.1). Jenom je nutné zaměnit přímku za plochu. Pro definování plochy potřebujeme 3 body. Hrana nám dává jenom dva (obr. 4.3.2). Tento problém řeší funkce *getNormalAtPoint*. Víme, že plocha má procházet dvěma body a její normála má být kolmá na normálu objektu v daném místě.



Obr. 4.3.2 – Hrana AB a proti ní testované body hrany CD

Normálu v bodu A a B získáme pomocí výše zmíněné funkce. Normálu plochy získáme takto:

$$N = (N_A + N_B) \times (A - B)$$

Normála spolu s bodem (některý z dvojice A nebo B) nám už jasně definuje plochu, takže můžeme snadno rozhodnout, zda přímka CD tuto plochu protíná.

Tento test v zásadě funguje podobně jako promítnutí bodů ABCD na plochu definovanou normálami v bodech A a B a následné rozhodování o průsečíku přímek v této ploše.

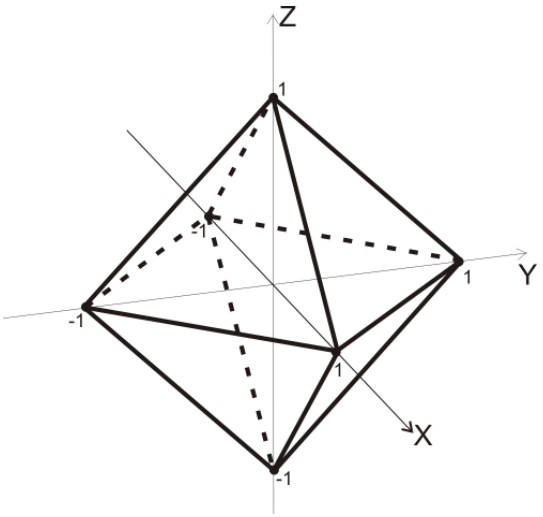
4.4 Triangulace koule

Pro triangulaci koule byla vytvořena třída *SphereProjector*. Ta, jakožto první třída vytvořená po opuštění triangulace na ploše, musí implementovat všechny funkce třídy *Projector*.

Popis algoritmu pro triangulaci na kouli je v kapitole 2.3 této práce. Zde budou jenom implementační detaily, které v dřívější kapitole chybí.

Počáteční triangulace

Počáteční triangulace je trojúhelníková síť, která je vytvořena ještě před vložením prvního bodu. Pokud při triangulaci na ploše stačil jeden trojúhelník, u koule už je situace složitější. Metoda *createInitialMesh* vytvoří téměř nejjednodušší možnou kouli. Tou je osmistěn, znázorněný na obr. 4.4.1.



Obr. 4.4.1. – Osmistěn vytvořený před začátkem triangulace koule

Generování náhodných bodů

Náhodně se vygeneruje bod, jehož všechny tři složky jsou v intervalu <-1,1>. Tento bod je potom normalizován, takže leží přesně na kouli.

Získání normály a promítnutí bodu na těleso (funkce *getNormalAtPoint* a *getPositionAtPoint*)

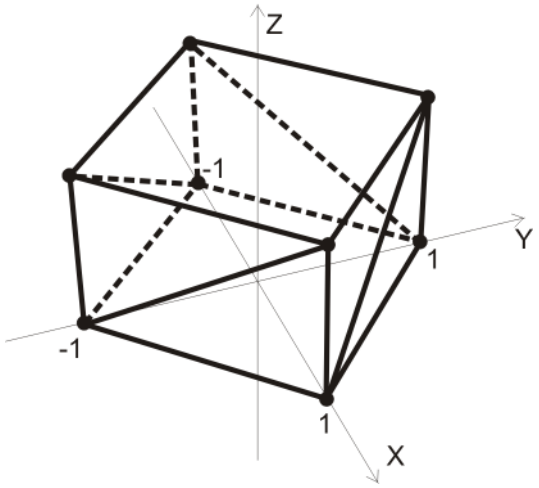
Tyto funkce fungují shodně tak, že normalizují vstupní bod.

4.5 Triangulace válce

Triangulace na válci j zařízena pomocí třídy *CylinderProjector*. Tato třída je potomkem třídy *SphereProjector*. Díky tomu není nutné znovu implementovat vyhledávání trojúhelníků. To je naprosto stejné jako u koule. Jedinou nutnou úpravou je funkce provádějící znaménkový test hrany a bodu (viz 2.4.2.).

Počáteční triangulace

Počáteční trojúhelníková síť je, podobně jako u koule, téměř nejjednodušší možný válec. V zásadě jde o krychli bez vrchní a spodní stěny otočenou o 45° podle osy Z (obr. 4.4.1.)



Obr. 4.4.1. – Čtyřstěn vytvořený před začátkem triangulace válce

Generování náhodných bodů

Generování bodů probíhá podobně jako u koule. Je vygenerován bod, jehož složky X a Y jsou v intervalu <-1,1>. Složka Z potom v intervalu <0,1>. Tento bod je potom znormalizován v rovině XY.

Získání normály a promítnutí bodu na těleso (funkce *getNormalAtPoint* a *getPositionAtPoint*)

Jako u koule tyto funkce normalizují vstupní bod. Ovšem pouze v rovině XY. *getPositionAtPoint* navíc omezuje výstup v ose Z na interval <0,1>.

4.6 Triangulace obecných objektů

Pro experimentování s triangulací obecných těles byly vytvořeny dvě třídy. Jedna (*FreeFormProjector*) se pokouší provést triangulaci s tělesem vytvořeným pomocí hledání maxim ve vstupních bodech (první část kapitoly 2.5.2). Druhá třída (*SortedFreeFormProjector*) se potom pokouší řešit stejný problém se základní trojúhelníkovou sítí mimo vstupní body (druhá část kapitoly 2.5.2).

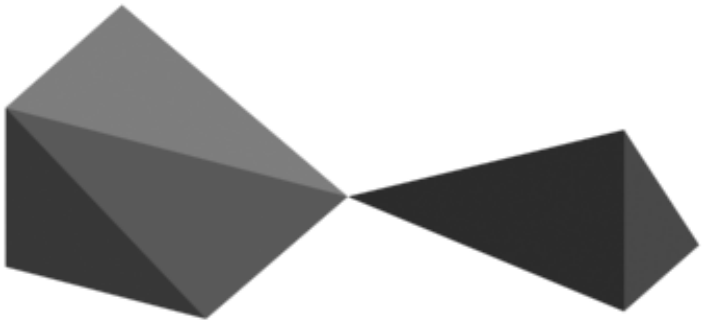
4.6.1 FreeFormProjector

To je opět potomek třídy *Projector*. Většina funkcí je popsána v kapitole 2.5. Protože je tato třída experimentální, implementuje čtyři možná triangulační kritéria. Konkrétně jde o minimalizaci maximálního úhlu, minimalizaci délky střední hrany (obě popsány v kapitole 2.1.3), minimalizaci úhlu mezi normálami trojúhelníků a minimalizaci obsahu plochy trojúhelníků (kap. 2.5.3). Jednoduchou změnou je možné změnit použité kritérium, případně i použít kombinaci kritérií.

Upravena bylo i chování funkce *getRandomPoint*. Ta by měla generovat náhodné body. Protože ale potřebujeme vstupní body řadit, je nutné je nejprve načíst a seřadit. Tyto body jsou potom uloženy v členské proměnné třídy a postupně předávány touto funkcí do aplikace. Díky tomu nemusíme dělat velké úpravy uvnitř aplikace. Ta si myslí, že body jsou náhodné. Ve skutečnosti jsou ovšem načtené z textového souboru.

Tyto body jsou řazeny tak, aby byl vždy vybrán bod, který je nejvzdálenější od všech již použitých. Tím se snažíme vytvářet těleso postupně od nejdůležitějších částí k menším detailům.

Poslední nepopsanou změnou v této třídě je možnost pozdržení legalizace pro určitou první část vstupních bodů. Při vložení několika prvních bodech je možné, že se díky legalizaci některé části tělesa téměř oddělí (vzniknou dvě tělesa, která jsou propojena jenom jedním společným bodem (obr 4.6.1).



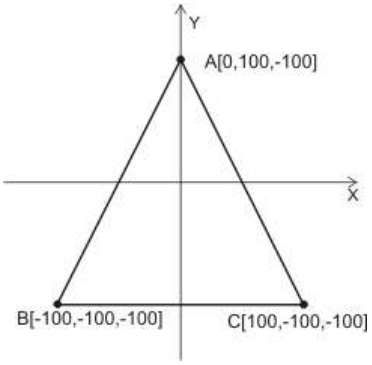
Obr. 4.6.1 – Příklad tělesa spojeného jediným bodem

To je způsobeno tím, že těleso je v počátku triangulace velmi jednoduché. Tím pádem může mít přehození jedné hrany velké následky v konečném tvaru. Protože celá legalizace postupuje na sousedy trojúhelníků, takovouto degradaci se už nepodaří v průběhu triangulace napravit.

Proto *FreeFormProjector* sleduje, kolik bodů už bylo vloženo (pro každé vložení je zavolána funkce *getTriangleAtPoint*). Když počet bodů přesáhne určitou mez, provede se legalizace celé trojúhelníkové sítě a povolí se i pro nově vložené body. Při správně zvolené mezi je trojúhelníková síť už dostatečně složitá, takže legalizace nezpůsobí problémy s degradací sítě.

4.6.2 SortedFreeFormProjector

Tato třída je odvozena od předcházející (*FreeFormProjector*). Je určena pro experimentování s počátečním tělesem mimo vstupní body. Oproti třídě *FreeFormProjector* je změněno jenom generování bodů, kdy jsou body seřazeny sestupně podle osy Z. Další změnou je vytvoření počáteční triangulace. Ta nezávisle na vstupních bodech vytvoří trojúhelník načrtnutý na obrázku 4.6.2.



Obr 4.6.2 – Startovací trojúhelník využitý ve třídě SortedFreeFormProjector

4.7 Ovládání aplikace pomocí haptického zařízení

První pokusy s haptickým zařízením byly provedeny v rámci práce [Pur09] bc. Václavem Purchartem. Ten upravil aplikaci tak, aby bylo možné použít toto zařízení pro rytí na ploše XY. Úkolem této práce bylo opět rozšířit tuto funkčnost na kouli a válec.

4.7.1 Haptické zařízení PHANTOM Omni® Haptic Device

Obr. 4.7.1 – Haptické zařízení

Všechny experimenty probíhaly se zařízením PHANTOM Omni® Haptic Device (obr. 4.7.1) vyrobeným firmou SensAble Technologies. Toto zařízení umožňuje snímání jak polohy hrotu, tak jeho natočení. Zároveň dokáže samo působit na hrot silou, která může v naší aplikaci simulovat odpor při zarytí nástroje do povrchu tělesa.

4.7.2 Komunikace programu s haptickým zařízením

Aplikace komunikuje s haptickým zařízením pomocí knihovny OpenHaptics. Pomocí této knihovny je při startu aplikace navázáno spojení se zařízením. Poté je spuštěno samostatné vlákno, které opakovaně spouští (při standardním nastavení 1000x za vteřinu) metodu *gravityWellCallback*. V této metodě dochází ke čtení pozice nástroje a jeho uložení pro další potřebu. Také je zde nastavována síla, kterou má zařízení působit na hrot. Tyto operace se provádějí pomocí knihovních metod *hdGetDoublev*(s parametrem HD_CURRENT_POSITION získá okamžitou pozici nástroje) a *hdSetDoublev*(s parametrem HD_CURRENT_FORCE nastaví okamžitou sílu). Detaily o komunikaci je možné nalézt v [Pur09].

Tato operace musí běžet asynchronně s aplikací. V hlavní smyčce aplikace by došlo za vteřinu k maximálně několika desítkám spuštění této funkce v závislosti na složitosti sítě a

prováděných akcích. To je ale hlavně pro nastavení odporové síly pro lidskou ruku velmi málo.

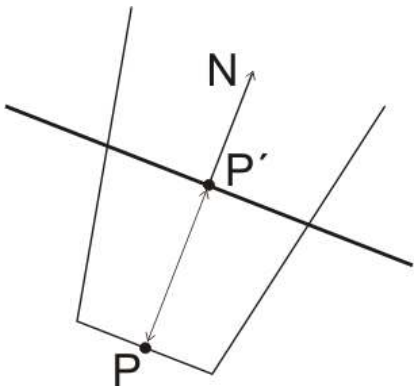
4.7.3 Úprava aplikace pro využití haptického zařízení

Bylo nutné upravit tři základní věci. První je pohyb nástroje, který již nemůže být vázaný na povrch tělesa. Druhou změnou je nutnost detekce, zda nástroj prochází tělesem, a případné automatické spuštění vkládání obtisků nástroje do trojúhelníkové sítě. Třetí úpravou je upravit výpočet odporové síly tak, aby odpovídala použitému tělesu (v původní verzi se pouze porovnávala výška hrotu podle osy Z)

První změna je triviální. Všechno může zůstat, jak je popsáno v kapitole (4.2), ale pozice nástroje se mění absolutně v závislosti na pozici hrotu haptického zařízení. Otáčení nástroje tak, aby hrot mířil vždy kolmo k povrchu tělesa, zůstává.

Druhá změna si vyžádala doplnění funkce *isPointInside* do třídy *Projector*. Při každé změně polohy nástroje se potom pomocí této funkce zjistí, zda je hrot uvnitř tělesa. Pokud ano, zapne se provádění obtisků (rytí).

K nastavení síly byly znovu použity funkce třídy *Projector getPositionAtPoint* a *getNormalAtPoint*.



4.7.2 – Zjištění míry zanoření nástroje do tělesa pro vypočítání odporové síly

Bod P (na obr. 4.7.2), což je momentální pozice nástroje, je promítnut pomocí *getPositionAtPoint* na povrch tělesa (bod P´). Poté je pomocí *getNormalAtPoint* získána normála tělesa v tomto bodě. Spočítáním $D = (P - P') \bullet N$ získáme vzdálenost, která ukazuje, jak hluboko je hrot uvnitř tělesa. Odporová síla vždy působí směrem ven z tělesa. Tudíž je nastavena jako vektor $F = N \cdot (CD)^2$, kde C je konstanta, určující tuhost tělesa (osvědčila se mezi 20 a 100).

Tuto sílu je ovšem lepší nastavovat postupně od [0,0,0] do F po určitý čas. Nastavením síly okamžitě může zařízení způsobit citelný ráz pro lidskou ruku.

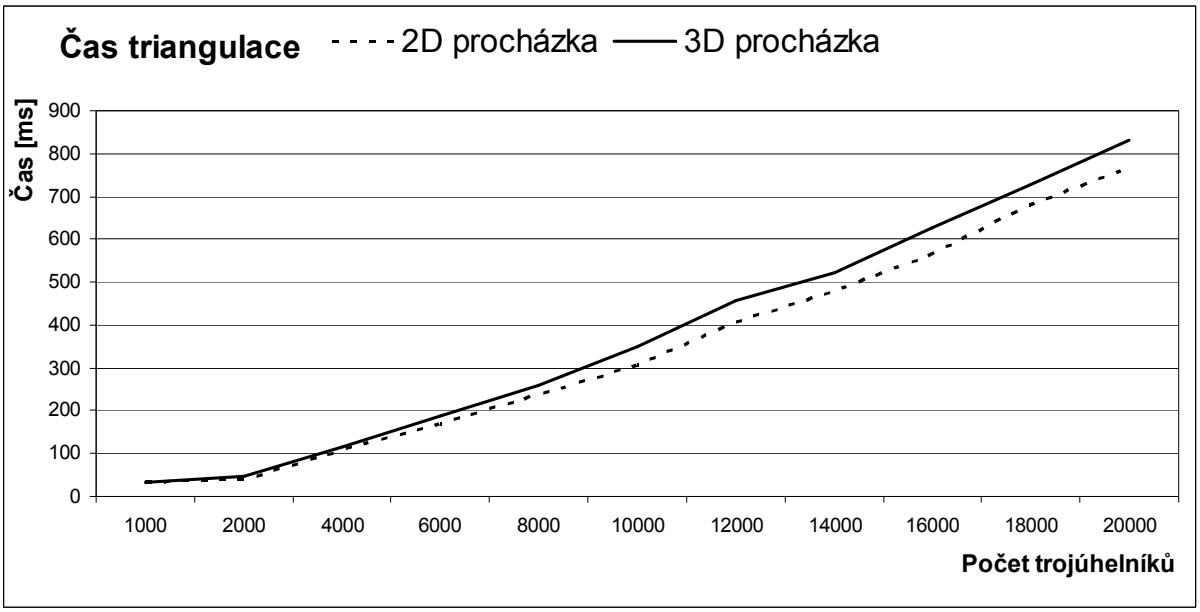
5. Výsledky experimentů

5.1 Porovnání procházky v polárních souřadnicích s procházkou ve 3D

V průběhu upravování procházky pro využití na povrchu koule jsme narazili na zajímavý problém. Je rychlejší procházka ve třech rozměrech, nebo je rychlejší mít body převedené do polárních souřadnic a používat 2D procházku? První možnost si ušetří převod bodů do sférických souřadnic, který není úplně jednoduchý (jsou použity goniometrické funkce – kap. 2.3.1), a zároveň teoreticky kratší cestu. Okolo pólů totiž procházka ve sférických souřadnicích zvolí delší cestu. Stejně tak není schopná přejít šev na přelomu úhlu φ. Procházka ve 2D zase nabízí jednodušší test hrany oproti bodu (determinant matice 2x2 oproti 3x3 v prostoru.

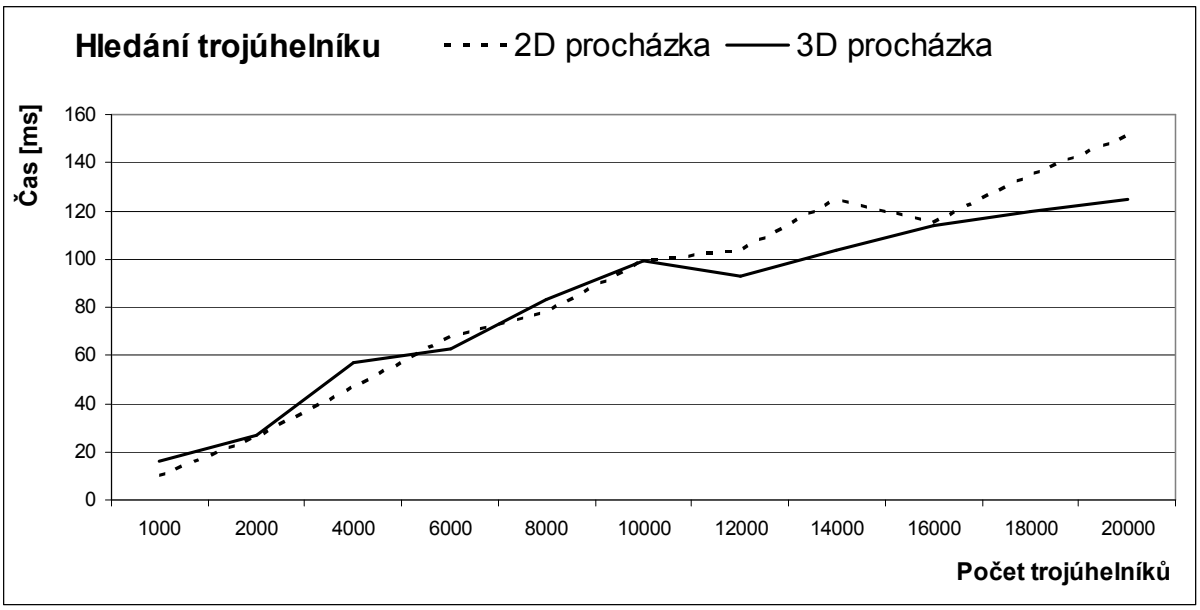
Testován byl čas triangulace koule. Potom bylo na této kouli spuštěno 2000x hledání trojúhelníku a odečten čas hledání a počet trojúhelníků prošlých procházkou. Testování probíhalo na kouli vytvořené z 1000-20000 trojúhelníků. Pro každou metodu a složitost trojúhelníkové sítě byl test spuštěn alespoň třikrát a výsledkem je průměr z těchto tří hodnot.

Jak se ukázalo, čas konverze koordinátů do sférických souřadnic byl i pro 10000 bodů prakticky zanedbatelný. To naznačuje, že výsledky 2D procházky budou lepší.

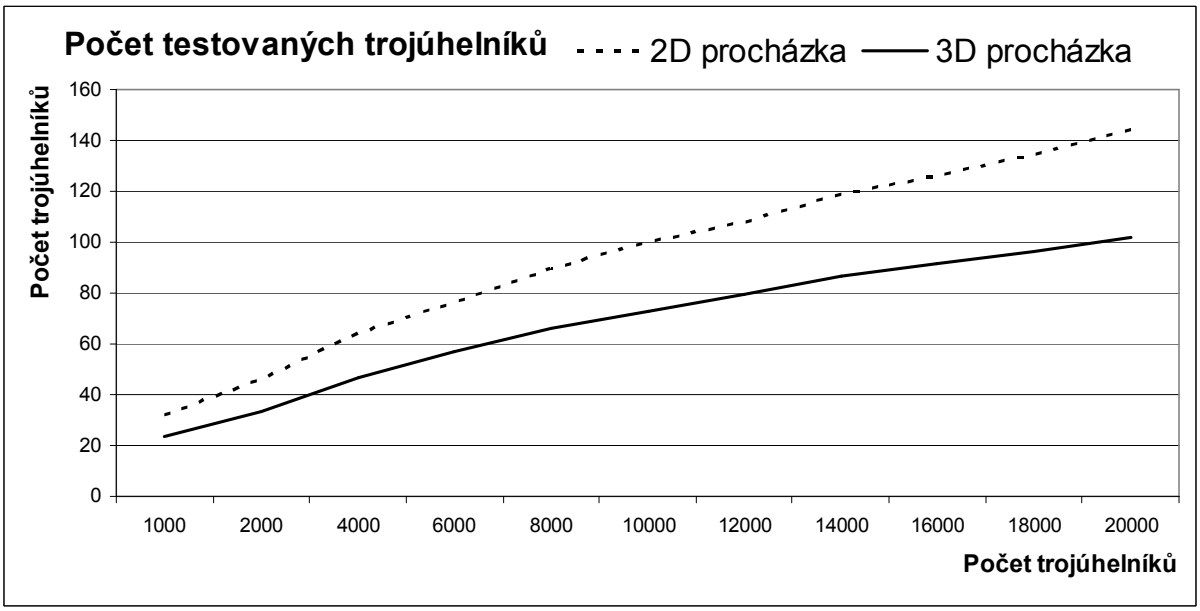


Graf 5.1.1 – Čas triangulace koule.

Z grafu 5.1.1 je vidět, že počítání jednoduššího determinantu je znát i ve výsledné aplikaci. Proto také 2D procházka vychází rychlejší.



Graf 5.1.2 – Čas vyhledávání trojúhelníku pro 2000 náhodně generovaných bodů.



Graf 5.1.3 – Počet trojúhelníků průměrně testovaných pro jedno hledání.

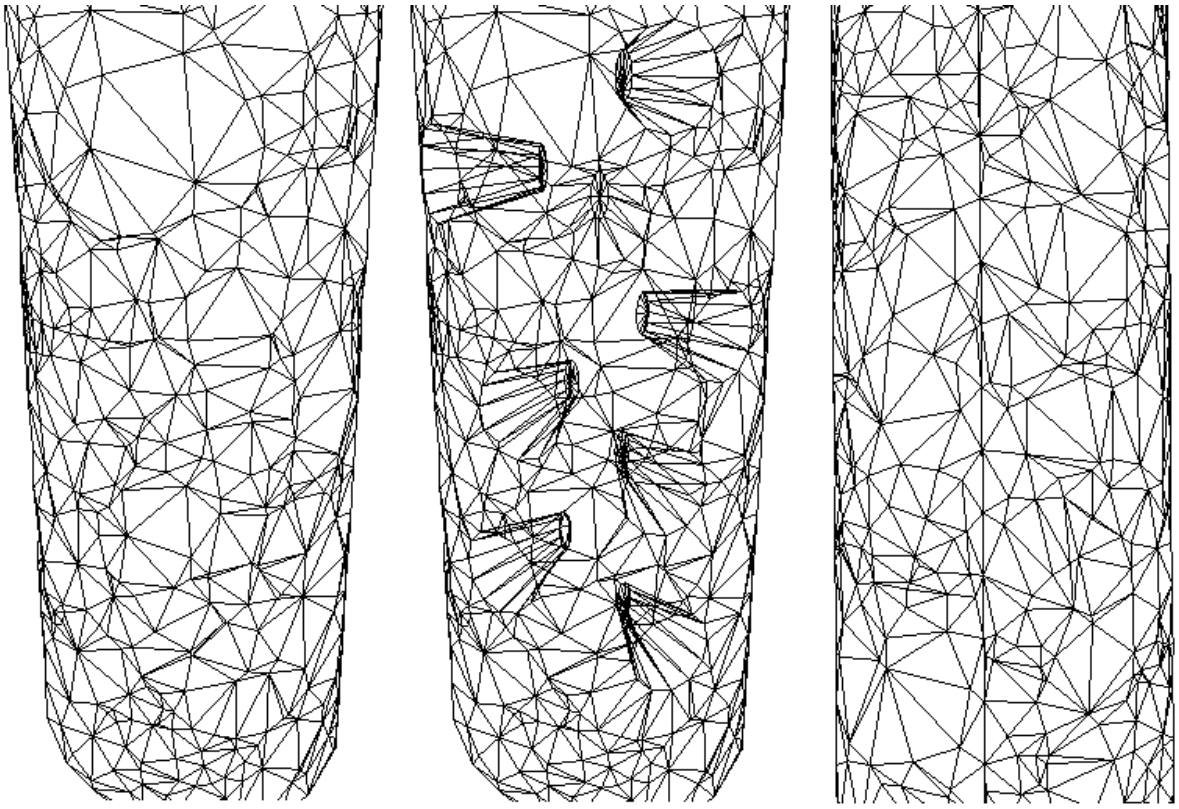
V grafu 5.1.2 se projevuje náhodnost při vybírání prvního trojúhelníku pro hledání. Zároveň je zde taky vidět nepřesné měření času. Proto se výsledky se zvyšující složitostí sítě ne vždy zvyšují. V průměru je ale vidět, že 3D procházka se začne asi kolem 10000 trojúhelníků vyplácet. Důvod pro toto chování je vidět v grafu 5.1.3 3D procházka totiž podle očekávání prochází menší počet trojúhelníků. Tím by mohla být triangulace s touto procházkou pro velký počet trojúhelníků dokonce rychlejší, i když to podle grafu 5.1.1. vypadá jinak.

5.2 Výsledky triangulací

5.2.1 Výsledky použití transformací

Výsledky transformace vypadají přesně podle předpokladů z kapitoly 2.3.1 a 2.4.1 (obr. 5.2.1).

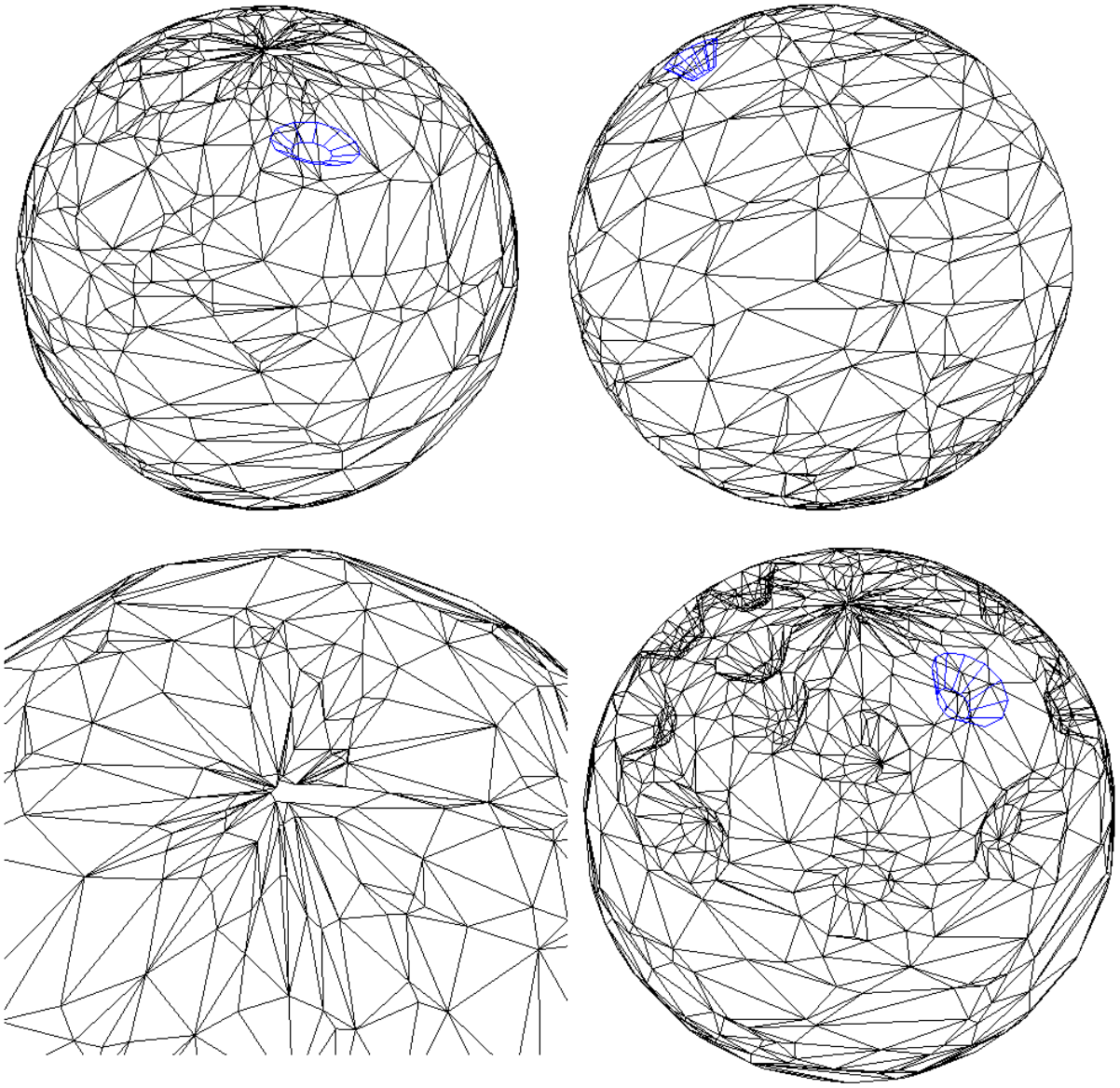
Válec je použitelný bez problémů všude, kromě oblasti švu. Tam jsou problémy s triangulací. Šev také narušuje funkčnost virtuálního nástroje. Ta se dá obnovit, pokud by byly akce nástroje v oblasti švu hodnoceny zvlášť jako speciální případ.



Obr. 5.2.1 – Ukázka triangulace válce vytvořeného náhodnými body. Vlevo – triangulace s druhé strany, než je šev. Uprostřed – triangulace s několikrát obtisknutým nástrojem. Vpravo – Triangulace se švem.

Na obrázku 5.2.1.1. je ukázka válce vytvořeného z náhodných bodů. Na prostředním obrázku je vidět další problém této jednoduché transformace. Protože na nástroj se používá stejná metoda transformace, jako na celý válec, je tento nástroj směrem ke středu válce zdeformován.

Triangulace koule pomocí transformace je ještě problémovější než u válce (obr. 5.2.2). K problému švu se ještě přidává deformace geometrie okolo pólů koule. Proto jsou oproti válci možnosti jejího využití velmi omezené.



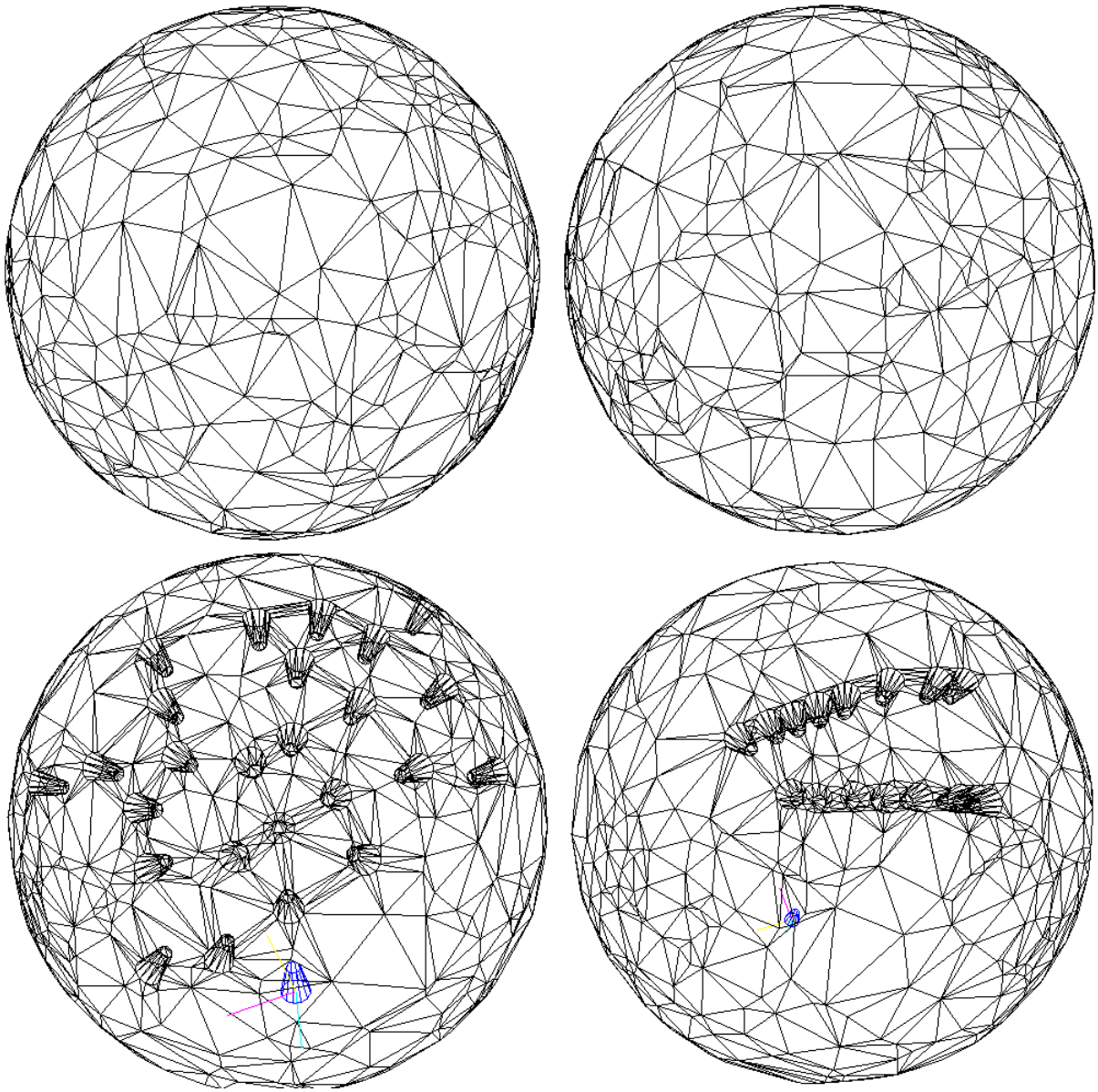
Obr. 5.2.2 – Ukázka triangulace koule vytvořené náhodnými body. Vlevo nahoře – celkový pohled na koulie, Vpravo nahoře – viditelný šev. Vlevo dole – detail trojúhelníků kolem pólu. Vpravo dole – několik ukázkových obtisků nástroje.

Na obrázku 5.2.2 vlevo nahoře a vpravo dole je možné vidět problém deformace nástroje, který je ještě výraznější než u válce.

Po zhodnocení výsledků triangulace koule a válce pomocí transformace bylo jasné, že tato metoda asi nebude pro naše potřeby ideální. Proto jsme přešli ke zcela jiným možnostem triangulace na povrchu těchto těles.

5.2.2 Triangulace na kouli

Při triangulaci podle algoritmu popsaného v kapitole 2.3.2 nebyly očekávány žádné problémy. Speciálně problém švu a deformace okolo pólů je vyřazením transformace do sférických souřadnic vyloučen.

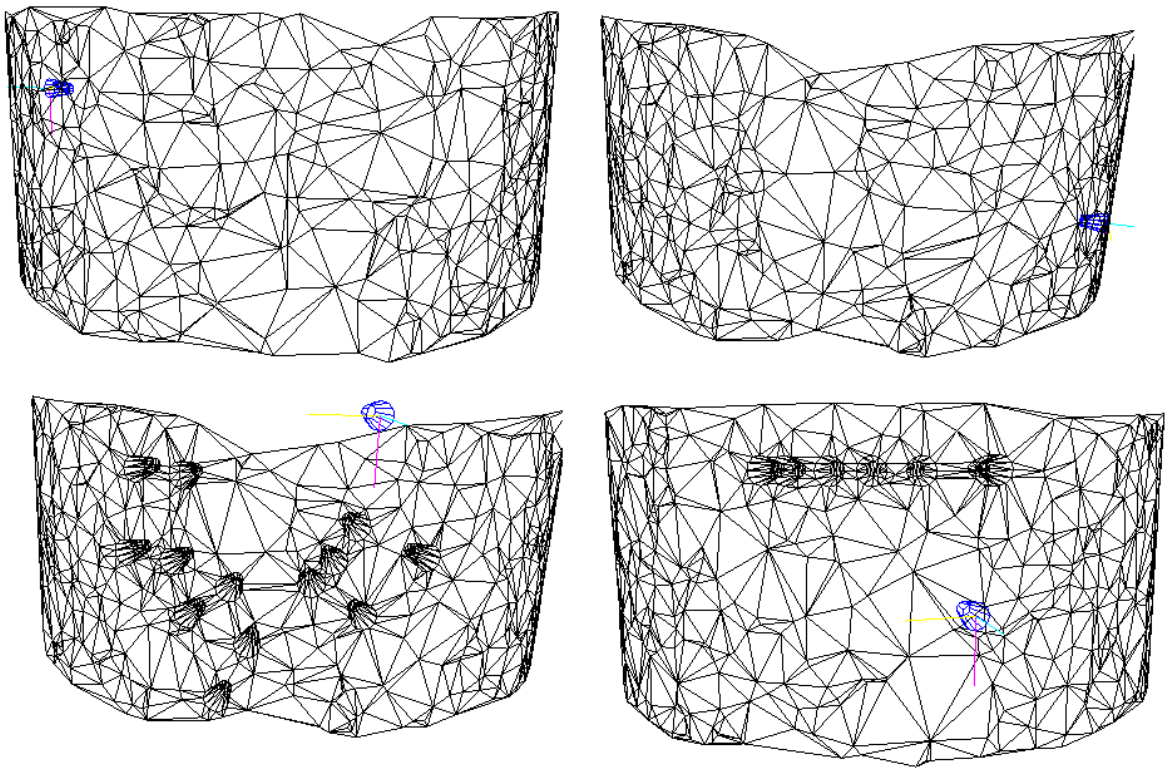


Obr. 5.2.3 – Triangulace na povrchu koule ve 3D.

To se také (jak je vidět na obr. 5.2.3) potvrdilo. Nejsou vidět žádné problémy při triangulaci z jakéhokoliv pohledu. Ani nedochází k deformaci nástroje. Jedinou nevýhodou tohoto řešení je větší pracnost úprav.

5.2.3 Triangulace na válci

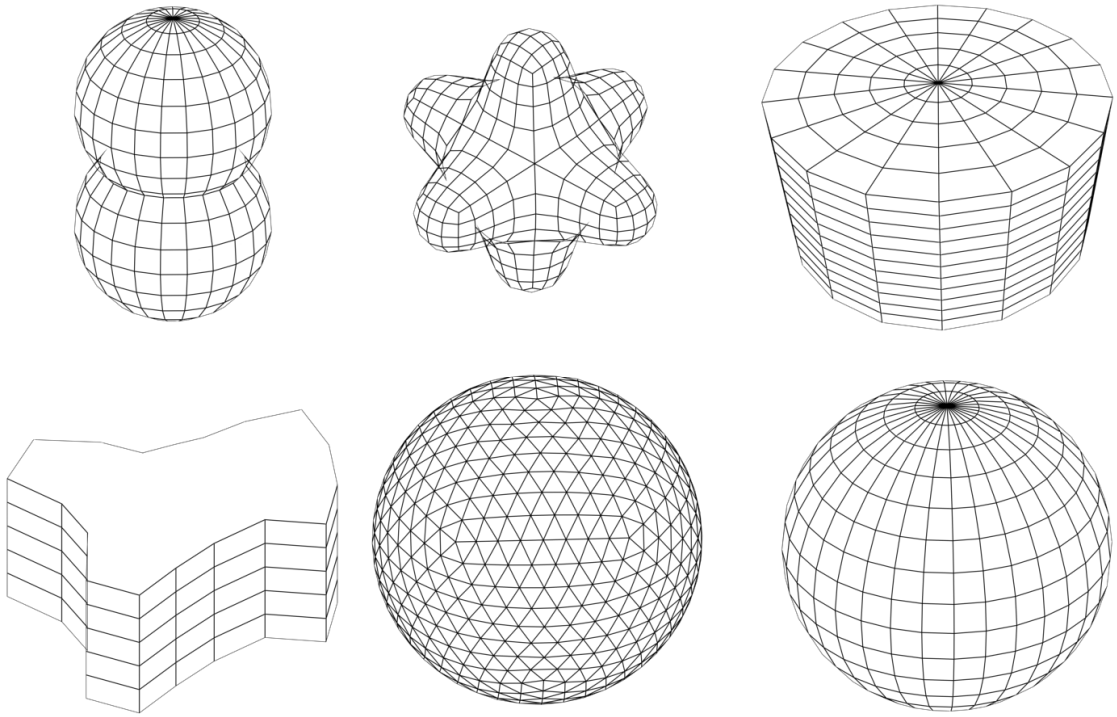
Při triangulaci na válci (popsána v kap. 2.4.2) jsme se bohužel použití polárních souřadnic nevyhnuli. Pořád jsou použity pro rozhodnutí o Delaunayově kritériu. Tam je ale problém přechodu úhlu φ z 0 na 2π vyřešen, takže také nedošlo k žádným problémům (obr. 5.2.4).



Obr. 5.2.4 – Triangulace na povrchu válce ve 3D.

5.2.4 Triangulace obecných těles

Zde již nejsou výsledky tak jednoznačné. Testovali jsme několik těles (obr. 5.2.5) a několik kritérií. Zároveň byly testovány dvě metody. První s počáteční triangulací vytvořenou ze vstupních bodů a body seřazenými tak, aby se vždy vkládal bod nejvzdálenější od všech vložených bodů (*FreeFormProjector*). Druhá metoda má počáteční triangulaci definovanou jako trojúhelník mimo všechny vstupní body a body seřazené od nejvzdálenějšího od tohoto trojúhelníku (*SortedFreeFormProjector*). Obě tyto metody jsou popsány v kapitole 2.5.

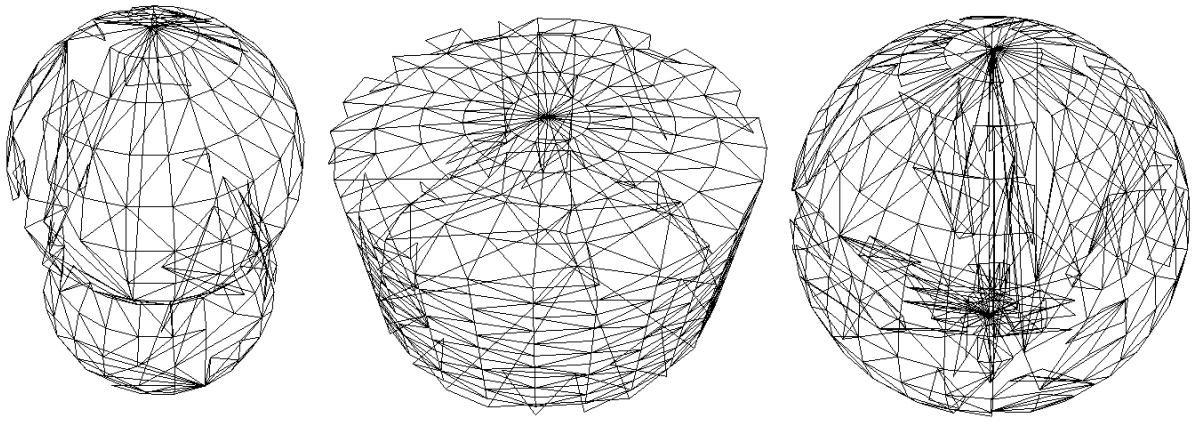


Obr. 5.2.5 – Tělesa použitá pro testování triangulace.

5.2.5 FreeFormProjector

Spolu s tímto způsobem triangulace jsme zároveň testovali několik možných kritérií pro triangulaci ve 3D. Testovali jsme kritéria popsaná v kapitolách 2.1.3 a 2.5.3 Konkrétně šlo o maximalizaci minimálního úhlu, minimalizaci délky hran, minimalizace úhlu mezi normálami trojúhelníků a minimalizace obsahu ploch trojúhelníků. Všechny testy byly provedeny se zpožděnou legalizací, kdy se legalizace začala provádět až po vložení n/6 bodů, kde n je počet vstupních bodů.

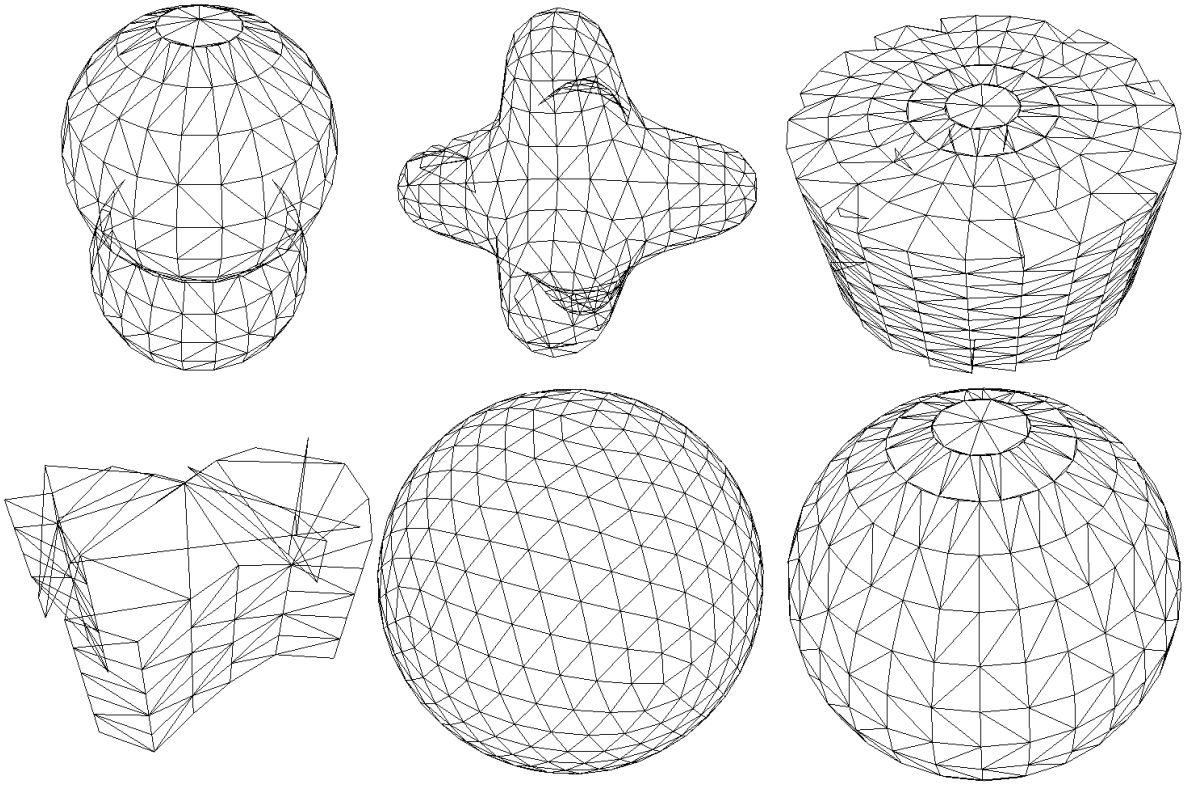
Minimalizace maximálního úhlu



Obr. 5.2.6 – Výsledky minimalizace maximálního úhlu

Toto kritérium je asi nejvíce podobné hodně používanému Delaunayovu kritériu prázdné kružnice. Přesto se pro naše účely naprosto nehodí. Pomocí tohoto kritéria se nepodařilo zrekonstruovat ani kouli, která patřila k nejjednodušším tělesům (obr. 5.2.6).

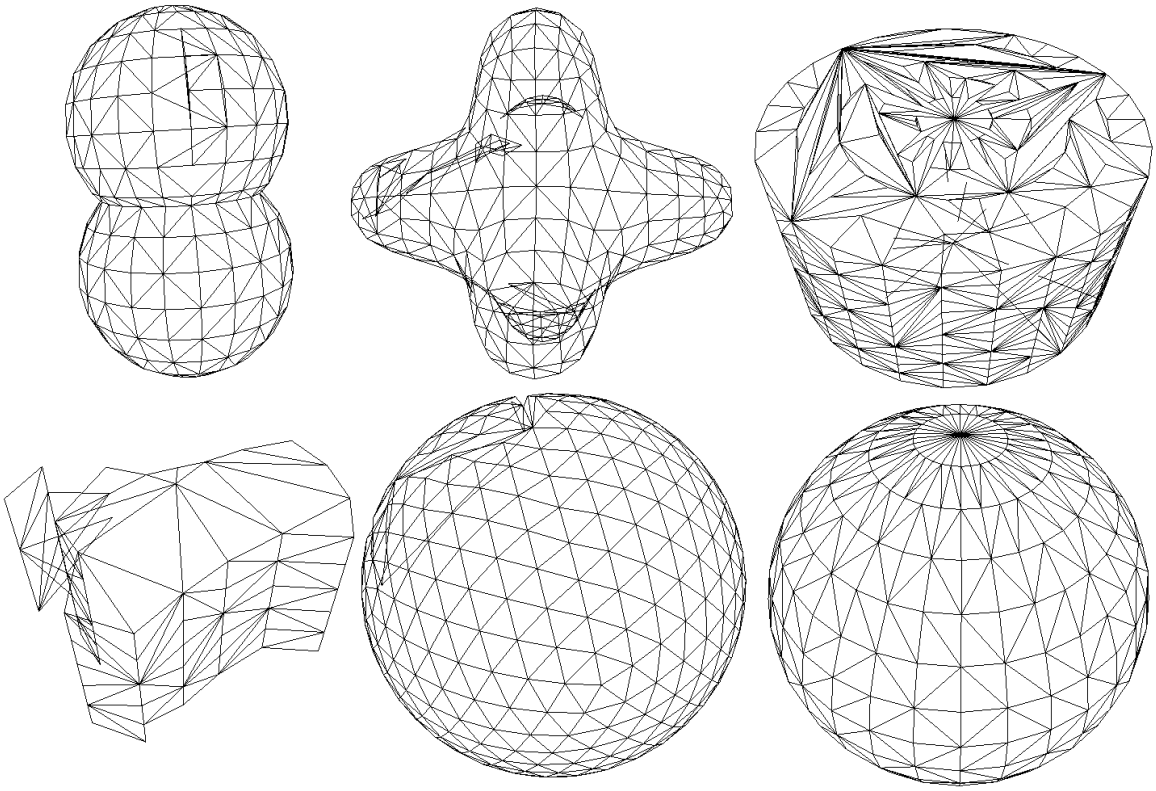
Minimalizace součtu délek hran trojúhelníků



Obr. 5.2.7 – Výsledky minimalizace součtu délek hran

Toto kritérium je velmi vhodné pro triangulaci ve 3D. Například kouli tvořenou rovnostrannými trojúhelníky se podařilo rekonstruovat naprosto bez chyby. Na ostatních tělesech jsou vidět vetší čí menší chyby (obr. 5.2.7).

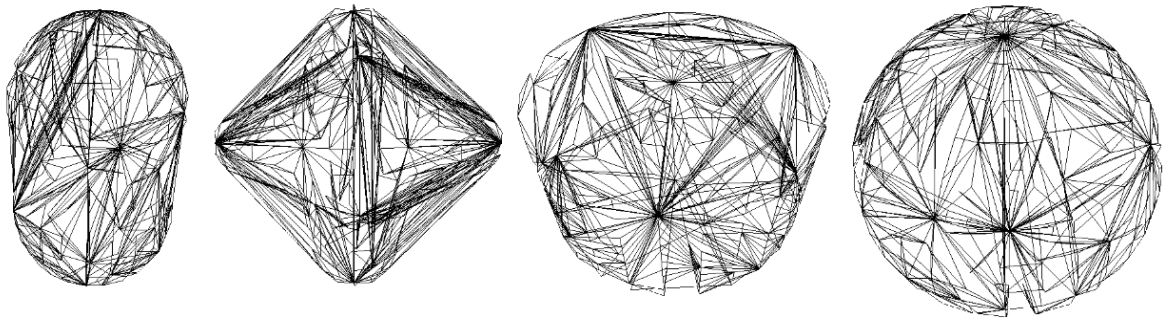
Minimalizace úhlu mezi normálami trojúhelníků



Obr. 5.2.8 – Výsledky minimalizace úhlu mezi normálami trojúhelníků

Opět velmi dobře použitelné kritérium. Obyčejnou kouli zvládá lépe než předchozí kritérium. Naproti tomu na kouli složenou z rovnostranných trojúhelníků je vidět problém. Válec má tvar bližší originálu, ale výsledné trojúhelníky nevypadají dobře (obr. 5.2.8).

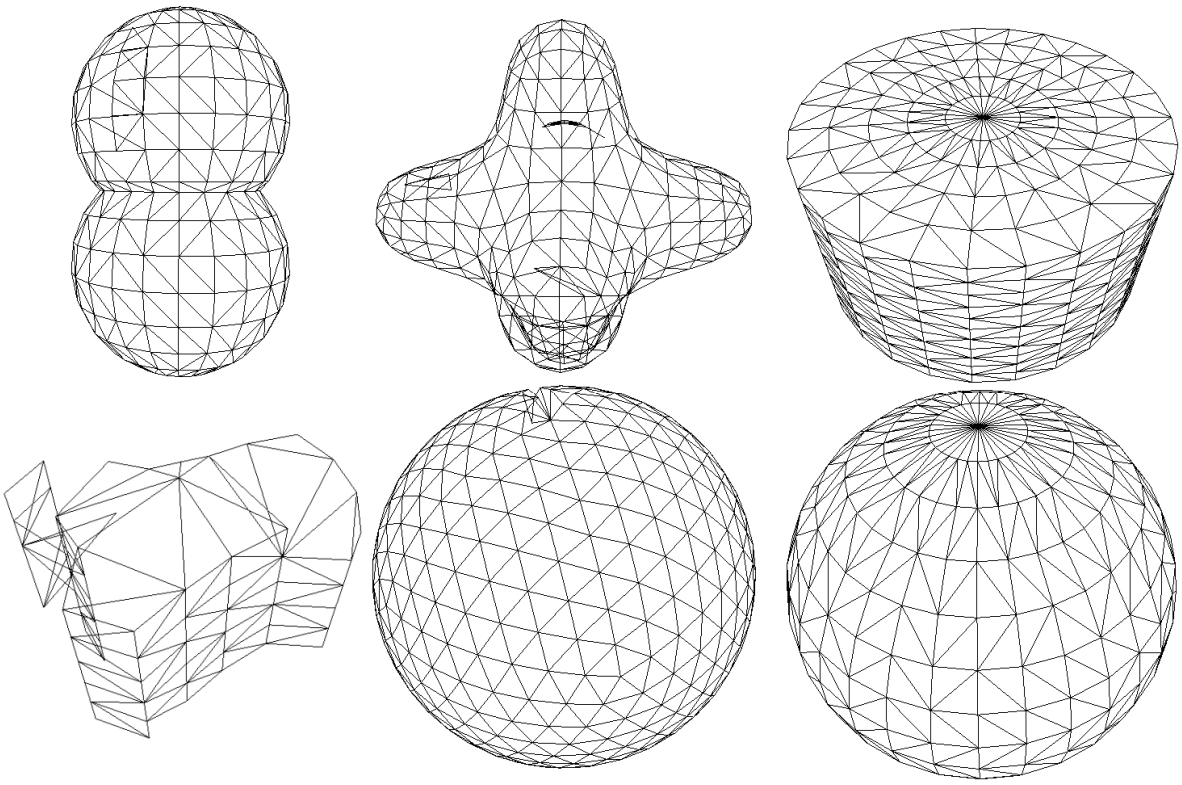
Minimalizace obsahu ploch trojúhelníků



Obr. 5.2.9 – Výsledky minimalizace obsahu ploch trojúhelníků

Toto kritérium je jednoznačně nejhorší ze všech testovaných. Pomocí něj se nepodařilo zrekonstruovat ani částečně žádný objekt. (obr. 5.2.9)

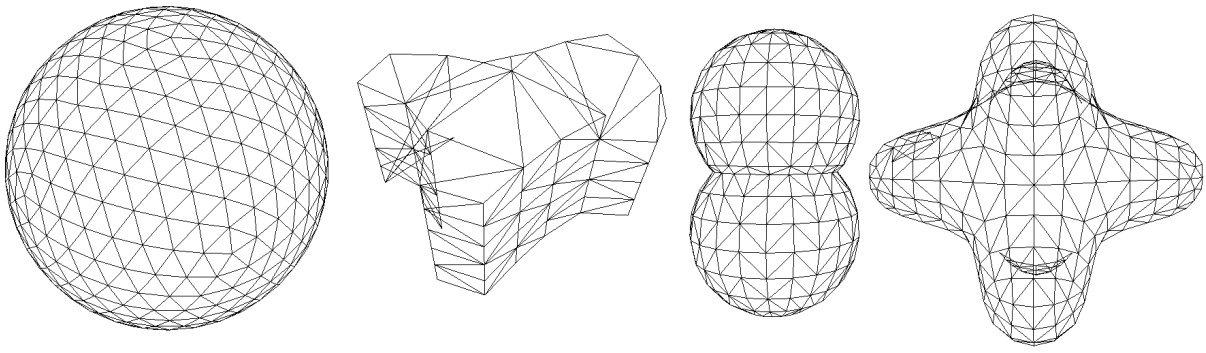
Kombinace minimalizace úhlů mezi normálami a maximalizace minimálního úhlu



Obr. 5.2.10 – Výsledky kombinovaného kritéria

Tato triangulace využívá obě kritéria. Jako hlavní kritérium používá úhel mezi normálami trojúhelníků. Pokud se tento úhel blíží k nule (oba dva trojúhelníky leží téměř na ploše), použije kritérium pro maximalizaci minimálního úhlu. Tím zůstanou výhody původního kritéria. Navíc ve sporných momentech (rozdíl je vidět hlavně na podstavě válce) využije kritérium původně navržené pro práci v rovině. Takto zkombinováno dává toto kritérium obecně asi nejlepší výsledky (obr. 5.2.10).

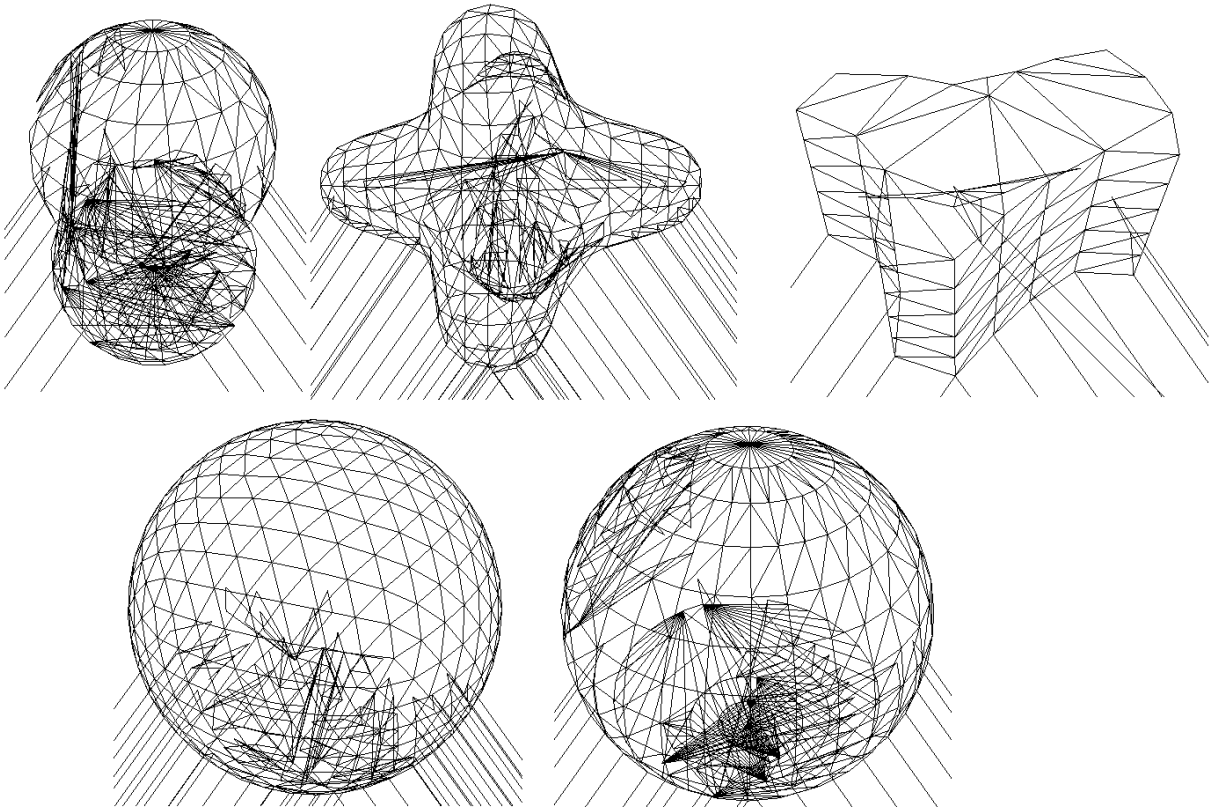
Změnou nastavení počtu bodů, po kterém se spustí legalizace, lze dosáhnout ještě lepších výsledků (obr. 5.2.11).



Obr. 5.2.11 – Výsledky dosažené změnou zpoždění legalizace

5.2.6 SortedFreeFormProjector

Protože se kombinované kritérium v předchozí kapitole ukázalo jako nejlepší, testovali jsme jenom pomocí tohoto kritéria (obr. 5.2.12).

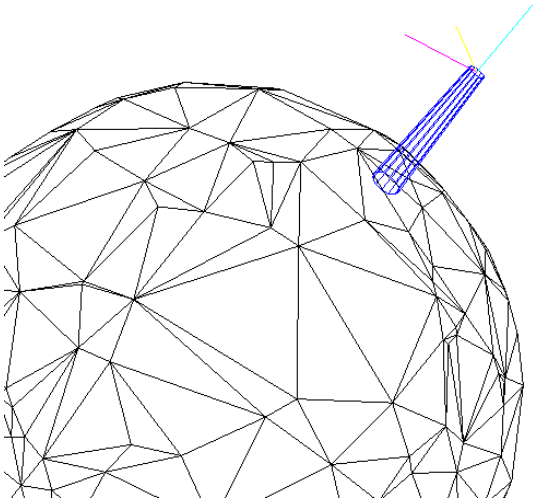


Obr. 5.2.12 – Výsledky triangulace pomocí SortedFreeFormProjector. Přímky vedoucí ke krajům jsou hrany vedoucí k počátečnímu trojúhelníku.

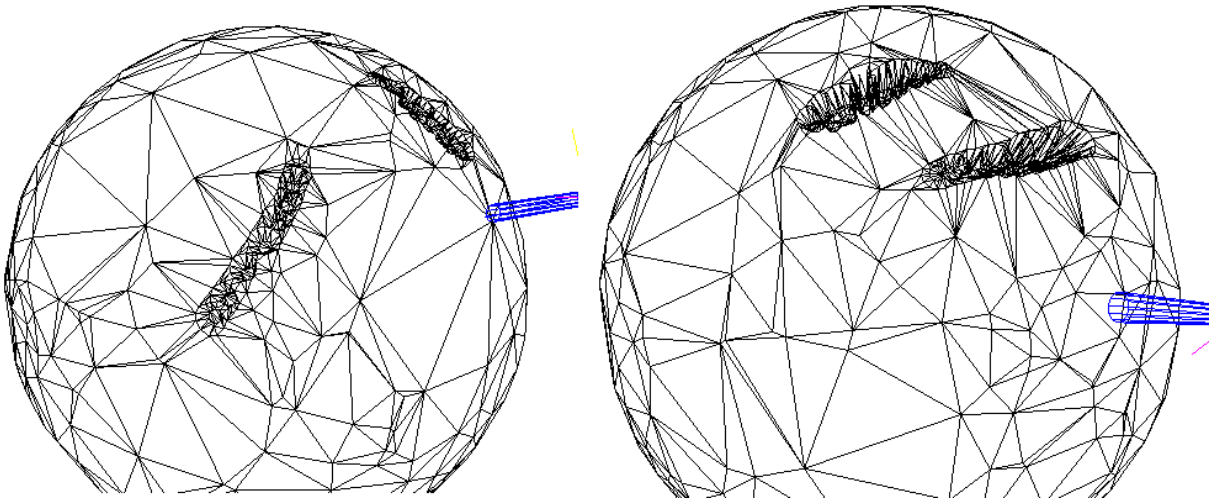
Tento přístup se nezdá zdaleka tak použitelný, jako předchozí. Velké problémy mu způsobují trojúhelníky natočené směrem k původnímu trojúhelníku. Na kouli jsou vidět artefakty i ze strany od tohoto trojúhelníku odvrácené. Naproti tomu si tento přístup jako jediný poradil s nepravidelným tělesem (na obrázku nahoře vpravo).

5.3 Výsledky použití haptického zařízení

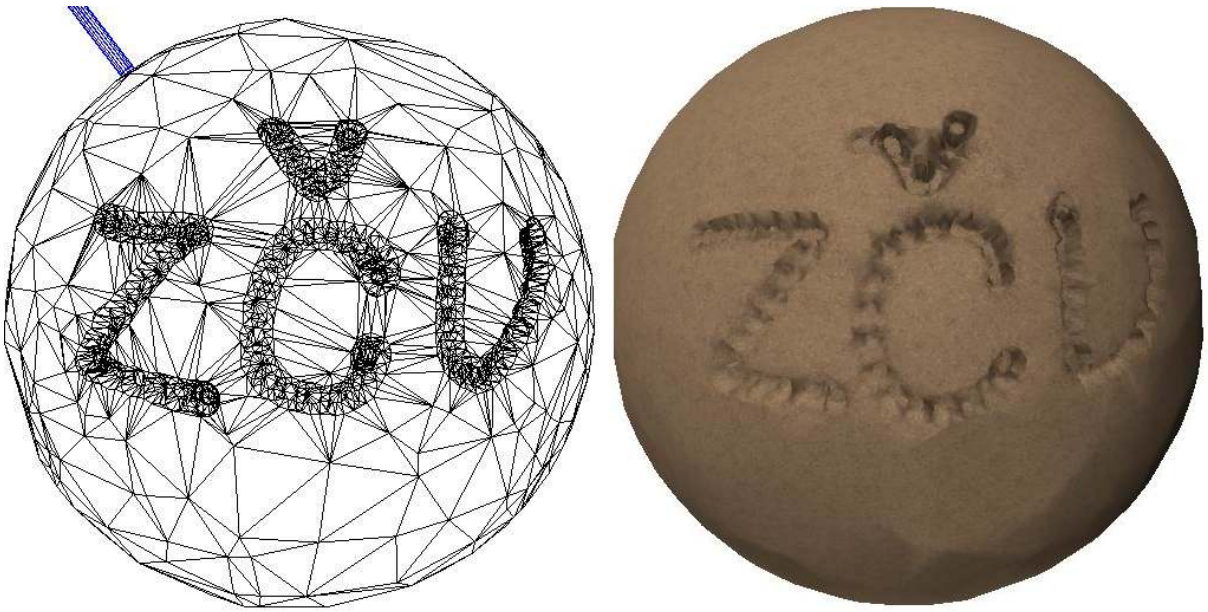
Haptické zařízení se podařilo bez větších problémů zprovoznit (obrázky 5.3.1, 5.3.2 a 5.3.3). Díky odporu kladenému zařízením bylo znát, když hrot zasáhl do tělesa (koule). Nicméně je zde ještě velký prostor pro vylepšení jak samotného rytí, tak i výpočtu odporových sil. Použitý model je prakticky nejjednodušší možný a například vůbec nereaguje na změny sítě, ale pořád používá triangulační těleso (koule nebo válec).



Obr. 5.3.1. – Nástroj je mimo kouli. Vnější okraj nástroje je neustále promítán na trojúhelníkovou síť.



Obr. 5.3.2. – Několik pokusných tahů deformujících povrch koule



Obr. 5.3.3. – Jeden z posledních výsledků

6. Závěr

6.1 Triangulace na povrchu trojrozměrných těles

Pokud se zaměříme pouze na povrch koule a válce, tak jsou výsledky velmi slušné. Triangulace a používání nástroje je téměř bezproblémové. Nicméně by pořád šlo zapracovat na stabilitě aplikace. Dále změnou všech výpočtů tak, aby bylo možné je použít ve 3D, došlo ke zpomalení celé aplikace, což by se také dalo ještě vylepšit.

Pokud jde o triangulaci složitějších objektů, dosáhli jsme částečného úspěchu. Výsledky některých metod se celkem blíží skutečným tělesům, ale tyto výsledky jsou velmi závislé na použitých parametrech. Zároveň jsou výsledky závislé na vstupním tělese. Proto se prakticky nedá mluvit o univerzálním řešení pro triangulace těles. Na druhou stranu jde o první pokus o využití triangulace tímto směrem, takže se určitě nedal čekat dokonalý výsledek.

Výsledky odpovídají upravenému zadání, kdy šlo o první pokusy s triangulací složitějších objektů, než je plocha. Triangulace na kouli a válci jsou, až na některé detaily, připraveny k použití v aplikaci.

6.2 Použití haptického zařízení pro ovládání

Většinu práce s připojením haptického zařízení měl na starost V. Purchart. Ten upravil aplikaci tak, aby bylo možné používat toto zařízení pro rytí v ploše. Mou prací bylo potom toto opět rozšířit pro rytí na kouli a válci. Po provedení několika změn se toto nakonec podařilo. Výsledný dojem z používání zařízení je mnohem lepší, než s použitím pouze myši. Jedním důvodem jsou odporové síly. Člověk okamžitě ucítí, kdy se nástroj dotkne tělesa. Samozřejmě pro rytí v prostoru, o kterém je celá tato práce, je takové zařízení mnohem vhodnější.

V tomto bodu se zadání podařilo splnit, i přes minimum času (asi jeden měsíc) na úpravy a experimenty.

7. Použité zdroje

[Kad07] KADLEC, J., Deformace terénu pro virtuální realitu – geometrická část modelu, *Bakalářská práce* KIV/ZČU, 2007

[Pur07] PURCHART, V., Deformace terénu pro virtuální realitu – datová, logická a vizualizační část modelu, *Bakalářská práce KIV/ZCU*, 2007
herakles.zcu.cz/~skala/MSc/Diploma_Data/2007_BP_Purchart_Vaclav.pdf

[Pur09] PURCHART, V., Modelování písčitého terénu pro virtuální realitu, *Diplomová práce KIV/ZCU*, 2009

[Sed07] SEDMIHRADSKÝ, J., Modelování eroze a deformací terénu, *Diplomová práce KIV/ZČU*, 2007

[VAM07] KOLINGEROVÁ, I., Rovinné triangulace a jejich aplikace,
přednášky KIV/VAM
http://iason.zcu.cz/~kolinger/VAM/VAM7.zip

[1] ATTENE, M., FALCIDIENO, B., SPAGNUOLO, M., WYVILL, G., Mapping independent triangulation of parametric surfaces, *Shape Modeling International*, 2002

[2] RENKA, R., J., Algorithm 772: STRIPACK: Delaunay Triangulation and Voronoi Diagram on the surface of a sphere. 1997

[3] DEVILLERS, O., PION, S., TEILLAUD, M., Walking in a Triangulation, *International Journal of Foundations of Computer Science*, 2002

[4] Tetrahedron Circumsphere, článek elektronicky publikovaný pomocí systému Wikipedia, *http://www.cgafaq.info/wiki/Tetrahedron_Circumsphere*

8. Přílohy

8.1 Postup prohazování hran pro vynucenou hranu (kap. 2.2.1)