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

Bakalářská práce

Detekce a optimalizace zajímavých oblastí v digitálním modelu terénu

Plzeň, 2014 Ondřej Byrtus

Abstract

This work explores possibilities in the area of altering triangulated irregular networks. While fast, edge swapping offers little flexibility in terms of getting stuck in nearest local extreme. The Simulated Annealing solves mentioned issue at the cost of high complexity. This work focuses on mid-step by altering network by groups of three and four triangles in hope of offering better results than edge swapping at computationally low cost.

Abstrakt

Tato práce zkoumá možnosti v oblasti úprav nepravidelných trojúhelníkových sítí. Ačkoli má prohazování hran nízkou výpočetní náročnost, tato metoda často uvázne v lokálním extrému. Simulované žíhání řeší tento problém za cenu vysoké výpočetní náročnosti. Tato práce se snaží najít kompromis mezi zmíněnými způsoby pomocí úprav skupin tří a čtyř trojúhelníků. Cílem je dosažení lepších výsledků než prohazováním hran a to se stále přijatelnou výpočetní náročností.

Obsah

Prohlášení

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

V Plzni, dne ..... Ondřej Byrtus

1 Úvod

Pro digitální reprezentaci terénu se praxi často používají buď čtvercové anebo nepravidelné trojúhelníkové sítě. Sítě čtvercové se snáze implementují, jejich nevýhodou je však vyšší paměťová náročnost pro dosažení vysokého detailu. Nepravidelné trojúhelníkové sítě se lépe přizpůsobují terénu a díky tomu dovolují vykreslit terén detailněji při stejné spotřebě paměti. Tato bakalářská práce se zabývá úpravami trojúhelníkové sítě digitalizovaného terénu.

Ne všechny sítě jsou vhodně triangulované pro výpočet vrstevnic a proto je nutné je následně upravit. Nejrozšířenější jsou dvě možnosti, a to simulované žíhání a prohazování diagonál na dvojici sousedních trojúhelníků. Každé z těchto řešení má však svou negativní stránku. Zatímco simulované žíhání je výpočetně náročné, prohazování diagonál nabízí pouze lokálně optimální řešení. Cílem této bakalářské práce je nalezení řešení a vytvoření aplikace, která nalezne určitý kompromis mezi těmito dvěma řešeními a nabídne řešení třetí, a to takové, které neuvízne v lokálním optimu a zároveň nedosáhne výpočetní náročnosti simulovaného žíhání.

Aplikace byla vytvořena za odborného dozoru vedoucí bakalářské práce Doc. Dr. Ing. Ivany Kolingerové z katedry informatiky a výpočetní techniky Západočeské univerzity v Plzni.

Obsah práce je následující:

- kapitola 2 Teoretická část - popisuje existující způsoby triangulace terénu, vhodnost pro výpočet vrstevnic a úprav sítě, v závěru jsou uvedeny a krátce vysvětleny použité algoritmy
- Kapitola 3 - Navrhované řešení - se věnuje kritériím úprav sítě, iterací sítí, sestavováním a testováním způsobů triangulace skupin 3 a 4 trojúhelníků stejně tak, jako návrhem a implementací aplikace
- kapitola 4 - Experimenty - se věnuje průběhu testování, popisuje experimenty a vyhodnocuje jejich výsledky
- kapitola 5 -Závěr - rekapituluje stanovené cíle a zkoumá, nakolik byly splněny

2 Teoretická část

Trojúhelníková síť je jednou z nejrozšířenějších a nejčastěji používaných digitálních reprezentací terénu. Dalším možným způsobem, který lze využít k popisu terénu, jsou vrstevnice – linie bodů se stejnou (nadmořskou) výškou, která je obvykle uchována jako z souřadnice vrcholů sítě. Ať již se použije jakýkoliv způsob, hlavní myšlenka zůstává stejná - sestavit funkci pro výšku daného bodu v terénu $z = f(x, y)$.

2.1 Trojúhelníková síť

Vrcholem v trojúhelníkové síti nechť je bod v terénu se známou polohou v rámci sítě a (nadmořskou) výškou, který byl vstupem triangulačního algoritmu. V bodech, kde se nacházejí vrcholy sítě, zpravidla předpokládáme nulovou chybu interpolace.

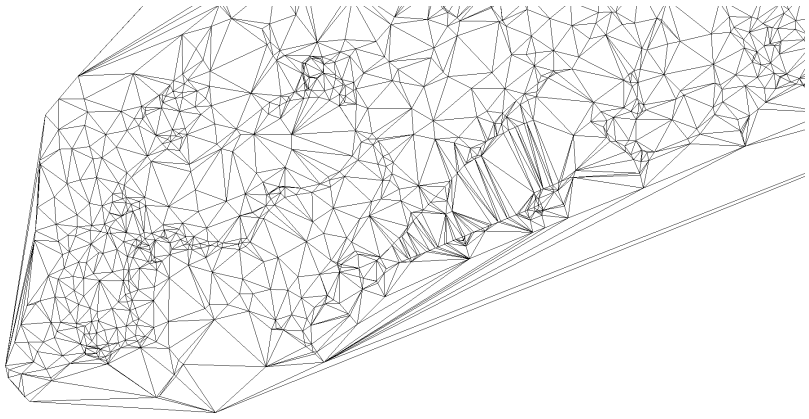
Hranou je pak spojnice mezi dvěma vrcholy sítě, jež neprotíná žádnou jinou hranu. Z předchozí definice vyplývá, že hrany se neprotínají v místech jiných než jsou vrcholy sítě. Trojice hran pak tvoří jeden trojúhelník sítě a každá vnitřní hrana (tzn. ta, která neleží na okraji sítě) je členem dvou trojúhelníků sítě. Je obecně známo, že celkový počet hran a trojúhelníků sítě nezávisí na použitém triangulačním algoritmu.

Samotná trojúhelníková síť [12] se sestává z množiny sousedících a vzájemně se nepřekrývajících trojúhelníků interpolujících zadané body. K popisu se obvykle využívá množina všech vrcholů a hran.

Podkladem pro vznik sítě je množina bodů. Zde jsou rozšířené hlavně dvě možnosti – pravidelné a nepravidelné vzorkování dat z terénu. Pravidelné sítě bývají v literatuře označovány pojmem GRID a jak již název napovídá, jedná se o síť pravidelně rozmístěných vrcholů. Pro pravidelnou síť se zpravidla využívá čtvercového rozmístění bodů v síti, ale můžeme se zřídka setkat i s např. šestiúhelníkovým rozmístěním. Výhodou tohoto typu sítě je zejména jednoduchá samotná triangulace bodů a stejně pravidelný výsledek. Nevýhodou je pak absence flexibility – nemožnost reagovat na oblasti vyžadující vyšší přesnost a nebo oblasti kde je vyžadováno vzorkování v určitých místech (např. linie hřebenu), což pak může způsobit nedokonalou interpolaci a nebo dokonce absenci zmíněného detailu.

Nepravidelná síť pak bývá označována jako Triangulated Irregular Netowork – TIN a jak z názvu vyplývá, rozmístění jejích vrcholů není pravidelné. Výhodou nepravidelné sítě je schopnost popisovat terén s proměnlivou přesností. Ačkoliv je triangulace této sítě složitější, přináší také jisté výhody - např. na rovném úseku není potřeba tolik vrcholů na přesnou reprezentaci, zatímco na zvlněné části terénu nebo blízko zlomu je koncentrace vrcholů

potřeba vyšší. Z toho dále plyne další výhoda – vyšší efektivita ukládání dat, není potřeba konstantní (vysoká) přesnost na celé síti pro zajištění podobné přesnosti modelu jako u husté pravidelné sítě. Příkladem by mohla být síť znázorňující úpatí hory a část jezera. Na části s rovnou hladinou jezera není potřeba tolik detailu jako na části s horou (nejedná-li se v této síti o dno jezera, samozřejmě). Nevýhodou je obtížnější manipulace, triangulace a její pozdější úpravy. Obrázek 2.1 je ilustračním příkladem zobrazení TIN.



Obrázek 2.1: Ilustrační ukázka TIN

Výše zmíněná množina vrcholů, ať již pravidelná nebo nepravidelná, ještě netvoří trojúhelníkovou síť. Proces převodu od množiny vrcholů na síť – množinu vrcholů a hran se nazývá traingularizací. Jednotlivé způsoby triangulace se od sebe liší a zpravidla se triangulovaný model vytváří s určitým cílem, vhodností, pro další využití sítě – globální kritérium, tzn. požadavek na výslednou síť. Ač možností, jak triangulovat konkrétní síť, je konečné množství. stále je to číslo velmi vysoké a tudíž metoda brute force při hledání optimální sítě nepřipadá v úvahu. Zde nastupují lokální kritéria, jež mají za úkol pomoci najít síť globálně optimální (pro účel), nebo alespoň síť blížící se optimu. Myšlenky jednotlivých algoritmů triangulace jsou povětšinou založeny na některém z lokálních kritérií.

Samotná lokální kritéria pak lze rozčlenit na kritéria zabývající se úhly v trojúhelníku – úhlová a na kritéria zohledňující délky hran - kritéria hranová. Úhlová kritéria bývají v praxi více rozšířená [6], a to díky výsledkům vyhýbajícím se příliš malým, velkým či „úzkým“ trojúhelníkům. Úhlová kritéria se zaměřují na maximalizování minimálního úhlu trojúhelníku (Delaunay triangulace - DT), minimalizování maximálního úhlu (tzn. minmax triangulace), minimalizování maximální excentricity, apod. Hranová kritéria se pak zaměřují na minimalizování sumy délek hran (triangulace s minimální va-

hou), její lokální aproximace, složená z nejkratších možných hran (Greedy triangulace -GT) a nebo minimalizování maximální délky hrany.

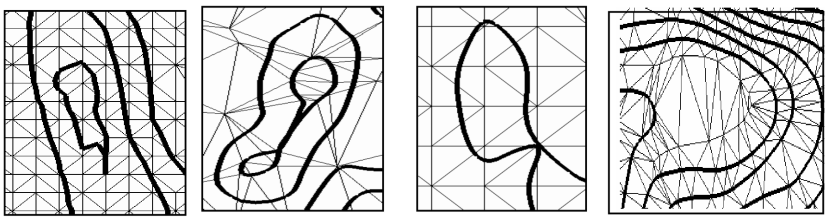
Jednou z nejčastějších metod triangulace sítě využívající úhlové kritérium je DT. Její výhodou je že produkuje „dobré“ trojúhelníky (tzn. blízké rovnostranným).

Hlavní myšlenkou greedy triangulace(GT) je vždy použít nejkratší hranu. Přesněji řečeno, algoritmus prochází množinu bodů sítě a vždy vytváří hranu trojúhelníku mezi dvěma nejbližšími nespojenými body a to takovou, že neprotíná žádnou z předešlých hran. Nevýhodou greedy triangulace je její vyšší složitost než u Delaunay triangulace, kterou srovnatelné výsledky s GT neospravedlňují a tudíž DT bývá upřednostňována před GT [6].

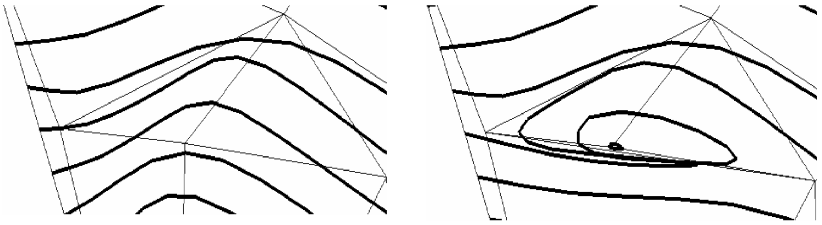
2.2 Vrstevnice

Vrstevnicí se obecně rozumí lomená čára na terénu spojující body se stejnou (nadmořskou) výškou - v případě trojúhelníkové sítě je to z souřadnice bodu. Výpočet vrstevnic na digitální reprezentaci terénu se většinou skládá z výpočtu průsečíku roviny paralelní s rovinou xy ve výšce z, která se pak „vyhladí“ pomocí interpolace bodů této linie. Na vrstevnicích mohou především vznikat tyto nedostatky (viz obr. 2.2, zleva a obr. 2.3):

- následující úsečky vrstevnice svírají hodně malý úhel
- vrstevnice sama se sebou v určitých oblastech splývá (na hřbetu terénu)
- vrstevnice se příliš přibližuje sama sobě v určitém bodě
- vrstevnice na rovinných oblastech neodpovídá tvaru terénu



Obrázek 2.2: Ukázky nedostatků na vrstevnicích [7]



Obrázek 2.3: Ukázka nedostatku na vrstevnicích (vpravo), oproti vhodnějšímu výsledku (vlevo) [7]

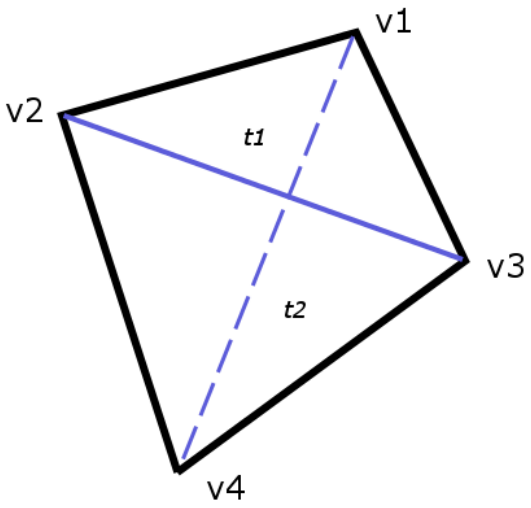
Přestože používané triangulizační metody optimalizují tvary výsledných trojúhelníků, nemusí výsledná trojúhelníková síť dávat vždy dobré vrstevnice. Problematická jsou například místa, kde je změna výšky terénu příliš velká. Takovými místy jsou například útesy, kde je změna výšky terénu velmi rychlá. Vzhledem k tomu, že v těchto místech jsou zpravidla potřeba trojúhelníky úzké a triangulační algoritmy se tomu typu trojúhelníků snaží vyhýbat, vznikají místa kde vrstevnice nemusí odpovídat skutečnosti (velká chyba interpolace). Dalším druhem problémových oblastí jsou oblasti skoro rovinné. V případě, že by přes skoro vodorovnou oblast měla procházet vrstevnice, i velmi malá změna triangulace v této oblasti způsobí velkou změnu tvaru vrstevnice.

Terénní schody také mohou být příčinou potíží. Jako terénní hranu lze označit linii, podél níž terén ostře (významně) mění svůj sklon. Za terénní hranu lze považovat i takovou, jejíž délka je nulová (např. vrchol nebo sedlo), nicméně tyto speciální případy nejsou tolik důležité pro tuto práci. V případě reprezentace terénu sítí lze za terénní hranu označit jakoukoli vnitřní hranu sítě, nicméně toto by v podstatě nemělo žádnou vypovídající hodnotu. Je tedy nutné definovat, co lze jako terénní hranu označit, a užít triangulační algoritmy, které terénní hrany respektují. Algoritmy s 2D kritérii sice vesměs zahrnutí terénní hrany umožňují (např. Constrained Delaunay Triangulation - CDT), ale lepší výsledky obvykle dosahují algoritmy zohledňující i souřadnici z (především datově závislé triangulace - DDT).

Jinou možností je využít výhod klasické rovinné triangulace a pouze se soustředit na předpokládané kritické oblasti – k tomu je potřeba provádět zásahy do již existující sítě. V souvislosti s tímto problémem je záhodno uvést lokální prohazování hran. Prohodit lze diagonály v konvexním čtyřúhelníku tvořeném dvěma sousedícími trojúhelníky sítě, obr. 2.4 znázorňuje obě možnosti využívající diagonály $t1$ nebo $t2$. Cílem lokálního prohazování hran je dospět k triangulaci s lokálním extrémem zvoleného kritéria – pokud by prohození přineslo zlepšení a je možné, prohazují se hrany tak dlouho, dokud dochází ke zlepšení sítě. Není možné prohodit hrany tak, aby triangulovaná

síť už neplnila svoji definici, jedná se o hrany, které by protínaly jinou hranu sítě, k tomu dochází u nekonvexního čtyřúhelníku.

Výhodou prohazování hran je jeho jednoduchost, rychlost a flexibilita. Nicméně i tato flexibilita je omezena na lokální kritéria. Metodu prohazování hran nelze použít na globální kritéria z důvodu, že vždy se rozhoduje pouze mezi dvěma možnostmi. Taktéž nelze aplikovat na některá kritéria a to zejména ta, která berou v potaz větší oblast než je čtyřúhelník. Toto omezení lze ovšem překonat za pomoci simulovaného žíhání.



Obrázek 2.4: Prohazování hran

Simulované žíhání [1] (Simulated Anneling – SA) je obecně využívaným algoritmem na optimalizaci dat, postupů, výsledků atd. - lze ho tedy využít i pro optimalizaci trojúhelníkové sítě dle zvoleného kritéria, v tomto případě na vylepšení interpolace terénu a výsledných vrstevnic ze sítě. Jedná se o pravděpodobnostní algoritmus - do sítě jsou prováděny zásahy, které se v případě, že by zhoršily celkovou váhu (kvalitu sítě), přijmou pouze s určitou snižující se pravděpodobností. Tento mechanizmus se snaží zabránit uvíznutí v některém z lokálních minim. Jeho další výhodou je univerzálnost a možnost aplikování globálních i složitějších kritérií a také to, že ze všech zatím zmíněných metod poskytuje výsledky nejbližší globálnímu optimu. Nevýhodou tohoto způsobu optimalizování sítě je jeho vyšší časová náročnost a nedeterminističnost.

Tato práce se zabývá možnostmi, jak řešit optimalizaci trojúhelníkové sítě bez nutnosti provádět simulované žíhání a přesto dosahovat výsledků lepších než při prostém prohazování hran. Hlavní myšlenkou je zapojení většího celku do optimalizace než je čtyřúhelník – pěti a šestiúhelníky by teoreticky mohly

dávat lepší výsledky díky většímu množství informací z okolí a přesto nedosahovat výpočetní náročnosti simulovaného žíhání.

2.3 Použité algoritmy

V této části jsou uvedeny a krátce vysvětleny algoritmy využité při vypracovávání práce, nicméně nesouvisejí přímo s problémem optimalizace trojúhelníkové sítě.

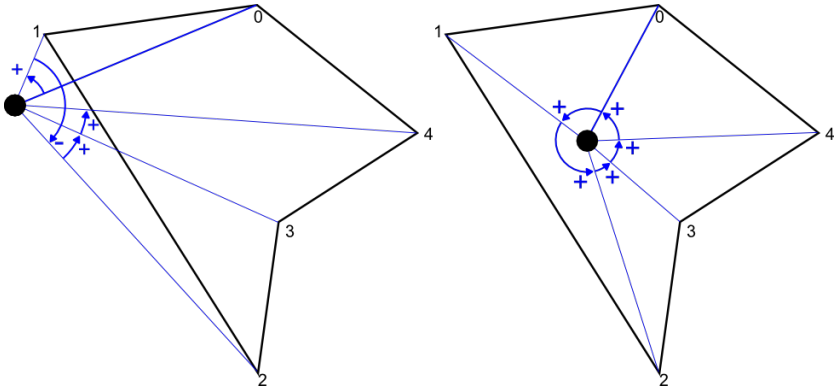
2.3.1 Znaménkový test

Vektorový součin dvou vektorů je využíván při výpočtu znaménkového testu, u kterého se porovnávají znaménka z souřadnice výsledku vektorového součinu. Znaménko z souřadnice výsledku v závislosti na směru vektoru přímky odpovídá na otázku, zda-li se nachází bod nalevo nebo napravo od vektoru přímky zadaného body (x_1, y_1) a (x_2, y_2). V případě, že se výsledek rovná nule, bod leží na přímce. Výsledná hodnota je kladná pro bod ležící nalevo od přímky, v opačném případě je výsledná hodnota záporná. Výsledek z znaménkového testu je :

$$z = \operatorname{sgn}(x_1 * y_2 - x_2 * y_1) \tag{2.1}$$

2.3.2 Winding number

Jedním z algoritmů používaným pro řešení, zdali bod p leží uvnitř 2D polygonu, je Winding Number (obr. 2.5)[2, 4] (WN). Alternativou k WN by mohl být algoritmus Crossing Number [2] (CN), nicméně WN podává intuitivnější výsledky u komplexních polygonů. Algoritmus WN počítá počet obtočení polygonu kolem bodu P. Bod je vně polygonu pouze v případě, že počet otočení (wn) je roven nule, jinak se tento bod nachází uvnitř. Na obrázku 2.5 je znázorněn případ pro bod uvnitř (vpravo) i vně (vlevo) polygonu. Znaménka udávají zda se jedná o "kladný"nebo "záporný"úhel, tzn. zda se jeho hodnota odečítá a nebo přičítá k wn.



Obrázek 2.5: Znázornění Winding Number

Obecněji lze definovat počet obtočení $wn(P, C)$ jakékoliv spojité uzavřené křivky C kolem bodu P na 2D ploše. Nechť je spojitá 2D křivka C definována body $C(u) = C(x(u), y(u))$, pro $0 \leq u \leq 1$ a $C(0) = C(1)$. A zároveň bod P neleží na C. Pak lze definovat vektor $c(P, u) = C(u) - P$ z P do $C(u)$, a jednotkový vektor $w(P, u) = c(P, u)/|c(P, u)|$, který tvoří spojitou funkci $W(P) : C \to S^1$ mapující body $C(u)$ na C bodům $w(P, u)$ na jednotkové kružnici $S^1 = \{(x, y)|x^2 + y^2 = 1\}$. To může být reprezentováno polárními souřadnicemi $W(P)(u) = (cos\theta(u), sin\theta(u))$, kde $\theta(u)$ je úhel v radiánech směru proti hodinovým ručičkám (Counter Clock Wise - CCW). Počet obtočení $wn(P, C)$ je pak roven počtu celých obtočení, kolika $W(P)$ ovíjí C kolem S^1. To odpovídá stupni homotopie S^1, a může být vyjádřeno integrálem (2.2):

$$wn(P, C) = \frac{1}{2\pi} \oint_{W(P)} \mathrm{d}\theta = \frac{1}{2\pi} \int_{u=0}^{1} \theta(u)\, \mathrm{d}u \tag{2.2}$$

Když je křivkou C polygon s vrcholy $V0, V_1, \ldots, V_n = V_0$, tento integrál se zjednodušuje na sumu úhlů (se znaménkem) tvořených body V_i, P, V_{i+1} pro každou hranu polygonu. Takže, pokud je $\theta_i = \angle(PV_i.PV_{i+1})$ vzniká (2.3):

$$wn(P, C) = \frac{1}{2\pi} \sum_{i=0}^{n-1} \theta_i = \frac{1}{2\pi} \sum_{i=0}^{n-1} \arccos\left(\frac{(V_i - P)(V_{i+1} - P)}{|(V_i - P)||(V_{i+1} - P)|}\right) \tag{2.3}$$

Tento zápis evidentně není efektivní pro výpočet z důvodu výpočetně náročné funkce arccos(), nicméně tento vzorec lze nahradit efektivnějším. Nechť je vybrán jakýkoliv bod Q na S^1, když se pak křivka $W(P)$ ovíjí kolem S^1, prochází několikrát bodem Q. Pokud se započítá (+1), když projde Q CCW a (-1) v případě, že po směru hodinových ručiček. Pak výsledná suma je rovna počtu ovinutí $W(P)$ kolem S^1, stejně jako winding number $wn(P, C)$.

Stejně tak, pokud zvolíme polopřímku R z bodu P ve směru Q, pak průsečíky této polopřímky s polygonem odpovídají bodům, kde $W(P)$ projde Q. Pro dokončení je potřeba ještě rozlišovat v průsečících R a C, zda C prochází R z prava do leva a nebo z leva do prava (neboli, za se jedná o "pozitivní"+1 a nebo "negativní"-1 průsečík).

Jakým směrem hrana prochází může být zjištěno například využitím znaménkového testu (část 2.3.1), nicméně i tato část lze ještě více optimalizovat. Vzhledem k tomu, že nezáleží na to, jaká polopřímka R bude vybrána, lze vybrat i polopřímku vodorovnou. Díky tomu není nutné počítat průsečík, ale pouze ověřit, zda bod P leží nalevo (+1) od testované hrany. Výsledek tohoto testu je ovšem nutné upravit v závislosti, zda hrana V_iV_{i+1} míří směrem vzhůru nebo ze shora dolů. Pokud hrana vedoucí směrem vzhůru protne R napravo od P, pak je P nalevo od této hrany a naopak, když míří hrana směrem dolů má bod P od sebe napravo.

2.3.3 Test průsečíku úseček

Dalším algoritmem použitým v této práci je test průsečíku přímek [3, 9]. V úvodu algoritmu je nutné definovat operaci $\times$ jako dvourozměrný znaménkový test (část 2.3.1).

Máme-li zadané dvě úsečky vedoucí z bodu p do bodu $p + r$ a z bodu q do $q + s$ (obrázek 2.6), pak lze jakýkoliv bod na první úsečce popsat pomocí vztahu $p + tr$ (pro skalární paramer t) a jakýkoliv bod na druhé úsečce jako $q + us$(pro skalární parametr u). Dvě přímky se protínají právě tehdy, když může nalézt parametry t a u takové, že platí (2.4):

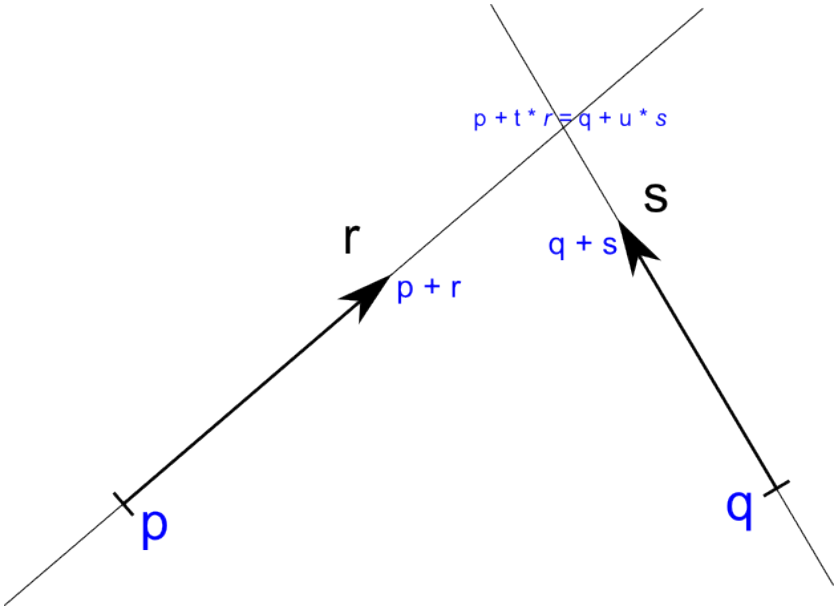
$$p + tr = q + us \tag{2.4}$$

Vektorovým násobením obou stran s získáme (2.5)

$$(p + tr) \times s = (q + us) \times s \tag{2.5}$$

A protože víme, že $s\ x\ s = 0$, můžeme upravit na (2.6)

$$t(r \times s) = (q - p) \times s \tag{2.6}$$



Obrázek 2.6: Znázornění problému průsečíku úseček

A tudíž vyřešit pro parametr t (2.7)

$$t = \frac{(q - p) \times s}{r \times s} \tag{2.7}$$

Obdobně se postupuje pro parametr u (2.8)

$$\begin{gathered} (p + t * r) \times r = (q + u * s) \times r \\ u(s \times r) = (p - q) \times r \\ u = \frac{(p-q) \times r}{(s \times r)} \end{gathered} \tag{2.8}$$

Tento vztah lze dále výpočetně optimalizovat na (2.9), protože platí $s \times r = -r \times s$.

$$u = \frac{(p - q) \times r}{r \times s} \tag{2.9}$$

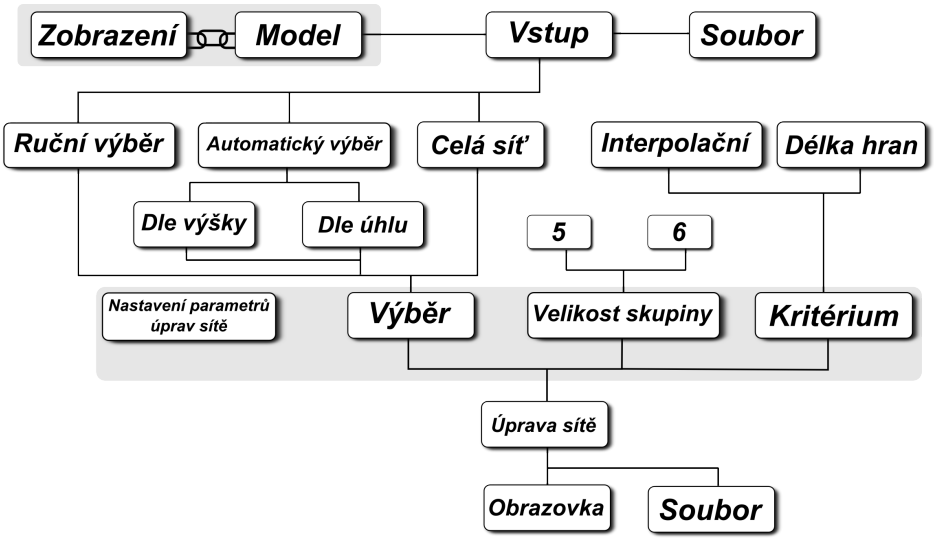
Lze rozlišit pět různých možností výsledků:

- Pokud $r \times s = 0$ a zároveň $(q - p) \times r = 0$, jsou tyto úsečky kolineární. Pokud je navíc buď $0 \leq (q - p)r \leq r * r$ nebo $0 \leq (p - q)s \leq s * s$, pak se tyto úsečky překrývají.
- Pokud $r \times s = 0$ a zároveň $(q - p) \times r = 0$, ale ani $0 \leq (q - p)r \leq r * r$ ani $0 \leq (p - q)s \leq s * s$, pak jsou tyto úsečky disjunktní.

- Pokud $r \times s = 0$ a $(q - p)x\ r \neq 0$, pak jsou tyto úsečky paralelní a neprotínají se.
- Pokud $r \times s \neq 0$ a $0 \leq t \leq 1$ a zároveň $0 \leq u \leq 1$, tak se tyto úsečky protínají v bodě $p + tr = q + us$.
- V každém jiném případě nejsou úsečky paralelní a ani se neprotínají.

3 Navrhované řešení

V rámci této kapitoly se nejprve představí aplikace a její možnosti názorně zachycené v diagramu na obrázku 3.1. Zbytek kapitoly je pak věnován postupnému detailnímu probírání jednotlivých částí, od aplikovaných lokálních kritérií (část 3.1), přes způsoby vnitřního uspořádání diagonál na skupině trojúhelníků (část 3.2) až po výběr oblastí na síti (část 3.3), na které lze následovně kritéria aplikovat. Závěr kapitoly (část 3.4) bude věnován stručnému popisu návrhu aplikace, jejímu vnitřnímu uspořádání a použitým knihovnám.



Obrázek 3.1: Přehled aplikace

Jak již bylo zmíněno na začátku předchozí kapitoly, trojúhelníky a trojúhelníkové sítě hrají významnou roli v digitální reprezentaci reálného terénu. Existuje sice více triangulačních algoritmů, nicméně ne vždy jsou všechny výsledné trojúhelníky vhodné pro výpočet vrstevnic. Problémové oblasti obsahující nevhodné trojúhelníky se proto dále upravují. Jednou z možností je použít simulované žíhání, které je ovšem výpočetně náročné. Druhou možností je lokální prohazování diagonál na dvojici sousedních trojúhelníků, nicméně jak již název implikuje, tato metoda nalézá pouze lokálně optimální řešení. Vzhledem k výše zmíněným důvodům vzniká potřeba jistého kompromisu - postup, jenž bere v potaz větší oblast než je dvojice trojúhelníků ve snaze neuvíznout v nejbližším lokálním extrému, ale nedosahuje výpočetní náročnosti simulovaného žíhání.

Mezeru mezi SA a prohazováním diagonál se snaží zaplnit tato práce. Hlavní částí je tedy aplikace, jež umožňuje optimalizovat vybranou oblast

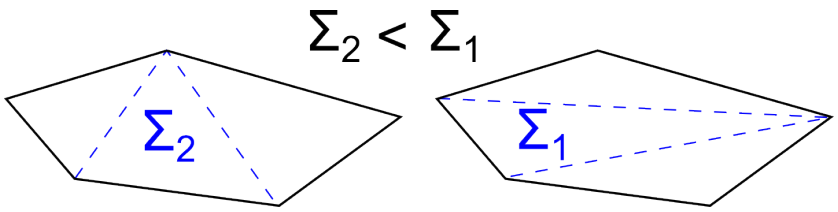
trojúhelníkové sítě. Optimalizace probíhá dle předem zvoleného kritéria, a to na skupinách tří nebo čtyř sousedících trojúhelníků – pětiúhelníku nebo šestiúhelníku. Výběr oblastí probíhá buď automaticky, nebo ručně. Automatický výběr lze realizovat různými způsoby - od výběru určité výškové hladiny přes náklon trojúhelníků (úhel mezi normálou a svislou osou z) až po úhly svírané s normálami sousedů. Celá aplikace je znázorněna na diagramu (obr. 3.1).

Pro úplnost je ještě nutno dodat, že v práci dále budou pěti nebo šestiúhelníky označovány jako „skupina“ trojúhelníků. Výběr pak bude také označován jako „oblast“ na síti. Výškou bodu se rozumí jeho z souřadnice v síti.

3.1 Kritéria

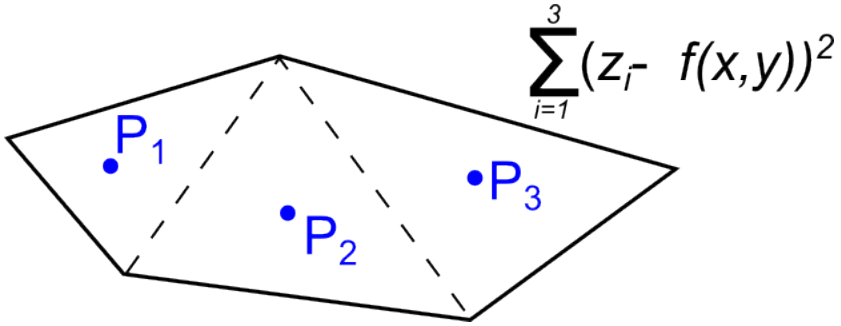
Jak je zobrazeno v přehledu aplikace (obr. 3.1), jsou implementována dvě rozhodovací kritéria pro lokální optimalizaci na vybrané oblasti. Protože tento počet nemusí být pro každého dostatečný, aplikace je navržena tak, aby umožňovala co nejsnadnější pozdější přidávání dalších kritérií. Tato lokální kritéria mají za úkol rozhodovat, která z možných kombinací (část 3.2) je optimální.

Prvním z kritérií (obr. 3.2) je kritérium rozhodující se dle sumy délky všech hran skupiny trojúhelníků. Vzhledem k tomu, že součástí každé triangulace jsou hrany konvexní obálky, tyto hrany se neupravují. Díky tomu lze tyto hrany ze sumy kritéria vyloučit a nadále pracovat pouze s délkou diagonál jednotlivých kombinací. Myšlenkou tohoto kritéria je tedy minimalizovat sumu délek hran skupiny, sumy hran lze přirozeně přiřadit jako váhu kombinacím. Se známou váhou je rozhodování o optimu již záležitostí triviální.



Obrázek 3.2: Demonstrace použití délkového kritéria

Druhé kritérium (obr. 3.3) je vhodné převážně pro umělé terény generované (známou) funkcí z důvodu, že u reálného terénu jen těžko zjistíme reálnou hodnotu výšky mimo body triangulace. Myšlenkou tohoto kritéria je minimalizovat chybu (přesněji řečeno kvadratickou odchylku) interpolace reálných dat zvolenou sítí.



Obrázek 3.3: Interpolační kritérium. z_i je výška sítě v bodě P_i

Hlavním problémem u výpočtu tohoto kritéria je zvolení míst v trojúhelníku, kde se bude porovnávat odchylka výšky sítě a reálné hodnoty. Řešení toho problému není zdaleka triviální. Jednou z možností by bylo zvolit vždy nové body na skupině čistě náhodně, nicméně pak by se lokální vylepšení stalo z části dílem náhody. Další možností je nejprve na síti zvolit pevné body pro všechny iterace sítí, zde by ovšem bez sofistikovaného algoritmu docházelo k neúměrnému zvýšení složitosti z důvodu nutnosti nalezení, kterému trojúhelníku aktuálně daný bod náleží. Třetí možností by bylo vždy vybrat pro každý trojúhelník nějaký bod (například jeho těžiště) a to i pro každou možnost úpravy skupiny. Poslední zmíněnou, a v aplikaci použitou možností pro šestiúhelníkové skupiny je kompromis mezi předchozími dvěma možnostmi – výběr specifického bodu (použito bylo těžiště) pro každý trojúhelník skupiny, ale dále tento body v rámci skupiny v jedné iteraci neměnit.

Nedostatkem toho způsobu, jež se ukázala při experimentech s tímto kritériem, je nejednoznačnost globálních výsledků. Třebaže bylo vždy vybráno minimum na skupině, globální výsledek ukazoval celkové zhoršení váhy, protože byla výška sítě sečtena v bodech jiných, než probíhalo rozhodování jednotlivých iterací. Ruku v ruce s tímto problémem jde nešťastně zvolený způsob průchodu sítí pro šestiúhelníky, kdy je velice obtížné udržovat statické pole testovaných bodů. Bohužel, pro šestiúhelníkové interpolační kritérium se tento problém nepodařilo do deadline vhodně vyřešit a proto se jeho výsledky v závěrečných experimentech nevyskytují.

V případě pětiúhelníkové skupiny je tento způsob vyřešen udržováním statického pole vzorkovaných bodů - na začátku dotazu na interpolační kritérium se z pole vyberou předem zvolené body, které se nachází v této skupině (indexy v poli odpovídají indexům trojúhelníků sítě). Na konci dotazu se dle zvolené možnosti vrátí body do pole na správné (po úpravě potenciálně jiné) indexy. Tento způsob je vhodnější a dává na rozdíl od způsobu pro šestiúhelníky vždy kladné relativní zlepšení váhy.

3.2 Skupiny

V aplikaci je možné uplatnit kritéria buď na skupiny tří nebo čtyř sousedících trojúhelníků. Důvodem pro vynechání skupiny dvou trojúhelníků byl fakt, že se vlastně jedná o již dobře prozkoumané prosté prohazování hran. Stejně tak byl omezen počet trojúhelníků ve skupině na čtyři z důvodu, že pro prozkoumání a demonstraci možností vývoje tímto směrem zmíněný počet postačuje. Nezanedbatelným důvodem této volby je též prudce zvyšující se složitost implementace iterací možnostmi uspořádání.

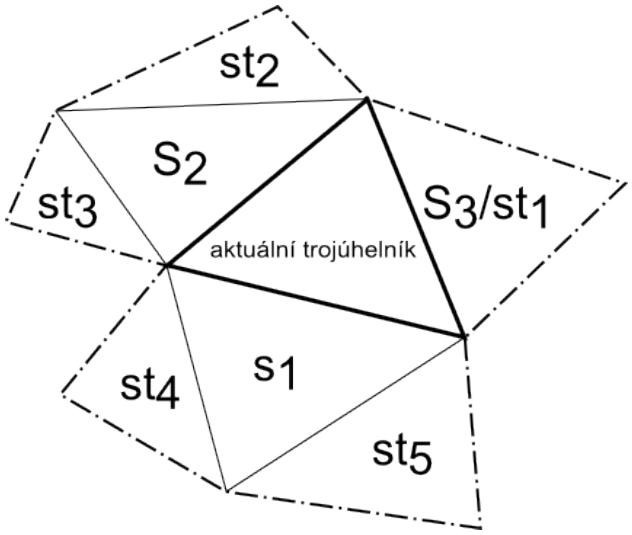
Před aplikováním lokálního kritéria je podstatné vyloučit neplatné možnosti uspořádání skupiny. Jednoduše řečeno, neplatné uspořádání je takové, které by porušilo definici a tím i integritu sítě. V tomto konkrétním případě je nutné vyřadit všechny možnosti, u nichž alespoň jedna z vnitřních hran by protínala nějakou hranu polygonu skupiny. Způsob generování možností vylučuje, že by se vnitřní hrany protínaly navzájem. Prvním testem hrany je, zda leží v úhlu vytvářeném oběma přilehlými hranami polygonu, neboli zda míří „dovnitř“ polygonu skupiny. Pokud diagonála projde tímto testem, je ještě nutné ověřit, zdali neprotíná nějakou z hran nepřilehlých koncovému ani počátečnímu bodu diagonály.

Pro generování jednotlivých možností vnitřního uspořádání skupin je potřeba zajistit dva druhy iterace. Za prvé vnější iterace generující skupiny - iterace sítí. Tato iterace je řešena průchodem sítí trojúhelník po trojúhelníku a vybíráním postupně všech (tří) dvojic jeho sousedů, což stačí pro generování skupin trojic. Pro generování skupin čtveřic se poté pro každou takto vybranou trojici k ní postupně přidružuje každý ze sousedů skupiny. Jak je patrno z obrázku (obr. 3.4), trojúhelník může být v rámci jedné iterace jak sousedem, tak i sousedem trojice. V těchto případech se s ním postupně počítá jako s obojím, aby bylo zajištěno co nejvíce prošlých možností. Výše popsaný způsob iterace není dokonalý, neboť v případě změny v síti, a tudíž změně indexů trojúhelníků, může dojít k přeskočení určitých možností. Tento nedostatek se ovšem vytrácí spolu s dalšími iteracemi celou sítí, které je ovšem nevyhnutelně nutné provádět bez ohledu na nedostatky způsobu iterace, protože je nereálné, že bude optimum nalezeno po prvním průchodu sítí. Zápis jedné iterace sítí v pseudokódu vypadá následovně:

```
1: for all trojúhelník do
2:   for all dvojiceSousedů do
3:     skupina ADD dvojiceSousedů
4:     skupina ADD trojúhelník
5:     if iterujProČtyřúhelník then
6:       for all sousedTrojice do
7:         ohodnoť (skupina + sousedTrojice)
```

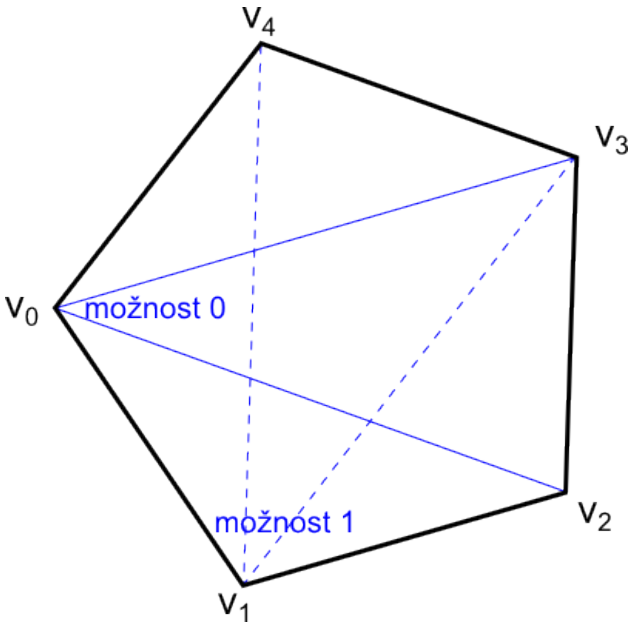
```
8:          end for
9:          CLEAR skupina
10:       else
11:         ohodnoť skupinu
12:         CLEAR skupina
13:       end if
14:     end for
15:   end for
```

Je třeba dodat, že výše uvedený pseudokód je pouze zjednodušenou verzí a samotná implementace je o něco složitější. Největší změnou oproti kódu je, že čtyřúhelníkem iteruje místo prostého cyklu dedikovaná metoda pro zajištění bezpečného průchodu sítí a vyřazení neplatných kombinací (okraje sítě atd.).



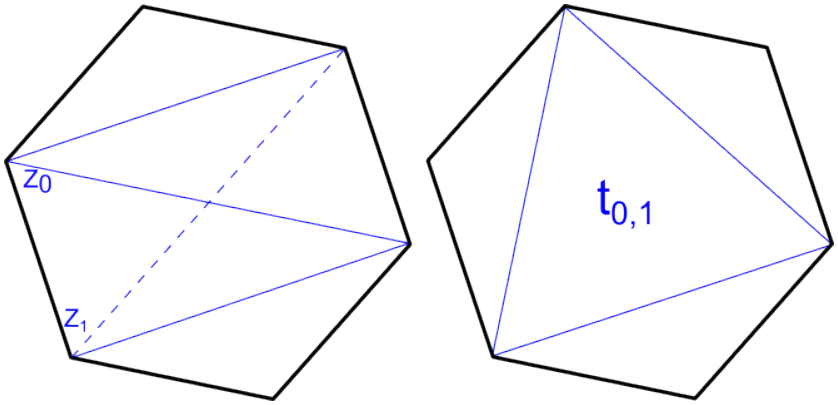
Obrázek 3.4: Příklad generování skupin

Vnitřní iterace skupinou je podstatně složitějším úkolem, u kterého se složitost ještě rapidně zvedá s velikostí skupiny. Od nejjednodušší možnosti – skupiny tří trojúhelníků tvořící pětiúhelník, který lze rozdělit vždy dvěma diagonálami. Tuto možnost lze označit za uskupení „vějíř“, protože vždy obě diagonály povedou z jednoho z vrcholů pětiúhelníku. Vějíř může vycházet z kteréhokoli vrcholu pětiúhelníku. Z toho vyplývá, že pro pětiúhelník existuje pět různých možností vnitřního uskupení, ale ovšem pouze za předpokladu, že každá hrana je validní. Vějíře v pětiúhelníku jsou znázorněny na obrázku 3.5.



Obrázek 3.5: Ukázka prvních dvou uspořádání typu "vějíř"

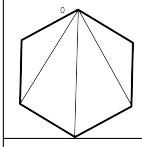
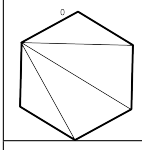
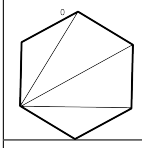
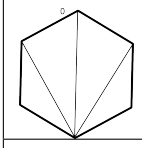
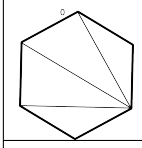
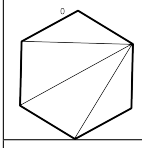
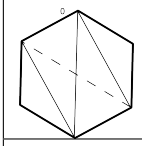
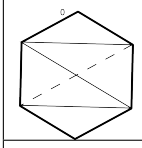
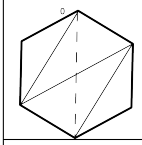
Iterace možnostmi vnitřního uspořádání šestiúhelníku je podstatně složitější. Ač je zde analogie s předchozím příkladem – na šestiúhelníku existují vějíře pro každý jeho vrchol (tzn. šest možností uspořádání), je možné navíc identifikovat možnosti typu „Z“ a „vnitřní trojúhelník“. Možnosti typu „Z“ se skládají z dvou diagonál spojujících vrcholy vzdálené jeden vrchol na polygonu a diagonály spojující vrcholy „naproti“, nicméně lépe než slovní popis tuto možnost ukazuje obrázek 3.6. Těchto možností se opět vyskytuje na šestiúhelníku celkem šest – pro dvojici kratších spojnic je možné zvolit dvě různé delší diagonály, ovšem možnosti naproti sobě splývají, tudíž je celkem 3x2 možností typu „Z“. Poslední možností je „vnitřní trojúhelník“ a jak již název napovídá, diagonály u ní vevnitř šestiúhelníku formují trojúhelník. Mohlo by se zdát, že tyto možnosti existují dvě v závislosti na natočení tohoto vnitřního trojúhelníka, nicméně vzhledem ke způsobu generování skupin na natočení nezáleží (vždy jsou vybíráni sousedi nějakého trojúhelníka) a tyto dvě možnosti z pohledu výběru splývají. Celkem je tedy možné vygenerovat 13 možností vnitřního uspořádání skupiny (samozřejmě za předpokladu, že jsou všechny hrany validní). Při generování jednotlivých možností je nutno postupovat pro každý typ uspořádání zvlášť, univerzální postup se nepodařilo nalézt.



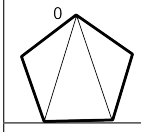
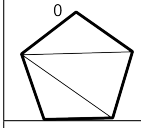
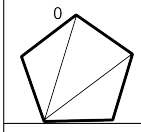
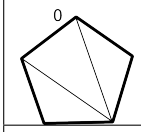
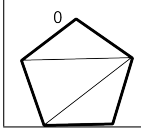
Obrázek 3.6: Další uspořádání na šestiúhelníku

Poslední, co zbývá uvést, je způsob zakomponování těchto způsobů do aplikace. Při implementování nemalou roli hrála znovupoužitelnost jednotlivých částí a tak bylo každé možnosti pevně přiřazeno ID. V případě pětiúhelníku je ID možnosti rovno pořadí vrcholu, z něhož vychází vějíř, v polygonu (viz tabulka 3.2).

Případ šestiúhelníka je opět podstatně složitější, prvních 6 možností jsou typu vějíř, a tudíž toto ID odpovídají indexu jejich společného vrcholu. Dalších šest možností se pak indexuje po dvojicích, jež využívají stejné dvojice kratších hran. Ve dvojici je vždy sudá možnost ta, která využívá diagonálu z i do $i+3$ pro i odpovídající pořadí dvojice, tedy $i = 0$ až 2. Lichá možnost pak využívá zbylou delší diagonálu. Poslední možnost, "vnitřní trojúhelník", má pak ID 12. Indexy možností jsou znázorněny v tabulce 3.1.

	hrana 1		hrana 2		hrana 3		ID možnosti
	z	do	z	do	z	do	
	0	2	0	3	4	0	0
	1	3	1	4	5	1	1
	2	4	2	5	0	2	2
	3	5	3	0	1	3	3
	4	0	4	1	2	4	4
	5	1	5	2	3	5	5
	1	3	4	0	0 a 1	3 a 4	6 a 7
	2	4	5	1	1 a 2	4 a 5	8 a 9
	3	5	0	2	2 a 0	5 a 3	10 a 11
	0	2	2	4	4	0	12
	1	3	3	5	5	1	13

Tabulka 3.1: možnosti uspořádání v šestiúhelníku a jejich ID, možnosti dvojice možností 6,7 až 10,11 jsou v jediném řádku, protože se liší pouze diagonálou. V případě neplatnosti některé z diagonál, se s kombinací nepočítá

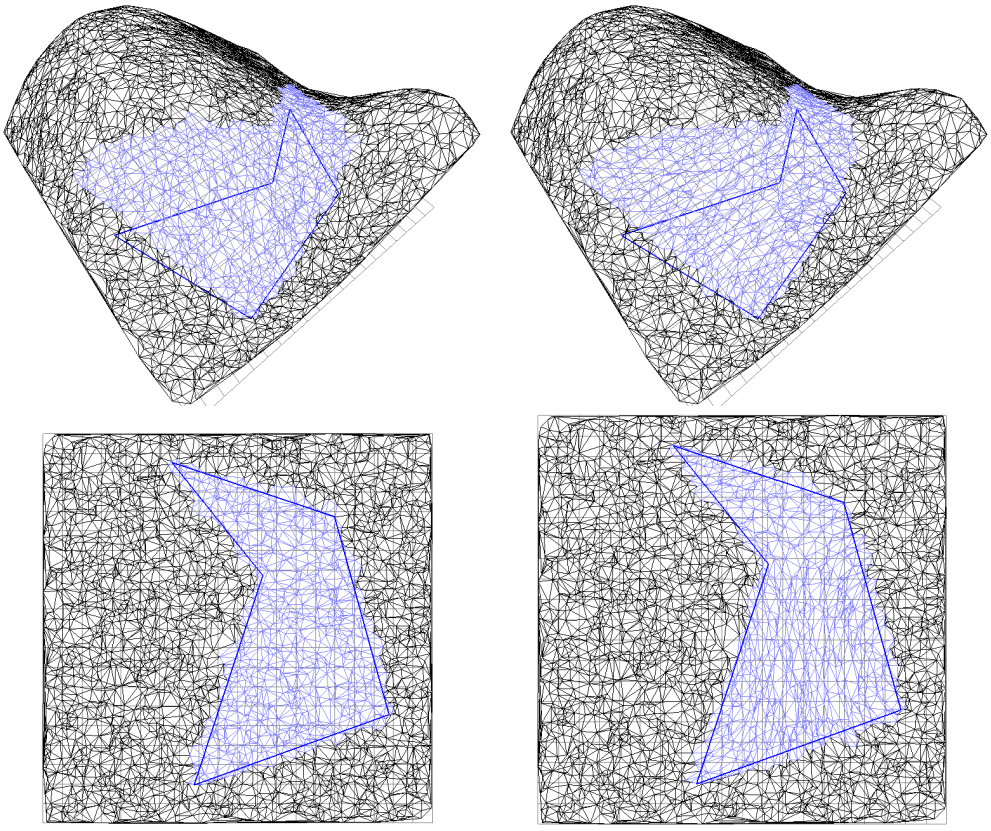
	hrana 1		hrana 2		ID možnosti
	z	do	z	do	
	0	2	3	0	0
	1	3	4	1	1
	2	4	0	2	2
	3	0	1	3	3
	4	1	2	4	4

Tabulka 3.2: možnosti uspořádání v pětiúhelníku a jejich ID

3.3 Výběry

Výběrem se rozumí určitá množina trojúhelníků sítě, kterou pak lze iterovat a aplikovat kritéria (část 3.1). Výběrem se dá označit i celá síť, ovšem výběr celé sítě je úloha triviální, proto se tato část věnuje ostatním způsobům výběru. Pro zahrnutí do výběru stačí, aby jakýkoli jeho bod splňoval podmínku zahrnutí do kritéria, není-li u daného kritéria výslovně uvedeno jinak. Výběr tedy vždy přesahuje vybrané meze, namísto druhého možného způsobu – nedosahování mezí. Toto je čistě rozhodnutí návrhu.

Jedním ze způsobů, který aplikace umožňuje je přímý, ruční výběr oblasti. Jeho použití demonstruje následující příklad, jako data byla použita síť tvaru "sedlo". Původní triangulace s výběrem se nachází na obrázku 3.7 vlevo, nová pak napravo (polygon výběru je dvourozměrný, v 3D náhledu je proto zkreslen projekcí). Na obrázku 3.7 vpravo je patrná mírná nesouvislost hranice.

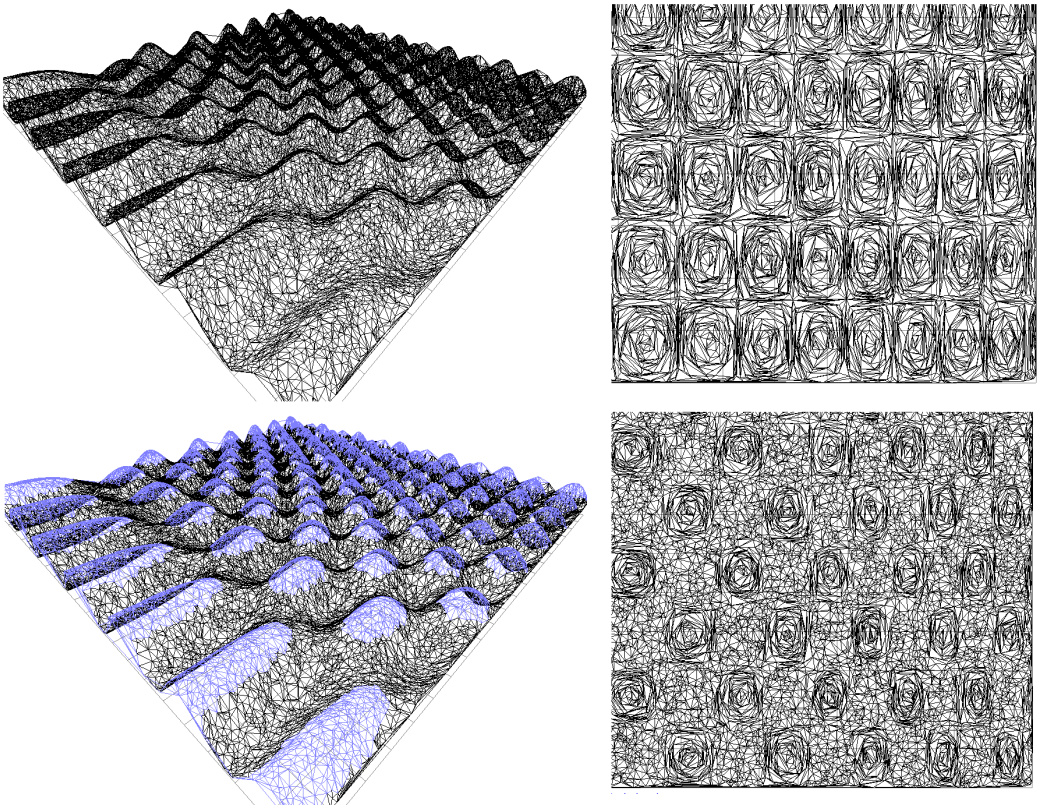


Obrázek 3.7: Ruční výběr na síti. Vlevo se nachází původní síť s výběrem, vpravo je nová síť se stále aktivním výběrem. Jak v 3D náhledu, tak i v 2D náhledu se jedná o stejný výběr

Aplikace umožňuje také několik způsobů výběru automatického. Prvním způsobem automatického výběru je výběr dle rozdílu výšek v trojúhelníku. Přesněji řečeno, do tohoto výběru patří každý trojúhelník splňující podmínku, že maximální rozdíl z souřadnic jeho vrcholů je menší než předem stanovené epsilon. Jedná se tedy o výběr trojúhelníků skoro vodorovných, nehledí se na jejich velikost. Dalším způsobem výběru založeném na výšce bodů je výběr všech trojúhelníku v daném výškovém rozsahu. Jeho využití se přímo nabízí při zaměření se na určitou vrstevnici terénu.

Obrázek 3.8 demonstruje použití automatického výběru pomocí rozsahu výšky. Na obrázku vlevo je zachycena síť (s upraveným měřítkem osy z). Na síti byla vybrána oblast ve výškovém rozmezí 0,71 až 1 a posléze na ní bylo aplikováno délkové kritérium na pětiúhelníkových skupinách. Výsledek je možné vidět na obrázku 3.8 vpravo. Ač by z horního pohledu mohlo zdát,

že suma délek hran se zvýšila, je nutné brát v potaz že se jedná o horní pohled na síť s velkými výškovými rozdíly.



Obrázek 3.8: Srovnání automatické výběr výškou a následná úprava sítě (dole) s aplikováním kritéria na celou síť (nahoře).

Jak název napovídá, automatické úhlové výběry jsou založené na úhlu, který svírají normály trojúhelníků a to buď s normálami sousedů anebo se svislou osou z. První zmíněný způsob je dále možné kaskádovitě rozšířit na sousedy sousedů atd. Aplikace ovšem nezachází dále než zmíněné sousedství sousedů. Tento způsob výběru lze například využít na hledání zlomů v terénu a nebo naopak podobně jako předchozí výběry na hledání rovinných oblastí. Posledním automatickým způsobem výběru je výběr dle úhlu mezi normálou trojúhelníku a svislou osou z, určuje tedy spád terénu a lze opět využít jak na hledání rovinných oblastí, tak i oblastí příkrých.

3.4 Aplikace a implementace

Protože se uživatelská příručka k aplikaci se nachází na přiloženém DVD, tato část se věnuje převážně implementaci a vzoru rozvržení aplikace. Návrhovým vzorem této aplikace je Model-View-ViewModel (MVVM) [10]. Jedná se o model dovozený z tradičního návrhového vzoru Model View Controler (MVC). Myšlenkou MVVM je využití mocného nástroje, který Windows Presentation Foundation (WPF) nabízí - databindingu. View, zprostředkovatele grafického náhledu na Model (data), a Model není nutné zdlouhavě popisovat, protože jejich názvy odpovídají jejich funkci.

Úkol ViewModelu se značně liší dle zdroje definice a není přesně definován. V aplikaci je použita definice, jež ViewModel popisuje jakou zprostředkovatele databindingu k View a zprostředkovatele vizualizace modelu. Funkce ViewModelu není tedy nepodobná původnímu Controlleru, ale využívá svázání jednotlivých částí GUI ke svým vlastnostem. Příkladem může svázání textu tlačítka k proměnné typu string v kódu (které může být obousměrné), pro změnu textu v GUI pak stačí pouze změnit danou proměnnou. Ke každému View by měl existovat minimálně jeden VeiwModel, toto pravidlo je ovšem v aplikaci není dodrženo pro View typu vyskakovacího okna.

Nespornou výhodou WPF jsou prakticky neomezené možnosti úprav vzhledu jednotlivých kontrolek. Tato výhoda má ovšem i svoji stinnou stránku - i zdánlivě jednoduchá úprava může vyžadovat rozsáhlé úpravy celého stylu kontrolky (příkladem může být zvýraznění ComboBoxu při přejetí myší). Dalším příkladem by mohl být i posuvník se dvěma jezdci [8].

3.4.1 MVVM Light

Toolkit MVVM Light navíc registruje a spravuje instance jednotlivých ViewModelů ve statickém lokátoru, který v DataContextu jednotlivých View nahrazuje (a zastřešuje) ViewModely. Díky tomu lze univerzálně přistupovat (i k jiným než aktuálnímu) ViewModelům.

V základním WPF nelze vytvářet jiné příkazy (Command, obdoba handlerů událostí), než jsou příkazy spojené událostmi myši, proto MVVM Light přidává navíc mechaniku EventToCommand a RelayCommand, které toto omezení odstraňují.

MVVM Light dále nabízí systém zpráv (přístup do jiného než přiřazeného ViewModelu z View by porušil návrhový vzor), který umožňuje komunikaci mezi komponentami, která by jinak nebyla možná. Příkladem použití je přeposílání událostí GUI, které by jinak byly nezachytitelné (dvojici EventToCommand a RelayCommand nelze použít na některé vlastní události).

3.4.2 SharpGL a zobrazení sítě

Stejně jako pro zobrazení GUI se využívalo WPF, pro zobrazení světa (trojúhelníkové sítě) se využívá knihovny SharpGL [5]. Knihovan Sharp GL je wrapperem OpenGL umožňujícím použití OpenGL v jazyce C#, se syntaxí co nejpodobnější tomuto jazyku.

Umístění sítě do zorného pole nezávisle na rozsahu hodnot má několik kroků. Před zobrazením sítě se nejprve spočítají ve funkci $getDataMetrics()$ informace o síti, minima a maxima ve všech osách. Tyto informace se pak využijí ke správnému posunutí počátku, aby bod $(X_{min}, Y_{min}, Z_{min})$ byl v původním počátku $(0, 0, 0)$. Z rozdílů mezi maximy a minimy v jednotlivých osách se určuje měřítko, aby vyšší rozdíl odpovídal délkou velikosti mřížky.

Pro pohyb v prostoru se využívá funkce $glLookAt()$, vektory Eye, $Focus$ a Up udržuje aktuální dle podnětů z GUI implementovaná třída $SphereCamera$. Jak již název napovídá, $SphereCamera$ umožňuje pohyb na kouli a to v zadané vzdálenosti kolem zaměřeného bodu. Dále pak umožňuje pohyb na rovině dané vektory Up a $Right$.

Aplikace také umožňuje přepínání mezi 3D módem a pseudo2D módem. 2D mód je ovšem pouze natočení kamery do směru opačnému svislé ose a změny měřítka svislé osy na nulu. Pouze ve 2D módu je umožněn výběr oblasti polygonem.

3.4.3 Stručný popis tříd

Aplikace se sestává mimo jiné z následujících tříd:

- $FileParser$ - čte a zapisuje sítě do souboru
- $InterpolationCriterion$ - délkové kritérium
- $LenghtSumCriterion$ - interpolační kritérium
- $Selection$ - popisuje výběr
- $TINModel$ - popisuje model s výběry a sítí
- $TINModelItem$ - popisuje síť
- $Triangle$ - popisuje trojúhelník s vrcholy a sousedy
- $TriangulationSettings$ - uchovává všechny potřebné informace z TriangulateDiagu
- $VectorD$ - popisuje vektor a implementuje vektorovou algebru

- *VertexD* - popisuje vrchol sítě
- *MainViewModel* - ViewModel hlavního View
- *NativeMethods* - nativní metody, čtení a nastavování kurzoru myši
- *SphereCamera* - kamera uchovávající aktuální vektory Eye, Focus a Up pro glLookAt
- *StringConvertor* - konvertor pro převádění typu double na string
- *ViewModelLocator* - Lokátor, část MVVM Light

GUI popisují XAML (a jejich codebehind) soubory:

- *MainSkin* - styly hlavního okna aplikace
- *MainView* - hlavní okno aplikace
- *RangeSlider* - posuvník se dvěma jezdci
- *TriangulateDialog* - okno pro nastavení triangulace
- *SettingsDialog* - okno pro nastavení aplikace

V neposlední řadě je nutné ještě zmínit rozhraní ICriterion, které má za úkol pomoci s další tvorbou kritérií. Smyslem tohoto rozhraní je usnadnit případnou další tvorbu kritérií. Kritéria mají k dále k dispozici funkce ověřující platnost hrany. Implementace rozhraní se skládá z těchto metod:

- double Evaluate(Triangle[] triangulation) vyhodnocující váhu dané oblasti trojúhelníků
- int Apply5(List<VertexD> polygon, Triangle[] trj) vyhodnocující kritérium na pětiúhelníku, druhým parametrem jsou samotné trojúhelníky tvořící tento pětiúhelník
- KeyValuePair<int, double> Apply6(List<VertexD> polygon, Triangle[] trj) vyhodnocující kritérium na šestiúhelníku, druhým parametrem jsou samotné trojúhelníky tvořící tento pětiúhelník

Pro správné fungování je ovšem nutné ještě navíc dodržet vnitřní konvenci číslování možností (podrobně popsáno v kapitole 3.2). Ačkoli není vysloveně nutné využít těchto nástrojů, tyto nástroje práci velice usnadňují a urychlují. Nespornou výhodou tohoto rozhraní je, že kód aplikace je připraven pracovat s jakoukoliv implementací (která dodrží vnitřní konvenci indexování možností) tohoto rozhraní a tudíž není nutné implementovat logiku procházení sítí při přidávání dalších kritérií.

4 Experimenty a výsledky

Kapitola Experimenty je členěna do částí podle jednotlivých kritérií. Důvodem oddělení je není pouze přehlednost, ale i fakt, že každé kritérium je značně unikátní po implementační stránce a tudíž se dá předpokládat, že to co platí pro jedno kritérium nemusí platit pro kritérium druhé.

V části 4.1 je kritérium nejprve testováno na datech různé velikosti pro rozmístění vrcholů typu "shluky vrcholů", "Gaussovo pravděpodobností rozložení", pravidelné rozložení do mřížky a rovnoměrné náhodné rozložení. Poté je kritérium na sítích "sedlo"a "špičky"testováno na možné zlepšení interpolace.

Část 4.2 se věnuje výsledkům interpolačního kritéria pro pětiúhelníkové skupiny na výše zmíněných sítích "sedlo"a "špičky", výsledky šestiúhelníkových skupin jsou vynechány záměrně (z důvodů vysvětlených v části 3.1).

Poslední část 4.3 je věnována orientačním zátěžovým testům obou kritérií a jejich výsledkům. Z důvodu lepší dostupnosti dat je délkovému kritériu v části 4.3 věnováno o něco více prostoru.

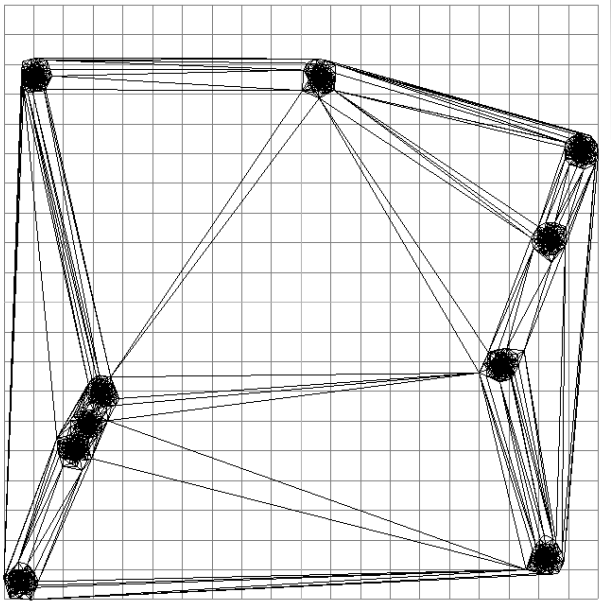
Zlepšením se rozumí procentuální zlepšení výsledné váhy sítě - $z = ((s/n) - 1) * 100$, kde z je zlepšení, s je stará váha a n je váha nová. Samotná absolutní váha vždy odpovídá použitému kritériu, není-li výslovně uvedeno jinak. Jak již bylo popsáno v části 3.1, váha délkového kritéria určena sumou délek hran. U interpolačního kritéria je vahou suma kvadratické odchylky od generující funkce v testovaných bodech. Jednotkami jsou sítí blíže nespecifikované délkové jednotky.

Experimenty probíhaly vždy na několika sítích rozdílných velikostí a to od 5×10^3 do 100×10^3 vrcholů. Zastavovací podmínka algoritmu byla nastavena na zlepšení 0,001 %. Jinými slovy, síť byla procházena dokud bylo relativní zlepšení mezi průchody sítí vyšší než 0,001 %. Výpočet probíhal na čtyřjádrovém procesoru Intel i5-4670k @3.4GHz.

4.1 Experiment 1: Výsledky délkového kritéria

Následující testy délkového kritéria se zaměřují na globální výsledky různých druhů sítí pro různé rozložení jejich vrcholů v rovině. V každém testu byla vždy zaznamenána původní váha, nové váhy (pro pětiúhelník i šestiúhelník) a dosažená zlepšení. Testy probíhaly na sítích různých velikostí, přesněji o velikosti 5, 10, 50 a 100 tisíc vrcholů.

První testovaná data mají body rozložené do shluků, viz obrázek 4.1. Překvapivé je zejména to, že s rostoucí velikostí sítě rapidně kleslo relativní zlepšení. Tento jev se vyskytuje také u jiných druhů sítě, ovšem zpravidla v menší míře.

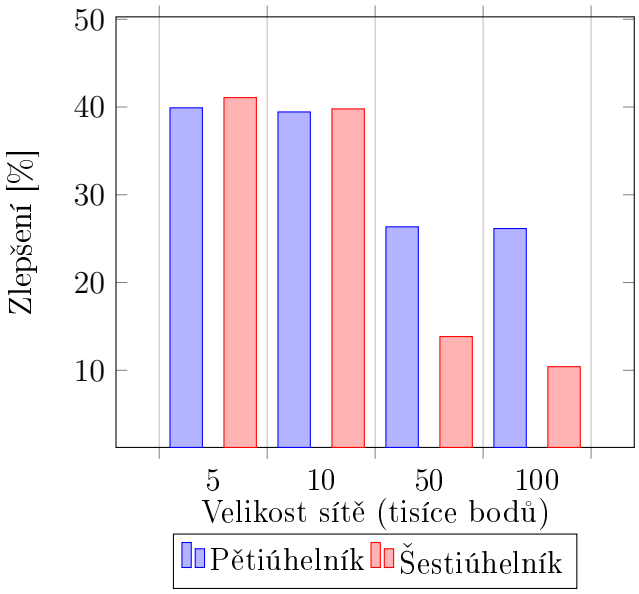


Obrázek 4.1: Síť tvořená shluky vrcholů, 5000 vrcholů

Přesné výsledky tohoto experimentu jsou uvedeny v tabulce 4.1 a grafu 4.2.

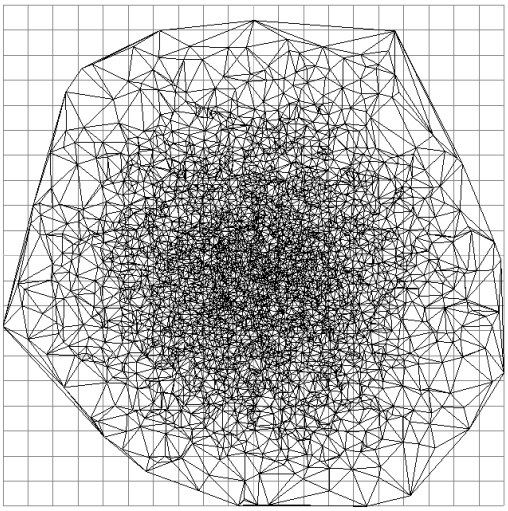
		pětiúhelník		šestiúhelník	
počet vrcholů	původní váha	nová váha	zlepšení [%]	nová váha	zlepšení [%]
5000	527,29005854	376,88227318	39,908427	373,7681049	41,074118
10000	890,43103038	638,58248575	39,438687	636,99891359	39,78533
50000	3626,13095653	2869,9469405	26,348362	3185,2529006	13,841226
100000	7967,13880298	6315,48242386	26,152497	7215,93756014	10,410307

Tabulka 4.1: Výsledky délkového kritéria pro shluky vrcholů.



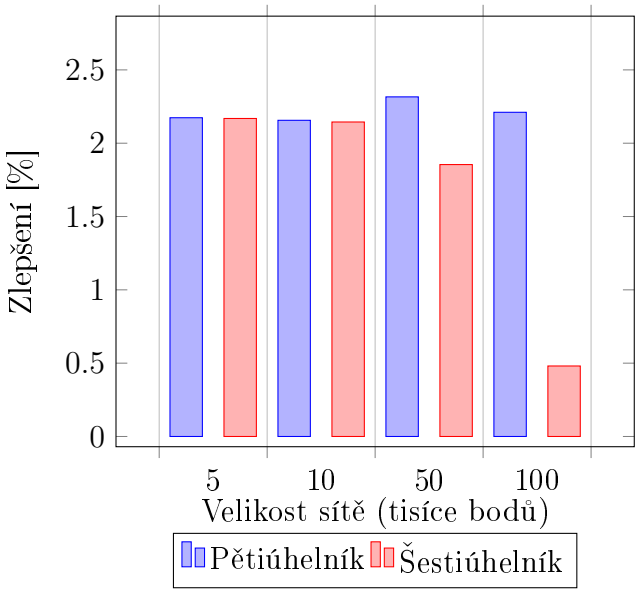
Obrázek 4.2: Výsledky pro shluky bodů

Druhý typ dat je tvořen vrcholy s Gaussovým pravděpodobnostním rozložením, viz obrázek 4.3. I zde dochází k mírnému propadu relativnímu zlepšení s růstem velikosti. Vyjímkou je pouze největší síť u šestiúhelníkových skupin, kde došlo k propadu výraznějšímu. Zlepšení je ale mnohem menší než u typu "shluk vrcholů".



Obrázek 4.3: Síť s Gaussovým pravděpodobnostním rozložením vrcholů, 5000 tisíc vrcholů

Kompletní výsledky jsou zaznamenány v tabulce 4.2 a grafu 4.4.



Obrázek 4.4: Výsledky pro Gaussovo pravděpodobnostní rozložení bodů

		pětiúhelník		šestiúhelník	
počet vrcholů	původní váha	nová váha	zlepšení [%]	nová váha	zlepšení [%]
5000	344,77234457	337,43735064	2,173735	337,45408517	2,168668
10000	490,96869134	480,60586034	2,156202	480,66015418	2,144662
50000	1103,85634212	1078,8700522	2,315968	1083,75992627	1,854324
100000	1559,15465084	1525,4296717	2,210851	1551,69501887	0,480741

Tabulka 4.2: Výsledky pro síť s Gaussovým pravděpodobnostním rozložením vrcholů.

Rozložení vrcholů u třetího typu dat do pravidelné mřížky dává tušit jen malé možnosti změn. Experiment tuto domněnku jen prokázal - nedošlo k vyššímu relativnímu zlepšení než $0,5 \times 10^{-3}$ % a u sítě s deseti tisíci vrcholy nedošlo k zlepšení vůbec žádnému. Za zmínku také stojí, že pro obě velikosti skupin jsou výsledky (kromě doby výpočtu) identické. Z důvodu nezajímavosti výsledků zde graf ani obrázek není uveden, ovšem přesné výsledky uvádí tabulka 4.3.

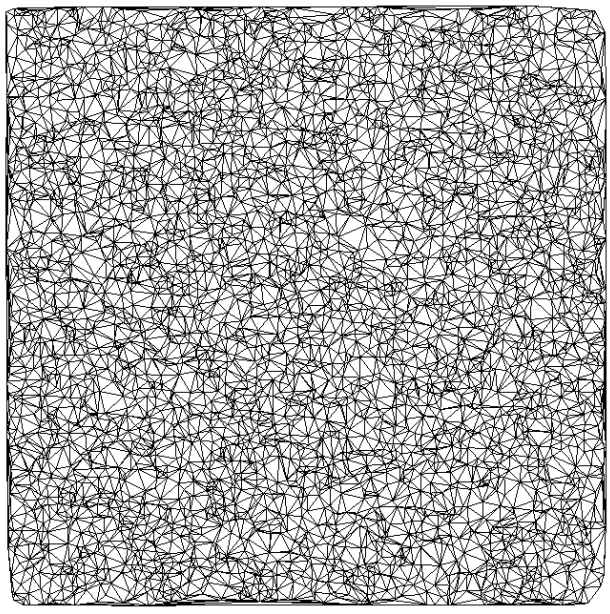
		pětiúhelník		šestiúhelník	
počet vrcholů	původní váha	nová váha	zlepšení [%]	nová váha	zlepšení [%]
5000	475,71582552	475,71580952	0,000003	475,71582552	0,000003
10000	675,00424064	687,00424064	0	675,00424064	0
50000	1519,63975265	1519,63969265	0,000004	1519,63969265	0,000004
100000	2151,64399676	2151,64389876	0,000005	2151,64389876	0,000005

Tabulka 4.3: Výsledky pro síť s rozložením vrcholů do pravidelné mřížky.

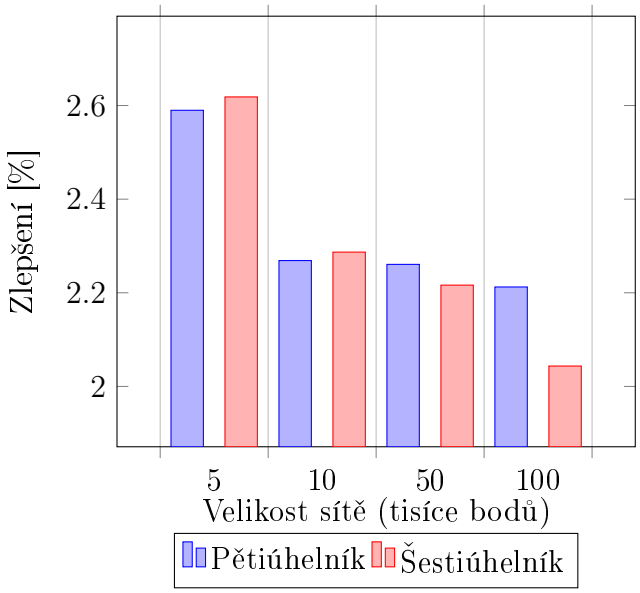
Rovnoměrné náhodné rozložení vrcholů je podkladem pro čtvrtý druh testovaných sítí. Na obrázku 4.5 je takováto neupravená síť. Hodnoty relativního zlepšení jsou zhruba stejné jako u Gaussovo pravděpodobnostního rozložení vrcholů, v hodnotách v tabulce 4.4 a v grafu 4.6 nejsou žádné překvapující hodnoty.

		pětiúhelník		šestiúhelník	
počet vrcholů	původní váha	nová váha	zlepšení [%]	nová váha	zlepšení [%]
5000	495,55303588	483,04296392	2,589847	482,90861644	2,618388
10000	698,87091185	683,36644301	2,268837	683,24573509	2,286904
50000	1536,87337111	1502,89738644	2,260699	1503,54939894	2,216354
100000	2170,46268906	2123,48190932	2,212441	2126,99586988	2,043578

Tabulka 4.4: Výsledky pro síť s rovnoměrným náhodným rozložením vrcholů



Obrázek 4.5: Síť s náhodným rovnoměrným rozložením vrcholů, 500 vrcholů



Obrázek 4.6: Výsledky pro náhodné rozložení bodů

Pomine-li se pravidelné rozložení sítě, hodnoty relativního zlepšení jsou mírně nižší pro větší sítě. Obě velikosti oblastí dávají přibližně stejné výsledky, ovšem zatímco šestiúhelníková je jen o trochu úspěšnější pro menší sítě, u větších sítí častěji zaostává. To může být způsobeno buď náhodou anebo skrytou nedokonalostí v generování možností. Ač se několik nedostatků (všechny nalezené) pro šestiúhelníky již opravilo, nelze v souvislosti výsledky s jistotou říci, že už žádné jiné neexistují. Případné nedostatky budou ovšem drobného charakteru, protože pro obě velikosti algoritmus prošel všemi vytvořenými testy.

Další část experimentu se snaží odpovědět na otázku, jak bude po aplikování délkového kritéria na síť reagovat hodnota váhy kritéria interpolačního. Pro tento test byla využita data "sedlo"(na obrázku 3.7) a "špičky"(na obrázku 3.8). Zmíněným datům se detailněji věnuje část 4.2.

	váha délkového kritéria			váha interpolačního kritéria		
velikost skupiny	původní váha	nová váha	zlepšení [%]	původní váha	nová váha	zlepšení [%]
pětiúhelník	551,0892178	504,31658328	9,274459	759,78771528	759,48429758	0,02995049
šestiúhelník	551,0892178	504,22754688	9,293755	759,78771528	759,4779019894	0,04079293

Tabulka 4.5: Výsledky pro síť "sedlo"délkového kritéria s výslednou vahou délkového i interpolačního kritéria

	váha délkového kritéria			váha interpolačního kritéria		
velikost skupiny	původní váha	nová váha	zlepšení [%]	původní váha	nová váha	zlepšení [%]
pětiúhelník	10738,71941217	5108,11288561	110,228702	7578,57662446	7404,29012504	2,35385832
šestiúhelník	10738,71941217	5086,49688842	111,122107	7578,57662446	7410,10530721	2,27354723

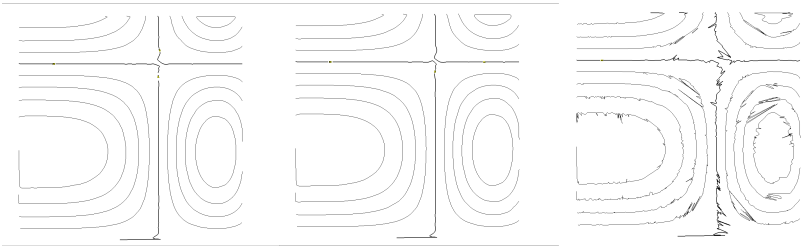
Tabulka 4.6: Výsledky pro síť "špičky"délkového kritéria s výslednou vahou délkového i interpolačního kritéria

Z výsledků v tabulkách 4.5 a 4.6 není patrné, že by délkové kritérium nějak výrazně ovlivňovalo váhu kritéria interpolačního. Pozoruhodné ovšem je, že v žádném z případů nedošlo ke zhoršení váhy interpolačního kritéria.

4.2 Experiment 2: Výsledky interpolačního kritéria

V testech interpolačního kritéria jsou použité pouze dvě sítě z předchozí části, protože u ostatních nejsou známé jejich generující funkce a nebo je jejich funkcí $f(x, y) = 0$ (a z souřadnice všech jejich bodů je rovna nule), tudíž na zmíněných datech není co optimalizovat z hlediska interpolace.

První testovanou je síť "sedlo", generující funkcí je funkce $f(x, y) = 0.5 * (\cos 4x^2 * \sin 4y^2) + 0.5$. Kompletní výsledky pro tuto síť jsou v tabulce 4.7. Zajímavým vedlejším efektem pro interpolační kritérium je, že výsledné vrstevnice jsou "zubaté", oproti neupravené síti i délkovému kritériu. Vrstevnice (vzpočtené pomocí programu [11]) jsou na obrázku 4.7.



Obrázek 4.7: Vrstevnice na síti "sedlo". Zleva: původní, délkové kritérium, interpolační kritérium pro pětiúhelníky

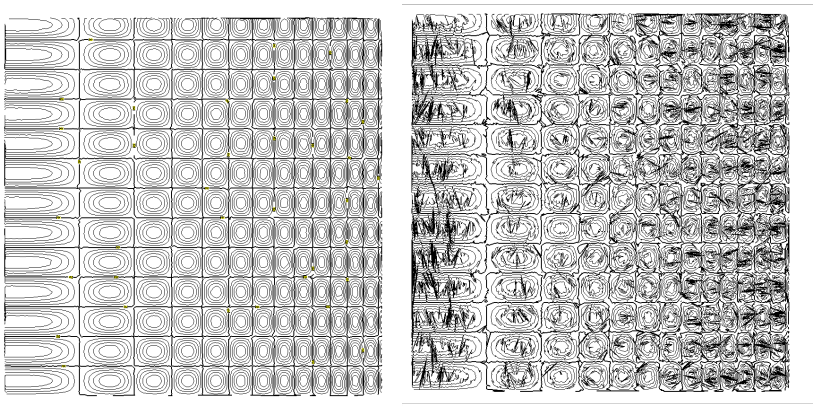
síť	původní váha	nová váha	zlepšení [%]
sedlo	759,78771528	741,35149967	2,486839
špičky	7578,57662446	6680,78735762	13,438375

Tabulka 4.7: Výsledky pro síť "sedlo"a "špičky"

Druhá síť se známou funkcí je síť "špičky", na které jsou velmi razantní změny sklonu terénu tvořícího špičky. Její funkcí je $f(x, y) = 0.5 * (\cos 40x^2 *$

$\sin 40y^2) + 0.5$, síť má 60×10^3 bodů. Z tabulky 4.7 je patrné, že zde interpolační kritérium podalo výsledky podstatně lepší.

Obrázek 4.8 je podobný předchozímu - aplikování interpolačního kritéria výsledné vrstevnice pouze zhoršilo.

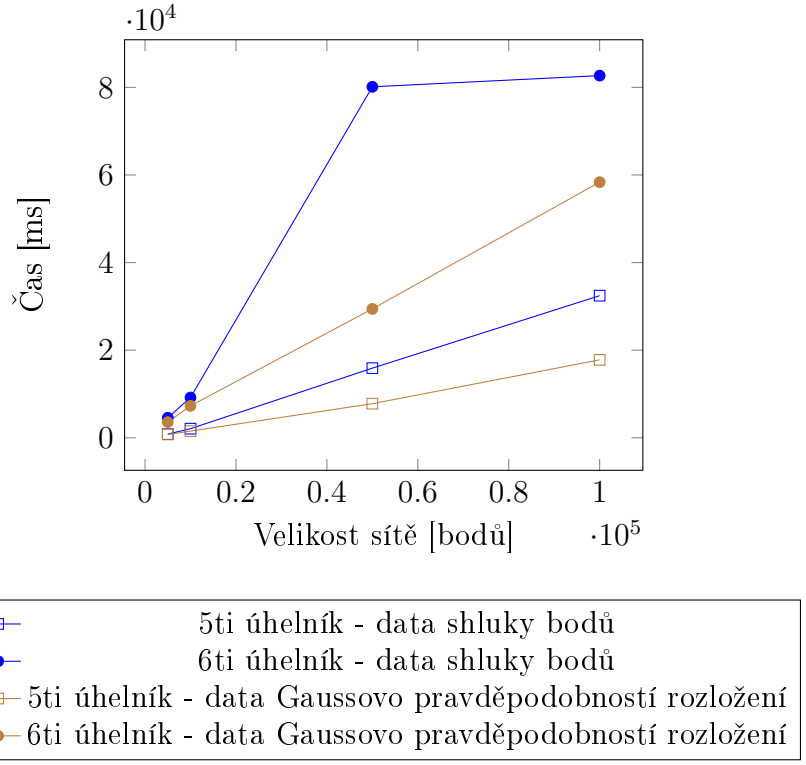


Obrázek 4.8: Vrstevnice na síti "špičky". Vlevo se nachází původní vrstevnice, vpravo pak upravené interpolačním kritériem pro pětiúhelníky

4.3 Experiment 3: Orientační zátěžový test

Cílem toho experimentu bylo (orientačně) otestovat rychlost, jakou se algoritmus vykonává. Vstupem byla data "skupinky vrcholů"a "Gaussovo pravděpodobnostní rozložení"(použitá v části 4.1) postupně o velikosti 5×10^3 až 100×10^3 bodů. V grafu 4.9 je zobrazeno délkové kritérium. Vzhledem k nedostatku vhodných dat se známou funkcí pro interpolační kritérium jsou tyto výsledky zpracovány pouze tabulkou 4.8. Do výsledného času nebyl započítán čas potřebný na aktualizaci aktivních výběrů, protože se přímo netýká testovaného algoritmu.

Výsledky v grafu 4.9 ukazují několik poznatků. Nejvýraznějším poznatkem je, že u šestiúhelníkových skupin se čas potřebný pro výpočet nezvedá lineárně s velikostí sítě, zatímco u pětiúhelníkových skupin tomu tak přibližně je. Dále, že čas pro výpočet silně závisí (obzvláště u šestiúhelníku) na konkrétní síti. Překvapivé je také to, že šestiúhelníková oblast se chová mnohem nevyzpytatelněji. Tyto testy je nutno brát pouze jako orientační.



Obrázek 4.9: Doba výpočtu délkového kritéria pro sítě různých velikostí

	sedlo	špičky
pětiúhelník	2753	38966

Tabulka 4.8: Výsledky pro síť "sedlo"a "špičky", čas uveden v ms

5 Závěr

Po implementování a provedení experimentů se mimo jiné ukázalo, že metoda fungovala pro délkové kritérium na všech datech a její výsledky se značně odvíjí od typu dat. U interpolačního kritéria z důvodu nedostatků v implementaci pro šestiúhelníkové skupiny byly experimenty omezeny pouze na skupiny pětiúhelníkové, které ovšem vykazovaly výsledky dobré. V experimentech došlo u interpolačního kritéria paradoxně ke z horšení kvality vrstevnic, nicméně těchto testů nebylo dostatek k vytvoření závěru v tomto ohledu.

Ač je výpočetně a implementačně podstatně složitější, procházení sítí po šestiúhelníkových skupinách nepřineslo lepší výsledky než procházení pro skupiny pětiúhelníkové.

Také se projevila nepružnost algoritmu procházení sítí v oblasti přidávání pomocných dat kritériem, což způsobilo nedobré výsledky interpolačního kritéria pro šestiúhelníkové skupiny a bylo by vhodné použít způsob jiný.

Reference

[1] G Bobáková, A Ferko, and L Niepel. On minimum weight triangulation. In *Proceedings of the 10th International Conference Spring School of Computer Graphics*, pages 226–232, 1994.

[2] Sunday Dan. Inclusion of a point in a polygon. `http://geomalgorithms.com/a03-_inclusion.html`, 2012.

[3] Ronald Goldman. Intersection of two lines in three-space. In *Graphics Gems*, page 304. Academic Press Professional, Inc., 1990.

[4] Paul S Heckbert. Point in polygon strategies. In *Graphics gems IV*, pages 24–35. Morgan Kaufmann, 1994.

[5] David Kerr. SharpGL. `https://sharpgl.codeplex.com`, 2011.

[6] Ivana Kolingerová. On triangulations. In Antonio Laganá, MarinaL. Gavrilova, Vipin Kumar, Youngsong Mun, C.J.Kenneth Tan, and Osvaldo Gervasi, editors, *Computational Science and Its Applications – ICCSA 2004*, volume 3044 of *Lecture Notes in Computer Science*, pages 544–553. Springer Berlin Heidelberg, 2004.

[7] Ivana Kolingerová, Matyáš Dolák, and Václav Strych. Eliminating contour line artefacts by using constrained edges. *Computers & Geosciences*, 35(10):1975–1987, 2009.

[8] Timo Korinth. Creating a range slider in wpf and silverlight. `http://timokorinth.de/creating-range-slider-wpf-silverlight/`, February 2013.

[9] Gareth Rees. How do you detect where two line segments intersect? `http://stackoverflow.com/a/565282`, February 2009.

[10] Josh Smith. Wpf apps with the model-view-viewmodel design pattern. `http://msdn.microsoft.com/en-us/magazine/dd419663.aspx`, February 2009.

[11] Václav Strych. Výpočet vrstevnic na trohúhelníkové síti, May 2003.

[12] Robert Weibel and Martin Heller. *Digital terrain modelling*. Oxford University Press, 1993.