

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

METÓDY GENEROVANIA GEORELIÉFOV
DIPLOMOVÁ PRÁCA

2022
MARCEL PECKO, Bc.

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

METÓDY GENEROVANIA GEORELIÉFOV
DIPLOMOVÁ PRÁCA

Študijný program: Počítačová grafika a geometria
Študijný odbor: Matematika
Školiace pracovisko: Katedra algebry a geometrie
Školiteľ: RNDr. Róbert Bohdal, PhD.

Bratislava, 2022
Marcel Pecko, Bc.



ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Marcel Pecko

Študijný program: počítačová grafika a geometria (Jednooborové štúdium, magisterský II. st., denná forma)

Študijný odbor: matematika

Typ záverečnej práce: diplomová

Jazyk záverečnej práce: slovenský

Sekundárny jazyk: anglický

Názov: Metódy generovania georeliéfov
Methods of generating georeliefs

Anotácia: Prehľad procedurálnych metód pre vytváranie georeliéfov. Kombinovanie reálnych súradníc bodov zemského povrchu s metódou diamant-štvorec pre generovanie terénu. Využitie splajnových plôch a ich kombinácia s ďalšími procedurálnymi metódami. Prehľad používaných aplikácií pre generovanie georeliéfov. Vytvorenie aplikácie pre generovanie georeliéfov. Porovnanie jednotlivých metód.

Ciel: Opísť metódy pre generovanie georeliéfov. Vytvoriť aplikáciu pre generovanie georeliéfov. Porovnať jednotlivé metódy.

Kľúčové slová: georeliéf, terén, procedurálne modelovanie, fraktál, diamant-štvorec

Vedúci: RNDr. Róbert Bohdal, PhD.

Katedra: FMFI.KAG - Katedra algebry a geometrie

Vedúci katedry: doc. RNDr. Pavel Chalmovianský, PhD.

Spôsob sprístupnenia elektronickej verzie práce:

bez obmedzenia

Dátum zadania: 03.12.2020

Dátum schválenia: 03.12.2020

prof. RNDr. Július Korbaš, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Podakovanie: Ďakujem vedúcemu diplomovej práce RNDr. Róbertovi Bohdalovi, PhD. za usmerňovanie pri písaní práce, jeho cenné rady a celkový dohľad. Podakovanie taktiež patrí doc. RNDr. Andrejovi Ferkovi, PhD. za pomoc popri celom magisterskom štúdiu, jeho dohľadom nad úspešným ukončením štúdia a za poskytnutie pomoci a cenných rád, vždy keď to bolo potrebné. Taktiež mu ďakujem za trpezlivosť, ktorú so mnou mal počas rôznych problémov, ktoré som spôsoboval počas štúdia.

Ďalej by som chcel podakovať svojím kamarátom a rodine za trpezlivosť, ktorú so mnou mali popri písaní práce a rovnako za motiváciu a podporu, ktorú mi poskytovali. Zároveň sa im ospravedlňujem za viaceré odrieknuté akcie, ktorých som sa nezúčastnil kvôli školským povinnostiam a verím, že si to po úspešnom obhájení práce vynahradíme.

Abstrakt

Skúmanie sveta okolo nás a vytváranie jeho virtuálnej reprezentácie je v dnešnej dobe veľmi rozšírenou oblastou počítačovej grafiky. Vďaka širokej škále využitia 3D modelov v rôznych sférach života, sa snažíme neustále zlepšovať dosiahnuté výsledky. Automatizované vytváranie terénov je stále otvoreným problémom počítačovej grafiky. V tejto práci sa venujeme procesu vytvárania digitálnych modelov terénov. Prechádzame historickým vývojom zaznamenávania terénu a vyzdvihujeme jednotlivé prvky, ktoré si našli uplatnenie aj v súčasnosti. V tejto práci oboznamujeme čitateľov s viacerými procedurálnymi prístupmi tvorby terénu a popisujeme ich dôležitosť a jednotlivé výhody. Zameriavame sa najmä na techniku diamant-štvorec, pomocou ktorej si vytvárame chýbajúce dátá terénu, ktoré neboli navzorkované s dostatočnou hustotou. Hlavnými vstupnými dátami našej práce sú reálne zemské súradnice, ale pracujeme aj s vlastnými synteticky vytvorenými dátami získanými zo šumov, 3D skenov a z vhodných matematických funkcií. Výsledkom práce je aplikácia, ktorá používateľom umožňuje generovať zvolený tvar terénu a následne s ním pracovať. Používateelia môžu v aplikácii meniť rôzne parametre vstupu a tým ovplyvňovať výsledný tvar terénu. Môžu do terénu pridávať prvky ako hladina vody, prípadne terén upravovať ďalšími parametrami.

Kľúčové slová: georeliéf, terén, procedurálne modelovanie, fraktál, šum, diamant-štvorec

Abstract

Exploring the world around us and creating its virtual representation is a very widespread area of computer graphics today. Thanks to the wide range of applications of 3D models in various spheres of life, we constantly try to improve the results. Automatized terrain creation is still an open problem in computer graphics. In this work we focus on the process of creating digital terrain models. We go through the historical development of terrain mapping and highlight the individual elements that can be used today. In this work, we acquaint readers with several procedural approaches to landscaping and describe their importance and individual benefits. We focus mainly on the diamond-square technique, which we use to create missing terrain data that has not been sampled with sufficient density. The main input data of our work are real earth coordinates, but we also work with our own synthetically generated data obtained from noise, 3D scans and suitable mathematical functions. The result of the work is an application that allows users to generate the selected terrain shape and then work with it. Users can change various input parameters in the application and thus influence the resulting terrain shape. They can add elements such as the water level to the terrain or modify the terrain with other parameters.

Keywords: georelief, terrain, procedural modeling, fractal, noise, diamond-square

Obsah

Úvod	1
1 Georeliéf a jeho reprezentácia	3
1.1 Motivácia	3
1.2 Klúčové pojmy	4
1.2.1 Formy georeliéfu	6
1.3 História reprezentácie georeliéfu	7
1.3.1 Reprezentácia pomocou máp	7
1.3.2 Reprezentácia pomocou 3D modelov	11
1.4 Digitálne modely	12
1.4.1 Reprezentácie digitálneho modelu	12
1.4.2 Typy digitálnych modelov	14
1.4.3 Metódy získavania reálnych súradníc	16
2 Metódy tvorby digitálnych modelov	23
2.1 Klúčové pojmy	23
2.1.1 Procedurálne techniky	23
2.1.2 Fraktály	24
2.1.3 Náhodnosť	28
2.1.4 Brownov pohyb	29
2.2 Rekurzívne delenie	30
2.2.1 Posúvanie stredného bodu	30
2.2.2 Metóda diamant-štvorec	32
2.2.3 Metóda štvorec-štvorec	33
2.2.4 Trojuholníková metóda	34
2.3 Generovanie pomocou šumu	35
2.3.1 Perlinov šum	36
2.3.2 Simplexný šum	38
2.4 Ďalšie metódy generovania terénu	39

3 Špecifikácia softvérového diela	45
3.1 Požiadavky na aplikáciu	45
3.2 Výber jazyka a platformy	46
3.3 Zber dát	48
3.3.1 Reálne zemské súradnice	48
3.3.2 Vlastné vytvorené dátá	50
4 Implementácia	53
4.1 Knižnice, pluginy, moduly	53
4.2 Triedy	54
4.3 Postprocessing	57
4.4 Vstupné a výstupné dátá	59
4.5 Používateľské rozhranie	60
5 Výsledky práce	63
5.1 Využiteľnosť informácií	63
5.2 Verifikácia a testovanie softvérového diela	64
5.3 Možnosti budúcej práce	69
Záver	71
Prílohy	77

Zoznam skratiek

CHM Canopy Height Model

CPU Central Processing Unit

DEM Digital Elevation Model

DHM Digital Height Model

DSM Digital Surface Model

DTM Digital Terrain Model

fBm fractal Brownian motion

GAN Generative Adversarial Network

GIS Geographic Information System

GPU Graphics Processing Unit

GUI Graphical User Interface

InSAR Interferometric Synthetic Aperture Radar

TIN Triangulated Irregular Network

USGS United States Geological Survey

WinForms Windows Forms

WPF Windows Presentation Foundation

Zoznam obrázkov

1.1	Porovnanie geoidu a elipsoidu/sféroidu	5
1.2	Rôzne formy georeliéfu	6
1.3	Mapy zo 16. storočia	8
1.4	Šrafovanie	8
1.5	Vrstevnice	9
1.6	Rekonštrukcia tvaru terénu z vrstevníc	9
1.7	Hypsometrické odtiene	10
1.8	Tieňovaný reliéf	11
1.9	Nepravidelná trojuholníková siet a jej 3D model	13
1.10	Výšková mapa a jej 3D model	14
1.11	Rozdiel medzi DSM, DTM a DHM	15
1.12	Satelitná interferometria	17
1.13	Obrázok zhotovený technikou interferometrie a jej interferogram	17
1.14	Fotogrametria	18
1.15	Model zhotovený technikou fotogrametrie	19
1.16	LiDAR	19
1.17	Mračno bodov zhotovené technikou LiDAR	20
2.1	Príklady fraktálov vyskytujúcich sa v prírode	25
2.2	Príklady fraktálnych matematických štruktúr	25
2.3	Sierpiňského koberec	26
2.4	Príklad multifraktálu	27
2.5	Príklad rovnomerného a Gaussovho rozdelenia	28
2.6	Príklady funkcií Gaussovho rozdelenia	29
2.7	Ovplyvnenie terénu Hurstovým koeficientom	31
2.8	Metóda posúvania stredného bodu	31
2.9	Metóda diamant-štvorec	33
2.10	Metóda štvorec-štvorec	34
2.11	Trojuholníková metóda	34
2.12	Príklady najznámejších šumov	35

2.13 Objasnenie Perlinovho šumu	36
2.14 Objekt vytvorený zložením viacerých funkcií	40
2.15 Princíp siete GAN	43
2.16 Spojenie údajov výškovej mapy a textúry	44
3.1 Časti USA s rozlíšením 1/9 oblúkovej sekundy	49
3.2 Matematické funkcie s tvarmi zemského povrchu	50
3.3 Objekty skenované skenerom CRUSE	51
4.1 Potrebné balíčky na chod aplikácie	54
4.2 Rozhranie pre prácu s vodnou hladinou	58
4.3 Vstupné a výstupné formáty aplikácie	59
4.4 Prvá časť používateľského rozhrania	60
5.1 Terény generované profesionálnymi softvérmami	64
5.2 Formy georeliéfu vykreslené aplikáciou	65
5.3 Terény vykreslené na základe šumu	65
5.4 Terény vykreslené na základe matematických funkcií	66
5.5 Terén zobrazujúci časť Vysokých Tatier	66
5.6 Rôzne nastavenia parametrov aplikácie	67
5.7 Terén na základe obrázka, ktorý neboli výšková mapa	68
5.8 Terény vyrenderované v aplikácii Blender	68

Zoznam tabuliek

3.1 Prevod najčastejšie používaných oblúkových dĺžok	48
--	----

Úvod

Cieľom tejto diplomovej práce je oboznámiť čitateľov s rôznymi procedurálnymi technikami tvorby georeliéfu. Ukázať ako sa postupom času menil spôsob zaznamenávania terénu a aké možnosti nám ponúka súčasná digitálna doba. Popisujeme spôsob získavania reálnych súradníc bodov zemského povrchu, akou formou je povrch najčastejšie reprezentovaný a následné využitie modelov pri tvorbe georeliéfu. Okrem reálnych súradníc v práci vytvárame aj vlastné syntetické dátá pomocou šumov, 3D skenov a vhodných matematických funkcií.

Problém, ktorý nám nastáva najmä pri reálnych zemských súradničiach je nedostatočné vzorkovanie skutočného terénu. Preto využívame metódu diamant-štvorec, pomocou ktorej chýbajúce dátá vytvárame, vždy keď to potrebujeme.

Súčasťou diplomovej práce je aplikácia, ktorá umožňuje používateľom generovať rôzne typy terénov a na základe zvolených parametrov ich patrične upraviť do želanej podoby. Po vytvorení terénu môžu používatelia meniť rôzne parametre alebo pridať do terénu vodnú hladinu.

Aplikácia je určená pre rôzne cieľové skupiny. Jednou z nich sú bežní používatelia, ktorých zaujíma geometria a procedurálne techniky a chcú si v aplikácii vyskúšať, ako niektoré parametre ovplyvňujú výsledný tvar terénu a získať nové vedomosti a zručnosti. Druhou skupinou sú skúsenejší používatelia, ktorí majú predstavu, aký výsledný typ terénu chcú dosiahnuť, a následne si ho môžu vyexportovať do zvoleného formátu. Exportovaný model terénu následne môžu ďalej využiť pri tvorbe hier, simulácií, tvorbe filmov a v iných odvetviach. Poslednou skupinou používateľov sú programátori, ktorí môžu využiť voľne dostupný zdrojový kód nachádzajúci sa v prílohách.

Celá diplomová práca je členená do piatich kapitol, ktoré združujú dôležité informácie o tvorbe georeliéfu do chronologicky nasledujúcich tematických celkov. Čitatelia tak postupne získavajú potrebné poznatky a vedia samy zreprodukovať využitý postup.

V prvej kapitole sa venujeme georeliéfu a jeho reprezentáciám. Definujeme tu základné pojmy z oblasti tvorby georeliéfu, ktoré potom využívame vo zvyšku práce. Rozdeľujeme tu georeliéf na rôzne formy podľa tvarov, ktoré si potom používateľ má možnosť zvolať v aplikácii. Kapitola pokračuje historickým vývojom reprezentácie georeliéfu od máp po 3D modely. Spomenieme najčastejšie metódy vytvárania jednotlivých reprezentácií. V závere kapitoly sa osobitne venujeme digitálnym modelom, ktoré sú

dôležitou súčasťou celej práce. Opisujeme ich vizualizáciu, ale aj získavanie súradníc na ich vytvorenie. Predstavujeme najčastejšie metódy získavania reálnych súradníc a poukazujeme na ich jednotlivé výhody a nevýhody.

V druhej kapitole oboznamujeme čitateľov s jednotlivými procedurálnymi metódami. Definujeme pojmy ako fraktál a Hurstov koeficient. Popisujeme najčastejšie používame metódy v oblasti počítačovej grafiky, ale ukazujeme aj moderné prístupy, ktoré sú stále predmetom výskumu pomocou neurónových sietí.

V tretej kapitole s názvom *Špecifikácia projektu* popisujeme základné požiadavky na náš projekt. Výber vhodného programovacieho jazyka a platformy a dôvody, prečo sme sa tak rozhodli. Najdôležitejšou časťou tejto kapitoly je zber dát, kde vysvetľujeme ako môžeme získať vstupné dátá. Hovoríme o reálnych zemských súradničiach, ako aj o vlastných syntetických dátach, ktoré vytvárame buď matematickými funkiami alebo ich získavame zo skenov 3D modelov.

Štvrtá kapitola sa venuje implementácii softvérového diela. Popisujeme jednotlivé triedy, knižnice, používateľské rozhranie a výstupné dátá. V poslednej záverečnej kapitole sa venujeme funkčnosti aplikácie. Ukazujeme v nej dosiahnuté výsledky a načrt-neme v nej možné vylepšenia.

Kapitola 1

Georeliéf a jeho reprezentácia

V tejto kapitole predstavíme kľúčové pojmy z oblasti tvorby georeliéfu. Zadefinujeme pojmy reliéf, geoid a vertikálna referenčná hranica, ktoré sú nevyhnutnou súčasťou tejto práce. Popíšeme niektoré formy georeliéfu, ktoré majú zaujímavé tvary, a ktoré budeme v našej aplikácii vizualizovať.

V druhej časti tejto kapitoly si priblížime historiou reprezentácie georeliéfu. Od zaznamenávania povrchu na mapy až po vytváranie 3D modelov. Vyzdvihнемe pri tom najčastejšie používané princípy.

Záver tejto kapitoly venujeme osobite digitálnym modelom. Zameriame sa na reprezentáciu digitálnych modelov, rozdelíme ich na rôzne typy a nakoniec popíšeme najčastejšie používané metódy na získavanie reálnych zemských súradníč.

1.1 Motivácia

Zemský povrch sa v priebehu vekov vyvíja. Vplyvom rôznych okolností ako sú pohyby zemských platení, sopečné činnosti, vplyv vody a vetra, či dokonca dopad meteoritov, sa neustále mení jeho tvar. Prirodzeným inštinktom človeka je poznať prostredie, v ktorom žije a snaha o opísanie a predvídanie jeho ďalších zmien. Postupným vývojom ľudstva sme prešli od slovného opisu prostredia okolo nás, cez jeho grafickú reprezentáciu pomocou máp, až ku komplexnému vytváraniu jeho digitálnych 3D modelov.

Snaha o vytvorenie 3D modelu terénu sa v dnešnej digitálnej dobe nezameriava len na vytvorenie čo najpresnejšej kópie zemského povrchu, ale vytvárame terény aj do rôznych počítačových hier, virtuálnych svetov, pre simulácie prírodných javov a do mnohých ďalších odvetví, kde nemusí ísť o napodobenie reálneho zemského povrchu.

Dnešné technológie nám poskytujú viacero možností vytvárania modelov terénu. Máme vysokokvalitné senzory, ktorými môžeme skenovať členitosť zemského povrchu, výkonné počítače, na ktorých môžeme manuálne modelovať terén a dostatočnú výpočtovú silu procesorov aj pre automatické generovanie terénu pomocou vopred zadefino-

vaných funkcií a pravidiel. Veľkou výhodou toho, že sa stále posúvame technologicky vpred a používame efektívnejsie algoritmy je, že môžeme pracovať s vytvorenými modelmi v reálnom čase a nástroje na prácu s týmito terénmi sa stávajú dostupné aj pre bežného používateľa.

Základnou motiváciou tejto diplomovej práce bolo zhromaždenie informácií o tvorbe digitálnych modelov, od procesu zaznamenávania reálnych súradníc, cez ich reprezentáciu pomocou modelov, až po vlastnú úpravu v aplikácii. Zhrnút čitateľom všetky dôležité informácie na jednom mieste a poskytnúť im priestor na samostatné vyskúšanie modelovania nasledovaním krokov v tejto diplomovej práci. Široká miera uplatniteľnosti vytvorených modelov bola jednou z ďalších motivácií.

1.2 Klúčové pojmy

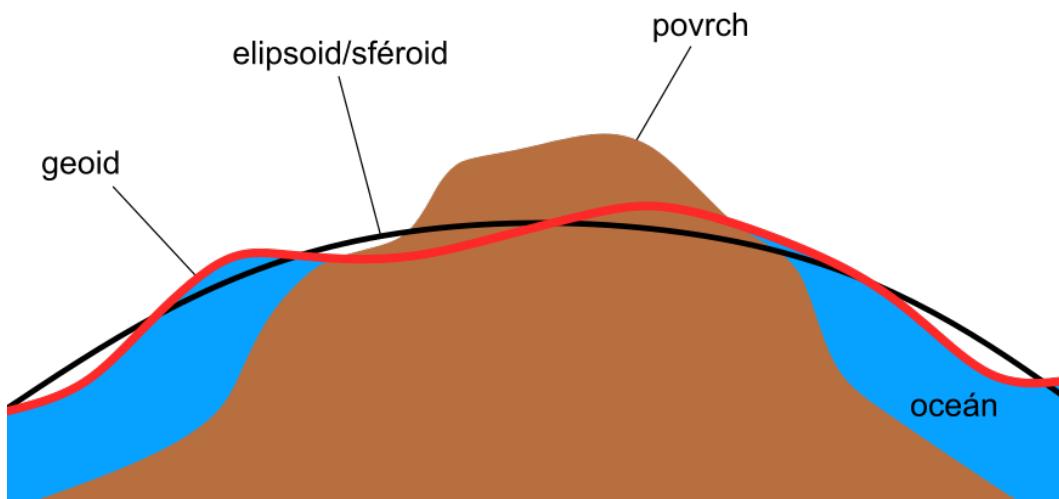
Ako najdôležitejší pojem potrebujeme zadefinovať čo je to povrch. Existuje viacero definícií na základe rôznych kritérií, pretože každá oblasť výskumu potrebuje čo najlepší opis pre svoju prácu. Ľudia povrch bežne chápú ako pevnú plochu, po ktorej chodia. Pri tejto definícii sa ale otvárajú otázky. Máme do definície zahrnúť oblasť pod hladinou mora? Poprípade povrch ľadovcov alebo snehovej prikrývky?

Z matematického hľadiska sa slovo povrch používa na označenie $(n - 1)$ rozmernej podmnožiny n rozmernej množiny [51]. Pre objekt v trojrozmernom Euklidovskom priestore to znamená, že jeho povrch je dvojrozmernou množinou bodov určených tromi súradnicami vo zvolenom súradnicovom systéme. Napríklad na telesie planéty Zem polohu bodu na povrchu určuje zemepisná dĺžka a šírka. Tretia súradnica (výška) definuje postavenie bodu vo vzťahu k Zemi ako celku, teda trojdimenzionálny priestor, v ktorom existuje Zem [19].

Odborným názvom označujeme zemský povrch **reliéf/georeliéf** a v tejto práci pod ním budeme chápať povrhy pevnej hmoty prírodného materiálu, pričom nebudeme zahŕňať ľadovú ani snehovú pokrývku a ani vegetáciu. Príkladom je Digital Terrain Model na obrázku 1.11. Niekedy použijeme aj slovo terén a budeme ho považovať za synonymum k týmto výrazom. Vedná disciplína, ktorá sa zaobrá štúdiom georeliéfu, jeho vznikom a formovaním, sa nazýva **geomorfológia**.

Zemský povrch má zložitý tvar a pre účely meraní sa zvykne approximovať jednoduchšími telesami. Zjednodušená reprezentácia zemského povrchu sa nazýva **geoid**. Je definovaná ako plocha, ktorá má voči gravitácií všade rovnakú potenciálnu energiu. Jej ploche sa najviac približuje stredná hladina morí a oceánov predĺžená popod súš. Stavuje nulovú hladinu nadmorskej výšky. Hlavnou vlastnosťou geoidu je kolmost ľažnice na plochu geoidu kdekoľvek na Zemi, vďaka čomu sa môžu merania výšok vzhľadom na geoid používať globálne.

Ak by naša planéta bola zložená z homogénneho materiálu, jej plocha určená gravitáciou by kopírovala tvar Zeme. Avšak presný výpočet geoidu je nemožný, pretože hustota materiálov sa lýsi v závislosti od meraného územia. Častokrát sa zvykne geoid approximovať jednoduchšími telesami. Najčastejšie je to rotačný elipsoid, sféroid a niekedy dokonca rovina. Vhodné approximácie sa vyberajú na základe veľkosti zobrazovaného územia a ich hlavným účelom je, aby sme mohli vyjadriť povrch analyticky, pomocou matematických funkcií. Approximácia rovinou sa zvykne voliť predovšetkým pre menšie územia, kde je rozdiel zanedbateľný. Porovnanie geoidu a elipsoidu/sféroidu vzhľadom na povrch môžeme vidieť na obrázku 1.1. Vidíme na ňom, že geoid kopíruje hladinu oceánov a ďalej pokračuje popod povrch.



Obr. 1.1: Porovnanie geoidu a elipsoidu/sféroidu

Určovanie polohy bodu na povrchu sa vykonáva vzhľadom na nejaký referenčný systém. Na základe neho sa potom určujú ďalšie hodnoty. Pri navigačných systémoch je dôležité poznať stred systému, čo sa najčastejšie berie ako ťažisko Zeme, aj keď pri lokálnych meraniach môžu byť referenčné body rozdielne. Najčastejšie sa pracuje s polárnymi súradnicami, kde sa zemepisná šírka meria od nultej rovnobežky a dĺžka od nultého poludníka.

Pre nás pri vytváraní modelov, nebude hrať úlohu to, kde sa nachádzajú konkrétné body na Zemi vzhľadom na zemepisnú šírku a dĺžku. Najdôležitejšou veličinou je pre nás výška mapovaných objektov. Preto potrebujeme vedieť, čo pokladá za **vertikálnu referenčnú hranicu** (*vertical datum*) pre určovanie výšky. To sa považuje ako základná nulová referenčná hodnota. Pri väčšine reálnych súradníc, s ktorými pracujeme je za vertikálnu referenčnú hranicu určený geoid. Ale niektoré vedné disciplíny pracujú aj s inými hranicami, takže pri práci s dátami si treba dávať pozor na čo sa odkazujú.

1.2.1 Formy georeliéfu

Jednotlivé tvary zemského povrchu sú ovplyvnené rôznymi procesmi. Geologickými procesmi – *vulkanizmus*, *tektonické pohyby*, *vrásnenie*, ďalej erozívnymi procesmi – *vplyvom vody*, *ľadu*, *vetra*, ale aj mimozemskými procesmi – *dopady meteoritov*. Za milióny rokov teda prešiel zemský povrch viacerými zmenami a dokázal vytvoriť zaujímavé formy.

Pre našu aplikáciu sú dôležité čo najrozmanitejšie tvary, aby si používateľ mohol vybrať povrch, ktorý mu najviac vyhovuje. Neklasifikujeme ich z hľadiska výšky alebo pôvodu, ale zamerali sme sa na tie, ktoré vyzerajú zaujímavo. Na obrázku 1.2 sa nachádzajú rôzne skutočné zemské tvary georeliéfu, z ktorých v našej aplikácii vytvárame digitálne modely. Niektoré z nich vytvárame na základe reálnych súradníc bodov zemského povrchu, iné vytvárame umelo pomocou matematických funkcií a skenov, tak aby odzrkadlovali čo najlepšie skutočnosť. Väčšinu foriem georeliéfu nájdeme v publikácii *Atlas vybraných foriem georeliéfu* [8] aj spolu s ich zaradením, charakteristikou a výskytom.



Obr. 1.2: Rôzne formy georeliéfu [39]

Týchto 12 foriem z obrázku 1.2, by malo čo najlepšie pokryť potreby používateľa aplikácie. Majú význačné tvary využiteľné v rôznych oblastiach. Niektoré tvary popíšeme, aby boli jasnejšie poznatelné rozdiely medzi mini. Pri kotlinie je rovinné územie zo všetkých strán obklopené pohoriami, naroziel od doliny, ktorú tiež obklopujú hory, ale nie je zo všetkých strán uzavretá. Tiesňava sa zaraďuje medzi doliny, ale je význačná svojím tvarom V. Úvaliny sú zaujímavé vlnovitým tvarom terénu. Podobne zaujímavý tvar vytvára forma thufur, ktorá je zložená z množstva malých kopčekov. Forma maar má tvar podobný stratovulkánu, ale leží pod úrovňou zemského povrchu a je naplnená vodou.

1.3 História reprezentácie georeliéfu

S prirodzeným vývojom ľudstva a zdokonalovaním dostupných technológií sa postupne menili aj spôsoby zaznamenávania okolitého prostredia. Najjednoduchšou, aj keď najviac subjektívou a nepresnou metódou, bol len jeho slovný opis. Keďže neposkytoval vizuálne informácie, tvar pre ďalšieho používateľa závisel na jeho predstavivosti, od subjektívnej interpretácie slovného prejavu. Takáto forma neposkytovala dostatok užitočných informácií a teda nemala veľké využitie.

Spomedzi našich piatich zmyslov prijíma človek najviac informácií pomocou zraku. Je teda prirodzené, že sa snažíme spracovať údaje predovšetkým vizuálne. Za prvé znázornenie základných tvarov krajiny môžeme považovať jaskynné maľby. No ich výpovedná hodnota je obmedzená vzhladom na nízku geometrickú presnosť. Maľby neboli podložené skutočným meraním a poskytovali len tvarové informácie. Postupom času a vývojom technológií prešlo ľudstvo cez 2D reprezentáciu pomocou máp až ku 3D modelom a ich zobrazovanie na zariadeniach ako sú počítače, tablety a zariadenia virtuálnej a rozšírenej reality.

1.3.1 Reprezentácia pomocou máp

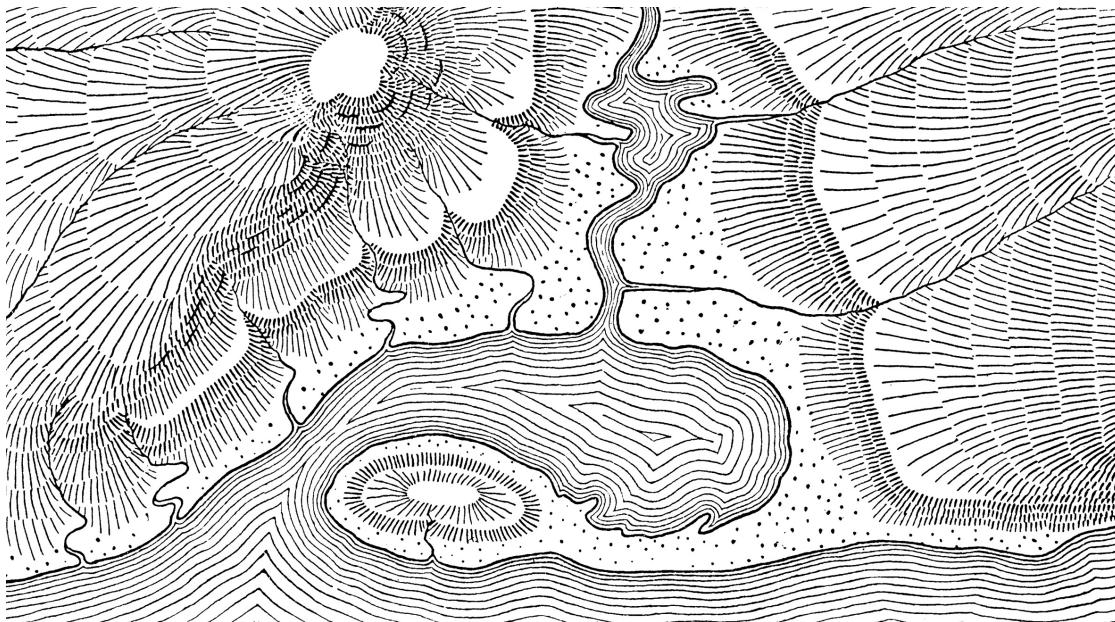
Znázorňovanie zemského povrchu pomocou máp je jednou z najstarších oblastí reprezentácie. Vyvinul sa aj vlastný vedný odbor – kartografia. Jeho názov pochádza z gréčtiny a je zložený z gréckych slov *chartis* (*mapa*) a *graphein* (*kresliť*). Za stáročia existencie odboru prešlo zaznamenávanie rôznymi zmenami, ktoré ovplyvňovali aj dostupné technológie danej doby.

Najskôr išlo len o jednoduché zaznamenávanie hôr a kopcov z profilu. Tieto mapy poskytovali len približný tvar terénu. Kým sa nezačali mapy popisovať prvkami, ktoré mali skutočné číselné ohodnotenie založené na meraní a výpočtoch, poskytovali mapy skôr estetickú ako funkčnú hodnotu. Príklad takýchto máp, ktoré zaznamenávali len tvarové informácie vidíme na obrázku 1.3.



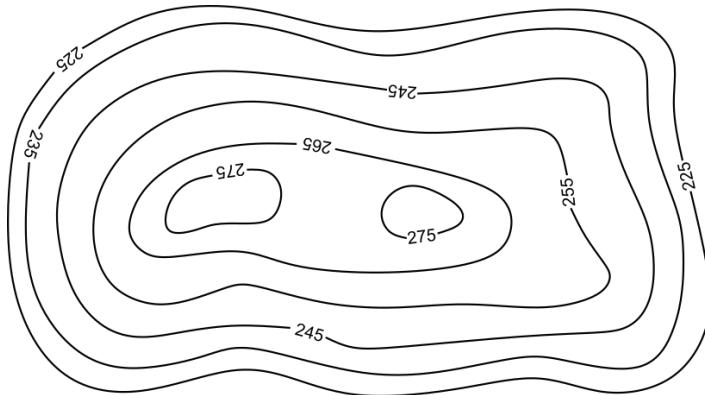
Obr. 1.3: Príklady máp zo 16. storočia [30]

Šrafovanie (*hachures*) je technika tieňovania pomocou čiar – krátkych úsečiek alebo krviek. Smer čiar ukazuje, akú orientáciu má svah, dĺžka čiar sa líši podľa dĺžky svahu, ich hrúbka navodzuje pocit strmosti a hustota čiar môže určovať stupeň sklonu. Keďže tieto čiary nemajú za sebou žiadny číselný podklad, sú menej užitočné ako napríklad vrstevnice, no napriek tomu sa nimi dajú dostatočne dobre popisovať tvary terénu. Príklad tejto techniky môžeme vidieť na obrázku 1.4. Využívali sa predovšetkým pri zobrazovaní nízkeho zvlneného reliéfu. Mierny svah alebo rovinatá oblasť zvyčajne zostáva prázdna bez čiar. V 20. storočí bola táto technika štandardom pre topografické mapy Nemecka. Zvykla sa kombinovať aj s inými typmi máp a vytvorila sa aj farebná modifikácia so zelenými čiarami pre nížiny a hnedými pre vysočiny. S technikou prišiel v roku 1799 rakúsky topograf Johann Georg Lehmann [11].



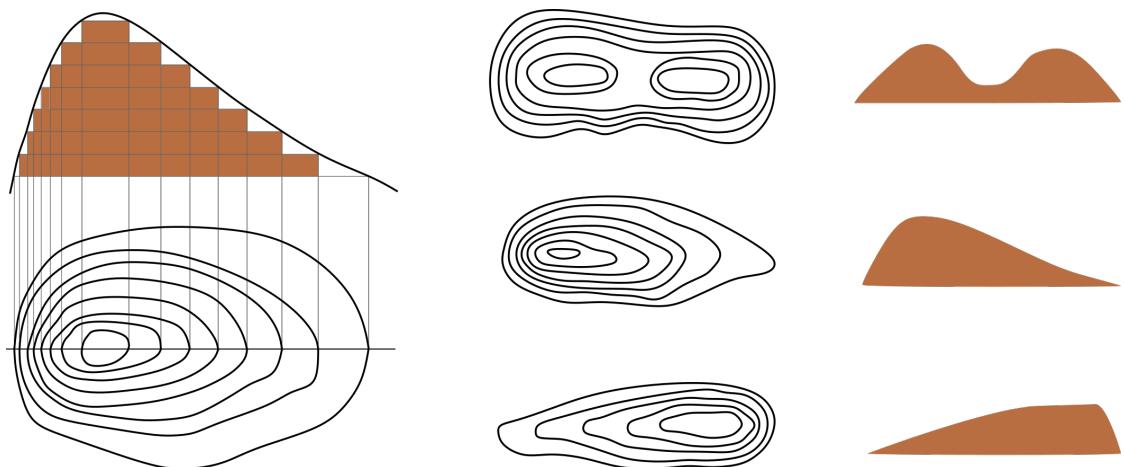
Obr. 1.4: Príklad techniky šrafovania [18]

Vrstevnice (*isohypsес*) je ďalší druh čiar, ktoré sa používajú na popis terénu. Ide o jednu z najpoužívannejších techník, pretože vyjadrujú nielen tvar, ale aj nadmorskú výšku terénu. Vrstevnice sú krivky, ktoré spájajú body rovnakej nadmorskej výšky. Matematicky je vrstevnica izokrívka, pozdĺž ktorej má funkcia konštantnú hodnotu, v našom prípade nadmorskú výšku. Najčastejšie sa vrstevnice kreslia v pôdoryse, teda akoby sa pozorovateľ díval na terén zhora. Príklad takejto mapy môžeme vidieť na obrázku 1.5. Ak sú vrstevnice blízko seba, tak je terén v tejto časti strmý. To znamená, že veľkosť gradientu je v tejto časti veľká. Dôležitou súčasťou vrstevníc je ich číselné označenie, ktoré určuje výšku a rýchlejšie nám pomáha predstaviť si tvar terénu. Avšak číslenie má určité pravidlá. Čísla sú otočené tak, aby pri ich čítaní používateľ mapy smeroval do kopca/údolia. Čísla by sa mali umiestňovať pozdĺž mierne zakrivenej čiary a ak je to možné z viacerých smerov, aby sa dali mapy rýchlejšie čítať.



Obr. 1.5: Príklad vrstevníc spolu s číselným označením

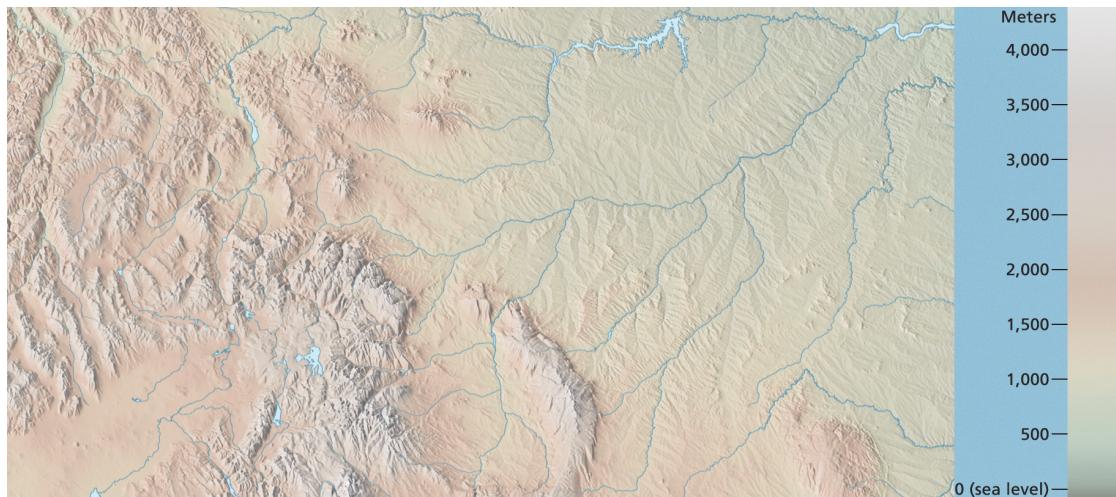
Technika je obľúbená, pretože rýchlo a jasne poskytuje potrebné informácie. Na obrázku 1.6 môžeme vidieť, ako ľahko môžeme zrekonštruovať výsledný tvar terénu len na základe niekolkých čiar ak poznáme výškový rozdiel medzi nimi. Výškový rozdiel medzi dvomi po sebe idúcimi vrstevnicami sa nazýva interval vrstevníc.



Obr. 1.6: Rekonštrukcia tvaru terénu z vrstevníc

Dôležitú úlohu v zobrazovaní terénu môžu hrať okrem čiar aj farby. Pojmom **hypsometrické odtiene** (*hypsometric tints*) označujeme farby, ktoré sa používajú na znázornenie nadmorskej výšky. Terén sa pri tejto technike zobrazuje rôznymi farebnými pásmi, ktoré sú odstupňované na základe výšky. Existujú dva prístupy, a to buď diskrétnie zafarbovanie jednotlivých častí príslušnými farbami, alebo spojity prechod od jednej farby k druhej. Tieto farby sa zvyčajne nazývajú aj falošné, pretože neodzrkadlujú skutočné farby zemského povrchu.

Pri výbere farebnej škály musíme dodržiavať určité pravidlá. Farby by nemali byť príliš živé, mali by byť tlmené, aby pri nich vynikol reliéf a nezakrývali ďalšie symboly na mape. V rôznych nadmorských výškach by sa nemal opakovať rovnaký odtieň farby, preto sa vo farebných škálach zvyčajne používa 6 až 10 rôznych farieb. Výrazné farby sa používajú vo vrchných zónach, aby boli jasne odlišiteľné. Najväčším problémom je výber konkrétnych farieb. Diskusie na túto tému sa vedú už niekoľko rokov a vytvorilo sa viacero farebných škál. Ich podrobný opis sa nachádza v 13. kapitole knihy *Cartographic relief presentation* od kartografa Eduarda Imhofa [30]. v súčasnosti najpoužívanejšou škálou je škála, kde je vrch zobrazený sýto hnedou farbou, a postupne prechádza naspodok ku sýto-modrej alebo šedo-zelenej. Príklad hypsometrickej mapy môžeme vidieť na obrázku 1.7.



Obr. 1.7: Hypsometrická mapa so stupnicou [41]

Ekvivalentom ku hypsometrickým odtieňom sú batymetrické odtiene (*bathymetric tints*), ktorými sa vyjadrujú rozdiely v hĺbke – mapuje sa nimi reliéf morského dna. Princíp je úplne rovnaký - čím sme hlbšie, tým je farba tmavšia. Zvyčajne sa používajú odtiene modrej. Problémom pri mapách s hypsonickými odtieňmi je, že ľudia si pri pohlade na mapu robia asociáciu medzi nadmorskou výškou a prirodzenou vegetáciou. To znamená, že napríklad v púštnych oblastiach je zelená farba, aj keď tam očakávajú žltú reprezentujúcu suché oblasti. Niektoré mapy preto kombinujú hypsometrické farby s vegetačnými alebo pridajú ku mapám viacero farebných škál pre jednotlivé oblasti.

Technika **tieňovaný reliéf** (*shaded relief*) sa snaží zobraziť tvar terénu tak, akoby vyzeral trojrozmerný povrch osvetlený bodovým zdrojom svetla. Zdroj svetla sa umiestňuje do ľavého horného rohu. Teda ak je mapa orientovaná na sever, tak svetlo prichádza zo severozápadu. Použitie južného zdroja by mohlo spôsobiť ilúzii viacstupňového vnímania a model by sa javil ako prevrátený. Matematický základ takého prístupu je vypočítanie normálového vektora v každom bode, a následným vypočítaním uhlu medzi týmto vektorom a vektorom smerujúcim k bodovému zdroju svetla. To sa dá jednoducho docieliť pomocou skalárneho súčinu. Čím menší je uhol týchto dvoch vektorov, tým viac je dané miesto osvetlené. Príklad môžeme vidieť na obrázku 1.8, kde sa v ľavej časti nachádza mapa oblasti Lake Mead a v pravej časti rovnaká oblast s použitím techniky tieňovania. Tieňovaný reliéf sa kedysi kreslil s dreveným uhlím. Dnes sa už generuje počítačom z digitálnych výškových modelov.



Obr. 1.8: Mapa bez a s použitím tieňovania reliéfu [36]

Popísali sme najčastejšie techniky, ktoré sa využívajú pri tvorbe máp. Samozrejme ich existuje oveľa viac. Proces akým sa postupne menili mapy aj s príkladmi nájdeme v knihe *Cartographic relief presentation* [30].

1.3.2 Reprezentácia pomocou 3D modelov

V dobách, keď ešte neexistovali počítače, využívali ľudia všetky dostupné materiály na to, aby vytvorili čo najlepšiu vizualizáciu svojho okolia. Prvé modely okolia vytiesávali do kameňa. Časom začali používať materiály, s ktorými sa im ľahšie manipulovalo. Napríklad v Číne modelovali hory a údolia pomocou zrniek ryže. Ďalšími používanými materiálmi pri vytváraní modelov boli piliny, včelí vosk, pšenica alebo drevo. Drevené modely mali veľkú obľubu a zaznamenávali celé provincie. Modely sa vytvárali po častiach, ktoré sa potom spájali ako skladačka. Po druhej svetovej vojne, s príchodom vákuovo tvarovaných plastových máp, nastala masová výroba modelov z plastu. Princípom bolo zahriatie plastovej fólie na tvarovateľnú teplotu, a potom sa pritlačila na formu pomocou vákua. Následne sa modely zafarbili na základe mierky. Farbili sa napríklad pomocou hypsometrických odtieňov.

V súčasnosti sú rozšírené predovšetkým digitálne modely. Vďaka digitálnej reprezentácii terénu sa otvorili viaceré možnosti, pretože s takýmito modelmi dokážeme ľahšie pracovať. Sú efektívnejšie z hľadiska ukladania, prenosu, analýzy a dnešné zariadenia poskytujú ich okamžitú vizualizáciu. Podrobne sa im venujeme v nasledujúcej časti Digitálne modely 1.4.

1.4 Digitálne modely

V tejto časti podrobnejšie popíšeme digitálne modely reliéfu. Najskôr v úvode priblížime reprezentáciu modelu. Potom zadefinujeme jednotlivé typy modelov, ich rozdiely a využitie. V závere tejto časti sa budeme venovať reálnym súradniciam, ktoré sú potrebné na vytvorenie modelu a jednotlivým metódam ich zaznamenávania.

1.4.1 Reprezentácie digitálneho modelu

Digitálne modely môžu mať viaceré reprezentácie. Niektoré presnejšie aproximujú zaznamenávaný terén, ale je ich obtiažne vizualizovať, iné sú jednoduchšie na spracovanie, ale strácajú niektoré informácie. My predstavíme najčastejšie reprezentácie digitálnych modelov, ktoré sa využívajú s reálnymi zemskými súradnicami pri mapovaní povrchu. Reprezentácie, ktorým sa nebudeme bližšie venovať sú pomocou funkcií a objemov, pretože sú pre našu prácu irelevantné.

Mračno bodov

Mračno bodov (*point cloud*) môžeme považovať za dátovú štruktúru, ktorá sa používa na reprezentáciu priestorových objektov a tvarov. Ide o množinu 3D bodov, ktoré majú zadefinovanú svoju polohu v priestore a môžu niesť informácie o ďalších vlastnostiach ako sú farba, intenzita alebo akékoľvek ďalšie abstraktné informácie [50]. Mračná bodov sa najčastejšie vytvárajú pomocou laserov a 3D skenerov, ktoré merajú veľké množstvo bodov na vonkajších povrchoch objektov.

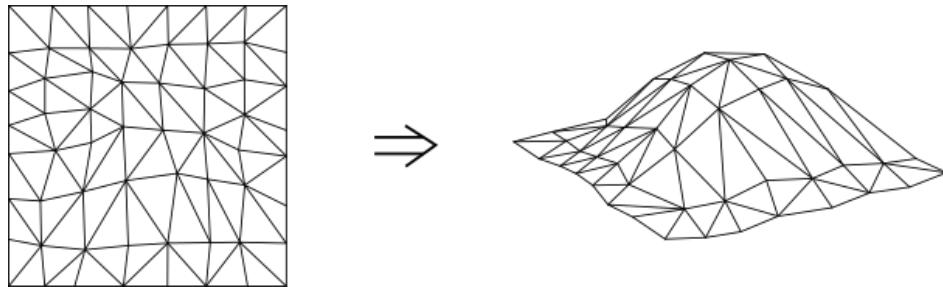
Pri terénoch nie sú mračná bodov vždy najlepší spôsob zobrazenia terénu, preto sa pri vykreslovaní konvertujú na modely polygónovej alebo trojuholníkovnej siete alebo na povrchové modely NURBS. Pri konvertovaní bodov na povrch sa zvyknú používať algoritmy ako Delaunayova triangulácia, alfa tvary (*alpha shapes*), otáčanie guľôčky (*ball pivoting*) alebo pomocou algoritmu pochodujúcich kociek (*marching cubes*). Príklad mračna bodov, ktoré je zhotovené z reálnych súradníc zachytávaných pomocou lasera môžeme vidieť na obrázku 1.17.

Nepravidelná trojuholníková sieť

Nepravidelná trojuholníková siet – TIN (*Triangulated Irregular Network*) je tvorená neprekrývajúcimi sa trojuholníkmi, ktoré majú spoločnú nanajvýš jednu hranu [29].

Vytváranie TIN sa väčšinou deje z diskrétnych bodov získaných z mračna bodov. Najbežnejší triangulačný algoritmus je Delaunayova triangulácia. Hustota bodov nie je na celom teréne rovnaká, pretože sa volia vzhľadom na tvar terénu. Väčší počet bodov sa nachádza na miestach, kde je terén členitejší. Pri vhodnej voľbe bodov approximuje skutočný terén lepšie ako rastrový model.

Modely TIN majú v porovnaní s výškovými mapami väčšie požiadavky pri ukladaní a sú komplikovanejsie na indexovanie. Výhodou trojuholníkov je ich rýchle zobrazenie. Kedže body trojuholníka definujú rovinu, tak je jednoduché v prípade potreby interpolovať výšku v každom bode trojuholníka. Príklad nepravidelnej trojuholníkovej siete a model, ktorý reprezentuje môžeme vidieť na obrázku 1.9.



Obr. 1.9: Nepravidelná trojuholníková siet a jej 3D model

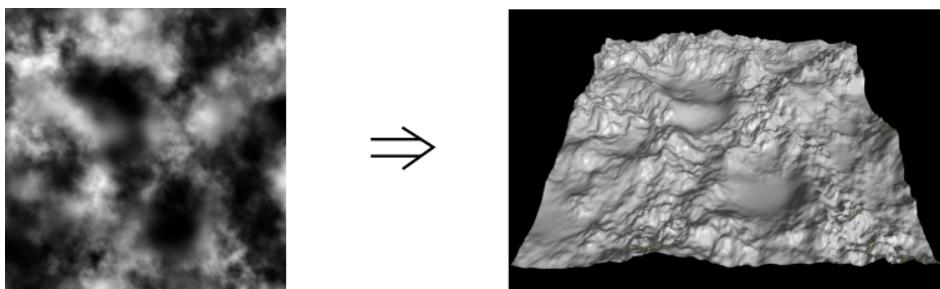
Rastrový model

Rastrový model je tvorený pravidelnými štvorcovými plôškami so spoločnými hranami. Tieto plôšky nazývame rastre a nesú v sebe informáciu o výške. Body majú medzi sebou konštantné rozostupy, čo umožňuje jednoduchšie počítačové spracovanie, a následnú vizualizáciu. Rastrový model je v podstate štvorcová sieť, ktorá vzniká pravidelným navzorkovaním bodov z mračna bodov. Jedna z možností, ako to urobiť je popísaná v článku *From point cloud to grid DEM: A scalable approach* [2]. Princíp využíva štruktúru quadtree, pričom nájde spoločné body v jednotlivých bunkách a urobí interpoláciu. Nevýhodou rastrového modelu je potreba väčšieho počtu bodov v porovnaní s TIN modelom, pretože voľba bodov sa neprispôsobuje skutočným potrebám tvaru. Niekde môže byť počet bodov nadbytočný, inde nedostatočný.

Rastrový model je pomenovanie štruktúry. Na jeho uloženie sa používajú výškové mapy (*heightmap*). Ide o najčastejší a najjednoduchší spôsob reprezentácie terénu. Je to dvojrozmerný rastrový obrázok, kde hodnota každého pixelu reprezentuje nadmorskú výšku povrchu v danom bode [13]. Nevýhodou tejto reprezentácie je, že v danom pixely môžeme vždy zachytiť len jednu výšku, takže ak by sa v teréne nachádzali jaskyne alebo

previsy, tak tie nie sme schopní reprezentovať. Výškové mapy sa zvyčajne nevykresľujú priamo, ale vykresľujú sa ako siete, najčastejšie štvoruholníkov alebo trojuholníkov. Sú obľúbené pre svoju výpočtovú efektivitu a jednoduché použitie v zložitých algoritnoch. Preto ich používa väčšina algoritmov pre generovanie terénu.

Výškové mapy sú najčastejšie v odtieňoch sivej, pričom biela predstavuje maximálnu výšku a čierna farba minimálnu. Takýto príklad je na obrázku 1.10. Jeden farebný kanál predstavuje iba 256 rôznych úrovní výšky, čo je pri potrebe väčších detailov nedostačujúce. Preto sa zvyknú používať všetky farebné kanály RGB. Ak sa využije aj alfa kanál, môžeme dostať až $256^4 = 4294967296$ možností čo je výrazne viac.



Obr. 1.10: Výšková mapa a jej 3D model

Ak by sme chceli príliš veľkú úroveň detailov, museli by sme uložiť výškovú mapu s veľkým rozlíšením. Ďalšou nežiaducou vlastnosťou výškových máp je vopred pevne určená úroveň detailov, ktorá je daná vzdialenosťou medzi rastermi. Ak sa priblížime príliš blízko k modelu terénu, stane sa lokálne plochým a nezaujímavým. Tieto problémy riešime v diplomovej práci.

1.4.2 Typy digitálnych modelov

Pri práci s reálnymi zemskými súradnicami musíme brať do úvahy to, aké dátá o povrchu nás zaujímajú. Pretože digitálne modely s reálnymi súradnicami môžu mať využitie vo viacerých oblastiach, vytvárajú sa rôzne typy. Ide o štrukturovaný súbor dát v tvare priestorových súradníc, ktoré sa od seba líšia podľa ponímania výšky v modeli. Ako vertikálna referenčná hranica sa najčastejšie berie geoid. V dnešnej dobe dobývania vesmíru už ale nejde výlučne o terény na Zemi, ale aj na iných planétach, mesiacoch alebo asteroidoch, takže referenčné hranice sa často vzťahujú na danú oblasť. V niektorých modeloch sú zahrnuté všetky objekty, ktoré sa na povrchu nachádzajú, v iných je zaznamenaný len tvar terénu bez vegetácie.

Digitálny model terénu – DTM (*digital terrain model*) predstavuje holý povrch bez akýchkoľvek objektov, akými sú napríklad budovy alebo vegetácia. Modely DTM sa využívajú na modelovanie krajiny, modelovanie miest a ďalšie vizualizačné aplikácie, akými sú predpovedanie rizika záplav a vymedzenie oblastí povodní.

Digitálny model povrchu – DSM (*digital surface model*) predstavuje zemský povrch spolu so všetkými prvkami, ktoré sa na ňom nachádzajú. Zahŕňa teda vegetáciu a aj umelé objekty vytvorené človekom. Tieto typy modelov sú užitočné napríklad pre plánovanie komunikačných systémov, pri inštalovaní vysielačov alebo pre letectvo, pretože sú v nich skutočné prírodné aj umelé prekážky v reálnych výškach.

Digitálny výškový model – DHM (*digital height model*) vyjadruje relatívnu výšku objektov k povrchu a nie hladiny mora alebo iných vertikálnych hraníc. Zobrazuje teda výšku všetkých objektov nachádzajúcich sa na zemskom povrchu a môžeme ho získať ako rozdiel modelov DSM a DTM. Používa sa na modelovanie záplav, modely miest, rôzne štúdie prekážok alebo archeologický výskum. V niektornej literatúre sa nachádza tento model pod pojmom CHM (*canopy height model*).

Všetky modely sa často využívajú v geografických informačných systémoch – GIS (*geographic information system*) a sú najbežnejším základom pre digitálne mapy reliéfu. Rozdiel medzi vyššie spomenutými digitálnymi modelmi je znázornený pre lepšiu predstavu na obrázku 1.11.



Obr. 1.11: Rozdiel medzi DSM, DTM a DHM

Digitálny výškový model – DEM (*digital elevation model*) v slovenčine nesie rovnaký názov ako DHM, a preto budeme v texte používať skratky, aby nedošlo k dezinterpretácii. Oproti zvyšným pojmom v tejto časti je trochu zložitejší. Má totiž viacero významov a v závislosti od geografickej oblasti, v ktorej ho používame, môže znamenať niečo iné. Jedna z definícií hovorí, že pojmom DEM sa myslia modely DTM. V tom prípade preberáme ich definíciu a ide o model, kde nie sú žiadne prírodné ani umelé prvky, len holý povrch. My budeme chápať tento pojem ako všeobecný termín pre DSM a DTM. Teda, keď povieme DEM modely, nepredstavíme si konkrétny typ, ale zastrešenie všetkých typov. Z hľadiska údajov sa používa pojem DEM ako 3D grafická reprezentácia výškových dát, ktoré sú usporiadané v pravidelnej mriežke – v rastroch. Takže aj pri stahovaní údajov budú mať pravdepodobne takúto štruktúru. Vo všeobecnosti môžu byť modely zložené aj z nepravidelne alebo náhodne rozmiestnených bodov spojených pomocou TIN, alebo iných dátových štruktúr. Jednotlivé definície sú z knihy *Lidar Base Specification* [28], kde sa nachádzajú aj ďalšie informácie o modeloch spolu s obrázkami.

Existuje viacero voľne dostupných zdrojov, kde si môže bežný používateľ bezplatne stiahnuť vyššie spomenuté modely a ďalej s nimi pracovať. Niektoré z nich sa nachádzajú v článku *5 Free Global DEM Data Sources* [24]. Okrem DEM údajov Zeme sa tam nachádzajú aj údaje o Marse, ktoré získavali pri mapovaní starých tokov riek a oceánov na tejto planéte. Výšky určovali na základe marťanského gravitačného pola.

1.4.3 Metódy získavania reálnych súradníc

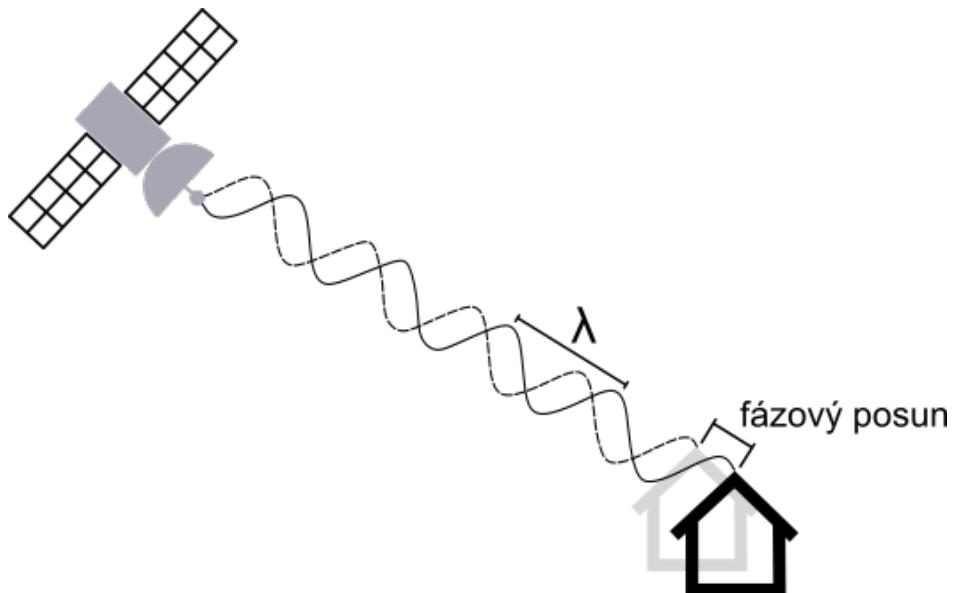
Aby mohli vzniknúť modely spomenuté v predchádzajúcej časti 1.4.2, potrebujeme najskôr získať vhodné dátá. Na získanie reálnych súradníc sa používa viacero techník. Získavanie dát priamo v teréne pomocou geodetického merania považujeme za techniky, ktoré majú veľkú presnosť, ale nedokážu pokryť veľké oblasti a je obtiažne získať dátá na ľahko dostupných miestach. Využívajú sa pri nich GPS, totálne stanice, laserové skenery a iné pozemné technológie. My sa zameriame na techniky, ktoré dokážu získať väčšie množstvo dát a sú schopné spracovať aj ľahko dostupné miesta.

Satelitná interferometria

Satelitná interferometria je jedna zo základných techník zaznamenávania reálnych tvarov vo veľkej mieri. Najčastejšie sa pri nej využíva interferometrický radar so syntetickou apertúrou - InSAR (*Interferometric Synthetic Aperture Radar*). Jej hlavnou výhodou je možnosť vytvárať snímky s vysokým rozlíšením za každého počasia a nezávisle od časti dňa (deň/noc), čo je pri iných technikách obtiažne. V súčasnosti je to jedna z najrýchlejšie rozvíjajúcich sa oblastí diaľkového prieskumu Zeme.

Techniku môžeme prirovnáť k echolokácii, ktorú využívajú netopiere pri navigácii v jaskyni. Zo satelitu sa vyšľú na Zem mikrovlnné impulzy, ktoré sa následne odrazia od povrchu a vrátia sa späť ku satelitu. V závislosti od zvolenej vlnovej dĺžky pulzov vedia prenikáť cez rôzne prvky. Preto je táto technika použiteľná za akéhokoľvek počasia, pretože pri vhodnej vlnovej dĺžke vedia pulzy preniknúť oblakmi a niektoré dokážu dokonca preniknúť aj do zeme.

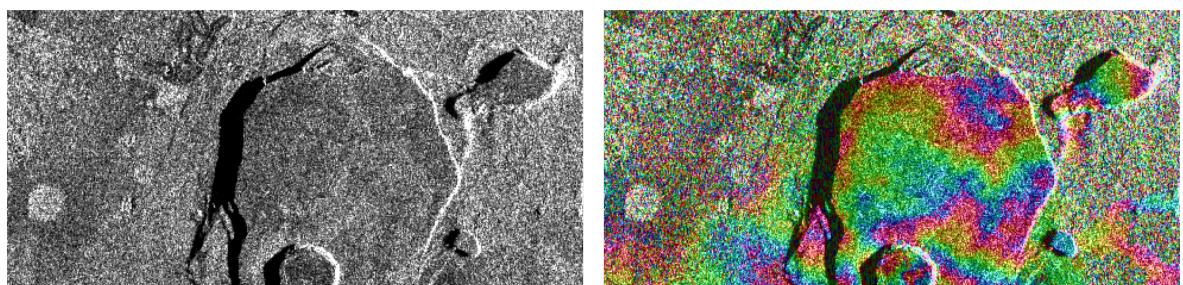
Pri vysielaní impulzov je známa ich fáza a tá sa porovnáva s fázou spätného signálu. Vypočítá sa fázový posun s vracajúcou vlnou a na základe času sa určí celková vzdialenosť k satelitu. Fáza spätného signálu môže byť ale ovplyvnená viacerými faktormi a môže dochádzať k nepresným informáciám. Z toho dôvodu sa nevyužívajú spätné vlny, ale vytvárajú sa radšej dve snímky tej istej oblasti. Princíp satelitnej interferometrie je zobrazený na obrázku 1.12. Písmeno lambda znázorňuje vlnovú dĺžku, čo je zároveň bežné vyjadrenie vlnovej dĺžky vo fyzike. Pri technike sa bud využijú dve antény, ktoré snímajú oblasť pod uhlom alebo sa využije jeden satelit. Pri zhotovení druhej snímky sa čaká niekoľko dní, kým prejde satelit po svojej obežnej dráhe a vráti sa na rovnaké miesto.



Obr. 1.12: Satelitná interferometria

Z vráteného signálu sa na základe prijatých informácií vytvárajú snímky. Každý pixel snímky vytvorenej satelitom nesie dve informácie – intenzitu a fázu. Farba pixelov závisí od toho ako sa pulzy pri dopade na zem rozptyľujú. Pri dopade vlny na povrch vzniká zrkadlový, difúzny a dvojitý odraz. Ku zrkadlovému odrazu dochádza pri hladkých povrchoch, ako sú cesty alebo vodné toky. Toto sa vyznačuje tmavo-čiernymi pixelmi. Pri drsnom povrchu, ktorý je typický pre polia a vegetáciu, dochádza k difúznomu rozptylu a takéto povrchy sa označujú sivou farbou. Najjasnejšou bielou farbou sa označujú povrchy, kde dochádza k dvojitému odrazu. Takéto povrhy sú budovy alebo iné umelo vytvorené objekty.

Technika nesie názov interferometria, pretože pri jej procese sa vytvárajú interferogramy. Interferogram je digitálne vytvorený obrázok, ktorý znázorňuje posuny zemského povrchu. Je to rozdiel fázových hodnôt určitej oblasti v rôznych časoch. Príklad snímky, ktorú vytvára radar ako aj jej interferometrická reprezentácia, je zobrazená na obrázku 1.13. Obrázky vytvorené týmto satelitom si netreba zamieňať s obrázkami výškových máp, ktoré sú v odtieňoch šedej a na tieto obrázky sa podobajú.

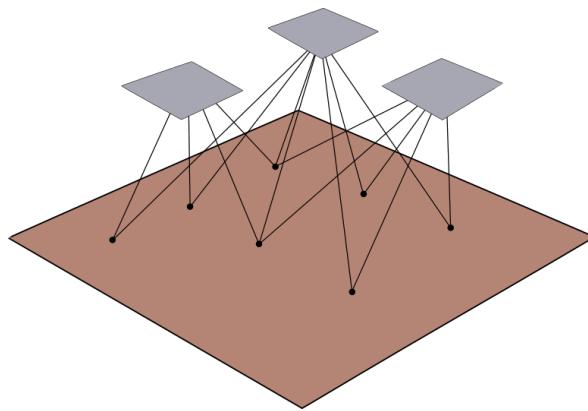


Obr. 1.13: Obrázok zhotovený technikou interferometrie a jej interferogram [32]

Proces vytvárania DEM dát z fotografií vytvorených pomocou InSAR je nesmierne náročný, pretože spracováva veľa údajov a prebieha v niekoľkých krokoch. Podrobnosti o všetkých krokoch spracovania ako aj počítaní fázových rozdielov vln jednotlivých snímok sa dočítame v článku *InSAR processing: a Mathematical Approach* [44]. Ďalšie príklady použitia radaru ako aj to, aké ďalšie informácie môže radar poskytnúť sa nachádzajú v článku *Learn Synthetic Aperture Radar (SAR) by Example* [26].

Fotogrammetria

Druhá z najčastejšie používaných techník sa volá fotogrammetria (*photogrammetry*). Tá využíva na získanie zemských súradníc fotografie z minimálne dvoch rôznych pohľadov. Funguje teda na princípe nášho zraku, pretože získava informácie o hĺbke a perspektíve vďaka oddeleným bodom pozorovania. Najskôr sa spraví veľké množstvo fotografií nad určitou oblasťou. Vzniknuté obrázky sa potom vo vhodnom počítačovom programe prekryjú tak, že ten istý bod na zemi je viditeľný na viacerých fotografiách z rôznych pohľadov. Príklad môžeme vidieť na obrázku 1.14. Na zemskom povrchu sa zvyknú vytvárať kontrolné body podľa známych geografických súradníc, pomocou ktorých sa potom ľahšie orientujú a umiestňujú snímky. Pri týchto kontrolných bodech sa zvykne zistiť ich skutočná poloha pomocou pozemného merania a získa sa tak väčšia presnosť výsledného modelu.



Obr. 1.14: Princíp techniky fotogrammetrie

Kedže sa pri tejto technike vytvárajú fotografie a nevyužívajú sa rádiové vlny, je nutné, aby pri fotografovaní bola čo najlepšia viditeľnosť a objektívu nič nebránilo. Techniku teda nemôžeme využívať v noci. Prekážkou pri zhotovovaní fotografií sú často prírodné úkazy ako dážď a sneženie. Presnosť snímok závisí aj od výšky ich zhotovovania, preto je veľmi dôležité správne nastavenie. Hlavnou výhodou tejto techniky je jej jednoduchosť spracovania, rýchlosť zberu údajov a dostupný softvér pre bežného používateľa.

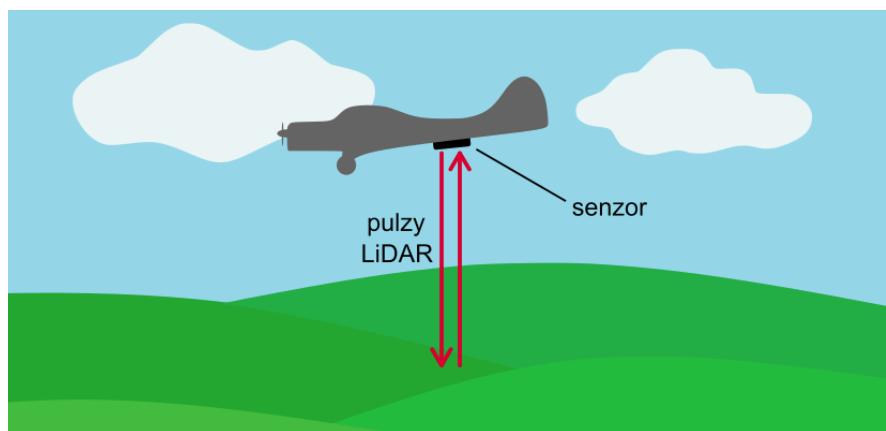


Obr. 1.15: Model zhotovený technikou fotogrammetrie [52]

V závislosti od výberu fotografií nemusia byť výsledkom iba informácie o nadmorskej výške, ale aj farba každého bodu oblasti čo nám poskytne textúru a aj jednoduchšiu interpretáciu výsledného modelu. Príklad pospájaných fotografií a tvar modelu môžeme vidieť na obrázku 1.15. Viac o tejto technike a o tom aké softvéry sa používajú pri spracovaní fotografií nájdeme v článku *What is Photogrammetry?* [25].

LiDAR

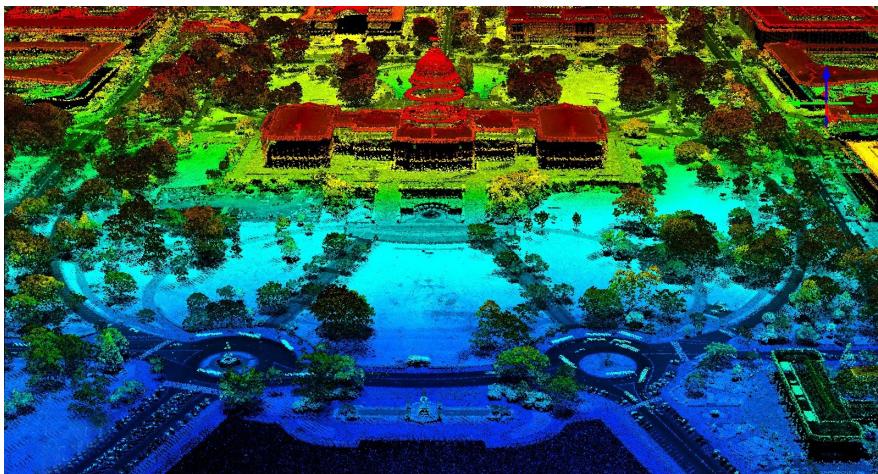
Ďalšia z používaných techník je technika LiDAR (Light Detection and Ranging). Z helikoptéry alebo lietadla sa vyšle pomocou lasera svetelný lúč na povrch zeme, a potom sa zaznamenáva čas trvania, kým sa svetlo vráti ku senzoru. Z času potrebného na návrat a údajov o výške a orientácii senzora sa potom vypočíta vzdialenosť. Technika je teda podobná satelitnej interferometrii, ale namiesto rádiových vĺn používa svetelné lúče. Táto technika má oproti fotogrametrii výhodu, že sa dá používať aj v noci, ale oproti InSAR zaostáva v tom, že sa nedá použiť za každého počasia. Predovšetkým kvapôčky dažďa môžu odrážať a absorbovať časť vyžarovaného svetelného lúča. Ukážku tejto techniky môžeme vidieť na obrázku 1.16.



Obr. 1.16: Princíp techniky LiDAR

Pomocou lasera sa vysiela viac ako 160 000 impulzov za sekundu. Kedže sa lietadlo pohybuje vo vzduchu nejakou rýchlosťou, musí sa zaručiť, že sa impulz vráti naspäť ku senzoru. Preto sa vysielajú niektoré senzory pod uhlom. Pri výpočte nadmorskej výšky sa následne berie do úvahy aj tento uhol. Význam vysielania lúčov pod uhlom je aj v prípade hustých stromov, kde by lúč nemusel prejsť cez vegetáciu pri jeho priamom vysielaní nadol, ale pod uhlom sa zachytia jeho boky.

Podobne ako pri technike InSAR, aj tu veľa závisí od vysielaných impulzov. Tie môžu podľa zvoleného typu prechádzať rôznymi povrchmi. Pri prechode lesným porastom sa postupne odráža od rôznych častí lesa až napokon dopadne na holú zem. Jeden impulz potom môže mať viacero návratov a dokáže zaznamenať informácie od koruny stromov až po zemský povrch. No pri veľmi hustých porastoch nedokáže preniknúť lúč až na zem, majú mikrovlnné impulzy výhodu. Pomocou tejto techniky dokážeme generovať modely typu DSM a DTM. Najčastejšie sa používa infračervený laser na zaznamenávanie povrchu. Zelené svetlo je vhodné pri prenikaní cez vodnú hladinu, a teda meranie výšok morského dna. Kedže sa využíva laser a nie fotografovanie, strácamo informácie o skutočnej farbe povrchu. Výstupy LiDAR sa teda zafarbijú na základe intenzity odrazených lúčov. Príklad takéhoto výstupu môžeme vidieť na obrázku 1.17.



Obr. 1.17: Mračno bodov zhotovené technikou LiDAR [22]

Ďalšie informácie o tejto technike sa môžeme dočítať v článku *A Complete Guide to LiDAR: Light Detection and Ranging* [22]. Nachádzajú sa v ňom informácie o nástrojoch využívaných pri spracovávaní dát, z akých komponentov sa skladá celý systém alebo aké ďalšie typy LiDAR existujú. Získanie LiDAR dát je pomerne jednoduché. Na svete existuje viacero bezplatných poskytovateľov LiDAR dát, tých najznámejších nájdeme v článku *Top 6 Free LiDAR Data Sources* [23]. Nachádza sa tam presný postup ako si vhodné dátá od jednotlivých poskytovateľov stiahnuť. LiDAR má vlastný binárny súborový formát (.las) určený na prácu a uchovávanie údajov mračna bodov. Tento formát sa považuje za priemyselný štandard pre LiDAR údaje.

Vzájomné porovnanie techník

Každá zo spomenutých techník má svoje určité výhody a nevýhody, či už pri rýchlosti spracovania, presnosti alebo pri možnosti zaznamenávania povrchu. Z hľadiska presnosti sú na tom lepšie techniky LiDAR a InSAR. LiDAR DEM je presnejší, s menším vychýlením ($\sim 0,3m$) a menším rozptylom ($\sim 0,9m$) ako InSAR DEM (odchýlka $\sim 0,9m$, rozptyl $\sim 3,3m$) [37]. Z hľadiska použiteľnosti vedie jednoznačne technika InSAR, ktorá sa môže v prípade potreby používať za každých okolností a počasia. Čo sa týka vizuálneho výstupu, technika fotogrametrie má navrch v poskytovaní skutočnej textúry povrchu a používateľ má tak lepšiu predstavu o tom na čo sa pozera.

Jednoznačne sa nedá povedať, že jedna technika je lepšia ako druhá. Všetko záleží od projektu, a dát potrebných na jeho dokončenie. Navyše sa technológie neustále vyvíjajú a tieto techniky sa v presnosti navzájom predbiehajú. Avšak techniky nemusíme stavať proti sebe. V súčasnosti sa využívajú ich kombinácie a integrujú sa spolu, a tak poskytujú širokú škálu monitorovacích informácií, ktoré sú stále lepšie a presnejšie.

Kapitola 2

Metódy tvorby digitálnych modelov

Napriek možnostiam, ktoré nám ponúkajú moderné počítačové systémy, vyžaduje proces tvorby terénov zásah zo strany človeka. Automatické vytváranie realisticky vyzejajúcich terénov je stále otvoreným problémom. Cieľom je čo najviac zautomatizovať proces tvorby terénu, ale zároveň poskytnúť používateľovi čo najväčšiu kontrolu nad generovaným obsahom, aby si ho mohol upraviť podľa predstáv.

V tejto kapitole popíšeme niektoré z týchto prístupov, ktoré sa využívajú. Povieme ich jednotlivé výhody a nevýhody. V úvode kapitoly sa nachádza sekcia kľúčové pojmy, kde zhrnieme nevyhnuté základné pojmy, ktoré sa potom vyskytujú v ďalšej časti práce. Objasníme pojmy ako procedurálne techniky, fraktály alebo náhodnosť.

2.1 Kľúčové pojmy

Zadefinujeme výrazy pre ďalšiu prácu – procedurálne techniky, fraktály, náhodnosť...

2.1.1 Procedurálne techniky

Už v skorých začiatkoch počítačovej grafiky sa začali využívať rôzne procedurálne techniky. Ich princíp spočíva v tom, že pomocou algoritmov sa snažia čo najvernejšie napodobiť požadovaný obsah. Využívajú sa v rozličných oblastiach počítačovej grafiky, napríklad pri tvorbe realistických textúr ako je mramor, drevo, kameň, alebo napodobenie ďalších prírodných materiálov. Iným využitím môžu byť animácie, kde pomocou procedurálnych techník vieme simulať horenie ohňa, pohyby vody, pary alebo dokonca srsti. Oblast, ktorá zaujíma nás je ich využitie na generovanie 3D modelov, predovšetkým tvorba terénu. Modelovanie terénu je v počítačovej grafike stále otvoreným problémom. Už niekoľko desaťročí sa vytvárajú čo najefektívnejšie algoritmy, ktoré by vedeli rýchlo generovať realistické tvary nerozoznateľné od skutočnosti. Viac o rôznych procedurálnych technikách a ich využití sa môžete dočítať v publikácii *Texturing and Modeling: A Procedural Approach* [13].

Hlavnou výhodou týchto techník je snaha o využívanie abstrakcie. Namiesto ukladania zložitých informácií o objekte, využijeme vlastnosti, ktoré sme schopní napodobniť matematickými funkciami. Výrazne sa nám tým znižujú nároky na ukladací priestor, aj keď niektoré algoritmy môžu byť výpočtovo náročné. Dobrým príkladom je procedurálne vytvorenie textúry, kde namiesto ukladania informácie o každom pixeli textúry, zadefinujeme jej správanie a pomocou algoritmu a matematických funkcií v prípade potreby vypočítame požadované hodnoty. Takto vieme vytvárať nekonečnorozmerné textúry s viacerými rozlíšeniami a úpravou parametrov jednoducho generovať jej nové inštancie. Keď číslam priraďujeme zmysluplný koncept, získavame silnú parametrickú kontrolu a už pri malých úpravách vieme dosiahnuť výraznú zmenu obsahu. Pri tvorbe terénu by sme chceli napríklad pomocou jedného parametra ľahko kontrolovať drsnosť resp. hladkosť pohorí a dosiahnuť tak požadovaný výsledok.

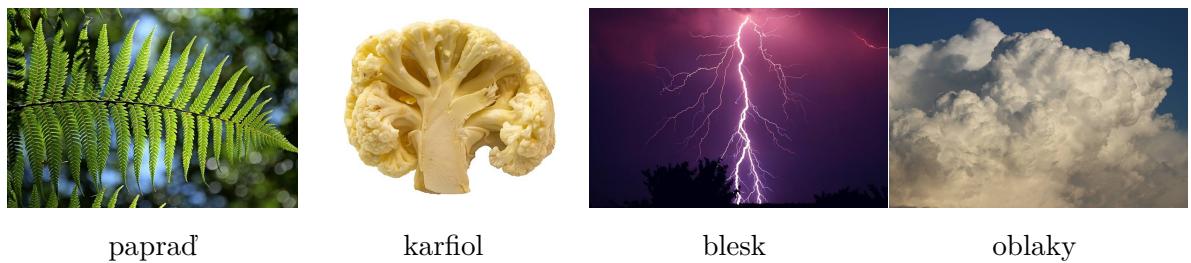
Pri procedurálnych technikách sa zvykne využívať aj náhodnosť. Mnohokrát sme príjemne prekvapení neočakávaným správaním procedúr. Dokážeme vytvárať efekty, ktoré kopírujú prírodné zákony alebo čisto umelecké veci. Pri generovaní terénu sa zvykne využívať náhodnosť pri posúvaní niektorých bodov o náhodnú výšku. Všetky procedurálne techniky by mali byť zreproduktovatelné, teda ak zadáme rovnaké parametre mali by sme dostať rovnaké tvary. Dôvodom prečo pri rovnakých parametroch dostaneme rovnaké výsledky, aj keď používame náhodnosť je ten, že v počítači neexistuje skutočná náhoda, ale len pseudonáhoda, určená na základne rôznych špecifikácií.

Procedurálne techniky sú v počítačovej grafike stále oblasťou aktívneho výskumu, keďže ponúkajú široké možnosti uplatnenia. Svoju oblubu si získava aj kvôli neustále stúpajúcemu výkonu procesorov a grafických kariet a ich cenovej dostupnosti. Moderné CPU a GPU dokážu generovať prvky pomocou procedurálnych techník aj v reálnom čase. To nám umožňuje interaktívne pracovať s procedurálnymi modelmi a efektami. Samozrejme procedurálne techniky záležia od použitého algoritmu a požadovanej úrovni presnosti. Stále sa nájdú oblasti počítačovej grafiky, kde sme obmedzení výkonom a musíme sa uspokojiť aj s nižšími nárokmi.

2.1.2 Fraktály

Viaceré významné osobnosti našej histórie našli inšpiráciu pre svoju prácu v prírode. Fyzik Isaac Newton s pomocou jablka alebo Mikuláš Koperník pri pozorovaní oblohy. Inak to nebolo ani v prípade fraktálov. Matematik Benoît Mandelbrot si všimol, že v prírode má viacero prvkov jednu zaujímavú vlastnosť – sebapodobnosť. To znamená, že väčšie časti daného objektu majú približne podobný tvar ako niektoré jeho menšie časti. Mandelbrot túto vlastnosť využil, aby zadefinoval matematickú konštrukciu – fraktál, z latinského názvu *fractus* (zlomený/rozbitý). Práve preto je Mandelbrot považovaný za otca fraktálnej geometrie.

V matematike sebapodobnosť rozlišujeme v dvoch podobách. Buď presná sebapodobnosť, kde každá časť je presnou kópiou inej časti ak ju správne natočíme a štatistická sebapodobnosť, kde fraktály vyzerajú podobne v rôznych mierkach. V prírode sa vyskytujú len štatisticky podobné fraktály, pretože neexistujú žiadne dva identické prvky. Príroda je príliš komplexná, takže zachytenie jej zložitosti a pokus o jej reprodukciu v syntetických obrazoch je zložitým problémom. V posledných rokoch sme urobili výrazný pokrok, no aj napriek tomu, výsledky ktoré dosahujeme sú menej zložité ako prírodná scéna. Medzi najznámejšie prírodné úkazy fraktálov patria hory, oblaky, blesky, snehové vločky, paprade alebo cievne systémy v živých organizmoch. Pri nich majú ich menšie časti tendenciu podobať sa na väčšie. Príklad niektorých prírodných fraktálov môžeme vidieť na obrázku 2.1. Ďalšie zaujímavé a prekvapivé príklady, kde všade môžeme vnímať fraktály nájdeme v knihe *Fractals everywhere* [6] od Michaela F. Barnsleyho.



paprad

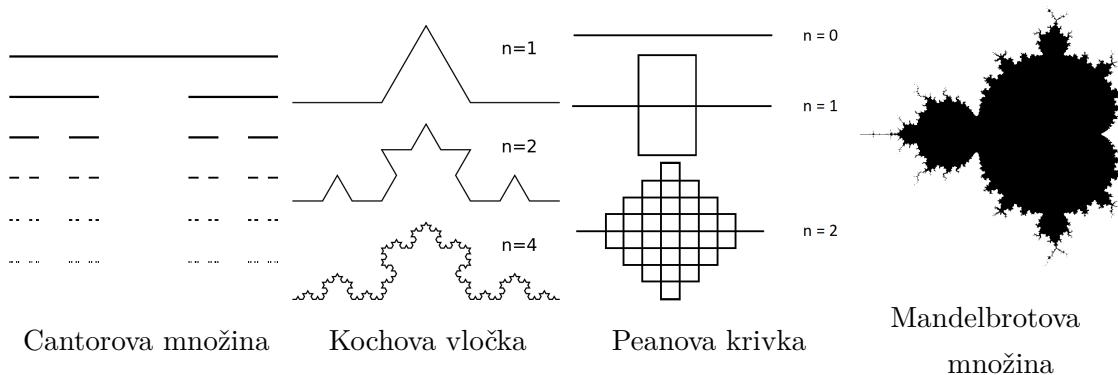
karfiol

blesk

oblaky

Obr. 2.1: Príklady fraktálov vyskytujúcich sa v prírode

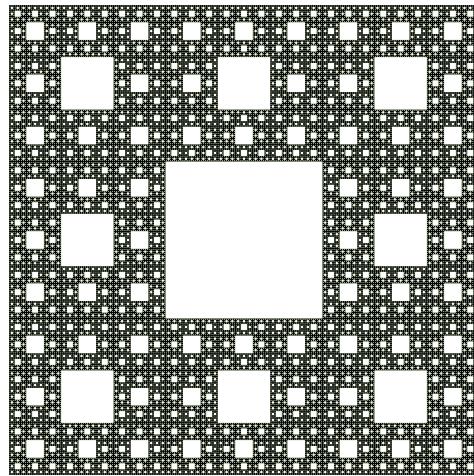
Zatiaľ sme spomínali len fraktály v prírode, ale vlastnosti fraktálov využili viacerí ľudia na vytvorenie vlastných matematických štruktúr, ktoré sú fraktálmi. Medzi najznámejšie patrí Cantorova množina, Kochova vločka, Peanova krivka alebo Mandelbrotova množina. Príklady týchto matematických štruktúr môžeme vidieť na obrázku 2.2. Bližšie informácie o týchto matematických štruktúrach aj s ich vyjadrením nájdeme v publikácii *Geometria fraktálov* [15]. Niektoré fraktály dokážu vytvárať veľmi zaujímavé tvary a môžeme ich dokonca zaradiť aj medzi abstraktné umenie.



Obr. 2.2: Príklady fraktálnych matematických štruktúr

Existuje veľa rôznych definícií toho čo je to fraktál. Dôležité je zapamätať si, že princípom fraktálov je popisovať zložité javy pomocou vzťahov. Fraktálna množina môže vo všeobecnosti obsahovať nekonečné množstvo bodov, ktorých štruktúra môže byť taká zložitá, že by bolo nemožné opísat, kde sa jej každý bod nachádza. Namiesto toho sa definuje vzťah medzi jednotlivými časťami. Veľmi pekným príkladom je, že na opisanie slnečnej sústavy nám stačí uviesť gravitačný zákon a určiť počiatočné podmienky.

Vlastnosť sebapadobnosti môžeme matematicky poňať – fraktály sú invariantné ku škálovaniu. Doteraz sme spomínali len túto jednu vlastnosť, ale fraktály ich majú viac. Ďalšou významnou vlastnosťou fraktálov je fraktálna dimenzia. Z matematického hľadiska je fraktál množina, ktorá má Hausdorfovskú dimenziu väčšiu ako topologickú dimenziu. Topologická dimenzia sa vyjadruje prirodzenými číslami, kde bod má dimenziu nula, krivka jedna, rovina má dimenziu dva a priestor má dimenziu tri. Fraktálna dimenzia rozširuje tento koncept a umožňuje aj reálne číselné hodnoty. Príkladom je Sierpiňského koberec, ktorý môžeme vidieť na obrázku 2.3. Jeho topologická dimenzia je 1, ale jeho fraktálna dimenzia je 1,8928 [31]. Prirodzene si to môžeme interpretovať tak, že čím je fraktálna dimenzia väčšia, tým daný fraktál zaberá viac priestoru, teda je hustejší. V prípade generovania terénov určuje fraktálna dimenzia drsnosť výsledného terénu. Označenie fraktálna dimenzia je teda synonymum pre neceločíselnú Hausdorfovú dimenziu. Objekty, ktoré nie sú fraktálmi majú typologickú a fraktálnu dimenziu rovnakú [14].



Obr. 2.3: Sierpiňského koberec

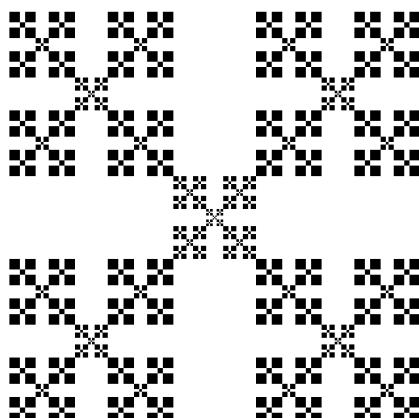
Treba si uvedomiť, že napríklad lopta je trojrozmerný objekt, ale jej povrch ním nie je. Ak sa priblížime k povrchu lopty dostatočne blízko stane sa lokálne rovinným. Povrch teda nemusí byť dokonale plochá euklidovská rovina, aby mal fraktálnu dimenziu 2 a ani nemusí vyplniť celý trojpriestor, aby mal dimenziu 3. Tieto spomenuté vlastnosti patria medzi základné definície fraktálov.

Ďalší pojem, ktorý si potrebujeme pri fraktáloch objasniť je základná funkcia. Je to počiatočný stav z ktorého vychádzame, a ktorého opakovaním zostavíme fraktál. Je to veľmi dôležitá vec, pretože voľba základnej funkcie ovplyvňuje výsledný fraktál. Pojem oktáva určuje počet opakovaní, kolkokrát vykonávame iterácie pri vytváraní fraktálu. Posledným dôležitým pojmom je lacunarita, ktorou meníme základnú funkciu pri každej iterácii. Pri tvorbe terénov je to zmena horizontálnej veľkosti základnej funkcie, ale všeobecne je to zmena frekvencie (môžeme to nazvať aj medzera). Fraktály s vyššou lacunaritou majú medzi vzormi väčšie medzery.

Multifraktály

Problém, ktorý nám nastáva pri generovaní terénov s fraktálmi je, že pri použití jednej základnej funkcie sa nám vytvorí fraktál, ktorý má všade približne podobné správanie, je teda homogénny. Takéto fraktály zvykneme označovať aj monofraktály. Skutočné terény sú ale rôznorodejšie, na niektorých miestach sú hladké na iných drsnejšie. Tento problém riešia multifraktály, ktoré nie sú všade rovnaké, teda sú heterogénne. Definovať ich môžeme ako fraktály, ktorých fraktálna dimenzia sa mení s umiestnením. Príklad multifraktálu môžeme vidieť na obrázku 2.4. Pri generovaní terénu môžeme získať multifraktály napríklad nejakým postprocesom. Ak do fraktálneho terénu, ktorý má určitú dimenziu pridáme vodnú hladinu, tak vznikne multifraktál. Dôvod je ten, že rovina bude mať inú dimenziu ako zvyšok terénu, vygenerovaného ako fraktál. Samozrejme len v prípade, že terén nie je rovina.

Multifraktály nám teda dovoľujú zachytiť väčšiu zložitosť a rozmanitosť. Príkladom multifraktálov je napríklad turbulencia. Turbulencia v nejakom bode vzniká súčtom šumu v tomto bode so zmenšenými hodnotami šumu v iných bodoch. Použitie turbulencií môže byť zaujímavým prvkom pre generovanie terénov. Ďalšie informácie o fraktáloch sa nachádzajú v podrobne spracovanej publikácii s názvom *The Science of Fractal Images* [7].

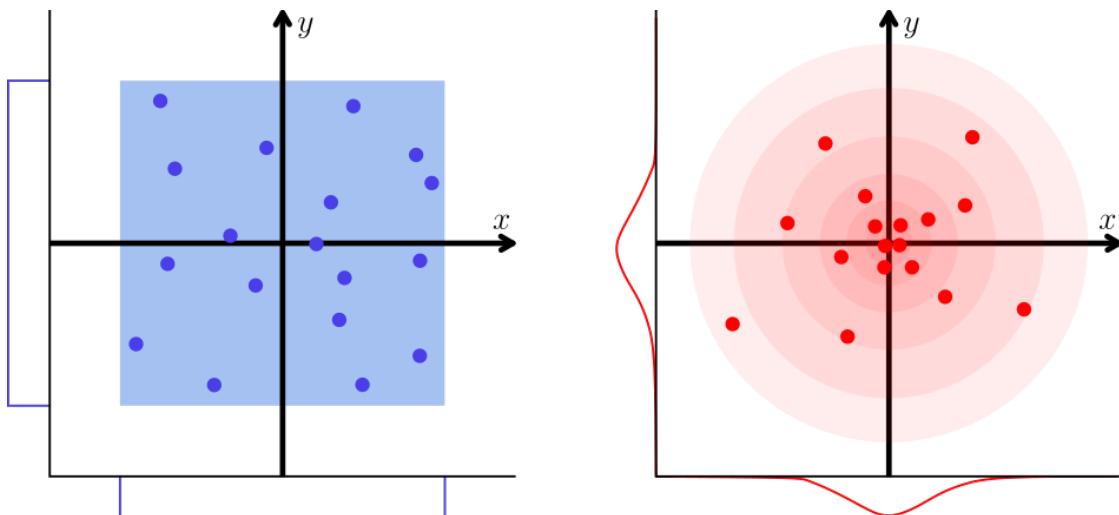


Obr. 2.4: Príklad multifraktálu

2.1.3 Náhodnosť

Dôležitou vlastnosťou, ktorá sa vyskytuje všade v našom živote a možno si to ani neuvedomujeme, je náhodnosť. Väčšina javov, s ktorými sa stretávame sa riadi náhodnými procesmi. Vedieť napodobniť túto vlastnosť je teda veľmi užitočným nástrojom pri vytváraní algoritmov simulujúcich prírodné procesy. Generovanie vhodných náhodných čísel je dôležité aj pri tvorbe terénov.

Gaussovo rozdelenie, (*Gaussian distribution*) niekedy pod pojmom normálne rozdelenie, (*normal distribution*) je typ rozdelenia pravdepodobnosti pre náhodnú premennú s reálnou hodnotou. Je založený na generovaní náhodnosti okolo určitej hodnoty. Najlepšie si to ukážeme na príklade s generovaním bodov v určitej oblasti. Bežné náhodné generovanie čísel z určitého rozsahu vyplní priestor bodmi rovnomerne s rovnakou hustotou, ako môžeme vidieť na obrázku 2.5 vľavo. No viaceré prírodné úkazy sa neradiad takýmto rovnomerným rozdeľovaním, ale náhodné body sa pohybujú okolo určitej hodnoty. Napríklad šírenie iskier alebo prachu pri práci sa šíri okolo určitého objektu, rovnako ako rozmiestnenie rastlín a vegetácie v blízkosti jazera a rieky. Príklad Gaussovho rozdelenia okolo určitej hodnoty môžeme vidieť na obrázku 2.5 vpravo.



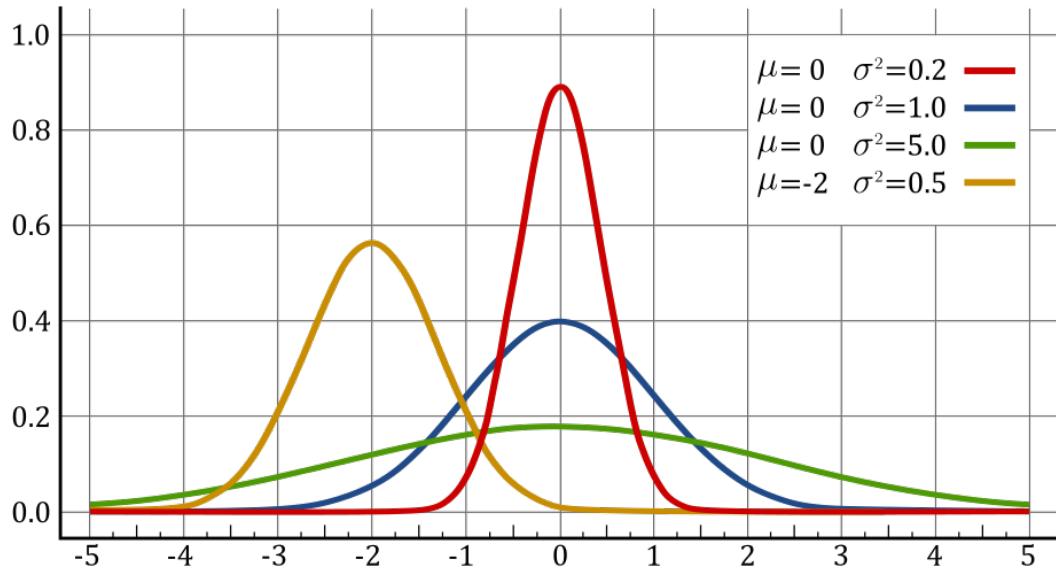
Obr. 2.5: Príklad rovnomerného a Gaussovho rozdelenia

Ak hádzeme bežnou hracou kockou, tak každé číslo na kocke má rovnakú pravdepodobnosť, že padne. Štatisticky povedané, hádzanie kockou je rovnomerné diskrétné rozdelenie, ktoré je definované počtom všetkých možností. Ak je možností n , tak je to $P(x) = \frac{1}{n}$. Pri Gaussovom rozdelení je to trochu zložitejšie, tam sa niektoré hodnoty vyskytujú častejšie ako iné. Pri Gaussovom rozdelení je všeobecný tvar funkcie výskytu danej hodnoty určený vzťahom:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \quad (2.1)$$

Je definované priemerom (*mean*) μ a rozptylom (*variance*) σ^2 . Parameter σ je štandardná (smerodajná) odchýlka (*standard deviation*), ktorá vyjadruje mieru rozloženia čísel a je určená ako druhá odmocnina rozptylu. Priemer posúva krivku po osi doľava alebo doprava, pričom ju vždy vycentruje na hodnotu, ktorá sa očakáva najčastejšie. Smerodajná odchýlka nám hovorí, aké ľahké je odchýliť sa od priemeru. Príklad môžeme vidieť na obrázku 2.6. Spodná os zodpovedá náhodnej premennej x a bočná nám určuje hodnotu pravdepodobnosti.

Ked' je premenná X generovaná určitým javom, ktorý je gaussovsky rozložený, zvyčajne sa označuje ako $X \sim N(\mu, \sigma^2)$. Podrobnosti o Gaussovom rozdelení, o odvodení vzťahu 2.1 a ďalších zaujímavých využitiach nájdeme v knihe *Handbook of the normal distribution* [40].



Obr. 2.6: Príklady funkcií Gaussovo rozdelenia

2.1.4 Brownov pohyb

Fraktálny Brownov pohyb – fBm (*fractal Brownian motion*) je zovšeobecnením Brownového pohybu, ktorý je definovaný neusporiadaným chaotickým pohybom častíc v kvapaline a ich vzájomným zrážaním. Generuje údaje, ktoré sú fraktálne v zmysle, že sú samopodobné. Pre dátá generované týmto procesom je charakteristický parameter H, ktorý sa nazýva **Hurstov exponent**. Je to číslo medzi 0 a 1, ktoré ovplyvňuje mieru korelácie medzi údajmi. Fraktály a prírodné javy zvyknú mať Hurstov exponent okolo hodnoty 0,75. Takže aj v našich algoritnoch sa budeme snažiť používať približne túto hodnotu. Kvantitatívne môžeme Hurstov exponent nazvať premennou hrubosti alebo náhodnosti údajov. Na obrázku 2.7 môžeme vidieť ako výrazne ovplyvňuje táto premenná tvary terénov. Môžeme vidieť, že nízky exponent vedie k veľmi náhodným údajom a vysoký k plynulejším.

2.2 Rekurzívne delenie

Rekurzívne delenie (*recursive subdivision*) je trieda fraktálnych algoritmov, ktorých princípom je buď delenie geometrického objektu alebo nejakej dátovej štruktúry na základe zadefinovaných pravidiel. Delenie sa opakuje kým nie je splnené nejaké vopred určené kritérium [9]. V starších publikáciách môžeme túto triedu algoritmov nájsť aj pod pojmom fraktálne delenie (*fractal subdivision*).

Vlastnosť delenia štruktúry, kým nie je splnená nejaká podmienka, je veľmi užitočná pri generovaní terénov. Pretože ak máme terén, ktorý nie je navzorkovaný dostatočným počtom bodov, je dobré si nejaké body vytvoriť pomocou definovaných pravidiel. Toto rieši naša aplikácia, ktorá využíva reálne zemské súradnice a chýbajúce dáta vypočíta podľa potreby. Existuje viacero algoritmov rekurzívneho delenia pri tvorbe terénov, ktoré sa snažia čo najvernejšie napodobniť reálne tvary. Všetky generujú terény od začiatku, nie s použitím reálnych súradníc, to je predmetom našej implementácie v kapitole 4.

2.2.1 Posúvanie stredného bodu

Algoritmus posúvania stredného bodu (*midpoint displacement algorithm*) je jeden z najzákladnejších algoritmov pri fraktálovo generovaní terénu. Princíp ukážme najskôr v jednej dimenzii, kde môžeme generovať pekné hrebeňové čiary, a potom ho rozšírimo do dvoch dimensií a ukážeme ako dokáže generovať povrch.

Algoritmus začína s úsečkou, ktorá má na začiatku inicializované svoje krajiné body na náhodné hodnoty. Ako napovedá názov, chceme posúvať stredný bod, takže nájdeme stred úsečky, ktorý dostaneme ako priemer krajných bodov a pripočítame Gaussovo náhodné číslo s priemerom 0 a rozptylom definovaným vzťahom

$$\sigma_1^2 = (1/2)^{2H} \sigma_0^2 \quad (2.2)$$

Pri každej iterácii nám vznikne dvojnásobný počet úsečiek. Na každej úsečke potom opakujeme rovnaké delenie, kym nedosiahneme požadovanú úroveň detailov. Pri každej iterácii sa zníži rozptyl náhodného čísla o faktor $1/2$, takže pre $i - tu$ iteráciu máme vzťah pre rozptyl daný ako:

$$\sigma_i^2 = (1/2)^{i2H} \sigma_0^2 \quad (2.3)$$

Táto metóda generuje 1 rozmernú čiaru s Hurstovým exponentom na vstupe. Tento parameter je veľmi dôležitý, pretože určuje drsnosť výsledného fraktálu. Ak by bol nastavený na hodnotu 0, tak sa nám rozsah, z ktorých sa vyberajú náhodné čísla vôbec nemení a vznikol by nám terén, ktorý je príliš hrbochatý a teda skáče hore a dole. Príklad toho ako Hurstov exponent ovplyvňuje drsnosť môžeme vidieť na obrázku 2.7.



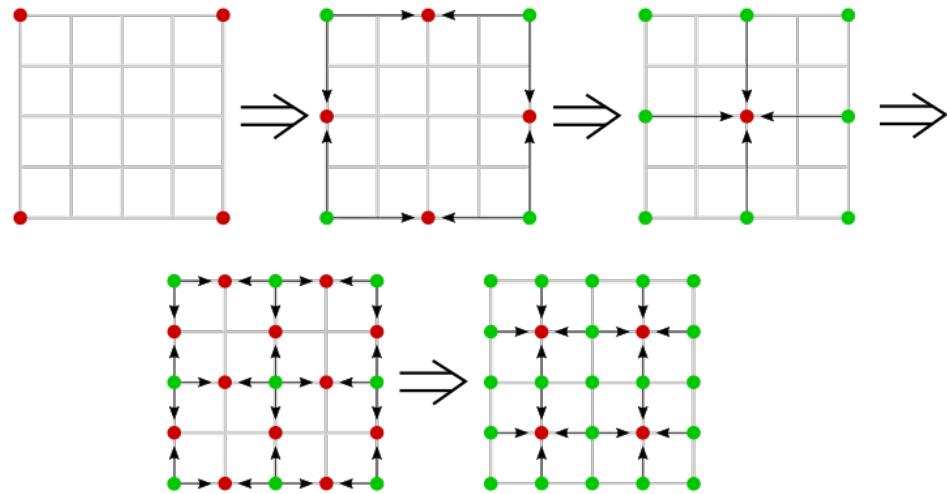
Obr. 2.7: Ovplyvnenie terénu Hurstovým koeficientom

Metóda je ľahko rozšíritelná do dvoch dimenzií. Generuje náhodné čísla, ktorými postupne zapĺňa výškovú mapu a tá sa následne použije na vykreslenie terénu. Výšková mapa sa zapĺňa postupným počítaním stredných hodnôt aktuálne spracovávaného štvorca.

Výpočet dĺžky strán výškovej mapy, l je dĺžka strany v počte bodov a i je počet iterácií:

$$l_i = 2^n + 1 \quad (2.4)$$

Algoritmus začína so štvorcovým poľom, pričom máme na začiatku vygenerované náhodné hodnoty 4 rohov. Pri každej hrane štvorca si určíme jej stred a vypočítame jej hodnotu ako priemer krajných bodov a pripočítame náhodné číslo. Práve vytvorené 4 body potom použijeme na výpočet stredného bodu, ktorý určíme ako ich priemer a znova pripočítame náhodnú hodnotu. Postup opakujeme, kým nevyplníme celú výškovú mapu. Príklad vyplňania výškovej mapy metódou posúvania stredného bodu môžeme vidieť na obrázku 2.8.



Obr. 2.8: Metóda posúvania stredného bodu

V našej aplikácii pracujeme s reálnymi súradnicami, teda už na nejakom vytvorenom teréne. Ako prvé si teda budeme musieť vypočítať v akej iterácii sa nachádzame, aby sme si následne mohli vypočítať rozptyl. To nebude problém, keďže poznáme počet bodov terénu v oboch smeroch. Ako môžeme vidieť v rovnici 2.4, tak výpočet počtu iterácií bude závisieť na logaritme počtu bodov so základom dva.

Tento rozptyl potom použijeme pri nasledujúcej iterácii. Ak by sme si nezistili iteráciu a pracovali s dátami akoby sa terén vytváral z roviny, tak by sa nám pri zvýšení iterácie pripočítavali príliš vysoké náhodné hodnoty a terén by bol nerealistický.

Metóda posúvania stredného bodu je jednoduchá a rýchla, ale pre dnešné náročné požiadavky nedostačujúca, pretože má nedostatky ako opakujúce sa vzory a viditeľné terénne záhyby. Metóda má viacero úprav, ktoré sa snažia riešiť grafické artefakty.

2.2.2 Metóda diamant-štvorec

Algoritmus diamant-štvorec (*diamond-square algorithm*) je vylepšenou verziou predchádzajúceho algoritmu posúvania stredného bodu, ktorý odstraňuje zle vyzerajúce terénne záhyby. Táto metóda je široko používaná vo vedeckých aplikáciách kvôli jej jednoduchosti a vďaka generovaniu realisticky vyzerajúcich terénov. Z tohto dôvodu sme sa rozhodli implementovať práve túto metódu v našej aplikácii. Ponúka totiž dobrú vyváženosť medzi jednoduchosťou a realistickosťou. Keďže je to rekurzívna metóda delenia, pracuje na princípe mriežky, čo je pre nás ideálne. My pracujeme s výškovými mapami, ktorých údaje sú uložené tiež v pravidelnej mriežke.

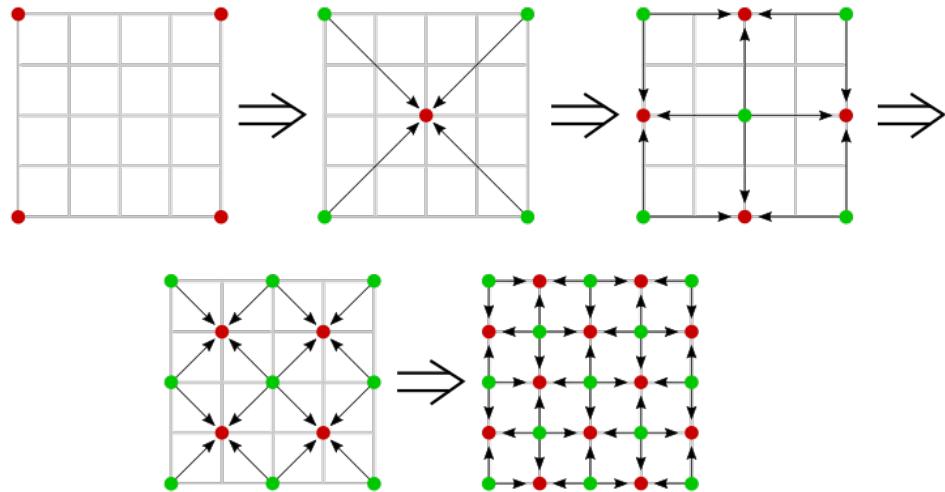
Algoritmus rieši problém terénnych artefaktov tým, že strieda pri vytváraní dva kroky – diamantový a štvorcový. Striedaním krovok sa v teréne vytvárajú plynulejšie prechody. Myšlienka bola prvýkrát predstavená v publikácii *Computer Rendering of Stochastic Models* [16] na SIGGRAPHe v roku 1982 . V roku 1986 Gavin S. P. Miller na SIEGGRAFHe opísal v publikácii *The Definition and Rendering of Terrain Maps* [35] niektoré chyby a začal v nej používať pojem diamant-štvorec.

Začiatok a koniec algoritmu je rovnaký ako pri posúvaní stredného bodu. Inicializujeme štvorec so štyrmi náhodnými bodmi a končíme, keď zaplníme celú mriežku resp. výškovú mapu na požadovanú úroveň. V algoritme sa striedajú nasledujúce kroky:

Diamantový krok zoberie vrcholy aktuálne spracovávaného štvorca a vygeneruje výšku prostredného bodu, ktorý leží na priesecníku uhlopriečok štvorca. Jeho výšku vypočíta z priemeru štyroch vrcholov, ktorú zvýši o náhodnú hodnotu. Vytvorené body sú potom vstupom do druhého kroku algoritmu.

Štvorcový krok zoberie štyri vrcholy aktuálne spracovávaného diamantu a vygeneruje výšku prostredného bodu, ktorý leží na priesecníku uhlopriečok diamantu. Jeho výšku vypočíta z priemeru štyroch vrcholov, ktorú zvýši o náhodnú hodnotu. Vytvorené body sú potom vstupom do prvého kroku algoritmu.

Po viacnásobnom opakovaní týchto krokov postupne vyplníme celú výškovú mapu. Problémy môžu nastaviť na hraniciach pola, kde sa môžeme zachovať dvojako. Bud použijeme len body, ktoré máme k dispozícii alebo namiesto chýbajúceho bodu použijeme bod na opačnej strane pola a vytvoríme tým opakovateľný terén.



Obr. 2.9: Metóda diamant-štvorec

Táto metóda generuje dobre vyzerajúce plochy, ktoré aproximujú fraktálny Brownov pohyb pričom je Hurstov koeficient ľahko ovládateľný. Výsledné terény po použití tejto metódy vyzerajú plynulo a hladko, takmer ako skutočný terén. Jednoducho vieme ovládať hrubosť aj výšku, ako uvidíme v aplikácii. Spomedzi metód rekurzívneho delenia je metóda diamant-štvorec najlepšia, lebo na nej nevidno artefakty, a preto sme sa ju rozhodli implementovať.

2.2.3 Metóda štvorec-štvorec

Ďalšou zaujímavou technikou rekurzívneho delenia je metóda štvorec-štvorec. Avšak v tomto prípade ide o nevnorené rozdelovanie (*unnested subdivision method*). Znamená to, že sa pri iteráciách nevnárame do jednej štruktúry výškovej mapy, ale vytvárajú sa stále nové pri každej iterácii.

Najskôr sa inicializuje výšková mapa veľkosti 3×3 , ktorá je vyplnená deviatimi náhodnými hodnotami. Hodnoty výšok pre nové body sa vypočítajú ako väžený súčet z bodov štvorca, do ktorého patria. Súčet je v pomere $9 : 3 : 3 : 1$ a je určený na základe relatívnej polohy nového bodu k bodom na predchádzajúcej výškovej mape. To znamená, že najbližší bod k vrcholu štvorca vynásobíme hodnotou 9, dva body, ktoré sa nachádzajú na uhlopriečke vynásobíme oba s hodnotou 3 a najvzdialenejší bod bude prenasobení len s hodnotou 1. Príklad tohto postupu môžeme vidieť na obrázku 2.10.

Tu sa iterácie líšia od predchádzajúcich postupov. Namiesto zníženia veľkosti kroku o polovicu v každej iterácii sa veľkosť kroku zvýší o jedna. Proces sa zastaví po dokončení zvoleného počtu iterácií.

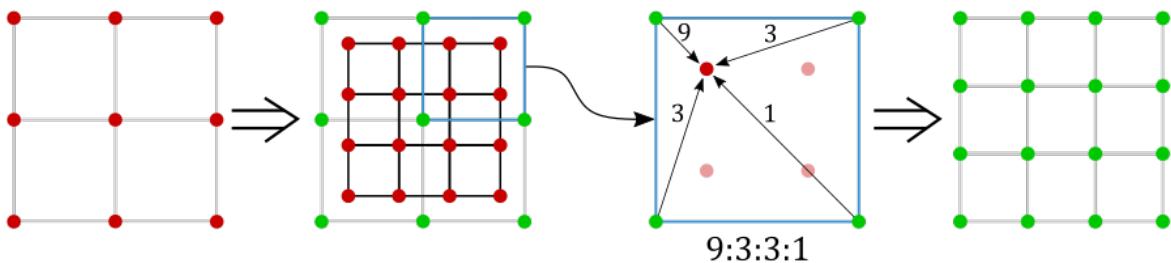
Dĺžka strany novej výškovej mapy:

$$l_n = 2(l_{n-1} - 1) \quad (2.5)$$

Počet bodov novej výškovej mapy:

$$a_n = (2^{(n+1)} + 2)^2 \quad (2.6)$$

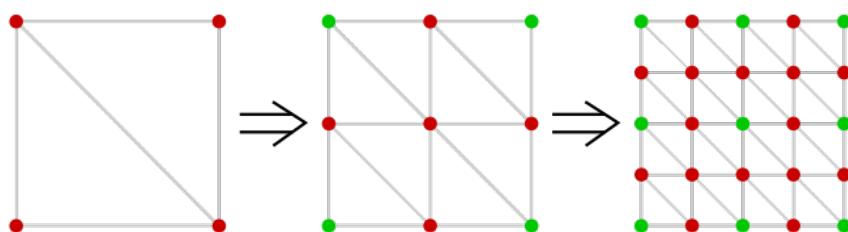
Po vypočítaní všetkých bodov v danej iterácii sa vypočíta nový maximálny výškový posun a veľkosť výškovej mapy pre ďalšiu iteráciu. Celkový počet bodov sa potom využíva na inicializáciu novej výškovej mapy. Ak sa zameriame na počet iterácií, tak pri rovnakom počte iterácií sú výškové mapy vytvorené metódou štvorec-štvorec o niečo menšie ako výškové mapy vytvorené predchádzajúcimi technikami rekurzívneho delenia. Metóda ponúka dobré výsledky a je rovnako vhodná ako metóda diamant-štvorec.



Obr. 2.10: Metóda štvorec-štvorec

2.2.4 Trojuholníková metóda

Známa je pod pojmom delenie hrán trojuholníka (*triangle edge subdivision*). Pri tejto metóde je každý trojuholník rozdelený na štyri nové. Hrany sú rozdelené na dve rovnaké časti, pričom stred sa berie ako priemer krajných bodov hrany. Ku stredu sa pripočíta Gaussova náhodná hodnota. Metóda je teda rovnaká ako posúvanie stredného bodu, len sa tu pracuje s trojuholníkmi. Príklad môžeme vidieť na obrázku 2.11.



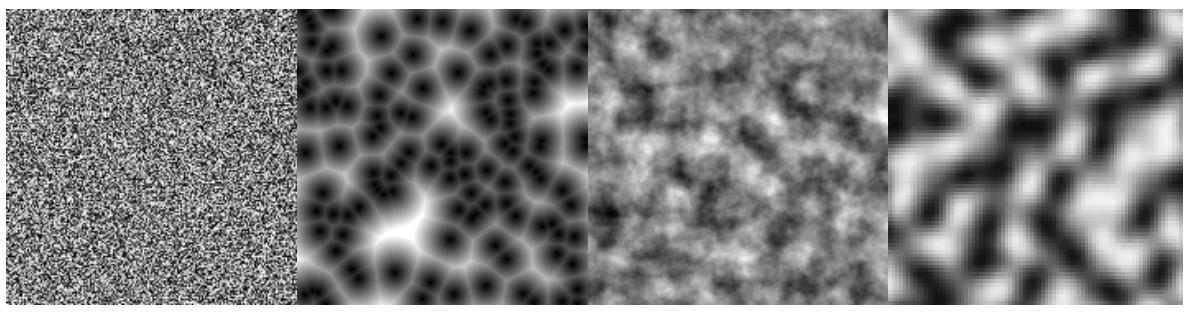
Obr. 2.11: Trojuholníková metóda

Výhodou metód rekurzívneho delenia je, že bežia v lineárnom čase na rozdiel od Fourierovej transformácie. Tá má časovú zložitosť $N \log(N)$. Takže ak porovnávame tieto dve metódy má navrh rekurzívne delenie. Nevýhodou týchto techník je, že pracujú len s diskrétnej pravidelnou sieťou. Pre našu aplikáciu to je výhoda, keďže tiež pracujeme s obrázkami na diskrétnej pravidelnej mriežke.

2.3 Generovanie pomocou šumu

Vo viacerých procedurálnych technikách sa využíva ako základ šum (*noise*), pretože narúša monotónnosť vzorov, čo je užitočné pri napodobňovaní prírodných objektov, lebo len málo z nich má pravidelné vzorky. Ide o jednu z najzákladnejších častí procedurálneho modelovania, na ktorú sa podobne ako pri fraktáloch aplikuje viacnásobne nejaké pravidlo. Šumy sú pseudonáhodné funkcie, ktoré nám pri rovnakých vstupných parametroch vygenerujú rovnaké hodnoty. Tým, že šum vzniká procedurálne môžeme vytvárať vzory v akomkoľvek rozlíšení a pristupovať k ich prvkom v konštantnom čase. Treba poznamenať, že šum je v skutočnosti periodický, ale väčšinou v takej miere, že to nie je pri aplikácii evidentné. Funkcie šumu v počítačovej grafike väčšinou generujú hodnoty na intervale [-1;1] a následne sa tieto hodnoty škálujú podľa potreby.

Existuje viacero druhov šumov v rôznych dimenziách, ale nás budú zaujímať len šumové funkcie v 2D priestore, ktoré každému pixelu mriežky priradia reálne číslo. Medzi najznámejšie šumy patria biely, ružový, hnedý, Perlinov a simplexný šum. Niektoré príklady môžeme vidieť na obrázku 2.12. Najjednoduchší z nich je biely šum, ktorý každému pixelu priradí náhodnú hodnotu, bez vzájomnej korelácie medzi hodnotami. No do našej aplikácie potrebujeme šum s plynulejšími prechodmi medzi jednotlivými hodnotami, pretože nám to umožní využiť šum na reprezentáciu terénu. Pri bielom šume by nám vznikal nepríjemný alias a to nechceme. Zameriame sa preto najmä na dva šumy, a to Perlinov a Simplexný šum. Bližšie informácie o ďalších šumoch nájdeme v publikácii (*Texturing and modeling: a procedural approach*) [13] na stranách 67-82.



biely šum

Voronoiov šum

Perlinov šum

simplexný šum

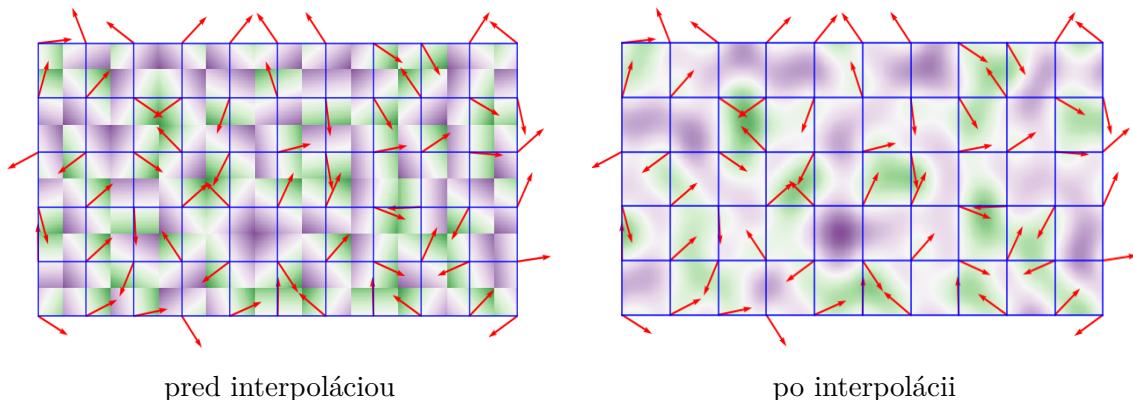
Obr. 2.12: Príklady najznámejších šumov [49]

2.3.1 Perlinov šum

Perlinov šum (*Perlin noise*) je jeden z najznámejších príkladov šumu v počítačovej grafike. Vynial ho Ken Perlin v roku 1983 a formálne ho opísal v článku *An Image Synthesizer* [42] na konferencii SIGGRAPH v roku 1985. Perlinov šum patrí do kategórie gradientných šumov, ktorý pracuje s pravidelnou štvorcovou mriežkou. Typy šumov, ktoré sú generované v nejakej priestorovej mriežke sa nazývajú mriežkové šumy. Ich náhodné hodnoty sa vypočítavajú v bodoch mriežky a počas vykreslenia celkového šumu sa potom jednotlivé body interpolujú. Tieto šumu sú rozširovateľné do viacerých dimenzií, tak pracujú aj s rôznymi mriežkami (1D, 2D, 3D).

Algoritmus vytvorenia Perlinovho šumu pozostáva z viacerých krokov. Najskôr sa na pravidelnej štvorcovej mriežke vygenerujú pseudonáhodné gradienty v každom vrchole mriežky. My pracujeme v 2D priestore, takže generujeme dvojrozmerný gradientový vektor, ale tento postup je ľahko rozširovateľný aj do viacerých dimenzií, rovnako ako väčšina algoritmov na generovanie šumu. Všetky vektory majú rovnakú dĺžku 1, takže záleží len na ich smere.

Ďalším krokom je vypočítanie skalárnych súčinov medzi gradientovými vektormi a vektormi určenými bodom, v ktorom počítame hodnotu a rohmi mriežky, do ktorej daný bod patrí. Pre lepšie pochopenie je tu obrázok 2.13. Červené šípky znázorňujú gradienty a tie farebné štvorčeky znázorňujú určené hodnoty.



Obr. 2.13: Objasnenie Perlinovho šumu

Po vypočítaní skalárnych súčinov sme dostali pre určovaný bod 4 skalárne hodnoty. Nasleduje posledný krok, a tým je interpolácia medzi týmito bodmi. Interpolácia sa robí pomocou funkcie s nulovou prvou deriváciou (prípadne aj druhou) pre rohy mriežky. To znamená, že hodnota šumu bude nulová v každom rohu mriežky. Ken Perlin pôvodne využíval ako interpolačnú funkciu Hermitovu funkciu $f(t) = 3t^2 - 2t^3$. Na obrázku 2.13 môžeme vidieť ako interpolácia pekne vyhladila výsledky. Na obrázku môžeme vidieť, že sa nám vytvorili akoby ostrovčeky v odtieňoch zelenej. To je presne to čo sme potrebovali, vytváranie takýchto ostrovčekov, ktoré sú spojité a môžeme nimi

reprezentovať terén. Princíp vytvárania šumu je teda taký akoby sme išli v smere šípky práve do kopca. Čím zelenšia časť tým sme vyššie a čím fialovejšia tým sme nižšie. My samozrejme budeme používať šedotónové obrázky a vytvárať výškové mapy. Tento obrázok slúžil len na ilustráciu.

Perlinov šum má viaceré parametre, ktoré budeme neskôr implementovať v našej aplikácii. Väčšina z nich je rovnaká ako v prípade fraktálov, čo by malo byť zrejmé, keďže šumy zaraďujeme medzi fraktálne algoritmy. Máme teda premenné:

- *seed*
- *offset*
- *scale*
- *octaves*
- *persistence*
- *lacunarity*

Seed slúži ako parameter pri generovaní náhodných hodnôt. Jeho zmenou sa teda vytvorí úplne nový šum a teda sa nám zmení aj výsledný tvar terénu. Offset je premenná posunutia, o koľko sa daný šum posúva v jednom aj druhom smere. Posúvame tým teda hodnoty v rámci výškovej mapy. Scale je parameter, ktorý slúži na škálovanie vygenerovaného šumu. Octaves vyjadrujú počet vrstiev šumu, ktoré sa budú generovať. Ak použijeme len jednu oktávu, šum nebude vyzerat realisticky, bude vyzerat príliš hladko, čo pri terénoch nemusí byť nevýhoda. Persistence je číslo v rozsahu 0 až 1, ktoré udáva ako sa každá oktáva podielala na konečnej hodnote šumu. Pri každej iterácii sa upravuje amplitúda šumu práve týmto parametrom. Ak je jej hodnota nastavená na 1, tak každá nová vrstva bude prispievať rovnakým množstvom. Ak je teda rôzna od 1, tak s každou oktávou sa prejavuje vplyv vygenerovaného šumu. Na začiatku je amplitúda nastavená na 1. Lacunarity udáva, koľko detailov pridáva každá oktáva ku konečnej hodnote šumu. Inak povedané o koľko sa zmenší v merítke každá nová textúra šumu. Mená parametrov sme nechali po anglicky, pretože aj implementáciu kódu píšeme po anglicky, tak aby sa jednoduchšie orientovalo, ktorá premenná patrí ku ktorej časti.

Vylepšený Perlinov šum

Predchádzajúci Perlinov šum má pár nedostatkov, ktoré Perlin odstránil vylepšením svojho predchádzajúceho algoritmu a nazval ho vylepšený Perlinov šum (*Improved Perlin Noise*). Chyby, ktoré mal predchádzajúci šum, opísal v článku *Improving Noise* [43] publikovanom v roku 2002.

S vyšším rozmerovým šumom, je generovanie výškovej mapy založenej na šume neefektívne a vytvárali sa smerové artefakty. Generujú sa aj defekty, ktoré sú známe ako problém vrásnenia, teda zle vyzerajúce záhyby pozdĺž hraníc. Na tieto problémy upozornil Gavin Miller v článku *The Definition and Rendering of Terrain Maps* [35].

Ken Perlin ako prvý vylepšil svoju interpolačnú funkciu v algoritme z Hermitovej funkcie $f(t) = 3t^2 - 2t^3$, ktorá nemala nulovú druhú deriváciu pre hodnoty $t = 0$ a 1 . Nenulovosť druhej derivácie spôsobovala nespojitost, ktorá sa prejavovala pri zatienení niektorých častí terénu vytvoreného z Perlinovho šumu. Nahradil ju teda funkciou $f(t) = 6t^5 - 15t^4 + 10t^3$, ktorej krivka je veľmi podobná predchádzajúcej, ale má v týchto hodnotách nulovú druhú deriváciu.

Druhou chybou bola zbytočne nákladná a čiastočne problematická metóda výpočtu gradientu. Vytváral sa efekt zhlukovania, kde sa blízke gradienty, ktoré boli takmer osovo zarovnané, a teda blízko seba, navzájom zarovnali. Spôsobovalo to vysoké hodnoty v týchto oblastiach. Perlin tento problém vyriešil zošikmením množiny smerov gradientov od súradnicových osí a uhlopriečok. V skutočnosti ale ani netreba počítať gradienty pomocou náhodných smerov. Dopredu si vytvoril množinu 12 vektorov, ktoré definovali smery a pristupoval k nim len keď potreboval. Odstránil teda potrebu viacerých násobení a ušetril približne 10 percent času.

Jedna vrstva šumu sama o sebe nemusí vytvárať realistické výškové mapy. Fraktálny Brownov pohyb (fBm) sa považuje za rozšírenie Perlinovho šumu. Pričom fBm kombinuje viaceré vrstvy Perlinovho šumu, pričom má rôzne amplitúdy a frekvencie. Rôznymi amplitúdami a frekvenciami sa generujú výškové mapy, ktoré vyzerajú realisticejšie.

2.3.2 Simplexný šum

Aj keď vylepšený Perlinov šum odstránil niektoré nedostatky pôvodného Perlinovho šumu, stále existovali aspekty, ktoré sa dali zlepšiť. Preto sa rozhadol Perlin vytvoriť nový šum, ktorý nazval Simplexný šum (*Simplex noise*).

Nový šum využíva menej násobení, a teda znížil výpočtovú náročnosť. Odstránil viditeľné smerové artefakty, ktoré sa vyskytovali v predchádzajúcich verziách. Tento šum má všade definovaný a súvislý gradient, ktorý sa dá lacno vypočítať a je ľahko implementovateľný do hardvéru.

Časovú zložitosť zrýchliл výberom vhodnejšieho tvaru (simplexu), ktorý pokrýva celý priestor. Napríklad v rovine využíval štvorce, ale existuje aj implementácia s rovnoramennými trojuholníkmi, čím sa znížil počet vrcholov medzi ktorými musíme interpolovať. Časová zložitosť sa výrazne znižuje najmä pri vyšších dimenziách. Simplexný šum sa najčastejšie implementuje ako $2/3/4$ rozmerná funkcia, ale dá sa definovať na ľubovoľný počet rozmerov.

Implementáciu môžeme zhrnúť do 4 krokov: zošikmenie súradníc, rozdelenie, výber gradientu a sčítanie jadra. Pri zošikmení sa vypočítajú nové súradnice pomocou vzorca závislého na dimenzii. Vypočítané súradnice sa používajú na určenie, že v ktorej zošikmenej bunke hyperkocky leží vstupný bod. V jednoduchom rozdelení sa súradnice

zoradia v zostupnom poradí, aby sa následne mohlo učiť, kde leží bod. Výber gradientu môže byť implementovaný viacerými spôsobmi. Najčastejšie sa ale používa dopredu určená permutačná tabuľka, z ktorej sa následne určia gradienty. Princíp je teda podobný ako sme mali pri vylepšenom Perlinovom šume, kde Perlin definoval 12 hodnôt poľa, tu ich ale definuje viac. Sčítanie jadra je posledný krok, kde sa pomocou viacerých vzorcov určuje aký príspevok má každý z vrcholov simplexu.

Týmto sme si približne zhrnuli ako daný algoritmus pracuje. Vyhli sme sa definovaniu jednotlivých funkcií využívaných pri výpočte, pretože ich definícia by nemala pre ďalšiu prácu veľký význam. Jednotivé vzorce sú implementované v kóde aplikácie. Podrobnosti o Simplexnom šume nájdeme v publikácii *Simplex noise demystified* [27]. Podľa tejto publikácie sme následne vytvárali náš kód v implementácii.

2.4 Ďalšie metódy generovania terénu

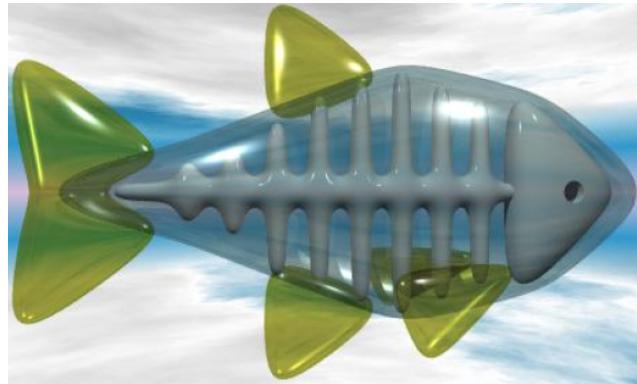
Metódy, ktoré sme spomínali boli založené na fraktáloch a využívali nejaké procedurálne postupy. Možnosti ako vytvárať terény je oveľa viac, nie len tie, ktoré sme spomenuli vyššie. Podrobnejšie sme sa venovali iba tým, ktoré budeme v práci implementovať, ale kvôli vytvoreniu celkového prehľadu spomenieme aj ďalšie používané metódy. Nebudeme sa, ale venovať ich implementačným detailom, ani definovaniu jednotlivých prvkov, ktoré sa v nich využívajú. Spomenieme len základný princíp, na ktorom pracujú a vyzdvihneme výhody resp. nevýhody.

Využitie rôznych funkcií a splajnov

Pri modelovaní terénu môžeme využiť rôzne matematické funkcie, ktoré reprezentujú zaujímavé tvary terénu. Pričom môžeme využívať rôzne druhy funkcií. Napríklad implicitné povrchové modely majú tvar $F(x, y, z) = 0$. Ide o stručnejšie znázornenie ako parametrické plochy a ponúkajú väčšiu flexibilitu pri modelovaní objektov. Pri vytváraní zložitejších tvarov môžeme zmiešať viac implicitných povrchov dokopy a tým sa nám vytvorí zložitejší tvar. Táto technika sa volá *F-rep* (objekty tvorené funkciami). Môžeme si to predstaviť tak, že pomocou jednoduchých funkcií popíšeme nakoniec zložitý tvar. *HyperFun* [1] je špeciálny programovací jazyk, ktorý je určený na modelovanie objektov pomocou funkcií. Pekný príklad môžeme vidieť na obrázku 2.14, kde táto celá ryba je definovaná pomocou funkcií.

Na modelovanie teda môžeme využívať rôzne funkcie. Často sa využívajú splajny buď interpolačné, ktoré prechádzajú danou množinou bodov ako kardinálny, kubický, kvintický splajn alebo aproximačné splajny, ktoré sa len približujú hodnotám ako B-splajn alebo Bézierova záplata. Možnosti využitia funkcií na tvorbu terénov je viacero. Môžeme napríklad definovať celú sieť kriviek a následne zvoliť nejakú vhodnú

implementáciu medzi nimi. To ako spájať viaceré krvinky dokopy, ako vytvoriť plynulé prechody a aké všetky podmienky na to treba dodržať sa dočítame v publikácii *Geometrické modelovanie krviek* [10]. V závislosti od toho akú spojitosť plôch potrebujeme tomu musíme prispôsobiť aj ich funkcie.



Obr. 2.14: Objekt vytvorený zložením viacerých funkcií [1]

Nevýhodou týchto prístupov, ktoré sú založené na funkciách zložitosť ich ovládania. Škálovanie, transformácie a podobné afinne princípy sú veľmi rýchle dátu sú invariantné. Problémy sú s manipuláciou. Ak si zmyslíme, že chceme novú plochu, vyjadrujúcu niečo úplne iné musíme ju vyskladať a definovať celú funkciu. Čiže oproti procedurálnym technikám, kde len zmeníme nastavenia parametrov, je tento prístup náročnejší.

Časticové systémy a simulácie

Základom časticových systémov je častica, ktorá sa v čase stochasticky mení. Výsledný objekt je definovaný veľkým množstvom jednoduchých častíc a tie sa menia. Rodia sa, pohybujú sa, vykonávajú ďalšie funkcie a následne umierajú. Väčšinou sa využívajú na animácie, pretože tam je užitočný pohyb častíc, ale ak by sme sa zamerali na pohyb v rovine vedeli by sme tento pohyb zaznamenať a parafrázovať ako výškovú mapu. Iné využitie časticových systémov môže byť pri simuláciach rôznych erozívnych prvkov. Môžeme mať na začiatku nejaký terén a pomocou časticových systémov ho budeme postupne deformovať. Napríklad dážď patrí medzi základné časticové systémy. Jednotlivé kvapky vody chápeme ako častice. Pomocou častíc môžeme napríklad vytvárať aj korytá riek. Definujeme trasu časticového systému a malé čiastočky budú postupne vymývať časti terénu. Pri simuláciách sa môžu využívať vektorové polia. Tie definujú každému bodu jednoznačne nejaký vektor, ktorý je určený na základe nejakého zákona. Môžeme napríklad určiť zákony erózie, vody, pohybov... Hlavnou výhodou týchto prístupov je, že sú realistické, pretože sa snažia napodobňovať prírodné zákony. Za túto realistickosť, ale platíme výpočtovou náročnosťou a tým, že ich úprava nie je veľmi interaktívna.

Evolučné algoritmy/Skladanie z primitív

Skladanie z primitív je podobný postup ako sme mali pri implicitných povrchových modeloch. Tu ale nevyužívame funkcie, ale hotové 3D objekty. Jednoducho vyskladáme výsledný objekt z jednoduchších objektov, ktoré si definujeme. Vytvorenie terénu si môžeme vysvetliť na princípe informatického stromu. Primitívy, z ktorých sa skladá výsledný terén sú uložené v listoch. Sú to napríklad hory, hrebene, údolia, prípadne cesty alebo jazerá. Následne sa postupuje tak, že sa vnútorné uzly spolu kombinujú. Na kombinovanie sa používajú deformácie, rôzne vzájomné vyrezávanie medzi objektami, miešanie viacerých objektov dokopy... Nadmorská výšku terénu sa určuje prechádzaním celého stromu a kombinovaním toho ako jednotlivé primitívy do celkového terénu prisievajú. Nevýhodou týchto systémov je príliš veľká časová aj pamäťová náročnosť. Aj keď môžu vytvárať dobre vyzerajúce terény, ktoré sa zakladajú na princípe prírodných prvkov, tak je ich ešte stále náročné používať. Druhý problém týchto systémov je, že nevieme jednoducho ovládať výsledné tvary.

Evolučné algoritmy pracujú na podobnom princípe. Definujú sa rôzne predpoklady, následkom ktorých sa vytvára želaný objekt tým, že as prepisuje a nahradza prvotný objekt. Príkladom evolúcie sú aj *L-systémy*, kde sa definujú predpoklady vplyvu prostredia a následne sa vytvorí objekt. Týmto spôsobom sa zvykne vytvárať napríklad flóra. Definujeme nech je viacero rastlín okolo vody a rastliny môžu vnikáť tak, že sa vytvorí stonka a v určitých častiach sa rozdelí a pridajú sa listy a nakoniec kvety. Príkladom fraktálu, ktorý sa vytvára princípom *L-systémov* je Kochova vločka. Podrobnosti o princípe evolučných algoritmov nájdeme v publikácii *Terrain generation using genetic algorithms* [38].

Medzi princípy, kde sa postupnou zmenou vytvára výsledok môžeme zaradiť aj gramatiky. Tie majú definovaný nejaký počiatočný stav a sériu prepisovacích pravidiel, pričom prepisovanie sa ukončí keď sa dosiahne požadovaný výsledok.

Metóda štetcov

Ide o jednoduchú metódu založenú na princípe pokus/omyl. Tieto terény sú vytvárané väčšinou modelármi, ktorí majú určité skúsenosti. Na začiatku je nejaká hmota a modelár ho pomocou funkcií štetca nejako upravujeme. Hmota terénu sa môže uberať, pridávať, rôzne deformovať, všetko záleží od dostupných funkcií softvéru, v ktorom pracujeme. Takúto možnosť ponúka napríklad softvér *Blender*. Výhodou je, že táto metóda pracuje v reálnom čase a niektoré štetce majú prednastavené rôzne funkcie, napríklad aj erózie a tým dokážu kopírovať prírodné zákony. Nevýhodou je prácnosť a nutnosť skúseností.

Softvéroví agenti

Neexistuje presný konsenzus o tom čo vlastne softvéroví agenti sú. My si ich definujeme ako počítačové programy, ktoré pracujú buď súbežne spolu alebo proti sebe. Fungujú bez priamej kontroly a dohľadu. Takže jednotlivý agenti majú úplnú autonómiu, až na obmedzenie toho, čo majú vyrábať. Pre pochopenie si môžeme uviesť príklad práce týchto agentov. Máme 3 agentov, z ktorých každá má inú úlohu. Je tu agent pobrežia, ktorý sa stará o vytvorenie obrysov pevniny, definuje teda plochu, ktorú môže napríkla obklopit aj vodou. Druhý agent, ktorý má na starosti terénnu fázu sa stará o zdvihnutie hôr, tvarovanie nízin, údolí, vytváraní pláží a podobne. Túto úlohu väčšinou robia viacerí agenti. Ďalšou úlohou, ktorú majú na starosti iní agenti je erózia povrchu. V nej sa vyhlbujú korytá, deformuje terén, pribúdajú a ubúdajú jednotlivé časti. Agenti pracujú naraz a asynchronne v rámci každej fázy vytvárania. Agenti môžu vidieť nadmorskú výšku ktoréhokoľvek bodu a môžu ho ľubovoľne upravovať. Podľa niektorých autorov, je prístup softvérových agentov lepší spôsob, ako generovanie pomocou fraktálov, pretože je vysoko parametrizovaný. Tento prístup vytvára dobre vyzerajúce terény. Jednotliví agenti môžu mať podľa publikácie [17] až 12 parametrov. Podrobnosti o tom, ako sa vytvára terén pomocou softvérových agentov sa nachádzajú v publikácii *Controlled procedural terrain generation using software agents* [12].

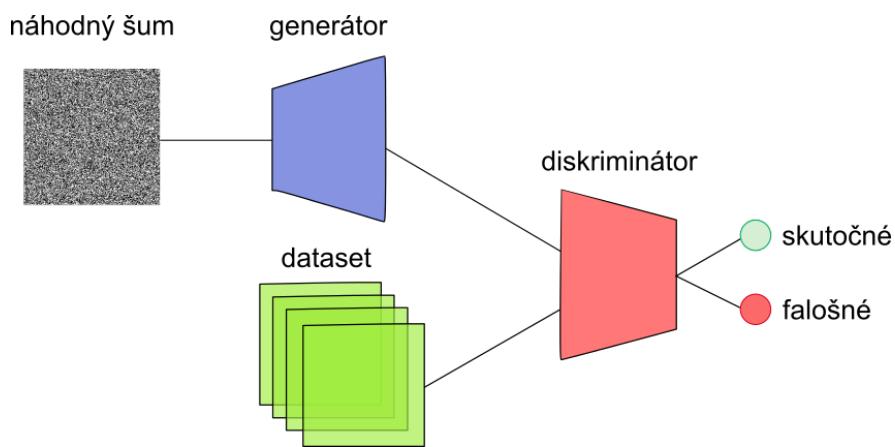
Neurónové siete

Techniku neurónových sietí si priblížme trochu podrobnejšie, aj keď sa nejedná o procedurálnu techniku, ale jedná sa o oblasť, ktorá je v posledných rokoch na silnom vzostupe, takže považujeme za vhodné si povedať trochu viac.

Neurónové siete (*neural network*) sú jedným z výpočtových modelov používaných v umelej inteligencii. Za posledné desaťročie sa stala oblasť hĺbkového učenia (*deep learning*), do ktorej spadajú neurónové siete veľmi populárna. Model sa snaží vychádzať z vlastností biologických nervových systémov. Zjednodušene môžeme povedať, že pracuje v dvoch krokoch. Najskôr prebieha fáza učenia, ktorej cieľom je nastavenie siete tak, aby dávala čo najpresnejšie výsledky. Po natrénovaní siete prebieha fáza vykonávania úlohy. Medzi základné úlohy patrí klasifikácia dát do rôznych tried, rozdelenie obrazu na segmenty, detekcia objektov obrazu a mnoho ďalších zaujímavých úloh. Ide o pomerne novú a rýchlo rozvíjajúcu sa disciplínu, ktorá sa snaží nájsť využitie v stále nových oblastiach.

Svoje využitie si našli neurónové siete aj v oblasti generovania objektov. Existuje viacero architektúr neurónových sietí a pre vytváranie modelov georeliéfov je najrelevantnejšia siet GAN (*Generative Adversarial Network*), pretože tento typ sietí je dobrý v zhodovovaní nových obrázkov. GAN fungujú na princípe postavenia dvoch neurónových sietí proti sebe, ktoré spolu súťažia. Jedna siet sa nazýva generátor, ktorá vytvára

stále nové príklady na základne naučených dát a náhodného šumu. Druhá sieť, ktorá je postavená proti nej, sa nazýva diskriminátor. Tá sa snaží určiť, či sú vstupy, ktoré dostáva, skutočné alebo falošné. Úlohou GAN siete je vyvinúť čo najlepší generátor tak, aby oklamal diskriminátor a ten neboli schopný rozlísiť, ktoré dátá vytvoril generátor a ktoré sú skutočnými dátami z datasetu. S každou iteráciou sa diskriminátor zlepšuje v odhalovaní falošných obrázkov, ale zároveň je generátor stále lepší vo vytváraní nerozoznateľných obrázkov od tréningovej sady. Chceme dosiahnuť, aby boli vytvorené dátá nerozlišiteľné od trénovacích. Príklad princípu GAN vidíme na obrázku 2.15.



Obr. 2.15: Princíp siete GAN

Sieť sa učí na základe vstupného datasetu. Tým sú väčšinou výškové mapy skutočných terénov získaných zo satelitných snímok. Pomocou konvolúcie sa potom hľadajú korelácie medzi údajmi v rámci jednotlivých obrázkov. Naučené prvky potom využívajú generátor na vytváranie nových výškových máp, z ktorých sa potom vizualizujú výsledné modely. V publikácii *Realistic and textured terrain generation using GANs* [46] využívajú autori siet priamo na generovanie výškových máp aj príslušných textúr. GAN sa bude učiť z hodnôt údajov textúry aj výšky zároveň. Kedže vstupom do sieti je vždy jeden obrázok a potrebovali na sieti trénovať výškovú mapu a jej prislúchajúcu textúru zároveň, tak využili farebný model obrázkov RGBA. Kanály RGB nechali pre textúru a kanál alfa využili na reprezentáciu výškovej mapy. Na obrázku 2.16 vidíme príklad spojenia výškovej mapy a príslušnej textúry.

Zahrnutím textúry do trénovania siete sa zvýšila zložitosť údajov o 3 kanály na pixel. To vyžadovalo hlbšiu analýzu tréningových dát a dlhší tréningový proces. Proces učenia závisí aj od hĺbky siete, to znamená kolko vrstiev daná sieť má. Vo všeobecnosti platí, že čím je siet hlbšia, tým viac štruktúr sa môže abstraktne naučiť. S rastúcim hardvérovým výkonom sa budú zväčšovať hlbky sietí a budú schopné detailnejších výstupov. Podrobnosti o tom ako fungujú neurónové siete pri generovaní, ich ďalšie architektúry a príklady nájdeme v publikácii *3D Terrain Generation using Neural Networks* [4].



Obr. 2.16: Spojenie údajov výškovej mapy a textúry [46]

Využívanie neurónových sietí na generovanie terénov môže zmeniť spôsob akým vytvárajú dizajnéri prostredia do hier alebo simulácií. Dnes sa vytvárajú na základe autorovho zámeru, ale zapojenie neurónových sietí prinesie automatizovaný spôsob generovania pri zachovaní témy dizajnu a navyše vytváranie veľkého množstva variácií prostredí, ktoré nebudú obmedzené poznaním autora. Umožní to teda odstrániť časť kreatívneho a vývojového bremena autorov. Neurónové siete ale čaká ešte ďalší výskum, kým bude používateľ schopný získať plnú kontrolu nad funkciami a štruktúrami.

Vzájomné prepojenie techník

Každá z vyššie spomenutých techník má svoje výhody a nevýhody. Niektoré sú rýchle, ale negenerujú terény v dostatočnej kvalite, iné generujú kvalitne vyzerajúce terény, ale máme nad nimi malú kontrolu. Stále teda musíme zaplatiť nejakú cenu. Môžeme si porovnať prístupy, ktoré budeme implementovať v našej aplikácii.

Algoritmus diamant-štvorec má vyššie pamäťové nároky v porovnaní s Perlino-vým a simplexným šumom. Na druhej strane ak porovnávame rýchlosť tak algoritmus diamant-štvorec je výrazne rýchlejší pri vytváraní hodnôt ako šumové algoritmy, ktoré sú v 2D dimenzií približne rovnako rýchle. Čo sa týka kvality výstupu tak v tom má navrch simplexný šum.

Viaceré zo spomenutých techník nemusíme stavať proti sebe, ale môžeme ich využívať súbežne a tým zmierniť ich niektoré negatívne črty. Môžeme vytvoriť terén jednou metódou a použiť ho ako vstup do ďalšej. Týmto prístupom môžeme využiť to najlepšie z každej metódy. Napríklad sa ukazuje ako rozumné, vygenerovať kvalitný šum pre pári hodnôt, ktorý následne môžeme rýchlo upravovať algoritmom diamant-štvorec a vytvoriť tak väčšie vzorkovanie terénu.

Takto môžeme kombinovať aj iné zo spomenutých metód. Napríklad vytvoríme rýchlo terén pomocou nejakej procedurálnej techniky, aj keď nebude kvalitný a použiť ho ako vstup do metódy štetcov. Iným príkladom je vytvorenie si malých, ale dokonalých častí terénu čo nám sice zaberie viac času, ale môžeme to dať ako vstup do algoritmov, ktoré pracujú na spájaní komponentov. Rovnako tak môžeme definovať rôzne splajnové plochy a použiť ich ako základ pre spájanie do väčších celkov.

Kapitola 3

Špecifikácia softvérového diela

V tejto kapitole predstavíme základné požiadavky na našu aplikáciu. Opíšeme, ako by mala fungovať a pre koho by mala primárne slúžiť. V tejto časti sa nebudeme venovať konkrétnym implementačným detailom, to je predmetom kapitoly 4. Popíšeme viaceré možnosti získavania vstupných dát. Najskôr si povieme, kde sa dajú nájsť skutočné zemské súradnice a ako ich môžeme stiahnuť. Potom sa zameriame na vlastné vytvorené dátá. Predstavíme si niekoľko matematických funkcií, ktoré majú tvary zemského povrchu a ukážeme si objekty, ktoré sme získali skenovaním pomocou skenera CRUSE v predchádzajúcej práci.

3.1 Požiadavky na aplikáciu

Pri vytváraní akejkoľvek aplikácie musíme vždy pred implementáciou zodpovedať niekoľko základných otázok.

Čo je cieľom aplikácie? Ako prvé si treba uvedomiť, že naša aplikácia nemá konkurovať softvérom, ktoré sú na trhu už niekoľko rokov a dokážu vytvárať terény na nerozoznanie od skutočnosti. Našim cieľom je poskytnúť používateľovi viacero zaujímavých zemských tvarov terénov, s ktorými môže pracovať a využiť ich neskôr v ďalšej práci.

Pre koho je aplikácia určená? Aplikácia by si mala nájsť širšie publikum. Jednou zo skupín sú bežní používatelia, ktorých zaujíma geometria a procedurálne techniky a chcú si na aplikácii vyskúšať, ako vstupné parametre ovplyvňujú tvar terénu. Druhou skupinou sú používatelia, ktorí využívajú aplikáciu za účelom vytvorenia konkrétneho terénu, aby ho mohli následne použiť podľa svojich potrieb. Či už ako vizualizáciu pre nejaký projekt alebo ako podklad pre hry a simulácie. Poslednou skupinou sú programátori, ktorých zaujíma implementácia aplikácie a chceli by kód, ktorý je voľne dostupný, využiť vo svojej vlastnej práci, prípadne ho ďalej rozšíriť.

Čo je na vstupe aplikácie? Používateľom sa po spustení aplikácie zobrazí možnosť výberu z viacerých foriem terénu. Možnosti reálnych tvarov sú zobrazené na obrázku 1.2. Tieto formy budú uložené v aplikácii pomocou výškových máp, o ktorých sme si hovorili v časti 1.4. Tento formát sme vyбрали z dôvodu jednoduchej reprezentácie a najmä rozšíritelnosti v ďalších aplikáciách. Poskytuje to používateľom väčšiu flexibilitu pri práci s vlastnými dátami. Jednoducho si môžu stiahnuť výškovú mapu terénu z nejakého vlastného zdroja a vizualizovať si ju v našej aplikácii.

Čo je výstupom aplikácie? Výstupom aplikácie je 3D model terénu, ktorý si môžu používatelia stiahnuť a exportovať do ďalších aplikácií. V implementácii poskytneme výber najčastejších formátov na reprezentáciu modelov, aby nenastal používateľovi problém s kompatibilitou pri exportovaní do iných aplikácií. Základným výstupom je výšková mapa, ktorú sme mali na vstupe, takže používateľ nemá problém opakovanie otvoriť terén na ktorom pracoval.

Aké sú ciele dizajnu? Dizajn aplikácie musí byť čo najviac prívetivý ku používateľovi. Hneď v úvode sa zobrazujú všetky terény, ktoré ponúka aplikácia a ak v nich nenájde tvar, ktorý potrebuje, má možnosť vytvoriť si vlastný pomocou šumu. Má možnosť načítania aj vlastného terénu vo forme výškovej mapy. Po zvolení tvaru sa aplikácia presmeruje na hlavný dizajn, kde sa vizualizujú konkrétnie 3D terény.

3.2 Výber jazyka a platformy

Pre vytvorenie našej aplikácie sme uvažovali nad viacerými programovacími jazykmi. Do úvahy pripadali jazyky C#, C, PYTHON, JAVA. Pretože v súčasnosti patria medzi najčastejšie používané programovacie jazyky. Nás program chceme predovšetkým spustiteľný v rodine operačných systémov Windows, čo je momentálne najrozšírenejší operačný systém medzi používateľmi.

Pri výbere sme sa rozhodli pre jazyk C# a to z viacerých dôvodov. Spolu s jazykmi C a C++ patrí medzi najbežnejšie pre vykreslovanie počítačovej grafiky. Pri zobrazovaní objektov v týchto jazykoch sa najčastejšie využívajú knižnice na prácu s grafikou ako OpenGL alebo DirectX. Obe knižnice sú určené na efektívne a rýchle spracovanie prvkov a každá má svoje výhody. Knižnica OpenGL je napísaná v jazyku C a jej hlavným rozdielom od DirectX je, že je multiplatformová. Využíva sa na všetko od videohier cez CAD nástroje, webové prehliadače až po mobilné zariadenia. DirectX je napísaný v jazyku C++ a je určený len pre operačné systémy Windows a XBox. Ak teda chceme vytvárať aplikáciu aj pre iný operačný systém ako je Windows, tak OpenGL je jasná voľba. Rozdielov v týchto knižniciach je viac a nájdeme ich v knihe *OpenGL Game Programming* [5]. Rýchle porovnanie v tabuľke vychádzajúcej z poznatkov tejto knihy sa nachádza na *OpenGL vs. DirectX: A Comparison* [3].

Kedže nechceme programátorov obmedzovať s kódom len na platformu Windows, rozhodli sme sa pre knižnicu OpenGL, aby v prípade budúceho pokračovania na tejto aplikácii mohli plynulo a bez problémov prejsť aj na iné zariadenia a jednoducho tam prekopírovať potrebné časti kódu. Knižnica OpenGL je naprogramovaná v jazyku C, čo posunulo tento jazyk do popredia pri rozhodovaní o voľbe. Nakoniec sme sa ale rozhodli pre jazyk C#, ktorý sa využíva v hernom engine Unity, a teda používateľ môže zobrať kód z implementácie a nechať si ho vykresliť v tomto prostredí. Hlavným aspektom pri výbere bolo predovšetkým poskytnúť používateľovi viaceré možnosti.

Ako vývojové prostredie sme si zvolili Microsoft Visual Studio, pretože medzi jeho vstavené jazyky patrí C#. Ide o vývojové prostredie bohaté na funkcie, v ktorom môžeme priamo skompilovať a spustiť náš vytvorený kód. Veľkou výhodu tohto prostredia je, že poskytuje viaceré open-source knižnice pre tvorbu grafického používateľského rozhrania – GUI (*Graphical User Interface*). Vo vývojovom prostredí Visual Studio máme pri začínajúcim projekte viacero možností šablón (*templates*), na ktorých môžeme začať budovať našu aplikáciu.

Visual Studio obsahuje vývojové platformy .NET Framework a .NET Core. Každá má svoje výhody, ale .NET Core je novšia, takže vývojári pri jej vzniku vychytali niektoré problémy .NET Framework a postupne sa ním budú snažiť nahradit celý .NET Framework. Hlavný rozdiel je, že .NET Core je multiplatformový a z hľadiska výkonnosti je lepší. Ďalsie porovnania nájdeme na *Differences Between .NET Core and .NET Framework* [20]. Aj napriek viacerým výhodám .NET Core sme sa rozhodli pre .NET Framework, pretože s ním máme viaceré skúsenosti a poskytuje dobré grafické šablóny. Ak však programátor nemá skúsenosti ani s jedným z týchto frameworkov a začína budovať novú aplikáciu, odporúčame používať .NET Core.

Potrebujeme si ešte zvoliť GUI framework, ktorý bude pracovať s .NET Framework. Vhodný framework nám poskytne jednoduchšiu prácu na aplikácii s možnosťou využívať viaceré prednastavené prvky GUI. Keby sme si nezvolili grafický framework museli by sme všetky prvky ako sú tlačidlá, textové polia a rôzne interakcie definovať samy. GUI frameworky robia túto prácu za nás. Pre .NET Framework vývojárov sú najčastejšie využívanými Windows Presentation Foundation – WPF a Windows Forms – WinForms .

WPF aj WinForms slúžia na rovnaký účel, ale sú medzi nimi viaceré rozdiely. WPF je novší, a tým zodpovedá viac súčasným štandardom a potrebám používateľov. Je flexibilnejší a umožňuje rozdeliť prácu medzi dizajnéra rozhrania a programátora. Preto sme sa rozhodli, pre aplikáciu vo WPF. Ďalšie rozdiely medzi týmito GUI frameworkami nájdeme na *Difference between WPF and WinForms* [21]. Aplikácia WPF potrebuje na svoje spustenie, aby bol v počítači rámc .NET. Naštastie Microsoft zahŕňa .NET framework do všetkých verzií Windowsu, takže so spustením aplikácie by nemal byť žiadnen problém.

3.3 Zber dát

Jednou z najdôležitejších častí našej aplikácie tvoria dátá. Bez nich by sme nemali na čom spúštať náš algoritmus na tvorbu nových dát. Aplikácia bude spracovávať rôzne druhy dát, všetky vo forme výškových máp, ale budú pochádzať z rôznych zdrojov. Pôjde o reálne zemské súradnice získané z mapovania povrchu a vlastné dátá vytvorené pomocou matematických funkcií, šumov a skenovania. Získavať výškové mapy vytvorené pomocou šumu dopredu nemusíme, pretože naša aplikácia bude šumové funkcie implementovať, a teda si ich vytvoríme za chodu aplikácie.

3.3.1 Reálne zemské súradnice

To ako sa zaznamenávajú reálne zemské súradnice a aké informácie o nich potrebujeme sme si podrobne zhrnuli v prvej kapitole v časti 1.4. Potrebujeme získať DEM dátá reálnych súradníc zemského povrchu. Možnosti ako ich získať je viacero. Existujú rôzne inštitúcie, ktoré takéto dátá poskytujú a väčšina z nich ich ponúka ako voľne dostupné, prípadne za registráciu na ich platforme. Jedna z najznámejších inštitúcií je USGS (*United States Geological Survey*), ktorá sa zaobráva štúdiom krajiny Spojených štátov amerických. Obsahuje rozličné dátá o našej Zemi využiteľné vo viacerých oblastiach. USGS sa zameriava na USA, ale Európska únia má podobný program nazývaný *Copernicus*, ktorý slúži na pozorovanie Zeme a spolupracuje s Európskou vesmírnou agentúrou. Toto sú teda dva hlavné zdroje údajov reálnych zemských súradníc.

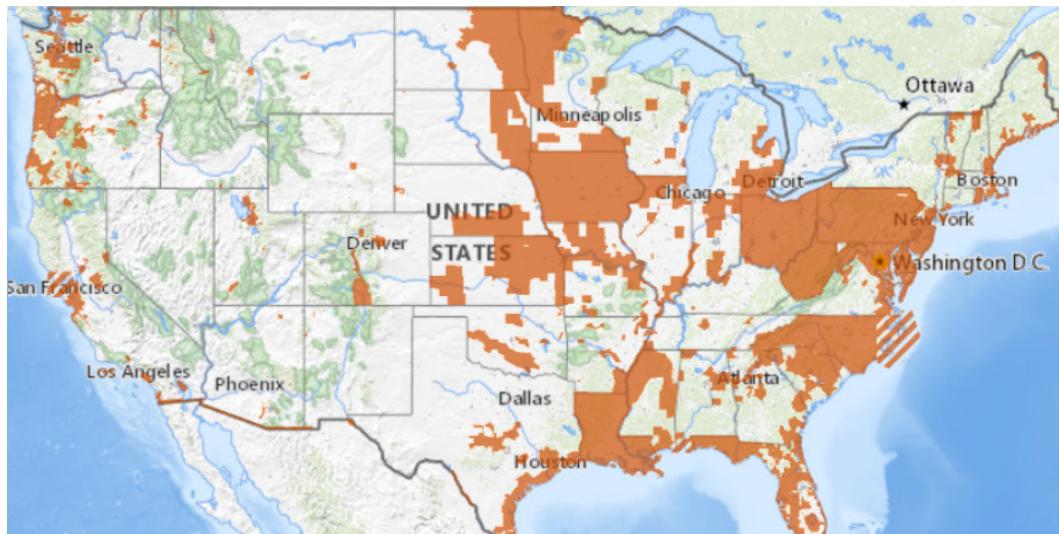
Väčšina inštitúcií, ktoré ponúkajú dátá, píše ich rozlíšenie v oblúkovej miere. Je teda dobré poznať, aké rozlíšenie dát vlastne stahujeme. Nižšie sa nachádza tabuľka 3.1, ktorá nám prevádzza najčastejšie používané oblúkové miery na metre.

Oblúk	Metre (približne)
1 stupeň	110 kilometrov
7.5 oblúkovej minúty	14 kilometrov
30 oblúkových sekúnd	1 kilometer
3 oblúkové sekundy	90 metrov
1 oblúková sekunda	30 metrov
1/3 oblúkovej sekundy	10 metrov
1/9 oblúkovej sekundy	3,4 metra

Tabuľka 3.1: Prevod najčastejšie používaných oblúkových dĺžok

U väčšiny zdrojov nájdeme krajinu mapovanú s rozlíšením 1 oblúkovej sekundy, teda 30 metrov. Pár častí na zemskom povrchu môžeme nájsť aj s rozlíšením 1/9 oblúkovej sekundy, ale je ich veľmi málo, koľko môžeme vidieť na obrázku 3.1. Treba si teda uvedomiť, že nie je jednoduché zozbierať dátá, ktoré opisujú ľubovoľné tvary terénov.

Prvý problém je, že ak si nájdeme terén, ktorého údaje by sme potrebovali, tak nemusí existovať jeho zaznamenanie. Druhý problém je, že môže byť navzorkovaný s malou presnosťou. Šírka a dĺžka väčšiny útvarov, ktoré potrebujeme sa pohybuje rádovo v stovkách metroch, to znamená, že výškové mapy budú mať len niekoľko pixelov. To spôsobí, že nebude vždy úplne jasne badateľný tvar terénu akoby sme si predstavovali.



Obr. 3.1: Časti USA s rozlíšením 1/9 oblúkovej sekundy [46]

Teraz si popíšeme ako môžeme dátá stiahnuť. Od USGS môžeme použiť stránku <https://earthexplorer.usgs.gov/>. Ako prvé sa zobrazí mapa, kde vyklikáme body oblasti, z ktorej nás zaujímajú údaje. Po vytvorení polygónu stlačíme tlačítko *Data Sets*. Následne sa nám zobrazí zoznam dát, z ktorých si vyberieme *Digital Elevation* a následne zvolíme možnosť *SRTM*. Tam si zvolíme dostupnú oblúkovú mieru a klikneme na tlačítko *Result*. Následne si stiahneme dátá vo formáte *.tif*. Môže sa stať, že zvolená oblasť neponúka dátá, prípadne, že sa skladá z viacerých častí. Vtedy si treba postahovať všetky časti, do ktorých daná oblasť zasahuje, a následne si vystrihnúť len tú, ktorú potrebujeme.

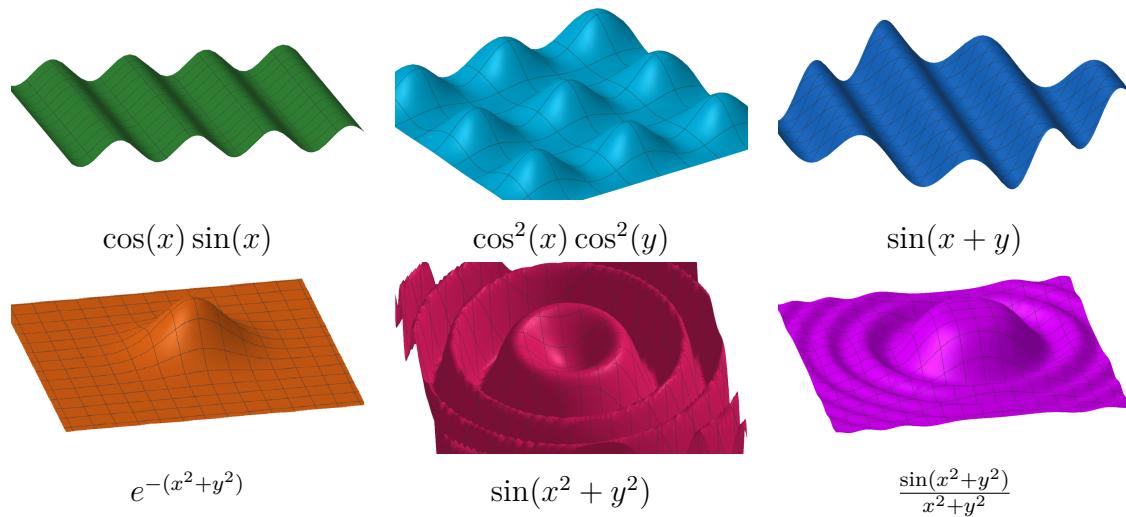
Existujú aj ďalšie stránky, ktoré pristupujú k DEM dátam od USGS. Z dôvodu jednoduchšieho prístupu k dátam, aby používateľ nemusel voliť oblasť, ktorú chce a následne pri nej zistil, že neexistujú dátá, vznikla stránka <https://prd-tnm.s3.amazonaws.com/LidarExplorer/index.html#/>, na ktorej sa dá jednoducho zistiť, ktoré časti Zeme sú zmapované akou oblúkovou presnosťou. Znova však treba stiahnuť viacero obrázkov ak sa oblasť skladá z viacerých častí. Tento problém rieši stránka <https://tangrams.github.io/heightmapper/>, ktorá je priamo prehliadač výškových máp sveta. Má jednoduché rozhranie a rýchlo a jednoducho sa dajú stiahnuť dátá. Nevýhoda je, že sa nedá zvolať a vyfiltrovať vlastná oblúková presnosť. Posledným zdrojom, ktorý si spomenieme sú európske dátá. Môžeme ich stiahnuť z <https://land.copernicus.eu/imagery-in-situ/eu-dem> a sú s rozlíšením 25 metrov.

Využili sme všetky spomínané zdroje, aby sme pripravili čo najvhodnejšie vstupné dátá do aplikácie. Pripravili sa ich v rôznych rozlíšeniach, vždy ale v každom rozmere menšom ako 1000 pixelov. Bežne to zodpovedá $30 \times 1000 = 30\text{km}$, čo už je príliš rozsiahla plocha a nie sú badateľné tvary, na ktoré sa zameriavame. Všetky pripravené dátá sa nachádzajú v priečinku *Heightmaps*.

3.3.2 Vlastné vytvorené dátá

Dátá vytvorené matematickými funkciami

Jednou z možností, ako si pripraviť vlastné dátá, je využiť rôzne matematické funkcie. Ak si totižto vhodne zadefinujeme funkciu, môže veľmi efektívne opisovať zemské tvary, ktoré potrebujeme. Najjednoduchším príkladom je funkcia *sinus*, ktorej vlnu vieme využiť pri viacerých terénoch. Úvalina a thufur sú príklady foriem georeliéfu s opakujúcimi sa kopcovitými tvarmi, takže funkcia síňus sa tu dá využiť. Existuje viačero matematických funkcií so zaujímavými tvarmi, niektoré z nich sa nachádzajú na obrázku 3.2.



Obr. 3.2: Matematické funkcie s tvarmi zemského povrchu

My potrebujeme tieto funkcie previesť na výškové mapy, aby sme ich vedeli načítať našou aplikáciou. Niektoré z funkcií implementujeme priamo do aplikácie a budeme ich patrične vzorkovať a vykreslovať. Výškové mapy z funkcií môžeme vytvoriť v implementácii tak, že si vypočítame súradnice z jednotlivých funkcií v pravidelných rozostupoch a potom tieto hodnoty len zakódujeme do obrázka. Stačí jednoducho naškálovať hodnoty na interval $[0, 255]$ a pridať každému farebnému kanálu RGB rovnakú hodnotu. Dôležité je zvoliť si, v akých intervaloch budeme vzorkovať funkciu. Všetky poskytnuté terény v našej aplikácii by sme chceli mať približne rovnaké. Treba teda zvoliť vzorku tak, aby 1 pixel zodpovedal zhruba 30 metrom.

Dáta získané pomocou skenovania objektov

Skenované dátá máme z predchádzajúcej publikácie *Možnosti využitia hĺbky ostrosti na analýzu obrazu zo skenera CRUSE a 3D rekonštrukciu* [34]. V práci sme využívali hĺbku ostrosti a vertikálny pohyb kamery pri skenovaní objektov a vytvárali tak sekvencie rezov, z ktorých sme následne vyhotovili výsledný obrázok. Dáta sú teda vhodné na využitie v našej práci, pretože máme informácie o výške každého bodu skenovaného objektu, a keďže sme využívali len vertikálny pohyb kamery, nemohlo dôjsť pri skenovaní ku zaznamenananiu previsov (v jednom pixeli obrázka uložené dve výšky). Príklad skenovaných objektov vidíme na obrázku 3.3.



Obr. 3.3: Objekty skenované skenerom CRUSE [34]

Ide o skeny prúteného košíka, štvorstenu a pologuli. Všetky tri objekty vieme využiť v našej práci, pretože ak ich vhodne naškálujeme majú tvary podobné zemskému povrchu. Prútený košík aj pologuľa môžu opisovať napríklad tvar osamej hory, ktorú chceme vykreslovať v našej aplikácii. Ich 3D údaje máme vo formátoch *.stl*, *.obj*, ale aj vo forme obrázkov výškových máp, takže sú priamo pripravené na spracovanie v našej aplikácii. Problémom ale je, že výškové mapy majú rozlíšenie v jednom rozmere od 1500 do 3000 pixelov. Ak by sme to prirovnali ku reálnym zemským súradniciam, kde 1 pixel zodpovedal často 30 metrom, tak by sme mali terény o veľkosti 45 kilometrov, čo je značne nerealistické na jednu osamelú horu. Potrebujeme teda zredukovať počet pixelov. Na redukciu sme využili dostupné softvéry na internete. Znovu sme si sprawili viacero typov rozlíšení, aby sme mali v aplikácii väčší dataset. Všetky vytvorené dátá sme pridali do priečinku *Heightmaps*.

Kapitola 4

Implementácia

V tejto kapitole popíšeme implementačnú časť zhotoveného softvérového diela. V kapitole 3 sme špecifikovali niektoré časti našej aplikácie, a teraz sa budeme venovať ich implementácií a budovaniu celistvej aplikácie. Celá aplikácia sa skladá z dvoch častí, prvá je výberová časť, v ktorej si používateľ zvolí požadovaný typ terénu alebo šumu a druhá, vizualizačná časť, kde sa celý terén zobrazuje a používateľ aplikácie s ním vie manipulovať a meniť parametre.

4.1 Knižnice, pluginy, moduly

Aplikácia je naprogramovaná v jazyku C# a bola vyhotovená v programovacom prostredí Visual Studio. V časti o špecifikácii sme si popísali, že chceme .Net Framework. Pracujeme teda s verziou .NET Framework 4.5.2. Ďalej sme si zvolili GUI framework WPF App, ktorý nám pomôže pri budovaní desktopovej aplikácie pre Windows.

V časti o špecifikácii sme popísali, že chceme pracovať s OpenGL. No táto knižnica je napísaná v jazyku C. To ale nie je problém, my použijeme wapper OpenTK, ktorý slúži na prístup k prvkom knižnice OpenGL prostredníctvom jazyka C#. Wrapper pridáme do aplikácie pomocou funkcie Manage NuGet Packages, ktorá slúži na pridávanie rôznych balíčkov. Potrebujeme nainštalovať balíčky OpenTK, OpenTK.GLControl a MathNet.Numerics. Používame verziu OpenTK 2.0.0 a OpenTK.GLControl s verziou 1.1.2349.61993 a MathNet.Numerics 4.5.1. Balíčky OpenTK slúžia na prácu s grafičkou a balíček MathNet.Numerics máme kvôli použitiu Gasussovej náhodnej premennej. Príklad ako vyzerajú tieto balíčky vidíme na obrázku 4.1. Po pridaní týchto potrebných balíčkov sa môžeme pustiť do implementácie zdrojového kódu. Pri práci v jednotlivých triedach netreba zabudnúť na pridanie direktív using, aby sme mohli s danými knižnicami pracovať, teda konkrétnie *using OpenTK;*, *using OpenTK.Graphics.OpenGL;* a *using MathNet.Numerics.Distributions;*.

	MathNet.Numerics by Christoph Ruegg, Marcus Cuda, Jürgen Van Gael	4.5.1
	Math.NET Numerics is the numerical foundation of the Math.NET project, aiming to provide methods and algorithms for numerical computations in science, engineering and every day use. Supports .Net Framework 4.0 or higher and .Net Stan...	5.0.0
	OpenTK by The Open Toolkit Team	2.0.0
	A set of fast, low-level C# bindings for OpenGL, OpenGL ES and OpenAL.	4.7.4
	OpenTK.GLControl by OpenTK Team	1.1.2349.61993
	The Open Toolkit Library (OpenTK)	3.1.0

Obr. 4.1: Potrebné balíčky na chod aplikácie

4.2 Triedy

Po zvolení predvolenej šablóny WPF sa vytvorilo okno MainWindow, ktoré je kombináciou súborov XAML (*.xaml*) a súboru CodeBehind (*.cs*). Súbory XAML slúžia na vytváranie používateľského rozhrania a do súborov s príponou (*.cs*) píšeme kód v jazyku C#. V našej aplikácii sme potrebovali vytvoriť dve používateľské okná, jedno je hlavné okno, v ktorom sa vykreslujú terény a druhé okno, ktoré sa zobrazí pri štarte aplikácie a ponúka možnosť voľby tvaru terénu podľa potreby. Na začiatku máme teda tieto dve okná:

- **StartWindow.xaml**
- **MainWindow.xaml**

StartWindow

Ide o jednoduché okno, ktoré si vyžadovalo prácu hlavne v súboroch XAML. Potrebovali sme, aby mal používateľ možnosť zvoliť si všetky tvary terénu definované v časti 1.2.1, teda minimálne 12 tlačidiel s možnostami. Okrem toho sme chceli ponúknut používateľovi v tejto časti možnosť načítať aj vlastný terén a vykresliť aj šumové funkcie, takže sme pridali ďalšie 4 tlačidlá a snažili sa, aby rovnomerne vyplnili plochu. Dva z nich sú určené na generovanie šumu, jeden na generovanie náhodnej matematickej funkcie a posledné tlačidlo je určené na otvorenie vlastnej výškovej mapy z počítača. Na zarovnanie tlačidiel sme použili element `<Grid>` a definovali výšku a šírku každého stĺpca a riadku tak, aby sa prispôsobili veľkosti okna. Každému tlačidlu sme potrebovali nastaviť akciu, ktorá sa má vykonať po jeho stlačení. Bolo by neefektívne vytvárať pre každé tlačidlo vlastnú funkciu, preto sme vytvorili jednu akciu `SelectButton`, ktorú sme priradili každému prvku. V triede `StartWindow.xaml.cs` sme tejto akcii priradili funkciu. Tá má zistiť meno aktuálne zvoleného tlačidla a poslať jeho text vo formáte `string` ako parameter pre druhé okno `MainWindow`. Akcia následne otvorí druhé okno, ktoré slúži na vykreslovanie a zavrie aktuálne okno.

MainWindow

Po zvolení niektorého z tlačidiel v predchádzajúcej časti sa otvorilo druhé okno. Jeho štruktúru bližšie popíšeme v časti o používateľskom rozhraní a v tejto časti sa zameziame najmä na implementáciu vykreslovania. Pri implementácii tejto časti sme použili niektoré nastavenia OpenTK zo šablóny [33]. Nastavili sme tu vzdialenosť kamery od počiatočného bodu $(0, 0, 0)$ a jej otočenie v rámci osi z a y . Rovnako sa tu nastavujú ďalšie verejné parametre potrené na vykreslovanie, ako sú napríklad základná farba terénu, farba vodnej hladiny, nastavenie počiatočných hodnôt, ktoré parametre vykreslovať a ktoré nie, a mnohé ďalšie. Tieto premenné nebudem popisovať a určovať ich hodnoty, ich nastavenie sa nachádza v triede *MainWindow.xaml.cs* a malo by byť jasné z názvov premenných a komentárov.

V tejto triede sa nachádza veľké množstvo ďalších funkcií potrebných na chod aplikácie a počítanie úprav terénu. Nebudeme vypisovať na čo všetky slúžia, pretože v kóde sú všetky funkcie nazvané podľa toho na čo slúžia a v častiach, kde je to potrebné sú pridané aj komentáre. Užitočné funkcie, ktoré sa tu vyskytujú sú *GetTerrainFromFunction()*; ktorá z matematickej funkcie urobí terén, alebo sa tu nachádzajú funkcie na manipuláciu s vodnou hladinou a otvorenie a uloženie terénnych dát.

Terrain.cs

Trieda *Terrain.cs* je základným kameňom našej aplikácie. Predstavuje objekt, do ktorého si ukladáme všetky potrebné informácie o aktuálne vykreslovanom teréne a základné funkcie, ktoré tento terén upravujú. Trieda obsahuje viaceré verejné premenné:

- *int Iteration;*
- *double Hurst;*
- *float[,] HeightmapAsArray;*
- *public List<Vector3> Coordinates;*
- *public List<int> Indices;*

Z názvov by malo byť jasné na čo slúžia, ale keby nie tak dvojrozmerné pole (matica) *HeightmapAsArray* uchováva aktuálne používanú výškovú mapu, ktorej hodnoty sme uložili do poľa. Význam toho, prečo posielame pole a nie obrázok, je ten, že nie vždy pracujeme s výškovými mapami, napríklad pri generovaní šumu sa nepoužívajú výškové mapy a hodnoty sa priamo generujú do matice. Pri práci s obrázkami si musíme dať pozor na jednu dôležitú vec, a to indexovanie. Ku pixelom na obrázku pristupujeme tak, že voláme súradnicu x a potom súradnicu y – pomocou funkcie *GetPixel(x,y);*. Súradnica x je hodnota zo šírky obrázka, takže definuje pozíciu stĺpca a súradnica y je

teda riadok. Avšak ak pracujeme s maticou, tak tá indexuje prvky zavolaním pozície riadku a potom pozície stĺpca. Pri ukladaní pixelov do matice teda musíme vymieňať indexy, aby sa nám zachovali totožné rozmery obrázka a matice. Treba na to myslieť, aj pri ďalšej implementácii, pretože takto sa môžeme ľahko dostať k spadnutiu aplikácie. Zamenením šírky a výšky by sme pristupovali mimo rozsahu matice.

Premenná *Coordinates* je dôležitá, pretože uchováva informácie o vrcholoch. Keď dostaneme hodnoty výškovej mapy resp. pola, tie nám určujú výšku, teda len súradnicu z . Potrebujeme si vytvoriť skutočné vrcholy, ktoré budú zo súradnicového systému. Začíname vo vrchole, ktorý je zo záporného kvadrantu roviny xy , a postupne vytvárame body v mriežke, až posledná hodnota výškovej mapy so súradnicami $[height - 1, width - 1]$ bude mať vrchol v kladnom kvadrante roviny xy . Teda presne oproti prvému vrcholu pozdĺž uhlopriečky. Pravidelným rozložením zabezpečíme, že súradnicové osi prechádzajú stredom plochy.

Dôležitou premennou v tejto časti je *Indices*. V nej máme uložený zoznam indexov vrcholov, v akom poradí sa majú vykresľovať. Dôvod prečo to robíme takto je, že je efektívnejšie uložiť indexy, a potom podľa nich pristupovať k vrcholom, akoby sme si vyrobili zoznam vrcholov v poradí v akom sa vykreslujú. Zabrali by sme zbytočnú pamäť a spomalili vykreslovanie aplikácie. Každú štvoricu vrcholov, ktorú získame z výškovej mapy, vykresľujeme ako dva trojuholníky. Teda pre jednu štvoricu vrcholov potrebujeme uložiť 6 indexov. Na čo musíme dať pozor je správna orientácia. V predvolenom nastavení OpenGL sú trojuholníky definované s vrcholmi idúcimi v proti smere hodinových ručičiek, ako tie, ktoré smerujú dopredu. Je dôležité, aby sme indexy uložili v tomto smere, aby sa materiál trojuholníkov zobrazoval správne.

Všetky premenné sa naplnia hodnotami v konštruktore triedy. Hodnoty *Coordinates* a *Indices* sa naplnia pomocou funkcií *GetCoordinatesFromHeightmap()*; a *GetIndices()*; ktoré pracujú na princípe popísanom vyššie. V triede sa ďalej nachádza funkcia *RecomputeCoordinates()*; ktorá slúži na prepočítanie súradníc. Treba si uvedomiť, že niekedy potrebujeme v programe prepočítať len súradnice, napríklad pri škálovaní. Inokedy, ale potrebujeme vypočítať nové vrcholy aj nové indexy, takže voláme konštruktor.

Prepočítanie vrcholov a indexov treba urobiť po zavolaní funkcie *DiamondSquareAlgorithm()*. Tá ako napovedá názov slúži na počítanie algoritmu diamant-štvorec. Funkcia sa zavolá pri každom zvýšení počtu iterácií. Algoritmus sme implementovali podľa krokov 2.2. Na vypočítanie náhodného Gaussovo posunu sme využili objekt *new Normal(mean, stdDev)*; z knižnice *MathNet.Numerics*. My sme len vypočítali štandardnú odchýlku podľa vzorca, ktorý sme spomínali v kapitole 2. Funkcia vracia na výstupe nový terén. Dôležité funkcie používané pri výpočte tejto funkcie sú *DiamondStep()*; a *SquareStep(float[,] heightmapAsArray)*;. Ako prvá sa zavolá funkcia *DiamondStep()*; tá vytvorí nové pole s rozmermi, akými potrebujeme a následne ho ide napĺňať hodnotami v správnych súradničiach. Vytvorili sme jeden prechod starým poľom, aby sme mohli

vypočítať priemery 4 vrcholov, ktoré potrebujeme. Následne sa ku priemeru pripočítá náhodná premenná. Po naplnení hodnotami, ktoré počíta štvorcový krok, sa následne toto pole odošle do funkcie *SquareStep(float[,] heightmapAsArray);*. Tá vypočíta zvyšné hodnoty. Všetky forsykly sme napísali tak, aby sme nepristupovali k hodnotám matice, ktoré nás aktuálne nezaujímajú a ušetrili tak nejaký čas výpočtu.

PerlinNoise.cs a SimplexNoise.cs

Tieto triedy sú implementované na počítanie šumov. Obe triedy napĺňajú v konštruktore dvojrozmerné pole, ktoré je následne posielané do konštruktora terénu. Následne sa vykreslí terén na základe tohto šumu. Tieto triedy majú v sebe parametre, ktoré sme si definovali v časti 2.3 a nazývame ich rovnako, takže bližší popis týchto tried netreba. Nie celý kód, ktorý sme použili je výhradne náš, vychádzame z článku *Simplex noise demystified* [27], ktorý sa venoval vysvetleniu jednotlivých premenných a ponúkol ku ním ukážky kódu, ktorý sme následne modifikovali. Pravdepodobne na generovanie šumov existujú aj viaceré externé knižnice, ktoré by možno vytvárali šum aj efektívnejšie. Avšak my sme chceli do aplikácie vložiť reálny kód, aby si mohol čitateľ v prípade záujmu prejsť jednotlivými krokmi výpočtu.

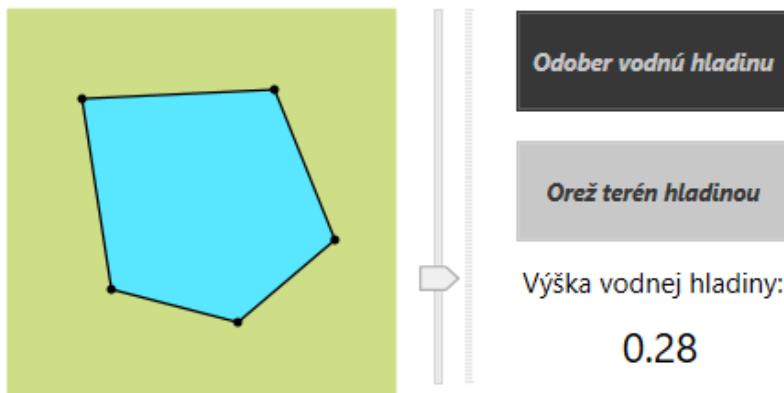
4.3 Postprocessing

V predchádzajúcej časti sme popísali funkcie, pomocou ktorých sme dosiahli vykreslenie terénu. Po vytvorení terénu môžeme uskutočniť postproces. Jednou z funkcií, ktorú sme implementovali, je pridanie vodnej hladiny. Pridaním vodnej hladiny do terénu sa stane z terénu multifraktál, pretože bude mať na rôznych miestach rôzne fraktálne dimenzie. V niektorých častiach bude vytvorený fraktálnou funkciou a na iných bude mať hodnotu roviny ako vodnej hladiny.

Pridávanie vodnej hladiny má v triede *MainWindow.xaml.cs* viaceré funkcie. Potrebovali sme používateľsky prívetivo vymyslieť zadávanie bodov ohraničujúcich vodnú hladinu. Pridali sme do používateľského rozhrania novú plochu, do ktorej môže používateľ zvoliť svoje body, a tie mu následne vykreslia polygón, ktorý sa transformuje na náš vykreslovaný terén. Príklad používateľského rozhrania na zadávanie vodnej hladiny môžeme vidieť na obrázku 4.2. Zadávacia plocha pre body je uzamknutá, kým používateľ neklikne na tlačidlo *Pridaj vodnú hladinu*. Rovnako je uzamknutý aj slider, ktorým ovládame výšku vodnej hladiny a tlačidlo *Orež terén hladinou*. Implementovali sme toto tlačidlo uzamykania, aby používateľ omylem nezvolil náhodné body, čo by v konečnom dôsledku zkomplikovalo jeho prácu.

Nastavenia vodnej hladiny

Volba bodov ohraničujúca
vodnú hladinu:



Obr. 4.2: Rozhranie pre prácu s vodnou hladinou

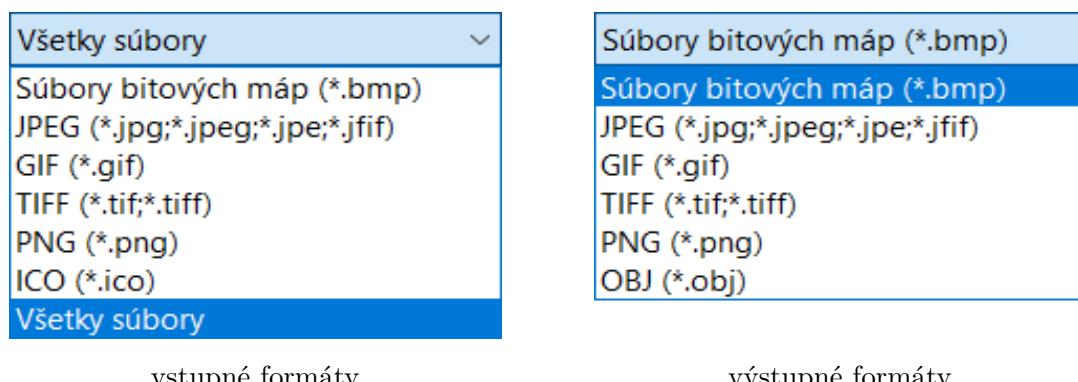
Vodná hladina sa začne vykreslovať, až keď používateľ zvolí aspoň 3 body. S bodmi následne môže manipulovať podľa predstáv. Kliknutím ľavého tlačidla môže body prešúvať podľa potreby a v prípade, že sa rozhodne, že by chcel mať menej bodov vodnej hladiny, môže ľubovoľný bod odstrániť, a to kliknutím ľavého tlačidla na daný bod. Následne sa hladina prekreslí. Zadávané body v používateľskom rozhraní sa naväzujú na body siete terénu. Nevytvárame teda nové body siete, ale zachovávame tie, ktoré už sú vytvorené a vodnú hladinu prispôsobujeme im.

Po stlačení tlačidla *Orež terén hladinou* sa terén oreže a z vodnej hladiny a terénu sa stane jeden celistvý prvok, ktorý uložíme do premennej *MainTerrain*. To je premenná, v ktorej máme uložený aktuálne používaný terén, ktorý vykresľujeme. Tým, že tieto časti splynú sa vodná hladina zafarbí farbou terénu. Následkom tohto sa reštartujú nastavenia pre vodnú hladinu a znova sa uzamkne pridávanie bodov hladiny aj slider na zmenu výšky hladiny. Pridaním ďalšej vodnej hladiny môžeme proces opakovať tolikokrát, kolko potrebujeme. Tým, že môžeme pridávať polynómy vodnej hladiny vlastnej veľkosti a nezaplavujeme rovno celú plochu terénu môžeme zaplaviť postupne jednotlivé časti, a to rôznymi výškami hladiny. Nevýhodou je, že vždy po pridaní ďalšej hladiny sa predchádzajúca hladina zafarbí a stane sa súčasťou terénu. Farebne teda nevidíme na teréne ten pekný efekt viacerých hladín v rôznych výškach.

Druhou funkciou, ktorú máme v rámci postprocesu je vyhladenie terénu. Funkcia má názov *TerrainSmoothing* a vytvára maticu, ktorá následne prejde výškovou mapou terénu a upraví jej hodnoty. Ide vlastne o princíp filtrácie obrazu. Existuje viacero druhov filtrácií, ktoré sa dajú aplikovať na dátu. Takýmto problémom sa venuje počítačové videnie a dobrou publikáciou aj s príkladmi je *Computer vision: algorithms and applications* [48].

4.4 Vstupné a výstupné dáta

V tejto časti preberieme vstupné a výstupné dáta, s ktorými pracujeme v aplikácii. Hlavnými vstupnými dátami sú výškové mapy. Pracujeme len so šedotónovými mapami, pretože v našej jednoduchej aplikácii nepotrebujeme na vizualizáciu viac. Výškové mapy chceme vedieť spracovať vo viacerých formátoch. Na načítanie obrázku používame *System.Drawing.Image.FromFile*, ktorý načíta naše obrázky a my ku ním následne pristupujeme cez jednotlivé pixely. Pri používaní Otváracieho dialógu na výber obrázkov sme nastavili do filtra najčastejšie používané formáty v rastrovej grafike. Prešli sme pritom formáty, ktoré ponúka softvér Skicár v našom počítači a implementovali rovnaké formáty. Všetky bežné formáty, ktoré poskytoval skicár zvládla aplikácia načítať, okrem formátov *.heic* a *webp*. Tie nezvláda načítať a je nutné ich pred použitím prekonvertovať na niektorý z dostupných formátov. Ako výstupné formáty sme samozrejme implementovali všetky obrazové formáty, ktoré sme mali na vstupe. Jediný problém bol s formátom *.ico*, pri ktorom sme pri testovaní aplikácie zistili, že ho nedokáže vytvoriť. Náš použitý postup vytvorí poškodený súbor. Medzi výstupné formáty sme zaradili aj jeden 3D formát, aby sme mohli naše výsledky vyskúšať vyexportovať aj do iných aplikácií. Rozhodli sme sa pre formát *OBJ* (*.obj*), pretože ide o jednoduchý formát podporovaný viacerými aplikáciami. Na serializáciu vrcholov a indexov do formátu *.obj* sme nepoužili žiadnu externú knižnicu, ale naprogramovali sme si vlastnú funkciu *SaveArrayAsOBJ(Terrain terrain, string path)*; Formát má totiž jednoduchú štruktúru, kde stačí vypísať vrcholy v tvare *v x z y*, každý v novom priadku. Následne môžeme pridať do nových riadkov ku zodpovedajúcim vrcholom normálly, ak je to potrebné *vn x z y*. Nakoniec sa pridávajú steny trojuholníkov, ktoré odkazujú na indexy riadkov *f i j k*. Treba myslieť na to, že každý formát má vlastné špecifikácie a môže mať inak pomenované súradnicové osi. V tomto prípade sme museli zameniť súradnicu *y* a *z*, a rovnako indexovanie sme museli posunúť o hodnotu 1. Na koniec súboru sa môžu pridať dodatočné informácie o textúrach, to sa ale netýka našej práce. Dostupné vstupné a výstupné formáty aplikácie sa nachádzajú na obrázku 4.3

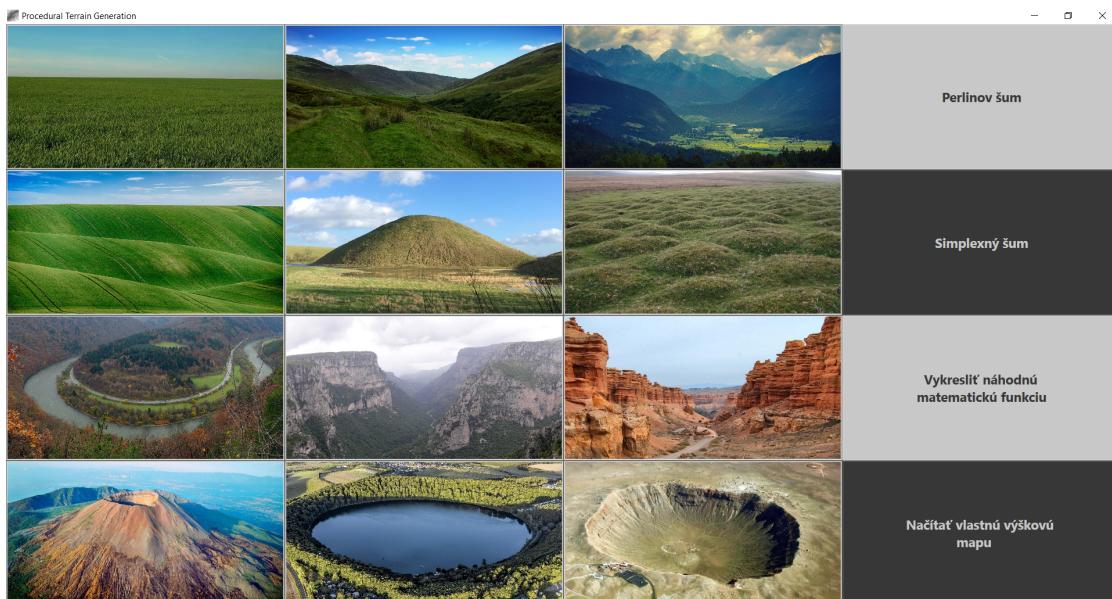


Obr. 4.3: Vstupné a výstupné formáty aplikácie

4.5 Používateľské rozhranie

Používateľské rozhranie sme sa snažili vytvoriť tak, aby bolo pre každého človeka, ktorí pracuje s danou aplikáciou čo najviac prívetivé a intuitívne. Na obrázku 4.4 sa nachádza vyššie spomínaná výberová časť, v ktorej ponúkame používateľovi možnosť vizualizovať terény z časti 1.2.1. Ak chce užívateľ vytvoriť náhodný terén, má možnosť použiť Perlinov alebo simplexný šum. Ďalšou možnosťou je vytvorenie terénu na základe náhodnej matematickej funkcie, ktorá sa vyberá z vopred určeného datasetu. Nachádzajú sa tu funkcie, ktoré pripomínajú zaujímavé terény, ako sme ukazovali v časti o príprave dát. Ak používateľovi nevyhovuje ani jeden z terénov, môže kliknúť na tlačidlo *Načítať vlastnú výškovú mapu*, ktoré otvorí dialógové okno prehliadania a môže si načítať vlastnú mapu alebo vybrať jednu z adresára *Heightmaps*, kde sme uložili všetky výškové mapy, ktoré sme nevyužili v aplikácii.

Farby používateľského rozhrania sme sa snažili zladiť do šedotónovej, aby sme aplikáciu vizuálne prepojili s použitými výškovými mapami. Ako ikonu softvérového diela sme nastavili jednu z výškových máp.



Obr. 4.4: Prvá časť používateľského rozhrania

Po vybratí zvoleného terénu sa aplikácia prepne do druhého okna, kde sa vybraný terén zobrazí a môžeme s ním ďalej pracovať a nastavovať jednotlivé parametre. V lavej časti sa nachádza okno na vykreslovanie a v pravej časti máme pohyblivý panel, kde môžeme meniť parametre terénu, šumu alebo postrocesu. Na obrázku 4.2 sme mohli vidieť súčasť používateľského rozhrania na pridávanie hladiny. Všetky prvky na používateľskom rozhraní sme sa snažili jasne pomenovať a spraviť ich čo najviac prívetivé pre používateľa. Napríklad na zvyšovanie a znižovanie počtu iterácií, sme pridali dva tlačidlá - a +, čo je používateľsky lepšie, akoby musel zadávať do textboxu číslo iterácie.

Spravili sme to tak, pretože väčšinou chceme terén upravovať iba o niekoľko iterácií viac alebo menej. Rovnako na násobenie výšky terénu škálovacím faktorom sa ukázal prívetivejší prístup pomocou slidera namiesto manuálneho zadávania hodnôt. Takto sme sa snažili vytvoriť všetky používateľské prvky na ovládanie hodnôt.

V spodnej časti rozhrania sa nachádza lišta, kde je v ľavej časti vypísaný názov aktuálne vykreslovaného terénu. Vedľa neho sa nachádzajú informácie o dĺžke a šírke terénu, ako aj o počte vykreslovaných trojuholníkov.

Na vrchnej časti je menu. To obsahuje položku súbor, kde sa nachádzajú možnosti vytvorenia nového terénu. Stlačením tlačidla *Nový terén* sa aplikácia prepne späť do režimu štartu. Ďalej sa tu nachádzajú tlačidlá *Otvorit vlastný*, *Uložiť terén* a *Ukončiť aplikáciu*, pričom z názvu je jasné na čo slúžia. Pre tlačidlo *Uložiť terén* sme implementovali bežne vyskytujúcu sa skratku *Ctrl + S* pre urýchlenie prístupu k ukladaniu.

Ďalej sa na lište menu nachádza položka *Terény*, kde si môžeme priamo zvoliť tvary, ktoré sa majú vykresliť a nemusíme sa vracať do prvej časti rozhrania. Je to rýchlejšie, ak si pamätáme, ako jednotlivé tvary vyzerajú podľa názvu. Nasleduje možnosť *Šumy*, kde si môžeme vybrať, ktorý šum sa má generovať. V položke *Funkcie* sa nachádzajú všetky matematické funkcie, ktoré môžeme v aplikácii vykresliť. Z nich sa pri spustení aplikácie vyberala náhodná funkcia, ak sme chceli vykreslovať na základe funkcie. V položke *Zobraziť* môžeme aplikácii prikázať či sa majú zobrazovať súradnicové osi, vykreslovať riadiace pletivo alebo môžeme zmeniť pohľad na terén. Zmenu pohľadu môžeme ovládať aj stlačením kláves *X*, *Y* a *Z*. Poslednou položkou na panely je *Pomoc*, kde sa nachádza manuál k aplikácii. Vyskočí jednoduché dialógové okno s párom základnými informáciami.

Poslednou vecou, ktorú si musíme ozrejmíť pri používateľskom rozhraní je jeho ovládanie. Popísali sme si ovládanie jednotlivých prvkov a potrebujeme ešte ovládanie celej aplikácie. Stlačením pravého tlačidla myši a jej pohybom ovládame zmenu pohľadu na terén. Pravým tlačidlom môžeme zvoliť jeden z vrcholov terénu a posúvať ho v smere osi *z*. Koliesko myši slúži na približovanie sa ku terénu.

Kapitola 5

Výsledky práce

V tejto kapitole zhrnieme dosiahnuté výsledky našej práce. Zhrnieme, kde všade majú využiteľnosť informácie zhromaždené v prvých dvoch kapitolách a aj to, ako by sa dali využiť výstupy z nášho softvérového diela. V druhej časti priblížime testovanie aplikácie, aby sme popísali, v ktorých častiach má problémy a ukážme si niekoľko výstupov, ktoré sme získali z našej aplikácie. V záverečnej časti tejto kapitoly popíšeme možnosti budúcej práce a vylepšenia vzniknutej aplikácie.

5.1 Využiteľnosť informácií

Výsledkom našej práce je podrobne spracovanie problematiky procedurálnej tvorby terénu. V prvej kapitole sme poskytli dostatočné informácie o rôznych možnostiach reprezentácie terénu, spôsobe zaznamenávania reálnych zemských súradníc a ich využitie. Je to vhodný základ pre ďalšie nadvážujúce práce v týchto oblastiach. Zhrnuli sme všetky potrebné informácie do jedného uceleného celku. Tieto informácie môžu nájsť uplatnenie v prácach zameraných na geomorfológiu, kartografiu alebo na iné prírodné vedy. Druhú časť práce prvej kapitoly sme venovali bližšej reprezentácií terénu. Popísali sme mračná bodov, trojuholníkové nepravidelné siete, aj rastrové modely spôsobom zrozumiteľným aj pre úplných začiatočníkov v tejto oblasti.

Druhú kapitolu sme sa snažili zamerať na technický aspekt. Je určená výhradne terminológii o generovaní terénu, a už sa nevenuje podrobnostiam z oblasti geomorfológie. Môže byť zaujímavá pre informatikov aj matematikov. Je samozrejme napísaná štýlom, zrozumiteľným pre akéhokoľvek čitateľa a sú v nej vysvetlené všetky potrebné termíny využívané v práci. A ak nie, sú vhodne doplnené citáciami, kde sa dajú dočítať podrobnosti pre danú oblasť.

Tretia a štvrtá kapitola sú venované softvérovému dielu. To nám vizualizuje viaceré terény a umožňuje ich úpravu. Výsledný softvér slúži ako pomôcka k tejto práci, ale poskytuje aj možnosť vyexportovať terén v 3D formáte.

Príkladom využitia vytvoreného terénu sú napríklad počítačové hry. Používateľ môže vyexportovať tvar terénu, a následne ho použiť vo svojej aplikácii podľa potrieb. Inou možnosťou využitia terénnych dát sú napríklad inštitúcie, ktoré potrebujú 3D modely pre rôzne simulácie prírodných prvkov. Aby si na rôznych tvaroch zobrazili napríklad správanie vody alebo vplyv iných prvkov. Ďalším využitím môžu byť simulácie určené na výcvik v rôznych oblastiach. Vojaci alebo piloti často trénujú v rôznych simulovaných prostrediach prepojených na virtuálnu realitu.

5.2 Verifikácia a testovanie softvérového diela

Aplikácia, ktorú sme vytvorili nemá konkurovať dostupným softvérovým dielam, ktoré sa venujú procedurálnemu generovaniu terénov. Tie vedia generovať zložité a realisticky vyzerajúce tvary v priebehu niekoľkých sekúnd a využívajú rokmi získané poznatky z oblasti počítačovej grafiky. Takými aplikáciami sú napríklad Terragen [45] alebo Outerra [47]. Oba softvéry dokážu vykreslovať vysokokvalitný terén spolu s textúrami, vodnými tokmi, flórou, a to všetko dynamicky a vo vysokom rozlíšení. Príklady výstupov z týchto programov môžeme vidieť na obrázku 5.1.



Terragen

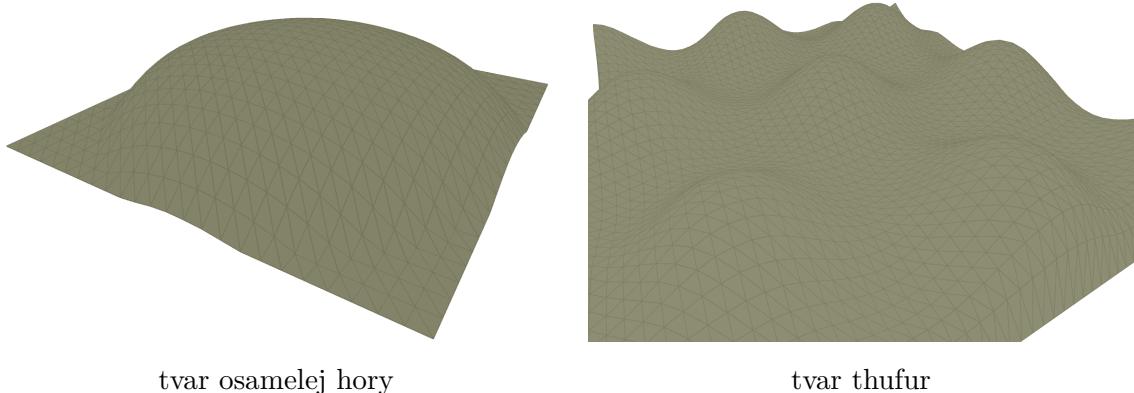


Outerra

Obr. 5.1: Terény generované profesionálnymi softvérvami

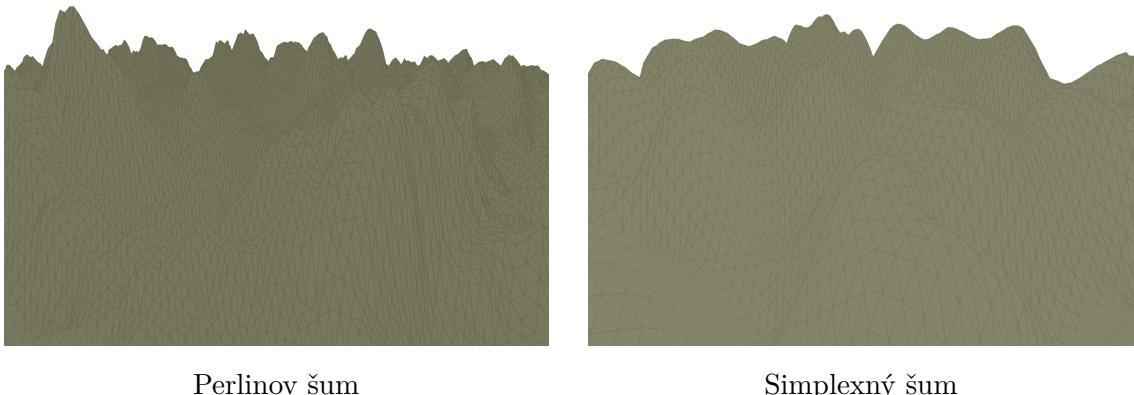
Naša aplikácia je jednoduchá a má poslúžiť čitateľovi tejto práce, aby si jednotlivé nadobudnuté informácie mohol spojiť s vizuálnymi reprezentáciami. V úvodnej časti softvérového diela sa zobrazujú rôzne formy georeliéfu, ktoré sme definovali v časti 1.2.1. Všetky vopred stanovené tvary sme skúšili pri testovaní aplikácie vykresliť. Každé z implementovaných tlačidiel generuje iný tvar terénu, aj keď možno nie vždy na prvý pohľad stotožniteľný so zvoleným tvarom. Príkladom sú tvary kotlina, dolina alebo tiesňava, v ktorých nemusia byť badateľné rozdiely pre bežného používateľa aplikácie. Tým myslíme, že nedokáže priradiť jednoznačne, ktorý terén reprezentuje daný tvar. Spôsobené je to vzorkovaním a obtiažnym získavaním reálnych zemských súradníč, ktoré sme si opisovali v časti 3.3.1. Avšak dôležité je, že po zvolení týchto tvarov sa

vykresľujú rôzne terény, čo poskytuje užitočnú rozmanitosť do našej aplikácie a rôzne tvary môžu mať využiteľnosť pre viaceré oblasti. Na obrázku 5.2 môžeme vidieť ako sú niektoré tvary vizualizované v našej aplikácii.



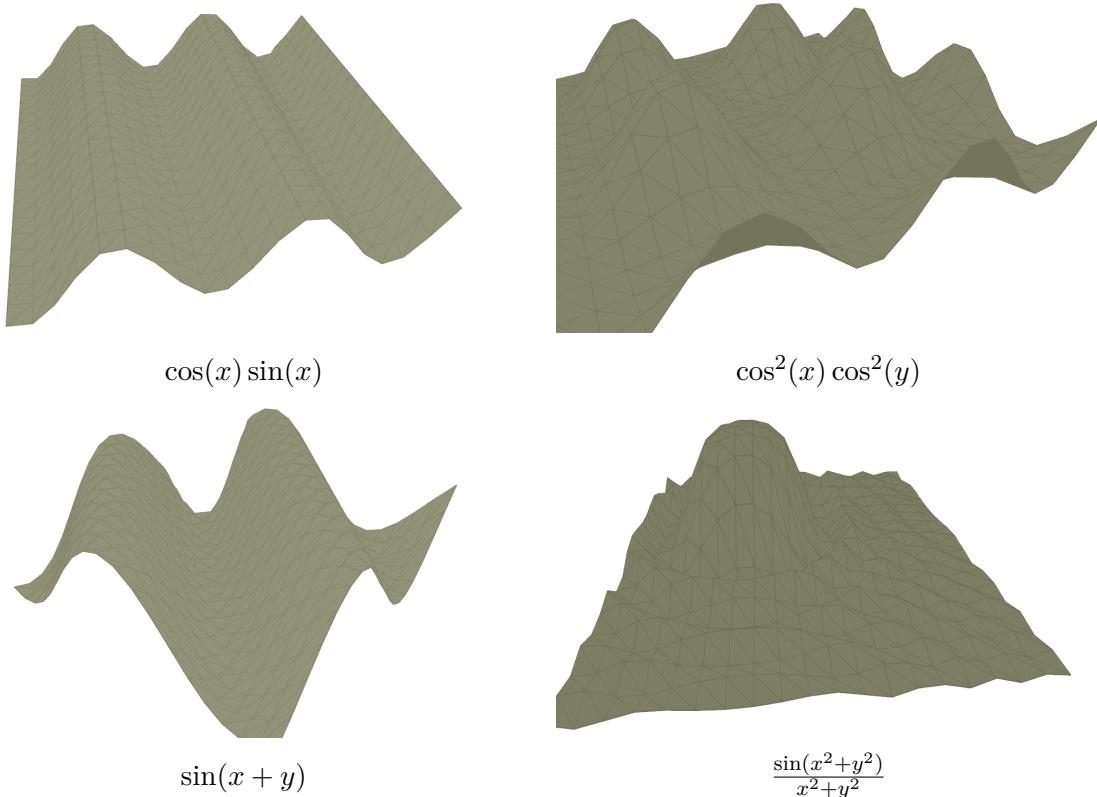
Obr. 5.2: Formy georeliéfu vykreslené aplikáciou

V úvode našej aplikácie sa ďalej nachádzajú tlačidlá na generovanie šumov. Konkrétnie Perlinovho a simplexného šumu. Po kliknutí na tlačidlo generovania zvoleného šumu, aplikácia otvorí druhé okno, kde sa nám zobrazí terén generovaný na základe vytvoreného šumu s počiatočne definovanými parametrami. Táto časť teda funguje správne. Terény vygenerované aplikáciou zo šumovej funkcie môžeme vidieť na obrázku 5.3. Na obrázku môžeme vidieť jemnejšie prechody pri simplexnom šume.



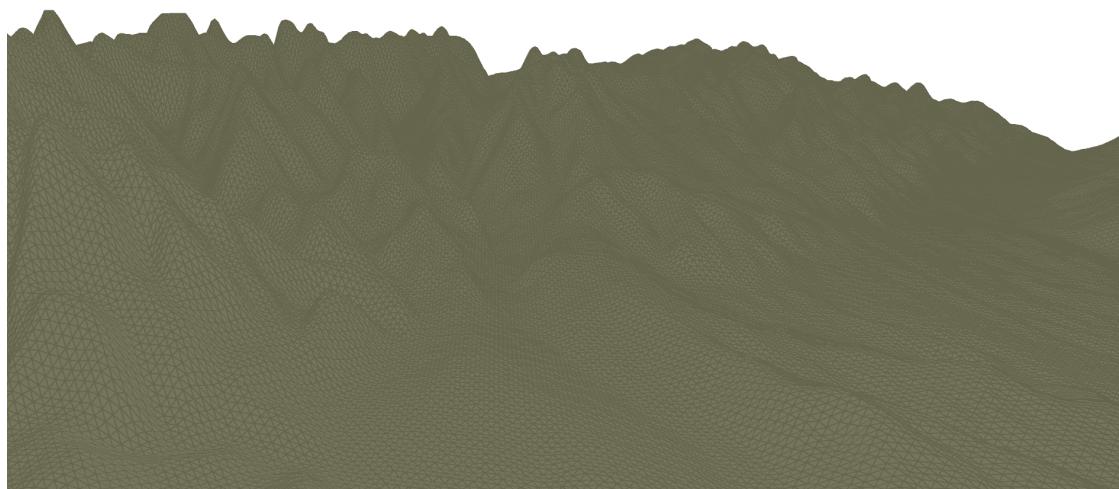
Obr. 5.3: Terény vykreslené na základe šumu

Ďalším tlačidlom úvodnej časti aplikácie, ktorý potrebujeme otestovať, je *Vykresliť náhodnú matematickú funkciu*. Po stlačení tlačidla sa nám vykreslí matematická funkcia z vopred definovaného datasetu. Otestovali sme si vykreslenie všetkých dostupných matematických funkcií, ktoré sme implementovali do aplikácie a všetky zvládla naša aplikácia správne vykresliť. Príklady terénov vykreslených na základe matematických funkcií môžeme vidieť na obrázku 5.4.



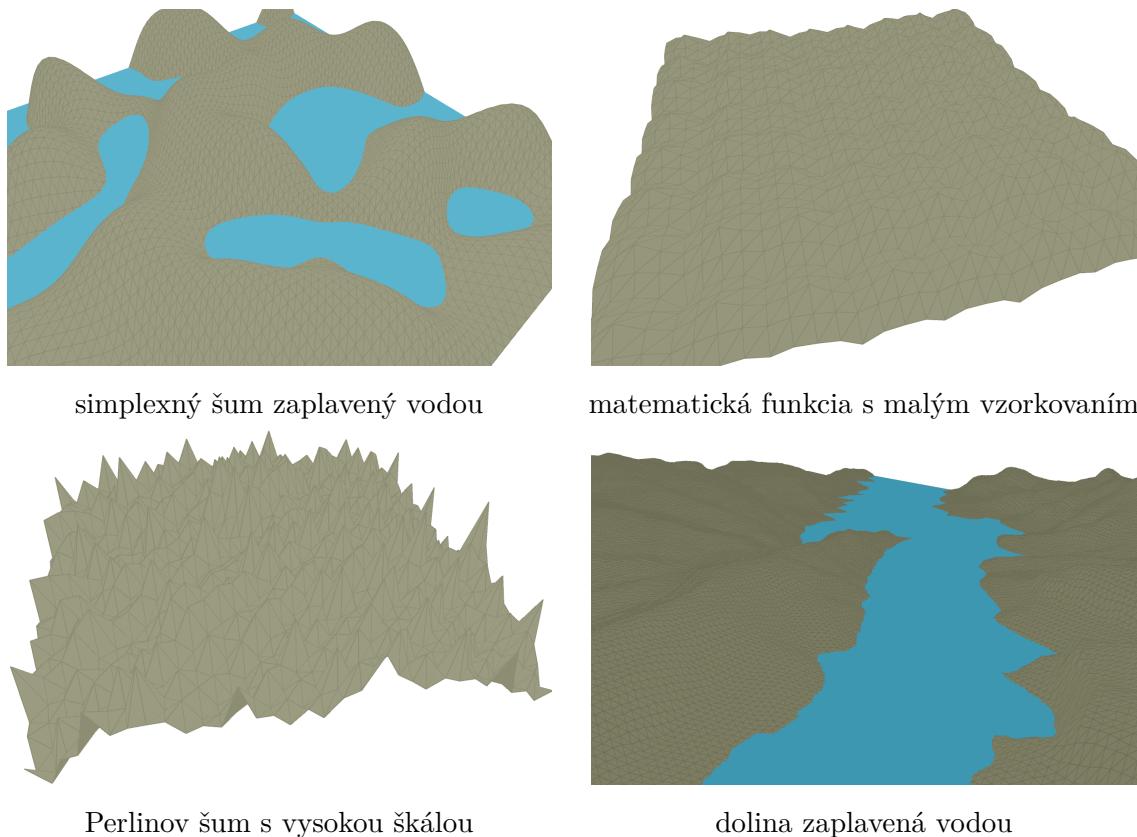
Obr. 5.4: Terény vykreslené na základe matematických funkcií

Posledným tlačidlom na úvodnej stránke aplikácie je otvorenie vlastného terénu z počítača. V priečinku *Heightmaps* sa nachádzajú terény, ktoré sa nedali zvoliť v úvode aplikácie. Nachádzajú sa tu výškové mapy, ktoré sme získali pri zbere dát v časti 3.3. Teda ďalšie reálne zemské tvary, matematické funkcie a skenované objekty. Vyskúšali sme niektoré náhodne vykresliť a všetky dokázala aplikácia zobraziť. Na obrázku 5.5, môžeme vidieť vykreslený terén získaný z výškovej mapy Vysokých Tatier.



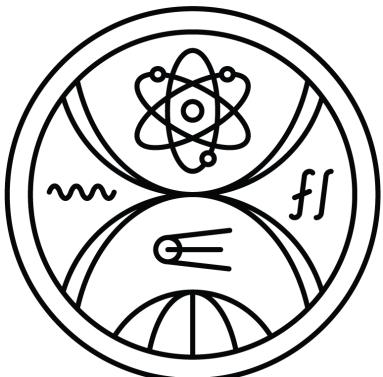
Obr. 5.5: Terén zobrazujúci časť Vysokých Tatier

Ako posledné treba otestovať rôzne parametre, ktoré môžeme na vygenerovaných terénoch meniť. To je otestovanie algoritmu diamant-štvorec, škálovanie výšky terénu, pridávanie vodnej hladiny alebo jednotlivé parametre patriace terénom šumu, funkcií a podobne. Vyskúšali sme rôzne nastavenia parametrov a aplikácia nespadla. Príklady vykreslených terénov po rôznych nastaveniach používateľského rozhrania môžeme vidieť na obrázku 5.6.

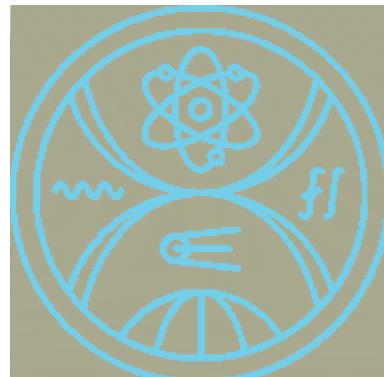


Obr. 5.6: Rôzne nastavenia parametrov aplikácie

Doteraz sme skúšali otvárať len výškové mapy, ktoré sme si samy pripravili. Ostáva otestovať či naša aplikácia zvládne otvoriť výškové mapy v rôznych formátoch a či vyplňe chybovú hlášku pri nepodporovanom formáte. Všetky formáty obrazových súborov nastavených pri implementácii zvládla aplikácia otvoriť bez problémov a pri pokuse o otvorenie textových súborov vo formátoch *.txt*, *.docx*, *.rtf*, *.pdf*, ako aj pri videoformátoch *.mkv*, *.mpg*, zvládla aplikácia vypísať chybovú hlášku. Na obrázku môžeme vidieť ako sa zobrazil terén na základe obrázka, ktorý neboli výšková mapa. Vybrali sme si logo fakulty matematiky, fyziky a informatiky Univerzity Komenského (FMFI). Po vykreslení terénu sme ho následne zaplavili vodou, aby bolo výraznejšie vidieť jeho obrys. Aplikácia ho zvládla bez problémov vykresliť.



logo FMFI

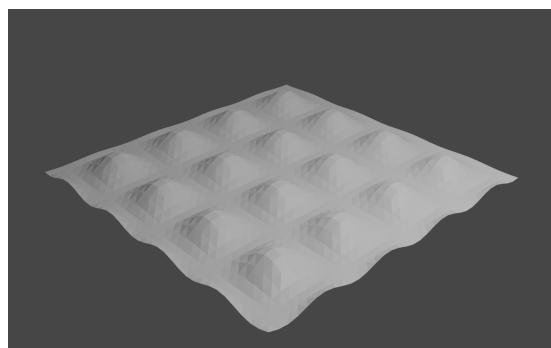
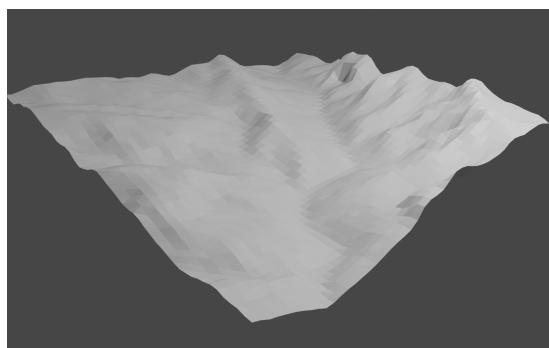


terén vytoverný z loga

Obr. 5.7: Terén na základe obrázka, ktorý neboli výšková mapa

Obmedzením, ktoré treba pri práci s aplikáciou brat do úvahy je, že nemôžeme donekonečna zvyšovať iterácie algoritmu diamant-štvorec, a tým zjemňovať pletivo. Kedže pamäť počítača je konečná, konečné sú aj naše možnosti zjemňovania. Rovnako sme obmedzení aj rýchlosťou vykreslovania. S narastajúcim počtom bodov sa zvyšujú nároky na vykreslovanie, takže aplikácia sa spomaľuje. Pri obrázku veľkosti 1000*1000 pixelov musí aplikácia zvládať vykresliť 1 996 002 trojuholníkov, a toto číslo rádovo rastie pri každej iterácii, takže sa aj výrazne spomaľuje naša aplikácia. Aplikácia dokáže ešte plynulo pracovať pri rozlíšení 300 bodov na dimenziu, čo nám pri 1 oblúkovej sekunde, v ktorej sa nachádza väčšina terénov, bohaté stačí.

Okrem vstupov by sme mali otestovať aj výstupy. Súbory *.obj* sa nám vytvárajú správne, ale potrebujeme zistiť či ich zvládnú otvoriť iné aplikácie. Otvorenie sme otestovali na jednom z najrozšírenejších programov na modelovanie *Blender*. Ten zvládol terény vytvorené v našej aplikácii otvoriť bez problémov. Na obrázku 5.8 sa nachádzajú obrázky vyrenderované pomocou programu *Blender*. Ak by sa vyskytol nejaký program, ktorý by nepodporoval otvorenie nášho terénu, môžeme ho otvoriť v programe *Blender* a následne vyexportovať podľa potreby daného programu. Týmto sme splnili veľmi dôležitý cieľ a to kompatibilitu našej aplikácie s inými softvérmi. Vďaka tomuto môžu byť naše terény využívané v iných aplikáciách.



Obr. 5.8: Terény vyrenderované v aplikácii Blender

Pri testovaní používateľského rozhrania sme narazili na chybu, kedy aplikácia spadne. Ak sa pri posúvaní oddelovača, ktorý sa nachádza medzi vykresľovacou časťou a panelom s parametrami, zakryje celá vykresľovacia časť, tak aplikácia spadne. Posun oddelovača na pravú stranu, a teda maximalizáciu vykresľovacej časti, aplikácia zvládla. Spôsobené je to tým, že ak zmizne vykresľovacia časť, aplikácia vráti niektorým parametrom chybné hodnoty a funkcie siahajú potom mimo definované časti. Túto skutočnosť treba opraviť v implementácii.

5.3 Možnosti budúcej práce

Zhotovené softvérové dielo je jednoduchá aplikácia určená pre potreby tejto práce. To znamená, že existuje viacero možností na vylepšenie. Jednou z možností rozšírenia je brať ako vstupné dátá, nielen obrázky vo forme výškových máp, ale možnosť načítania rôznych 3D formátov bežne používaných v iných aplikáciách. Rovnako je priestor na rozšírenie aj pri výstupných dátach, kde je možné implementovať viac 3D formátov a rozšíriť tak kompatibilitu aplikácie. Ak by sa aplikácia rozšírila o možnosť zafarbovania terénu, tak by bolo vhodné doimplementovať možnosť importovania a exportovania textúry.

Ďalšou možnosťou vylepšenia aplikácie je pridanie ďalších procedurálnych metód na vytváranie terénov. Pridanie ďalších typov šumov alebo implementácia algoritmov rekurzívneho delenia. Pridať sa dajú aj ďalšie funkcie reprezentujúce tvary terénu, ale aj ďalšie parametre upravujúce už hotový terén.

Oblastou s najväčším množstvom možností na rozširovanie je postproces. Dajú sa pridať ďalšie nástroje na manipuláciu s terénom, napríklad rôzne štetce ako ponúka napríklad softvér *Blender*, ktoré vedia zaujímať meniť tvary terénu. Dôležitým a užitočným rozšírením je pridanie možnosti spájania viacerých terénov dokopy. Výborným nástrojom by bola možnosť vytvoriť si jeden terén, uložiť si ho na pozadí aplikácie, a po vytvorení ďalšieho terénu len zvoliť hranicu spoločného napájania, kde by sa vytvoril plynulý prechod.

Medzi ďalší postproces môžeme zaradiť pridanie textúr, flóry alebo iných objektov. No týmto by sme sa už nezameriavalí na tvary terénu, ale snažili sa vytvoriť komplexné prostredie, čo nebolo cieľom našej práce. Možným rozšírením je aj prechod na iné platformy alebo vizualizácia terénu pomocou virtuálnej reality. Existuje rad ďalších užitočných funkcií, ktoré by sa dali implementovať a vytvoriť skutočne profesionálne softvérové dielo. To všetko môže byť predmetom budúcich prác.

Záver

V práci sme postupne zaznamenali celý proces vytvárania procedurálnych terénov. V prvej kapitole sme podrobne popísali proces získavania reálnych zemských súradníc spolu s ich digitálnou reprezentáciou a stručne sme prešli historiu vývoja jeho zaznamenávania. Vytvorili sme teda kvalitný základ pre budúce práce podobného charakteru a spojili najdôležitejšie informácie do chronologicky nasledujúceho postupu vytvárania terénov.

V druhej kapitole sme sa zamerali viac na digitálne modely a opísali najrôznejšie spôsoby vytvárania nových terénnych údajov. Čitateľovi to poskytuje základný prehľad dostupných metód a otvára možnosti do budúcej práce. Spravili sme tu prehľad procedurálnych techník, kde sme postupne popisovali ako sa vyvíjali a aké sú ich klady a zápory. Tým sme splnili dva ciele určené na začiatku našej práce, a to opisanie dostupných metód a ich vzájomné porovnanie.

Vo zvyšku práce sme sa venovali vlastnej aplikácii, ktorú sme vytvorili na vizualizáciu terénu a jeho dodatočného úpravu, aby mal čitateľ možnosť si niektoré veci vyskúšať. Aplikácia dokáže vizualizovať zvolené terény získané z reálnych zemských súradníc a dokáže ich kombinovať s procedurálnou metódou diamant-štvorec. Dokáže pracovať so vstavanými dátami, ale vie spracovať aj cudzie dátá vo vybraných formátoch. Zvláda export do 3D formátu a teda je kompatibilná s ďalšími aplikáciami. V aplikácii sú viaceré parametre, ktoré sa dajú upracovať a tým ovplyvňovať tvar terénu. Aplikácia teda splňa základné požiadavky, čím sme naplnili aj posledný ciela tejto práce.

Literatúra

- [1] Valery Adzhiev, Richard Cartwright, Eric Fausett, Anatoli Ossipov, Alexander Pasko, and Vladimir Savchenko. Hyperfun project: Language and software tools for f-rep shape modeling. *Computer Graphics & Geometry*, 1(2):75–100, 1999.
- [2] Pankaj K Agarwal, Lars Arge, and Andrew Danner. From point cloud to grid dem: A scalable approach. In *Progress in Spatial Data Handling*, pages 771–788. Springer, 2006.
- [3] Alex Allain. *OpenGL vs. DirectX: A Comparison*. Retrieved from <https://www.cprogramming.com/tutorial/openglvsdirectx.html>.
- [4] Rodrigo de Matos Pires Tavares de Almeida. *3D terrain generation using neural networks*. PhD thesis, Instituto Universitário de Lisboa, 2020.
- [5] Dave Astle and Kevin Hawkins. *OpenGL Game Programming*. Premier Press, 2001.
- [6] Michael F Barnsley. *Fractals everywhere*. Academic press, 2014.
- [7] Michael F Barnsley, Robert L Devaney, Benoit B Mandelbrot, Heinz-Otto Peitgen, Dietmar Saupe, Richard F Voss, Yuval Fisher, and Michael McGuire. *The science of fractal images*, volume 1. Springer, 1988.
- [8] Mária Bizubová. *Atlas vybraných foriem georeliéfu*. Bratislava, PriF UK, 2015.
- [9] Edwin Catmull and James Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-aided design*, 10(6):350–355, 1978.
- [10] Pavel Chalmovianský, Barbora Pokorná, and Martina Bátorová. *Geometrické modelovanie kriviek*. Univerzita Komenského v Bratislave, FMFI, 2017.
- [11] Peter Collier, David Forrest, and Alastair Pearson. The representation of topographic information on maps: the depiction of relief. *The Cartographic Journal*, 40(1):17–26, 2003.

- [12] Jonathon Doran and Ian Parberry. Controlled procedural terrain generation using software agents. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):111–119, 2010.
- [13] David S Ebert, F Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.
- [14] Marek Fabrika and Hans Pretzsch. *Analyza a modelovanie lesných ekosystémov*. Technická Univ., 2011.
- [15] Andrej Ferko, Martin Samuelčík, and Veronika Šprláková. *GEOMETRIA FRAKTÁLOV*. FMFI UK, Bratislava, 2015.
- [16] Alain Fournier, Don Fussell, and Loren Carpenter. *Computer rendering of stochastic models*. *Communications of the ACM*, 25(6):371–384, 1982.
- [17] Miguel Frade, Francisco Fernandez de Vega, and Carlos Cotta. Automatic evolution of programs for procedural generation of terrains for video games. *Soft Computing*, 16(11):1893–1914, 2012.
- [18] Thomas E French. *A Manual of Engineering Drawing*. New York: l'icGraw-Hill Book Company, 1911. Inc.
- [19] Michal GALLAY. Digitálne modelovanie reliéfu v open-source gis, 2015.
- [20] geeksforgeeks.org/. *Differences Between .NET Core and .NET Framework*, 2020. August 28. <https://www.geeksforgeeks.org/differences-between-net-core-and-net-framework/>.
- [21] geeksforgeeks.org/. *Difference between WPF and WinForms*, 2021. May 28. <https://www.geeksforgeeks.org/difference-between-wpf-and-winforms/>.
- [22] GISGeography. *A Complete Guide to LiDAR: Light Detection and Ranging*, 2015. August 23. Retrieved from <https://gisgeography.com/lidar-light-detection-and-ranging/>.
- [23] GISGeography. *Top 6 Free LiDAR Data Sources*, 2015. January 19. Retrieved from <https://gisgeography.com/top-6-free-lidar-data-sources/>.
- [24] GISGeography. *5 Free Global DEM Data Sources*, 2016. May 2. Retrieved from <https://gisgeography.com/free-global-dem-data-sources/>.
- [25] GISGeography. *What is photogrammetry?*, 2020. February 24. Retrieved from <https://gisgeography.com/what-is-photogrammetry/>.

- [26] GISGeography. *Learn Synthetic Aperture Radar (SAR) by Example*, 2020. November 27. Retrieved from <https://gisgeography.com/synthetic-aperture-radar-examples/>.
- [27] Stefan Gustavson. *Simplex noise demystified*. <https://weber.itn.liu.se/~stegu/simplexnoise/simplexnoise.pdf>.
- [28] Hans Karl Heidemann. *Lidar base specification*. Technical report, US Geological Survey, 2012.
- [29] Yih-ping Huang. Triangular irregular network generation and topographical modeling. *Computers in industry*, Elsevier, 12(3):203–213, 1989.
- [30] Eduard Imhof. *Cartographic relief presentation*. Walter de Gruyter GmbH & Co KG, 2015.
- [31] Michal Kuderjavý. *Procedurální generování terénu v reálném čase*. PhD thesis, Masarykova univerzita, Fakulta informatiky, 2010.
- [32] NASA Jet Propulsion Laboratory. *Space Radar Image of Kilauea, Hawaii - Interferometry 1*, 1994. Images retrieved from <https://www.jpl.nasa.gov/images/pia01763-space-radar-image-of-kilauea-hawaii-interferometry-1> and <https://www.jpl.nasa.gov/images/pia01762-space-radar-image-of-kilauea-hawaii>
- [33] Marcel Makovník. *OpenTK šablóna pre vykreslovanie plôch na kurze modelovanie kriviek a plôch (2)*, 2020.
- [34] Pecko Marcel. *Možnosti využitia hĺbky ostrosti na analýzu obrazu zo skenera CRUSE a 3D rekonštrukciu*. FMFI UK, Bratislava, 2019.
- [35] Gavin S. P. Miller. *The Definition and Rendering of Terrain Maps*. *SIGGRAPH Comput. Graph.*, 20(4):39–48, 1986.
- [36] Nasa.gov. *NASA WorldWind*, 2019.
- [37] RA Norheim, VR Queija, and RA Haugerud. *Comparison of LIDAR and INSAR DEMs with dense ground control*. In *Proceedings, 2002 ESRI International User Conference: San Diego, California*, 2002.
- [38] Teong Joo Ong, Ryan Saunders, John Keyser, and John J Leggett. Terrain generation using genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1463–1470, 2005.

- [39] Andrew Panoramio, Shane Torgerson and Lizz Roe. *Obrázky foriem georeliéfu.* Images retrieved from <https://pixabay.com/> and <https://i.pinimg.com/736x/8a/19/5c/8a195ce50cc27c688792bdc89bf6862e.jpg> and <https://www.amusingplanet.com/2016/07/charyn-canyon-grand-canyon-of-central.html> and <https://www.prirodovedci.cz/storage/images/800x600/1484.jpg> and https://commons.wikimedia.org/wiki/File:Neolithic_Silbury_Hill,_Wiltshire.jpg#metadata.
- [40] Jagdish K Patel and Campbell B Read. *Handbook of the normal distribution*, volume 150. CRC Press, 1996.
- [41] Tom Patterson. *Physical Map of the Coterminal United States*, 2006. Retrieved from <http://www.shadedrelief.com/physical/pages/download.html>.
- [42] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.
- [43] Ken Perlin. Improving noise. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 681–682, 2002.
- [44] Fabio Rocca, Alessandro Ferretti, Andrea Virgilio Monti-Guarnieri, Claudio Maria Prati, and D Massonnet. *InSAR processing: a Mathematical Approach*. ESA Publications, 2007.
- [45] Planetside Software. *Terragen*. <https://planetside.co.uk/>.
- [46] Ryan Spick and James Walker. *Realistic and textured terrain generation using GANs*. In *European Conference on Visual Media Production*, pages 1–10, 2019.
- [47] Outerra s.r.o. *Outerra*. <https://outerra.com/>.
- [48] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.
- [49] Melnychuk Vladyslav. *Landscape generation using procedural generation techniques*. Ukrainian Catholic University, Lviv, 2020.
- [50] Martin Weinmann. *Reconstruction and analysis of 3D scenes*, volume 1. Springer, 2016.
- [51] Eric W Weisstein. *Surface*. Wolfram Research, Inc., 2008.
- [52] Wingtra. *Drone photogrammetry vs. LIDAR: what sensor to choose for a given application*, 2019. October 18. Retrieved from <https://wingtra.com/drone-photogrammetry-vs-lidar/>.

Prílohy

Na externom pamäťovom médiu priloženému k práci sa nachádza zdrojový kód programu spolu so spustiteľným súborom vytvorenej aplikácie. Okrem výškových máp využívaných v aplikácii, nájdeme v prílohe aj modely terénov v ich 3D formáte. Nachádzajú sa tam aj niektoré 3D modely skenovaných objektov, ktoré sme v práci využívali na vytváranie terénov.

Na priloženom médiu sa ďalej nachádza elektronická verzia tejto diplomovej práce vo formáte *.pdf* spolu so všetkými obrázkami, ktoré boli použité. Obrázky, ktoré neboli prebraté z externých zdrojov a vytvárali sme ich samy sú v prílohe aj vo formáte *.svg* a sú voľne dostupné na ďalšie použitie a úpravu.

Zdrojový kód spolu s vyššie spomenutými dátami je zverejnený aj na stránke <https://github.com/marcelpecko/master-thesis>. V súbore *README.md* sa nachádzajú základné informácie o softvéri.