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

Bakalářská práce

Fotorealistický rendering terénu

Plzeň 2011 Milan Staffa

Prohlášení

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

V Plzni dne 27. dubna 2011

Milan Staffa

Abstract

Photorealistic terrain rendering

This thesis is focused on the problem of photorealistic terrain rendering. Main task is to create modules for photorealistic terrain rendering in RenderMan Shading Language (RSL) and make photorealistic output images of requested terrain. Output images of these modules have to be comparable with outputs of other programs with similar field of use.

In order to provide greater variability, graphical user interface (GUI) frontend was made in C#. This program simplifies creating these modules and offers wide range of options.

Obsah

1 Úvod

Cílem této práce je snaha o přiblížení se realistickému vyobrazení terénu (tak jak to dělají například k tomuto účelu určené profesionální programy). Nejde tedy o tvorbu simulace reálného terénu, jde pouze o čistě výtvarný prostředek vyobrazení terénu do obrázku. Ke splnění tohoto cíle je nutno seznámit se se specifikací RenderMan a pomocí této specifikace vhodně implementovat bloky pro tvorbu výsledného obrázku.

Vytvořené obrázky budou porovnávány jak se skutečnými fotkami tak i s uměle vytvořenými obrázky terénů (které byly vytvořeny pomocí jiných programů).

Moderní počítačová grafika se v dnešní době snaží dosáhnout co nejvíce fotorealistického zobrazování uměle vytvořených objektů. Tato snaha započala malířstvím, kdy se umělci snažili napodobit co nejlépe skutečné krajiny. Dnes je tato snaha patrná nejvíce ve vývoji počítačových her a v animovaných filmech, kde jsou tyto techniky podstatou realističnosti. Pro co nejpřesvědčivější zobrazování se klade důraz hlavně na přirozené osvětlení a vyobrazení povrchu objektů.

Před zobrazením terénu je nutno si zvolit reprezentaci terénu samotného, na který se následně pak aplikují shadery. Úkolem shaderů je napodobovat na terénu co nejvíce realistickou texturu (obarvení terénu) a dodatečné úpravy geometrie původního tvaru terénu.

Práce pojednává o vytvoření modulů a jejich aplikaci na jednoduchou geometrii reprezentující terén tak, aby výsledný efekt byl co nejvíce fotorealistický. Tyto moduly po složení dávájí dohromady shader, který se na terén aplikuje a obarví ho do výsledné podoby. Pro psaní shaderů byl zvolen programovací jazyk RenderMan Shading Language.

Pro uživatelskou přístupnost bude v jazyce C# vytvořen interaktivní GUI program (nazvaný *RTC - RenderMan Terrain Creator*), který podle nastavených hodnot vygeneruje všechny zdrojové soubory pro vytvoření výsledného obrázku s obarveným terénem.

2 Teoretická část

2.1 Fotorealistické zobrazování terénu

Pro vytvoření obrázku s fotorealistickým zobrazením terénu je potřeba si určit, z čeho a jak bude tento obrázek vytvořen. Je tedy nutné určit tvar terénu, jaká bude textura tohoto terénu (respektive jak definovat obarvení povrchu tak, aby co nejvíce pozorovateli připomínal reálný povrch terénu). Dále je potřeba určit, jakým světlem bude terén osvětlen, zda a jaké budou ve scéně s terénem působit atmosférické jevy (tedy mlha atd) a podobně.

V tomto textu si popíšeme, jak reprezentovat terén. Digitální model terénu (DMT) budeme reprezentovat v trojrozměrném prostoru. DMT se rozumí prostorový geometrický popis reliéfu terénu. Na tomto terénu se dají dále modelovat nejrůznější objekty (například budovy, vegetace a podobně). Pro samotný DMT jsou však tyto objekty nedůležité.

Mezi charakteristické vlastnosti DMT patří:

- terén je nepravidelná plocha. V této ploše se nacházejí jak místa s hladkou geometrií tak s narušením této hladké geometrie. Zvláštními částmi terénu jsou vrcholy, sedla, údolnice (což je křivka spojující místa největšího vyhloubení příčného řezu údolím) a hřbetnice (ta je opakem údolnice - tedy hřbetnice spojuje relativně nejvyšší body terénního tvaru), které mají podélně často hladký průběh, avšak v kolmém směru může dojít k ostrému lámání plochy. Těmto úkázům se říká „singularity“.

- jelikož se vychází z reálných terénů, výšková souřadnice (většinou jde o souřadnici z) je vzhledem k souřadnicím délky a šířky DMT v nižší hodnotách. Z toho důvodu jsou také v DMT nižší převýšení.

- k popisu plochy je potřeba velké množství dat (z důvodu rozsáhlosti digitálního modelu terénu).

- zobrazovaný terén lze většinou z matematického hlediska chápat jako funkci dvou proměnných $z = f(x, y)$, kde funkční hodnota z je jednoznačně přiřazená a vyjadřuje výšku terénu v daném bodě $[x,y]$. Problémem jsou pouze převisy a svislé plochy, které samozřejmě neodpovídají charakteru zobrazení (respektive funkce dvou proměnných),

protože jednomu bodu [x, y] je přiřazeno více výškových hodnot *z*. Singularity lze chápat z matematického hlediska jako místa s nespojitými parciálními derivacemi funkce popisující tvar terénu.

2.2 Datová reprezentace terénu

Pro snadný popis terénu se často používá rozdělení celé plochy DMT na menší části, které se dají jednodušeji geometricky popsat. Podle charakteristiky těchto plošek se rozlišují následující typy modelů:

- Polyedrický model
  - V tomto modelu jsou elementární plošky tvořeny trojúhelníky. Tyto trojúhelníky k sobě přiléhají a tvoří tak mnohostěn (bez podstavy), který se přimyká k terénu. Vrcholy mnohostěnu (tedy jednotlivé vrcholy trojúhelníků) jsou body na terénní ploše. Ty jsou souřadnicově předem určené.
  - Interpolace plochy terénu se provádí lineárně po trojúhelnících. Vrcholy trojúhelníků je však vhodné zvolit tak, aby vystihovaly nejen obecně průběh tvaru terénu, ale i jeho singularity.
  - Popis tohoto modelu je rozdělen do *geometrické* části, která udává souřadnice vrcholů trojúhelníků, a do části *topologické*, jejímž úkolem je přiřazovat vrcholy jednotlivým trojúhelníkům a udržet sousednost trojúhelníků.
- Rastrový model
  - Tento model je dán množinou elementárních plošek nad prvky pravidelného rastru[1]. Jde vlastně o čtyřúhelníky, které je možno rozdělit na trojúhelníky.
  - Výhoda tohoto modelu je v tom, že pracuje s pravidelnou maticí uzlových bodů, které se dají snadno vypočítat a není nutné o nich udržovat všechny údaje. Naopak nevýhodou může být různorodý terén, který obsahuje například rozsáhlé rovné plochy a zároveň vysoké pohoří. V tom případě je tedy nutné tento model rozdělit na menší modely, které reprezentují daný typ povrchu a zpracovávat je ve vhodném rozlišení, každého tohoto podmodelu.

[1]Rastr = mřížka bodů.

  - Rastrový model je v principu definován prostorovými souřadnicemi každého bodu rastru (tedy souřadnicemi bodu x, y, z). Při praktickém použití stačí určit vzdálenost bodů rastru a umístit jeden bod do souřadného systému. Všechny ostatní body se pak snadno dopočítají. Z toho tedy vyplývá, že prakticky použitelný rastrový formát může obsahovat pouze:
    * souřadnice jednoho rohu rastru
    * úhel natočení rastrové sítě
    * rozměr jednoho prvku rastru
    * matici výškových hodnot každého bodu rastru

- Plátový model
  - Tento typ modelu předpokládá, že se povrch rozdělí na nepravidelé, obecně křivé plošky trojúhelníkového tvaru, přičemž hranice se vedou po singularitách (v případě, že se v modelu vyskytují). Používají se také rovněž obecné n-úhelníky (např. čtyřúhelníky). Těmto n-úhelníkům se říká pláty (*patch*).
  - Rozdělení plochy pomocí plátů nám umožňuje zachycovat singularitu a charakteristické body terénu.
  - Pro další vysvětlování se bude předpokládat tento model.

Více o digitálních modelech terénu v [12].

2.3 RenderMan shading language

RenderMan shading language (dále jako *RSL*) je součástí specifikace RenderMan Interface a slouží k definici shaderů.

Shadery vytvořené podle *RSL* jsou přeložitelné v jakémkoli kompilačním nástroji RenderManu (například *AQSIS*, *Pixie*, *JrMan* a další).

RSL definuje pět typů shaderů:

- **surface shader** - definuje optické vlastnosti tělesa
- **light shader** - definuje vlastnosti světla

- **displacement shader** - modifikuje povrch tělesa
- **imager shader** - slouží pro konečné korekce
- **volume shaders** - skupina tří dalších shaderů:
  - **external volume shader** - pro modifikaci světla procházejícího skrz nějaké prostředí (např. atmosféru)
  - **internal volume shader** - při průhlednosti tělesa modifikuje světlo procházející skrz něj
  - **atmosphere shader** - ovlivňuje světlo jdoucí od tělesa ke kameře

Bližší informace v [1, 2, 3, 11].

V této práci se budu zabývat převážně surface shaderem, pomocí kterého se bude napodobovat fotorealistické obarvení porvchu terénu. Obarvení bude provedeno pomocí tzv. procedurální textury. Procedurální textura se dá definovat jako vyjádření textury pomocí matematické funkce. Její výhodou je, že nezáleží na jejím rozlišení, protože procedurální textura se přizpůsobí velikosti vykreslovaného (jinak řečeno renderovaného) obrazu. Nevýhodou je však, že ne všechny povrchy se dají pomocí matematických funkcí vyjádřit.

2.4 Reprezentace ploch terénu v RenderManu

V RenderManu se pro popis komplikovaných objektů používají pláty (viz kapitola 2.1). Při tvorbě ploch pomocí plátů používáme pro navazování jednotlivých plátů tzv. *plátování*. Pro plátování je potřeba vysvětlit pojem C spojitost.

Pojem C spojitost znamená: dva pláty mají napojení C^0, mají-li společnou krajní stranu (hranu), která je křivkou třídy[2] alespoň C^0.

RenderMan podporuje pláty bilineární a bikubické.

[2]Křivka má spojitost C^0, pokud pro každé dva sousední křivkové segmenty, ze kterých je tato křivka složena platí, že koncový bod jedné křivky je počátečním bodem křivky druhé. Dva segmenty křivky mají spojení C^1, pokud je tečný vektor v koncovém bodě prvního segmentu křivky roven tečnému vektoru v počátečním bodě druhého segmentu křivky (jsou si rovny první derivace v obou bodech). Stejným způsobem lze určit C^2 spojistost (jsou si rovny nejen první derivace, ale i druhé derivace). Obdobně lze definovat C^n spojitost.

2.4.1 Plochy tvořené bilineárními pláty

V RenderManu se používá dvou typů plátů - bilineárních a bikubických. Hlavním stavebním prkvem bilineárních plátů jsou úsečky (přímky). U bikubických plátů to jsou křivky třetího stupně (kubiky).

Plát bilineární plochy je obecně udán dvěmi okrajovými úsečkami a_0 a a_1. Tyto úsečky se dají charakterizovat jako vektorové funkce $a_0(t)$ a $a_1(t)$, kde parametr t je z intervalu $t \in \langle 0, 1 \rangle$. Detailnější popis je v [4] a konkrétní realizace pro RenderMan je popsána v [9].

Kvalitativně je použití těchto plátů ne příliš dokonalé (popis této nedokonalosti je popsán v [7]).

2.4.2 Plochy tvořené bikubickými pláty

Bikubické pláty se pro modelování plochy terénu hodí více než bilineární (viz předchozí kapitola). Popíšeme si Coonsovy B-spline pláty, které byly zvoleny v této práci za nejvhodnější pro reprezentaci povrchu terénu.

Tyto pláty se obecně snadno navazují a z tohoto důvodu jsou vhodné pro modelování (v našem případě pro modelování terénu). B-Spline plochy n-tého stupně mají C^{n-1} spojitost (viz kapitola 2.4) a to ve všech svých bodech. Při změně některého z řídících bodů měníme tvar vždy pouze určité části B-Spline plochy.

B-spline plát definuje $(m+1) \times (m+1)$ bodů (m odpovídá řádu plochy). Při plátování se použije pro nový plát m řad ze stávajícího plátu a přidá se další řada m bodů. Tedy pro přidání dalšího plátu se použije $m \times (m+1)$ bodů ze stávajícího plátu. Více v [4].

Nejednodušší z B-spline ploch jsou bikubické B-spline plochy. Jelikož v našem případě jde právě o kubiky (respektive *Coonsovy B-spline pláty*), tak se při tvorbě plochy jednotlivé pláty překrývají ve třech sloupcích a přidává se jeden nový sloupec nebo se překrývají ve třech řádcích a přidává se jeden nový řádek. Protože jsou třetího stupně (tedy jsou kubické), je zajištěno, že jsou C^2 spojité. Proto se používají ve složitějších modelovacích programech.

Podrobněji o tvorbě těchto ploch v RenderManu v [14].

Podrobněji o těchto plátech v [4] a o jejich implementaci v RenderManu v [9].

2.5 Šumové funkce

Pro vytváření přirozených (tedy náhodných) povrchů a pohybů je snaha často používat klasické generátory pseudonáhodných čísel, které mají za úkol simulovat nepředvídatelné výsledky.

V této práci využijeme šumu na napodobování přírodních jevů, jaké se objevují v reálném světě. Pro jejich napodobení se často používá spojitých funkcí (například tzv. Perlinova šumu a turbulence). Pro tuto práci jsem si vybral turbulenci, jelikož její výsledky více odpovídají reálné nepravidelnosti.

Pro pochopení turbulence je potřeba nejdříve vysvětlit, co je to Perlinův šum a skládání šumových funkcí.

Bližší informace o šumových funkcích v [4, 5].

2.5.1 Perlinův šum

Perlinův šum patří k nejpoužívanější a nejrozšířenější formě šumu v počítačové grafice. Pro popsání Perlinovy šumové funkce vycházím z [4, 10].

Ken Perlin navrhl šumovou funkci, kterou lze rychle vypočítat a která splňuje následující požadavky:

- Perlinova šumová funkce je spojitá
- je statisticky invariantní vzhledem k otáčení a posunutí
- má omezené frekvenční spektrum a je opakovatelná[3].

Podstatou Perlinova šumu je generování spojitého šumu, který je vypočítáván v diskrétní mřížce. Perlinův šum lze definovat v libovolné dimenzi. Pro další vysvětlení budeme předpokládat trojrozměrný případ.

[3]Při zavolání funkce s parametrem x vždy vrátí stejnou hodnotu y.

Základem je šumová funkce *noise(x,y,z)*, která pro hodnoty [x,y,z] vrací vždy stejné náhodné číslo z intervalu hodnot $\langle -1, 1 \rangle$. To, že tato funkce vrací vždy stejnou hodnotu je vlastně splněním podmínky, že má být výsledek opakovatelný.

Hlavní myšlenkou Perlinovy šumové funkce je rozdělení prostoru do pravidelné mřížky, jejíž vrcholy si označíme jako [i,j,k]. Souřadnice vrcholů mřížky nabývají celočíselných hodnot. V každém z těchto vrcholů je definována pseudonáhodná trojrozměrná funkce, která se označuje jako *wavelet* (neboli volně přeloženo *vlnka*). Vlnka má tzv. poloměr, který udává její rozsah (tzn. že hodnoty vlnky za tímto poloměrem jsou nulové). Vlnka zároveň prochází počátkem. To tedy znamená, že pro parametry [i,j,k] je její hodnota nulová. Proto se tedy nemusí brát v úvahu hodnota vlnky v počátku a definuje se pouze gradient vlnky.

Pro výpočet Perlinovy šumové funkce v bodě [x,y,z] je použit následující algoritmus:

- Určíme osm nejbližších vrcholů mřížky (tedy určíme buňku, ve které se bod [x,y,z] nachází).
- Pro každý z těchto osmi vrcholů spočteme tvar vlnky.
- Hodnoty vlnek (ovlivněné vzdálenostmi od daných vrcholů vůči bodu [x,y,z]) jsou sečteny.

Ken Perlin zjednodušil přiřazení gradientů vrcholům na jednorozměrné pole (256 vektorů), ze kterého se vybírá gradient (jednotkový vektor) pro daný vrchol. Všech osm vrcholů buňky má přiřazeny gradienty

$$g_{i,j,k} = G[P[P[P[i] + j] + k]],$$

kde P je předem stanovená permutace, která náhodně rozmisťuje gradienty do pole G. Pro vytvoření gradientů g je zvolen takový algoritmus, aby se tyto různé gradienty neopakovaly.

Po přiřazení gradientů okolním vrcholům je nutno vypočítat vliv těchto vrcholů na bod [x,y,z]. Vliv je dán relativní vzdáleností bodu [x,y,z] od vrcholu [i,j,k]. Tato vzdálenost se dá vyjádřit

$$[u, v, w] = [x, y, z] - [i, j, k], -1 \leq u, v, w < 1.$$

Podle Perlina je pokles hodnoty funkce se vzdáleností dán kubickou funkcí[4]

$$drop(t) = 1 - 3t^2 + 2t^3.$$

Celkový úbytek $\Omega(u, v, w)$ v bodě $[u, v, w]$ je dán jako

$$\Omega(u, v, w) = drop(u) \cdot drop(v) \cdot drop(w).$$

Na konec vynásobíme relativní úbytek $\Omega(u, v, w)$ náhodnou hodnotou určenou *G(i,j,k)* a tím získáme celkový relativní úbytek od daného vrcholu.

Výslednou hodnotu v bodě [x,y,z] pak určíme součtem všech relativních úbytků od okolních vrcholů (vůči bodu [x,y,z]).

Více o Perlinově šumu v [4].

2.5.2 Skládání šumových funkcí

Pro zlepšení výsledku generování šumu se používá tzv. skládání šumových funkcí. Skládání šumových funkcí se nejčastěji používá v počítačové grafice na co nejrealističtější napodobení reálné textury nebo na bump mapping pro napodobení zvrásněného povrchu. Často se používá pro napodobení textur například mramoru, dřeva, ohně, vody, struktury zeminy apod.

Pro vysvětlení této tematiky je potřeba definovat pojmy *amplituda* a *frekvence* šumové funkce. *Amplituda* definována jako rozdíl minimální a maximální hodnoty, kterou může funkce nabývat a *frekvence* je převrácená hodnota vzdálenosti mezi jednotlivými body, které jsou použity pro spojitou interpolaci.

Z matematického hlediska se nejedná o nic jiného, než o součet funkcí šumu, kde každá funkce se liší amplitudou a frekvencí. Tento součet šumových funkcí se často nesprávně označuje jako Perlinova funkce, ačkoli jde ve skutečnosti pouze o skládání libovolných šumových funkcí (ať Perlinových nebo jiných). Sčítancům (jednotlivým šumovým funkcím) se říká oktávy.

[4] V roce 2002 Ken Perlin předvedl na konferenci Siggraph vylepšení tohoto algoritmu na rovnici pátého řádu.

Matematické vyjádření vypadá takto:

$$sNoise(x, y, z, p, n) = \sum_{i=0}^{n-1} a_i \cdot noise((x, y, z) \cdot f_i),$$

kde n udává počet oktáv (sumarizovaných šumových funkcí) a a_i udává amplitudu, která je pro každou i-tou oktávu určena jako

$$a_i = p^i.$$

Proměnná p udává rychlost klesání vlivu každé oktávy na celkový výsledek a nazývá se persistence p, kde $p \in (0, 1)$. Frekvence f_i i-té oktávy se určí jako

$$f_i = 2^i.$$

Pokud je persistence rovna jedné, tak samozřejmě nedochází k žádnému útlmu výsledného šumu jednotlivými oktávami. Více o skládání šumových funkcí v [4].

Na obrázcích 2.1 a 2.2 je vidět skládání šumové funkce.

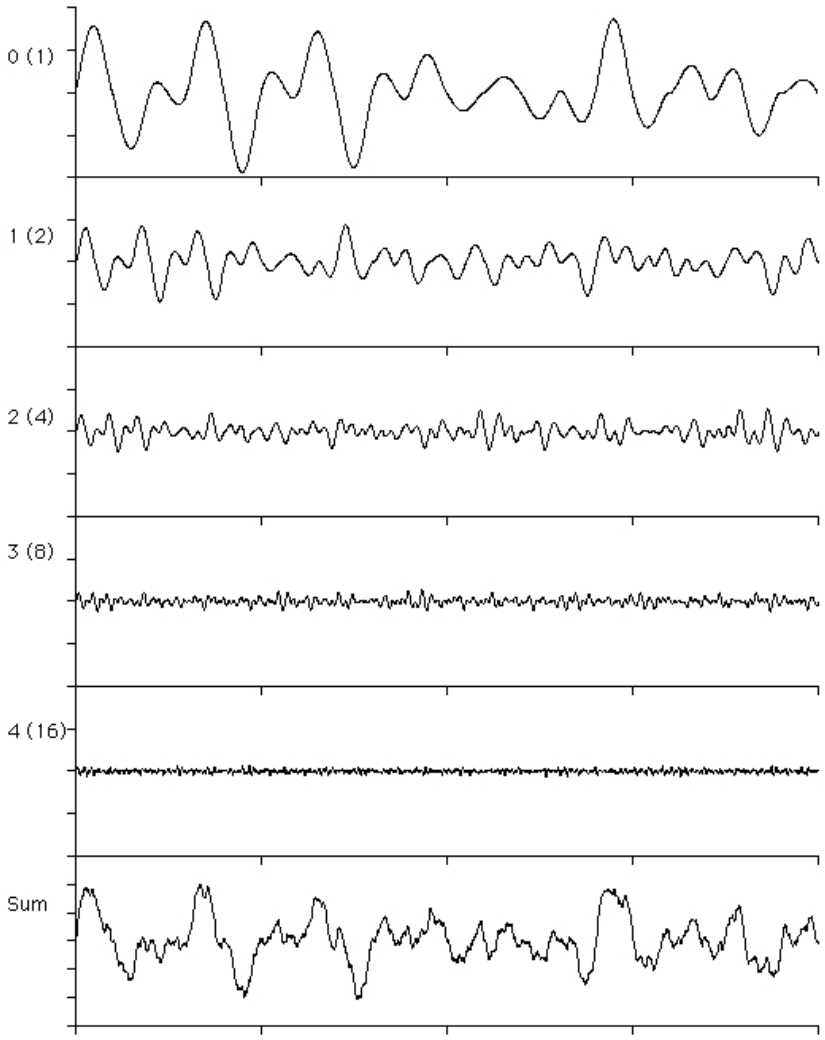
2.5.3 Turbulence

Turbulence je případem součtu šumových funkcí. Ken Perlin použil součet absolutních hodnot šumových funkcí a v roce 1984 jej pojmenoval Turbulence. Díky absolutní hodnotě se záporné hodnoty při výpočtu překlápí do kladných hodnot. Všechny výsledky se budou pohybovat v kladných hodnotách a ve funkci přibudou body, ve kterých nebudou existovat parciální derivace (zjednodušeně řečeno ve funkci dojde k „ostrým zlomům“).

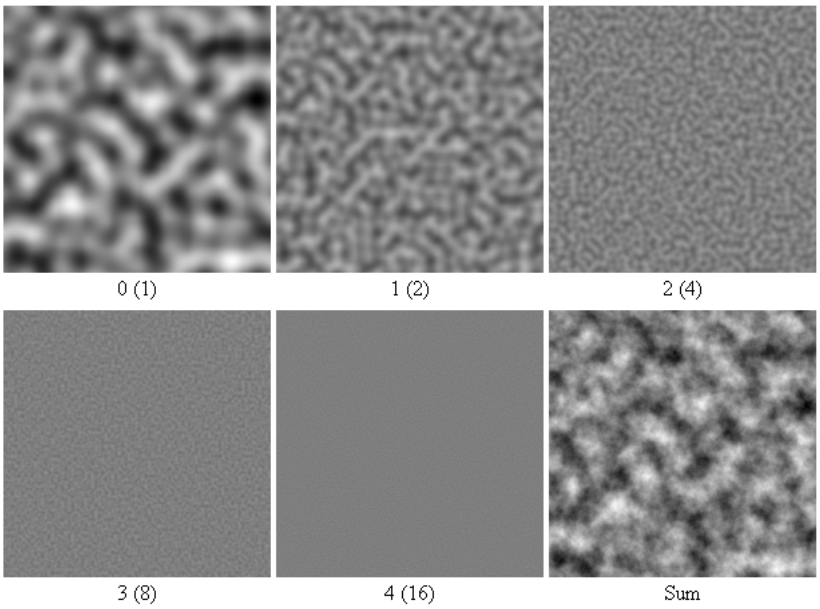
Důsledkem toho se při vykreslení turbulence jeví tento jev jako tmavé nepravidelné pruhy. Absolutní hodnota vytváří efekt, který se projevuje na narušení hladkosti šumu. Turbulence je často používána na napodobení textury dřeva, mramoru, dále také atmosféry a explozí.

Matematické vyjádření turbulence je:

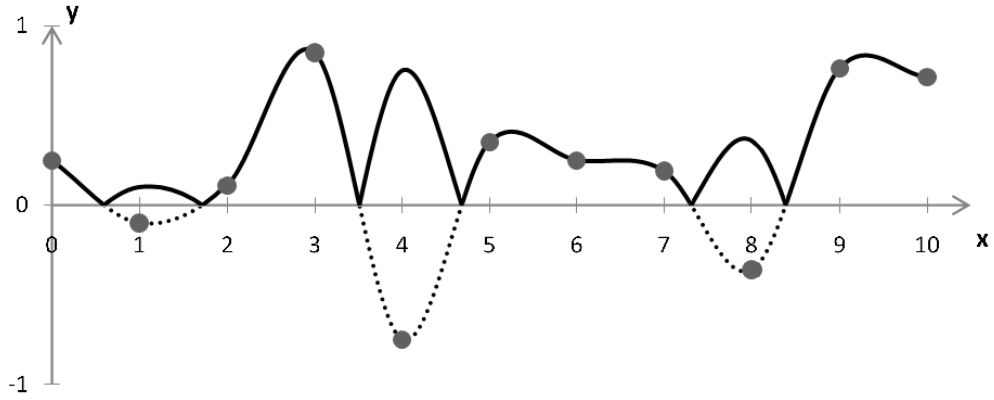
$$turbulence(x, y, z, p, n) = \sum_{i=0}^{n-1} a_i \cdot abs(noise((x, y, z) \cdot f_i))$$



Obrázek 2.1: Ukázka skládání šumových funkcí (v tomto případě pro 5 oktáv). Číslo před závorkou udává oktávu, číslo v závorce udává frekvenci.



Obrázek 2.2: Ukázka skládání šumových funkcí ve 2D (v tomto případě pro 5 oktáv je vidět, že poslední oktáva nemá na výsledek velký vliv). Číslo před závorkou udává oktávu, číslo v závorce udává frekvenci.



Obrázek 2.3: Ukázka proložení generovaných náhodných hodnot interpolační křivkou a použití absolutní hodnoty u turbulence (jedna oktáva).

Rychlost klesání vlivu každé oktávy je stejně definována jako u skládání šumových funkcí, tedy je to persistence p, kde $p \in (0, 1)$ a amplituda a_i i-té oktávy je v tomto vyjádření udána jako

$$a_i = p^i.$$

Frekvence f_i i-té oktávy se určí jako

$$f_i = 2^i.$$

Porovnání mezi turbulentním šumem a Perlinovým šumem je vidět na obrázku 2.4.

U turbulentního šumu se dají použít různé druhy vypočtů. Pro naši práci si turbulentní šum upravíme takto:

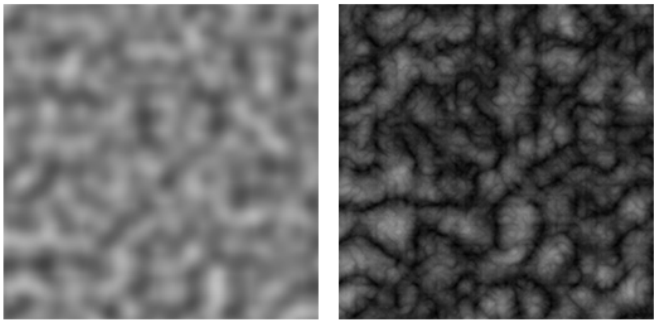
$$turbulence(x, y, z, p, n, f, a) = \sum_{i=0}^{n-1} a_i \cdot abs(noise((x, y, z) \cdot f_i)),$$

kde pro pro nultou oktávu je $f_0 = f$ a $a_0 = a$ a pro i-tou oktávu platí

$$f_i = f_{i-1} \cdot p^{-1}$$

$$a_i = a_{i-1} \cdot p$$

Více o tubrulenci v [5, 6].



Obrázek 2.4: Ukázka rozdílu mezi Perlinovo (*vlevo*) a turbulentním šumem (*vpravo*).

2.6 Programy pro rendering terénu

Při širokém rozšíření počítačové grafiky bylo vyvinuto velké množství komerčních i nekomerčních programů, které se zabývají fotorealistickým renderingem scén obsahujících terény. Některé z těchto programů se díky své úspěšnosti rozšířily i do oblasti filmů a počítačových her.

Mezi nejznámější programy tohoto druhu patří například Terragen, který se dočkal již druhé verze. Jeho rozšířenost je dána jak dostupností pro nekomerční účely, tak i snadnou ovladatelností, velkou kvalitou a realističností výstupních obrazů. V této části budou popsány některé z těchto programů.

Tyto programy lze rozdělit do základních kategorií:

- programy, které pro vytváření scény poskytují uživateli pouze řídící panely (bez možnosti ruční úpravy objektů přímo ve 3D scéně). Mezi tyto programy patří např. *Terragen Classic*, *VistaPro* a *World Machine*
- programy, které uživateli poskytují možnost upravovat objekty přímo ve 3D scéně, jako je tomu u 3D modelovacích programů. K těmto programům patří například *Terragen 2*, *E-ON Vue*, *Bryce* a *WorldBuilder*

Mezi nejznámější patří:

- Terragen 0.9.43 (také známý jako Terragen Classic)

Program Terragen Classic je založen na uprovování terénu pomocí řídících panelů (respektive nedovoluje zasahovat uživateli „přímo“ do 3D prostoru scény, kromě nastavení místa kamery).

- jeden z nejznámějších programů pro tvorbu fotorealistických terénů se širokým spektrem možností nastavení scenérie
- nedovoluje do scény umist'ovat vegetaci ani další objekty
- disponuje možností renderingu atmosféry
- k dispozici na Microsoft Windows (Windows 95 a novější), Mac OS 9 a Mac OS X
- pro nekomerční účely je zdarma
- tento program byl zvolen jako předloha této práce a vytvořeného GUI programu

- Terragen 2.2

  Tento program tvoří scénu v tzv. „Node Network“, což je schématické znázornění scény v podobě grafu. Každý objekt, který se ve scéně nachází (krajiny, vodní povrchy, světelné zdroje, kamery, externí objekty apod.), je reprezentován uzlem (tzv. *Node*) v grafu, vazby mezi těmito objekty jsou znázorněny šipkami. Jde o obdobu grafového znázornění scény, které je možno i vidět u dalších 3D modelovacích aplikací (například 3Ds Max). Díky tomuto grafu je zřejmé, z jakých objektů se tvořená scéna skládá a jakým způsobem ji bude renderer zpracovávat.

  Možnosti tohoto programu jsou podobné jako u *Terragen Classic*, avšak jsou rozšířeny o přidání objektů a vylepšené grafické rozhraní a renderovací jádro.

  - nástupce Terragen Classic, který oproti svému předchůdci využívá jiného grafického rozhraní a používá vylepšené renderovací jádro
  - tento generátor navíc také obsahuje možnost importu externích modelů, přidání objektů a vegetace do scény
  - disponuje možností renderingu atmosféry
  - dostupný na platformách Microsoft Windows (Windows 2000 a novější) a Mac OS X (10.4 a novější)
  - pro nekomerční využití zdarma s omezeními (velikost výsledného obrazu omezena do rozlišení 800 x 600, nepovolena nejvyšší kvalita detailů renderingu a kvality anti-aliasingu)

  - pro komerční účely placený (bez omezení zmíněných u nekomerční verze)

- E-ON Vue 9

  Ovládání tohoto programu je založeno na čtyřech pohledových oknech (kde jedno okno slouží k pohledu z libovolného úhlu a další tři okna pohlíží na terén ze směru os x, y a z). Do scény je možno umístit na uživatelem požadované místo libovolné množství terénů a objektů. Nastavování materiálu terénů, je možno vždy lehce kontrolovat pomocí „náhledové kuličky“ (ta svým porvchem simuluje povrch výsledného terénu).

  Program se dále také zabývá možností přidání tzv. ekosystému do terénu (různé druhy vegetace atd) pro zvýšení realističnosti.

  Ovládáním je tento program podobný např. programu *WorldBuilder*. Oproti programům *Terragen 2* a *Bryce* je možnost tedy vytvářer scénu z více pohledů.

  - E-ON Vue je high-endový generátor krajin
  - dovoluje přidání statických i dynamických objektů do scény (obsahuje například přes 170 typů vložitelné vegetace)
  - vytváří plně animované scény s obsáhlým nastavením atmosféry (obsahuje 160 typů předdefinovaných atmosfér a 140 typů typů oblačnosti)
  - dostupný na platformách Microsoft Windows (Windows XP, Windows Vista a Windows 7) a Mac OS X (10.5 a novější)
  - placený komerční program dostupný v několika verzích (např. Infinite a xStream)

- Bryce 7.1

  Program pro generování krajin založený čistě na jednoduchém umisťování a úpravě terénu v 3D grafickém prostředí. Pro tvorbu povrchu dává k dispozici ruční úpravu výškové mapy a širokou škálu možností pro nastavení vlastností materiálu terénu.

  Oproti *E-ON Vue* je většina ovládacích prvků tohoto programu zaměřena na tvorbu objektů ve scéně. A dále také oproti *E-ON Vue* disponuje pro pohled na vytvářenou scénu nahlížením pouze z jednoho pohledu, což může být umisťování objektů do scény nepraktické.

  - původně byl vytvořen hlavně na tvorbu fraktálních povrchů hor a pobřeží
  - podporuje tvorbu animací a 3D modelování
  - dovoluje umisťování objektů a vegetace do scény
  - dostupný na platformách Microsoft Windows (Windows NT4(SP6), Windows 2000(SP2), Windows XP, Windows Vista a Windows 7) a na Mac OS X (10.4 a novější)
  - placený komerční program

- VistaPro Renderer 4.2

  Program VistaPro oproti většině generátorů krajin nevyužívá 3D rozhraní na tvorbu scény. Oproti (například Bryce 7.1) tedy nedovoluje uživateli zasahovat „přímo“ do 3D prostoru. Uživatel si může nastavit požadovanou scénu pouze změnou hodnot v jednotlivých oknech nastavení.

  Program *Terragen*, který je také zaměřen na tvorbu scény přes řídící panely oproti tomuto programu nedisponuje možností vložit do scény například stromy a další objekty, avšak dává k dispozi širší nastavení atmosféry ve scéně.

  - generátor krajin a povrchů
  - možňuje umisťovat do scény další objekty (jako jsou různé druhy stromů a modely domů)
  - dovoluje přidání velmi jednoduchého typu atmosféry do scény
  - původně určen pro Amiga a PC
  - nejnovější verze dostupná pouze na platformě Microsoft Windows (Windows 98, Windows ME, Windows 2000 a Windows XP)
  - placený komerční program

- WorldBuilder Pro 4

  Tento program připomíná svým uživatelským prostředím 3D modelovací programy (např. *Blender*). Terén a další objekty se v tomto programu umisťují pomocí náhledových oken, která (přednastaveně) ukazují scénu ze tří pohledů určených osami (x,y,z) a oknem pro pohled na scénu z volitelného úhlu. Terén se upravuje pomocí nástrojů pro ovlivnění tvaru (tedy v tomto případě jde hlavně o křivky, které „průchodem“ skrz plochu terénu mění jeho geometrii).

Všechny nastavení se pak provádějí v postranní nabídce, kde je možno použít nástroje pro úpravu povrchu terénu, umístění světla a mnoho dalšího.

Program *WorldBuilder* je podobný programu *E-ON Vue*, avšak oproti němu je tvorba plochy reprezentující terén v tomto programu složitější.

- generátor krajin s návazností na 3D modelovací programy Max, Maya, LightWave a SoftImage
- podporuje tvorbu animací a možnost vložení efektu deště a sněhu
- dostupný na platformě Microsoft Windows (Windows NT4(Service Pack 5), Windows 2000(Service Pack 2) a Windows XP(Service Pack 1))
- placený komerční program

- World Machine 2.2

  Tento program zprostředkovává tvorbu terénu pomocí napojování jednotlivých „zařízení“ (angl. *device* - obdoba „Node network“ z Terragenu 2). Každé obsahuje své detailní nastavení podle toho, k čemu je určeno. Jejich postupným poskládáním (respektive propojením do hierarchické struktury) se tvoří výsledná scéna.

  Program *World Machine* je podobný programu *Terragen 2*, ale oproti němu je tento program určen převážně na tvorbu terénu (ve formě objetku nebo výškové mapy) pro použití v dalších programech (jako jsou *E-ON Vue* a další).

  - generátor pro tvorbu uměleckých scén, krajin pro počítačové hry a další
  - nabízí modelování erozí povrchu, velké možnosti v nastavení texturování povrchů
  - dostupný na platformě Microsoft Windows
  - placený komerční program

Z těchto programů byl první jmenovaný program Terragen vybrán jako předloha pro vytvoření interaktivního GUI programu, který bude generovat bloky, z nichž se sestaví surface shader pro obarvování uživatelem definovaného terénu.

3 Realizační část

Na fotorealistický rendering terénu (neboli vyobrazení terénu tak, aby pozorovateli připadalo jako co nejvíce přirozené) existuje mnoho programů (viz kapitola 2.6).

K tomu, aby se dal vygenerovat fotorealistický obrázek umělého terénu, je potřeba definovat výšková data, která určují geometrii terénu, jakým způsobem bude vytvořena textura terénu (tedy jak bude obarven), osvětlení scény a atmosférické jevy, které více přibližují výsledek realitě.

Mým cílem je realizovat interaktivní aplikaci, která vytváří surface shader, jehož úkolem je obarvení terénu ve výsledné scéně. Tento shader bude složen z několika kombinovatelných bloků, které mají za úkol ve výsledku vypočítat barvu každého bodu povrchu tak, aby výsledek byl kvalitativně porovnatelný s ostatními programy (které se zabývají renderingem přírodních povrchů terénu).

Pro názornost k tomu použiji jeden z nejznámějších programů zabývajícím se problematikou fotorealistického renderování obrazu terénů. Tímto programem je v kapitole 2.6 zmíněný Terragen Classic.

3.1 Fotorealistické terény v Terragenu

Terragen Classic je jeden z nejznámějších programů pro fotorealistický rendering terénu. Tento program dovoluje uživateli nastavovat pro scénu jak obarvení terénu tak i atmosféru a mnoho dalšího.

Mezi jeho nevýhody se však řadí to, že nedovoluje umist'ovat do vytvářené scény další objekty a vegetaci.

Nás bude především zajímat, jakým tento program zpracovává obarvení terénu pro generované povrchy.

Nejdříve si popíšeme, jak vypadá tvorba terénu, která je vidět na obrázku 3.1. Na tomto obrázku jsou popsány hlavní části grafického uživatelského prostředí (GUI).

1. Panel pro generování terénu. Pomocí tohoto panelu se dají importovat vstupní výšková data terénu do Terragenu a stejně tak se dají exportovat výšková data vygenerovaného terénu ve formátu RAW[1] obrázku.

   Dále je zde možno vidět náhled vygenerovaného terénu a například nastavit škálování výšky a další modifikace geometrie aktuálně zpracovávaného terénu.

2. Tento panel slouží k manuálnímu nastavení pozice a směru pohledu kamery v terénu.

3. Nastavení pozice kamery pomocí číselných hodnot.

4. Seznam druhů térénů pro vytvoření procedurální textury terénu.

5. Náhledové okno terénu s možností změny kvality zobrazení výsledku. Poskytuje možnost nastavení velikosti výstupního obrázku, vypnutí vykreslování oblohy a tvorby animace.

6. Panel k otevření oken s dalším nastavením. Od shora to jsou:

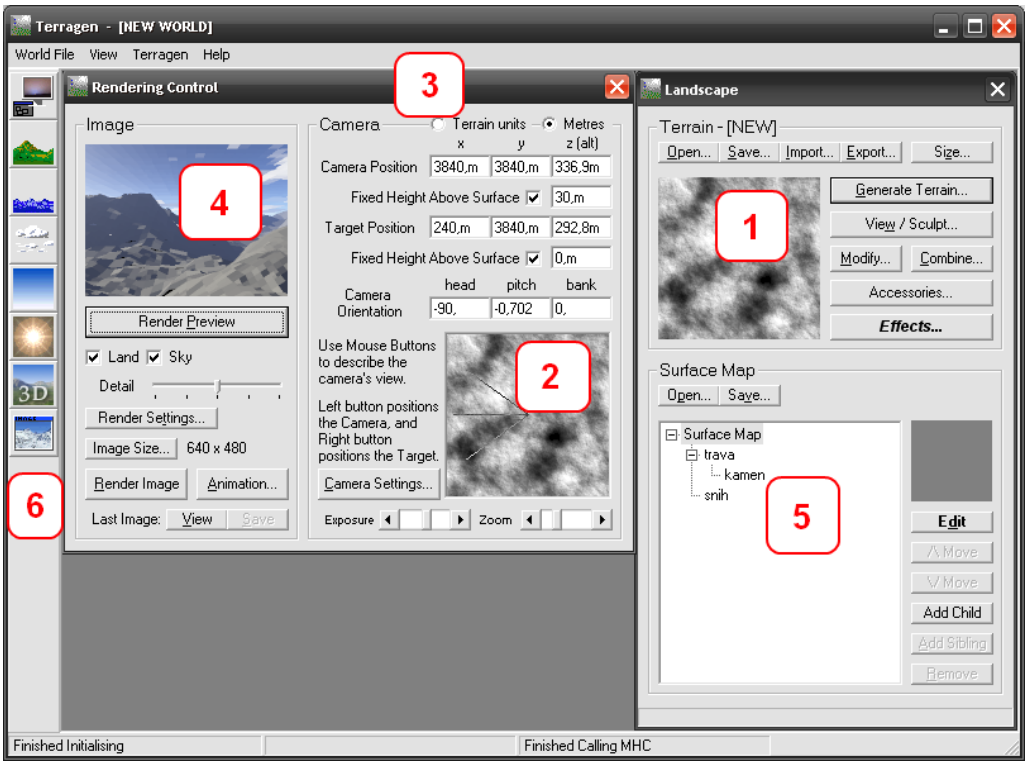
   - Nastavení renderingu
   - Povrch terénu
   - Nastevní vodní plochy
   - Nastevní oblohy (mraky)
   - Nastavení osvětlení
   - Interaktivní 3D náhled
   - Poslední vytvořený obrázek

3.1.1 Výšková data

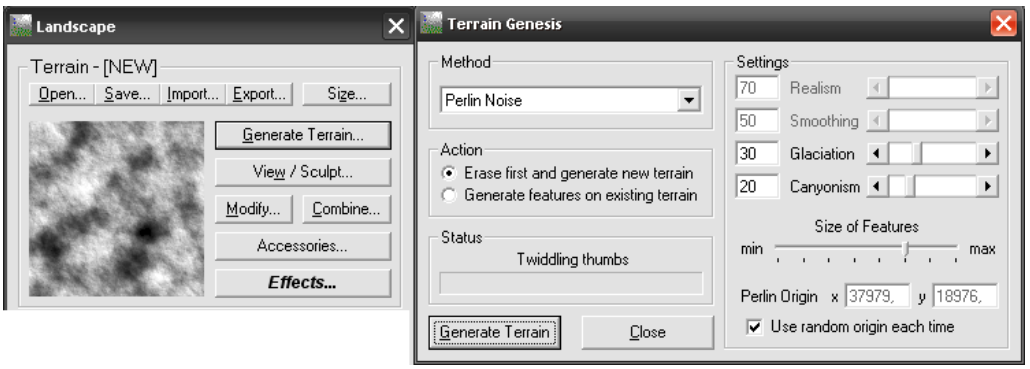
První, co v je v Terragenu nutno udělat, je získat výšková data, ze kterých bude vytvořen terén. Pro tuto úlohu obsahuje Terragen vlastní generátor výškových dat.

Generátor těchto výškových dat je vidět na obrázku 3.2 a spouští se z hlavního okna přes tlačítko *Generate Terrian* v panelu 1 na obrázku 3.1.

[1] formát RAW = jde o nekomprimované pole výšek s 8 nebo 16 bitovou přesností. Tento typ formátu může obsahovat doplňující informace v hlavičce soubor (pro naši práci však žádná data v hlavičce souboru neobsahuje).



Obrázek 3.1: Ukázka hlavního okna programu Terragen Classic.



Obrázek 3.2: Ukázka nabídky pro vygenerování nové geometrie povrchu terénu v programu Terragen Classic

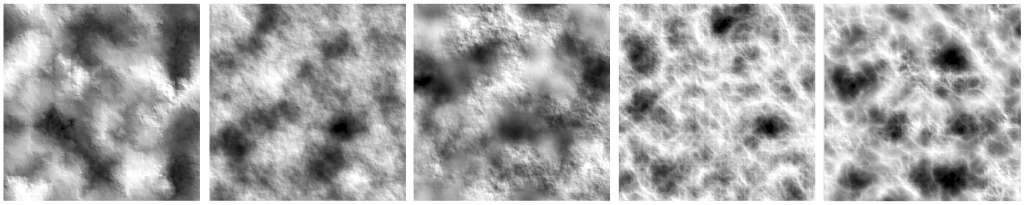
Na tomto obrázku je vidět, že Terragen disponuje možností nastavení metody pro generování povrchu a možností upravení paramametrů, z kterých je tento povrch generován. Tyto metody jsou:

- *Subdivide & Displace II* - „originální“ generování výškových dat pro Terragen
- *Perlin noise* - Perlinův šum
- *Ridged Perlin* - rozšíření Perlinova šumu s tím, že do térénu přidává více hřebenů[2]
- *Multi-versions of Perlin* - vytváří více hrbolatý a nepravidelný povrch než samotná Perlinova šumová funkce (do těchto metod patří *Multi Perlin* a *Ridged Multi Perlin*)

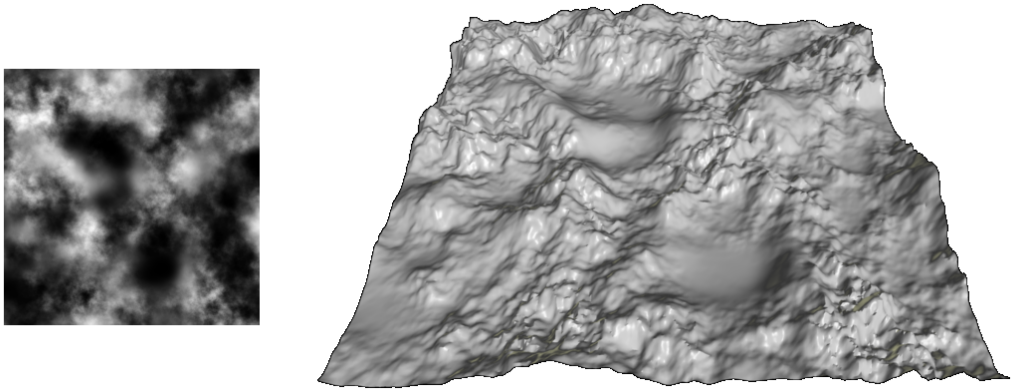
Pro názornou ukázku těchto metod je vidět na obrázku 3.3 příklad vygenerovaných výškových map.

Další možností je import výškových dat do programu Terragen z externích zdrojů (tedy ze souborů). Nejčastěji se používá formátu RAW, který obsahuje čistá data. Pro účel reprezentace výškových dat se využívá 8 bitový RAW, kde každých 8 bitů udává jeden pixel (šedotónového) obrázku. Pixely tohoto obrázku udávají tvar terénu (respektive reprezentují výšková data terénu). Jak tato reprezentace vypadá je vidět na obrázku 3.4.

[2]Hřeben je označení skalního tělesa, které se táhne mezi dvěma vrcholy povrchu, či jako dlouhé poměrně rovné místo, na jehož dvou stranách dochází k prudkému poklesu výšky.



Obrázek 3.3: Druhy metod generování terénů (z leva: *Subdivide & Displace II*, *Perlin noise*, *Multi Perlin*, *Ridged Perlin* a *Ridged Multi Perlin*).



Obrázek 3.4: Výšková data pomocí RAW obrázku (*vlevo*), 3D výšková mapa vytvořená z těchto dat (*vpravo*).

Tato data se kromě importu dají také exportovat ve stejném formátu pro použití v dalších programech.

3.1.2 Texturování

Na vygenerovaný terén je potřeba vytvořit obarvení (tedy přidat terénu texturu). Obarvování povrchu terénu se v Terragenu provádí pomocí budování procedurální textury ve formě výpočetního stromu. V programu je to reprezentováno stromovou strukturou, kde každý uzel této struktury představuje jeden typ subtextury terénu. Ve stromové struktuře se nachází jedna základní textura a ta se překrývá těmito subtexturami.

U základní textury se nastavuje jedna barva, která bude použita pro obarvení tohoto terénu v místech, kde nebude žádná subtextura. Na obarvovanání však neni použita čistě uživatelem definovaná barva, ale má na ni velký vliv i osvětlení (podle toho se barva definovaná uživatelem při výpočtu upraví) a bump mapping[3].

Pro subtextury, které budou překrývat základní texturu terénu, je nastavení rozšířeno. Je možno nastavit, v jakých podmínkách se bude subtextura na povrchu terénu vyskytovat. Tyto podmínky jsou:

- minimální a maximální výška výskytu subtextury na povrchu terénu
- minimální a maximální úhel subtextury, ve kterém je výskyt povolen

Při vytváření procedurální textury terénu si Terragen zjišťuje pro každý bod terénu, zda jsou tyto podmínky splněny a podle toho se rozhodne, zda bude v daném místě pro obarvení textury použita tato subtextura terénu.

Pro nejednoznačné obarvení (jako je tomu v přírodě) jsou zde možnosti ovlivnění přesnosti pokrytí při splnění těchto podmínek (tedy, že Terragen nemusí přesně dodržet při umisťování obarvení textury terénu dané podmínky se stoprocentní přesností).

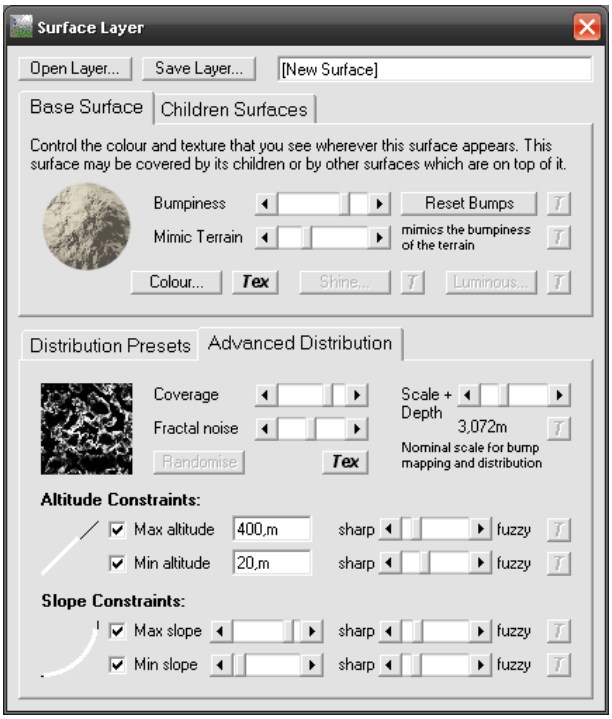
Jak tato nastavení subtextury terénu vypadají, je vidět v dolní části obrázku 3.5.

Procedurální textura je však tvořena z více než jedné subtextury a proto je ještě definována priorita obarvení terénů. Tato priorita je udaná pozicí ve stromové struktuře subtextur (viz obrázek 3.1 - položka 5). Tato stromová struktura má charakter uspořádaného stromu[4]. V uspořádaném stromu pro náš popis jsou subtextury vrcholy stromu.

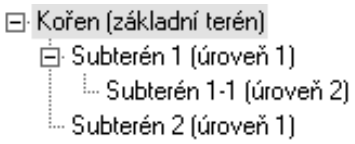
Priorita subtextur je udána takto:

[3] *Bump mapping* je technika texturování, která vytváří iluzi nerovnosti povrchu bez změny jeho geometrie. Iluze nerovnosti povrchu se dosahuje úpravou normály v každém pixelu plochy. Modifikovaná normála pak ovlivní výpočet osvětlení plochy. Citace z [5, 13]

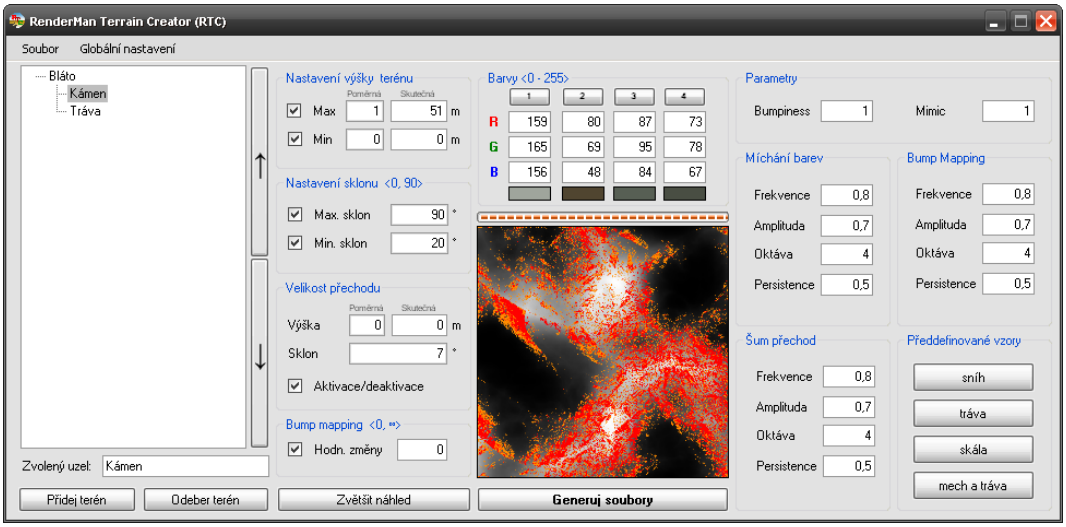
[4] *Uspořádaný strom* je takový, ve kterém jsou větve každého vrcholu uspořádané. Vrchol y, který je bezprostředně pod vrcholem x se nazývá (přímý) následník vrcholu x. Když je vrchol x na i-té úrovni, tak vrchol y bude na $(i+1)$ úrovni. Vrchol x se pak nazývá (přímý) předchůdce vrcholu y. Kořen stromu je na první úrovni. Více v [8].



Obrázek 3.5: Okno s nastavením subtextury terénu v Terragenu.



Obrázek 3.6: Uspořádaný strom subtextur pro tvorbu prodedurální textury terénu v Terragenu.



Obrázek 3.7: Ukázka vytvořeného interaktivního GUI programu

- Pokud má vrchol více následovníků, tak větší prioritu má ten, který je níže. Na obrázku 3.6 má větší prioritu vrchol *Subtextura 2* než *Subtextura 1*.
- Pro následníka a předchůdce platí, že větší prioritu překreslování má následovník.
- Pro následníka platí, že podmínky jeho zobrazení jsou navíc omezeny podmínkami jeho předchůdce.

Podle těchto pravidel se při renderování (vykreslování) obrázku tvoří procedurální textura terénu.

3.2 GUI program

Po vzoru Terragenu (viz kapitola 3.1) byl v této práci vytvořen GUI program nazvaný *RenderMan Terrain Creator* (dále už jen *RTC*), jehož úkolem je tvorba RIB souboru pro popis scény s terénem a surface shaderu pro obarvení terénu.

Tento program byl napsán v programovacím jazyce C# (s podporou .NET

Framework 4). Ukázka grafickéhu uživatelského prostředí tohoto programu je vidět na obrázku 3.7.

Surface shader generovaný tímto programem je vytvořen z menších bloků (respektive modulů).

Program je vytvořen tak, aby uživateli dával k dispozici co největší škálu nastavitelných parametrů s náhledem toho, na jakou část terénu ve scéně budou mít uživatelem nastavené parametry vliv. Pro zkušenějšího uživatele je možná i manuální úprava nastavení vygenerovaných modulů. Tato úprava však již není tak jednoduchá jako tvorba modulů přímo v GUI prostředí a je k ní potřeba alespoň minimální znalost toho, co který modul dělá.

Výstupními soubory této aplikace jsou vygenerovány moduly, které se prostřednictvím nástroje *AQSIS* přeloží do výsledného surface shaderu.

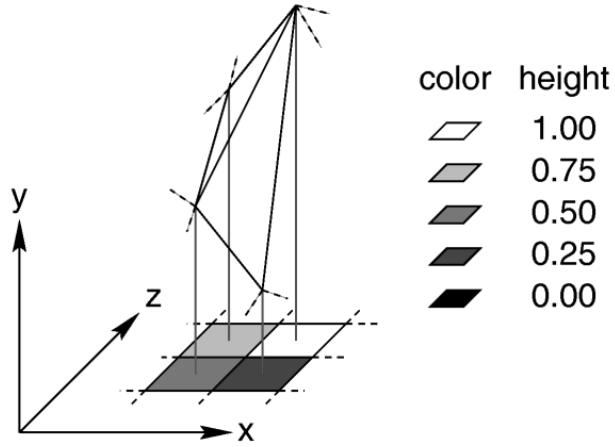
Dalším vygenerovaným souborem je i RIB soubor (pouze v případě, že byla uživatelem načtena výšková mapa). V něm je definována scéna s požadovaným nastavením a na plochu reprezentující terén je aplikován již zmiňovaný surface shader.

3.2.1 Výšková data

Výšková data pro vstup RTC program jsou ve formátu RAW, ve kterém nedochází ke ztrátě a zkreslení kvality uchovávaných dat. Jelikož datový formát RAW v sobě nenese žádnou informaci o šířce a délce obrazu (respektive mapy, která má být umístěna do scény), je předpokládáno, že budou mapy čtvercového charakteru (tedy šířka a délka obrazu jsou shodné). Stejně je tomu tak i v programu Terragen Classic, avšak ten dovoluje načítat pouze výškové mapy předem daných velikostí.

RAW soubor je tvořen Byty, kde každý Byte udává výšku v daném bodě terénu. Z toho vyplývá, že každý bod terénu nabývá hodnot 0 až $2^8 - 1$ (neboli 0 až 255), kde 0 je jeden výškový extrém a hodnota 255 je druhý.

Při načítání nové výškové mapy RAW se zpřístupní základní nastavení scény. V tomto nastavení si uživatel může definovat počáteční polohu kamery, výšku nad terénem v místě kamery a směr, kterým bude kamera natočena podobně jako tomu je v programu Terragen Classic.



Obrázek 3.8: Výšková data jako šedotónový obraz.
Obrázek převzat ze stránek *http://www.povray.org/documentation*.

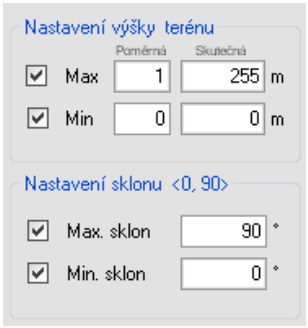
Hlavním rozdílem Terragenu Classic oproti vytvořenému RTC programu je zpracování výškové mapy ve formátu RAW. RTC program načítá výšková data tak, jak jsou v RAW uložena. Většinou jsou výšková data v RAW formátu uložena tak, že při jejich zobrazení je severní strana mapy nahoře a jižní dole. V programu Terragen jsou však výšková data načtená z RAW interpretována opačně (respektive je zaměněna severní a jizní strana). Proto je potřeba před importem výškových dat (výškové mapy) data pozměnit. Pokud chceme importovat do Terragenu výšková data a zachovat severní a jižní orientaci výškové mapy, tak je nutno RAW data horizontálně překlopit[5].

Tato výšková data lze získat například prostřednictvím programu *Quantum GIS*, který umí vizualizovat a exportovat data z topografických měření SRTM agentury NASA do různých formátu nebo exportem výškových dat z programu Terragen.

3.2.2 Podmínky povrchu

Jedním z důležitých nastavení parametrů terénu je určení toho, kde se bude část aktuálně zpracovávaného terénu v objektu (v ploše reprezentující terén)

[5]Co bylo v původním souboru výškových dat severní stranou povrchu se po načtení v Terragenu stane stranou jižní a naopak.



Obrázek 3.9: Nastavitelné podmínky zobrazení vybraného povrchu.

nacházet. Pro tuto subtexturu jsou důležité parametry výška a sklon. Ty udávají, při jakých podmínkách se může ve výsledném terénu daná subtextura vyskytovat (například sníh se nemůže držet ve velkých sklonech atd.).

Pro zjednodušení je možno zadávat terén dvěma způsoby. První je tzv. poměrem. Tento způsob udává výšku v povrchu pomocí poměru vůči celé výšce objektu. Tedy hodnota 0 udává nejnižší možnou výšku terénu a maximální výška 1.0 udává nejvyšší bod. Druhým způsobem je zadávání přes výšky v metrech.

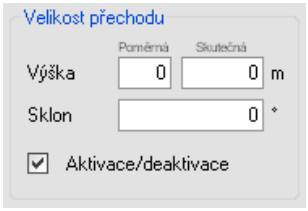
Sklon terénu je udáván ve stupních. Omezení sklonu je $\langle 0, 90 \rangle$ stupňů, kde 0 udává vodorovnou plochu a 90 kolmou plochu.

Panel s těmito nastaveními je vidět na obrázku 3.9.

Tyto parametry je v Terragenu také možno nastavit, avšak pro nastavení sklonu není možné pro daný terén definovat číselnou hodnotou, ale pouze změnou posuvníku (tedy není možné přesné číselné definování těchto hodnot a je nutno hodnoty nastavovat odhadem přes náhledové okno).

Výškové hodnoty je v Terragenu možno zadávat taky v metrech. Druhá možnost je však odlišná tím, že v Terragenu se zadávají hodnoty výšky v jednotkách terénu (*terrain units*) zatímco v RTC programu je možno nastavovat tyto hodnoty přes již zmíněné poměrové hodnoty, které pomáhají uživateli v představě, v jaké výšce bude daná hranice nastavena.

Do podmínek, ve kterých se má aktuálně upravovaný terén umístit je zařazen také šum na přechodu. Ten se skládá jak z aplikace turbulentního šumu na přechod mezi dvěma terény (aktuální a nadřazený) tak i z šířky



Obrázek 3.10: Nastavitelná výška a sklon přechodu mezi dvěma terény.

přechodu, která zabraňuje ostrému přechodu mezi terény (tento panel je vidět na obrázku 3.10).

Pro co nejlepší představu o místě, kde bude ve výsledku terén obarven, je použito barevného zvýraznění v náhledovém okně ihned po změně hodnot výšky a sklonu (viz kapitola 3.2.3).

3.2.3 Náhled vybraného terénu

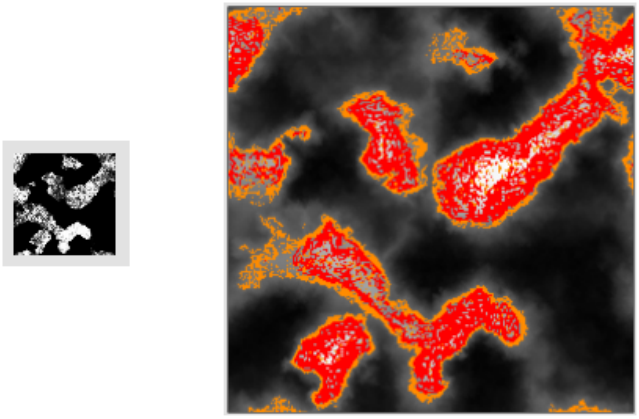
Aby měl uživatel možnost zjištění, zda hodnoty, které zadal budou ve výsledku vyobrazeny, je v programu implementováno náhledové okno. V tomto okně se automaticky zobrazují všechny změny provedené uživatelem na načteném mapovém podkladu.

Ta část terénu, kterou uživatel vybral pro aplikace změn obarvení, je v náhledovém okně zobrazena červenou barvou. Část, která má být navíc zahrnuta do přechodu (tedy míchání aktuálně upravovaného terénu s nadřazeným terénem), je označována oranžovou barvou.

Přechodová část je interaktivně měněna při zadávání podmínek do podmínek terénu.

V Terragenu je při úpravě terénu zobrazován náhled tak, že je na černém pozadí vyobrazena bíle pouze část, která bude výsledně obarvena. Oproti tomu v RTC programu je vidět obarvovaná část s kontextem okolního terénu a navíc je oproti Terragenu možnost tento náhled zvětšit.

Rozdíl mezi náhledovým oknem Terragenu a oknem RTC programu je názorně vidět na obrázku 3.11.



Obrázek 3.11: Náhledová okna Terragenu (*vlevo*) a RTC programu (*vpravo*).

3.2.4 Tvorba surface shaderu a texturování

V tomto programu je vytvářena procedurální textura, která je reprezentována jako výpočetní strom. Zásady, které jsou v Terragenu, jsou i zde podobné. Jedná se opět o uspořádaný strom, kde vrcholy stromu reprezentují jednotlivé subtextury. Výpočet a míchání barev vychází z pravidel, kterými se řídí Terragen (například tvorba výsledné textury, která vyhází z uspořádaného stromu, kde subtextury mají prioritu obarvování určenou svou pozicí).

Surface shader, generovaný RTC programem, je, jak již bylo zmíněno výše, složen z několik modulů. Každý z těchto modulů má jinou funkci.

Moduly se volají jako procedury s několika parametry. Po jejich zavolání a úspěšném výpočtu vrací zpět požadovaný výsledek, který je použit na další zpracování.

Tyto moduly jsou:

- *terrain_shader.sl*
  - jde o surface shader, jehož úkolem je vypočítat obarvení povrchu v závislosti na vstupních parametrech (viz kapitola 3.4.5)
  - v popisu scény (RIB souboru) je volán pro aplikaci na objekt (terén) příkazem *Surface "terrain_shader"*

  - jsou v něm volány dvě funkce: *colour()* pro míchání barev v každé subtextuře a funkce *resolve_function()*, která rozhoduje, jaká z dvojic barev subtextur bude na povrch umístěna (těchto dvojic je tolik, kolik je v terénu subtextur)

- *turbulence.h*
  - tento modul vypočítává podle paramatrů (uvedených v kapitole 3.4.1) hodnotu turbulentního šumu (viz kapitola 2.5.3)

- *colour.h*
  - jde o výpočet jedné barvy ze čtyř vstupních barev (4 barvy jedné subtextury) pomocí turbulentní funkce *turbulence()*
  - v tomto modulu je zároveň počítán bump mapping pro změnu obarvení daného terénu pomocí funkce *bump_mapping()*
  - je volán jako *colour()* - podrobný popis parametrů v kapitole 3.4.3

- *bump_mapping.h*
  - výpočet bump mappingu
  - je volán funkcí *bump_mapping()* a jeho výstupem je pozměněná normála povrchu terénu pomocí turbulentního šumu
  - podrobný popis funkce a parametrů je v kapitole 3.4.2

- *resolve_function.h*
  - funkce, která podle nastavených parametrů (dvou barev pro dva terény a podmínek jejich použití) určuje výstupní barvu
  - v této funkci se také počítají přechody mezi subtexturami terénu
  - podrobný popis funkce a parametrů je v kapitole 3.4.4

- *setting.h*
  - soubor se všemi potřebnými parametry, které určují jednotlivé subtextury terénu, jejich barvy atd.

3.3 RIB soubory

Pro definování scény, tedy objektů, materiálů, světel, kamery atd. jsou podle *RenderMan Iterface* dvě možnosti:

- pomocí procedurální metody - tedy programového rozhraní
- přes RIB (RenderMan Interface Bytestream)

Více o psaní RIB souborů v [1, 2, 3].

Pro ukázku si popíšeme jednoduchou tvorbu RIB souboru:

```
Format 1024 768 1
```

Tímto příkazem udáváme jako první a jde o formát výstupního obrázku. První dva parametry udávají šířku a výšku tohoto výstupního obrázku v pixelech. Třetí parametr udává poměr šířky pixelu k jeho výšce.

```
PixelSamples 2 2
```

Nastavení z kolika okolních bodů se bude počítat barva daného pixelu. Pro rozumně vypadající výsledky je doporučeno nastavit tento parametr přibližně na `3 3`.

```
Display "vystupniobrazek.tiff" "file" "rgb"
```

Nastavení typu výstupu, v tomto případě jde o výstup do souboru (v případě výstupu na obrazovku se místo `"file"` používá `"framebuffer"`).

```
Projection "perspective" "fov" [60]
```

Tento příkaz nastavuje kameru. Jak je zřejmé z jména příkazu, jde o nastavení typu projekce (v tomto případě perspektivní) a pohledového úhlu kamery.

```
WorldBegin
```

Návěští `WorldBegin` udává, odkud lze začít definovat scénu jako takovou (ukončením bude návěští `WorldEnd`).

```
LightSource "distantlight" 1 "from" [-10 10 10]
   "intensity" 8.2
```

Nastavení světelného zdroje. Parametr `"distantlight"` udává typ osvětlení, neboli název použitého light shaderu (další typy můžou být například `"ambientlight"` a `"pointlight"`). Číslo `1` udává o jaké jde světlo.

```
Translate 0 0 50
Rotate -30 1 0 0
```

Příkazy zadávající transformaci scéna → kamera. Transformace se dají zadat buď pomocí transformační matice a nebo pomocí těchto základních příkazů.

```
AttributeBegin
   Color [1 1 0]
   Surface "matte" "Kd" 0.85 "Ka" 0.35
   Translate 0 2 0
   Sphere 1 -1 1 360
AttributeEnd
```

Tyto příkazy slouží k vytvoření žluté kuličky s matným povrchem. Prvním uvedeným příkazem, tedy příkazem *AttributeBegin*, dojde k uložení platných atributů do zásobníku (tedy například barvy, která je v tomto místě nastavena). Následně jsou definovány objekty. Příkazem *Color* se definuje barva pro následující objekty (v tomto případě jde o žlutou barvu). Další příkaz *Surface* definuje povrch objektů. Za první parametr má název surface shaderu (v tomto případě *matte* neboli matný povrch), který se má na povrch aplikovat a parametry *Ka* a *Kd* určují vliv světel. Příkazem *Translate* dojde k posunu objektů. Dále je pak uveden příkaz pro zařazení objektu do scény. V tomto případě jde o kouli (příkaz *Sphere*).

```
WorldEnd
```

Na konci RIB souboru je nutné uzavřít návěštím `WorldEnd` scénu. Hlouběji o této tématice v [1, 2].

3.4 Moduly a jejich použití

V této kapitole jsou detailně popsány vytvořené moduly, jak tyto moduly fungují a jaké mají vstupní parametry.

3.4.1 Modul výpočtu Turbulence

Úkolem tohoto modulu je výpočet turbulence (viz kapitola 2.5.3) pro účely dalších modulů ze vstupních hodnot *amplitudy*, *frekvence*, *oktávy* a *persistence*.

Tento modul se používá například pro míchání dvou barev (výstupní hodnota tohoto modulu určuje, která hodnota se v daném místě použije).

Hlavní částí tohoto modulu je výpočet turbulence jako takové:

```
for(i = 0; i < octaves; i = i + 1)
{
   perlin = perlin + amplitude * abs(2.0f * (noise((PP_start
      + perlin_random) * frequency) - 0.5f));
   frequency *= 1/persistence;
   amplitude *= persistence;
}
```

V RenderManu standardně vrací funkce *noise()* hodnotu v intervalu $\langle 0, 1\rangle$. Jak jsme si uvedli v teoretické části, je pro turbulenci nutné, aby funkce generující náhodné hodnoty, tyto hodnoty generovala v intervalu $\langle -1, 1\rangle$. Proto tedy od vygenerované hodnoty funkce *noise()* odečteme hodnotu $-0.5f$ (tím interval funkčních hodnot posuneme do intervalu $\langle -0.5f, 0.5f\rangle$) a následně vynásobíme dvěmi (interval se změní na $\langle -1, 1\rangle$). Tím jsou splněny požadavky na to, aby náhodné hodnoty byly v tomto intervalu. Funkce *abs()* vrací absolutní hodnotu (je typická pro turbulenci).

3.4.2 Modul výpočtu Bump mappingu

Tento modul se jmenuje `bump_mapping.h`. Pomocí parametrů *amplitudy*, *frekvence*, *oktávy* a *persistence*, které byly zadány v RTC programu vypočítává hodnotu bump mapping pro daný bod terénu.

```
float turbulence_bump = turbulence(frequency_bump,
   amplitude_bump, octave_bump, persistence_bump,
   set_bump_mapping, PP_start);
```

Ze vstupních parametrů se vygeneruje hodnota pro změnu normály.

```
point PP = PP_start + turbulence_bump * normalize(Ng);
```

Výpočet nové hodnoty bodu z původního bodu a pozměněné hodnoty normály.

```
normal Nf = - calculatenormal(PP);
```

Výpočet výsledné normály `Nf` pro obarvení z pozměněného bodu. Tato normála je návratovou hodnotou tohoto modulu.

3.4.3 Modul míchání barev

V následující části bude popsána struktura modulu `colour.h`, jehož úkolem je ze čtveřice vstupních barev určit jednu barvu, vypočítat bump mapping v tomto místě a podle toho určit výstupní barvu. Důvod tohoto míchání je ten, že každý typ terénu definovaný uživatelem pro scénu je udán čtyřmi základními barvami, mezi kterými se míchá výstupní barva aktuálně počítaného bodu. Výběr této barvy se provádí pomocí výpočtu turbulence.

Vstupní parametry jsou (v tomto pořadí):

- `color c1, color c2, color c3, color c4`
  - čtyři barvy aktuálního povrchu určené pro míchání
- `float fr, float am, float oc, float pe`
  - frekvence, amplituda, oktáva a persistence pro výpočet turbulence při míchání barev
- `float fr_bump, float am_bump, float oc_bump, float pe_bump`
  - frekvence, amplituda, oktáva a persistence pro výpočet turbulence bump mappingu (tedy úpravy barvy povrchu terénu podle pozměněné normály)
- `point PP_start`
  - pozice aktuálně počítaného bodu
- `float cond_bump_mapping`

  - podmínka, zda má být aplikován bump mapping

- `float size_bump_mapping`
  - hodnota pro pozměnění bump mappingu
- `float bumpiness`
  - hodnota pro bumppiness - definování členitosti a nerovnosti povrchu
- `float mimic`
  - mimika terénu umožňuje upravit frekvenci pro výpočet turbulence bump mappingu
- `float Ka, float Kd`
  - difuzní a ambientní složky osvětlení

První částí tohoto modulu je vypočítat náhodnou hodnotu (z turbulence), která pro každou dvojici barev určí jaká barva se použije:

```
float turbulence_float = turbulence(fr, am, oc, pe, 0.0f,
   PP_start);
```

Tato funkce se bude volat pro míchání barev $1-2$, $3-4$ a $(12)-(34)$ (kde hodnoty (12) a (34) jsou dvojice namíchaných barev). Dostaneme tři hodnoty `turbulence_float`, `turbulence_float2` a `turbulence_float3`.

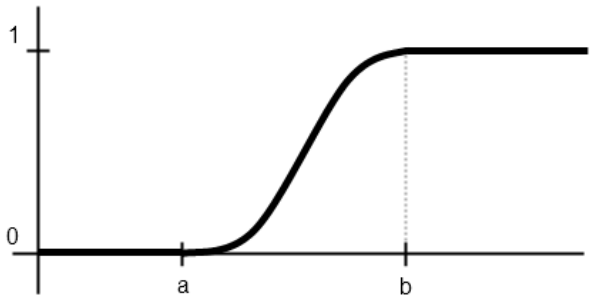
```
inTop = smoothstep(0,1,turbulence_float);
```

Z vypočtené hodnoty si pomocí funkce *smoothstep(a, b, x)* (která je ukázána na obrázku 3.12) si zjistíme hodnotu `inTop`.

```
color Ct1 = mix(c1, c2, inTop);
```

Funkcí *mix(color1, color2, inTop)* se podle hodnoty `inTop` namíchá z barev, které jsou uvedeny jako první dva parametry, výsledná barva.

Po zjištění, jaká barva se pro dané místo použije, přijde řada na výpočet bump mappingu. Tento výpočet se však provede pouze v případě, že byl buď v RTC programu a nebo v souboru s nastavením `setting.h` tento výpočet povolen. Tato podmínka vypadá takto:



Obrázek 3.12: Ukázka funkce *smoothstep(a, b, x)*.

```
if (cond_bump_mapping == 1.0f)
```

Pokud je splněna, tak je bump mapping daného terénu aktivní a vypočte se.

```
Nf = bump_mapping( mimic * fr_bump, bumpiness * am_bump,
   oc_bump, pe_bump, PP_start, size_bump_mapping);
```

V případě, že pro daný terén se počítat bump mapping nemá, použije se výpočet, kde `Ng` je normála v aktuálním bodě a `I` je vektor od kamery do aktuálně zpracovávaného bodu:

```
Nf = - faceforward( normalize(Ng) ,I );
```

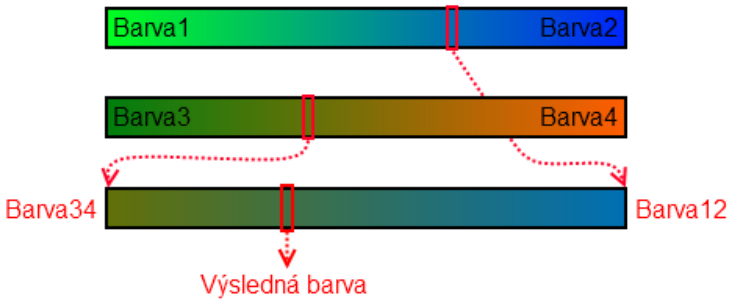
Po výpočtu normály se nastaví standardní opacita (jinak řečeno průhlednost) terénu.

```
Oi = Os;
```

Konečným krokem je výpočet výsledné barvy, která se předá zpátky jako výstup tohoto modulu.

```
color Ct_mix = Oi * ( Ct * (Ka*ambient()+ Kd*diffuse(Nf)));
return Ct_mix;
```

Samotný princip míchání barev je vidět i na obrázku 3.13.



Obrázek 3.13: Míchání jednotlivých barev v modulu `barva.h`.

3.4.4 Modul pro umístění terénu na povrch

Tento modul je volán po zpracování předchozího modulu `barva.sl`. Podle hodnot nastavených v RTC programu nebo `setting.h` se v tomto modulu určuje, jaká barva má být v daném místě (podle podmínek výšky, sklonu a přechodu) použita.

Všechny potřebné hodnoty jsou přenášeny do tohoto modulu přes parametry:

- `color Ct1, color Ct2`
  - Barva nadřazeného terénu (označena jako *Ct1*) a barva aktuálně zpracovávaného terénu (označena jako *Ct2*)
- `point PP`
  - Pozice aktuálně zpracovávaného bodu
- `float min_height_param, float max_height_param`
  - Minimální a maximální výška, která ve které je barva *Ct2* povolena
- `float min_angle_param, float max_angle_param`
  - Minimální a maximální úhel, ve kterém je barva *Ct2* povolena
- `float frequency, float amplitude, float octave, float persistence`

  - *Frekvence*, *amplituda*, *oktáva* a *persistence*, které se používají při počítání náhodné hodnoty (pomocí turbulence) pro přechody mezi terény

- `float cond_min_height, float cond_max_height`
  - Podmínky zda bude použit minimální nebo maximální výškové hranice pro přechod mezi terény

- `float cond_min_angle, float cond_max_angle`
  - Podmínky zda bude použit minimální nebo maximální úhlové hranice pro přechod mezi terény

- `float set_crossover_height, float set_crossover_angle`
  - Nastavení výšky a úhlu pro přechod mezi terény

- `float cond_crossover_noise`
  - Hodnota podmínky, zda se má použít pro přechod mezi terény turbulence

- `float scaleY`
  - Škála výšky terénu (hodnota *1.0f* odpovídá maximální výšce terénu 255 metrů)

- `float terrain_min`
  - Korekční hodnota pro výšky terénu. Jde o rozdíl pozice terénu ve scéně vůči poloze kamery.

Struktura modulu samotného je popsána v následující části textu. Nejdříve se zjistí, zda je aktivován v daném terénu (turbulentní) šum na přechodu.

```
if (cond_crossover_noise == 1)
```

Pokud je tato podmínka splněna (respektive pokud je přechod aktivován), tak bude vypočtena z turbulentní funkce hodnota `random_number` pro změnu nepravidelnou přechodu.

```
random_number = turbulence(frequency, amplitude, octave,
   persistence, 0.0f, PP);
```

V opačném případě je tato hodnota nastavena na 0 a na přechod nebude mít turbulentní funkce žádný vliv.

```
float min_height = min_height_param * scaleY * 255.0f;
float max_height = max_height_param * scaleY * 255.0f;
float crossover_height = set_crossover_height * scaleY
   * 255.0f;
```

Přepočet hodnot minimální výšky, maximální výšky a velikosti přechodu předaných parametry (které se pohybují od *0.0f* do *1.0f*) na výškové hodnoty ve scéně.

```
float crossover_angle = cos(radians(set_crossover_angle));
```

Přepočet úhlu pro přechod mezi terény (pro úhel 0° bude tato hodnota *1.0f* a pro úhel 90° bude *0.0f*).

```
if (cond_min_height == 1)
min_height_var = smoothstep(min_height + terrain_min
   + random_number , min_height + terrain_min
   + crossover_height + random_number, ycomp(PP));
```

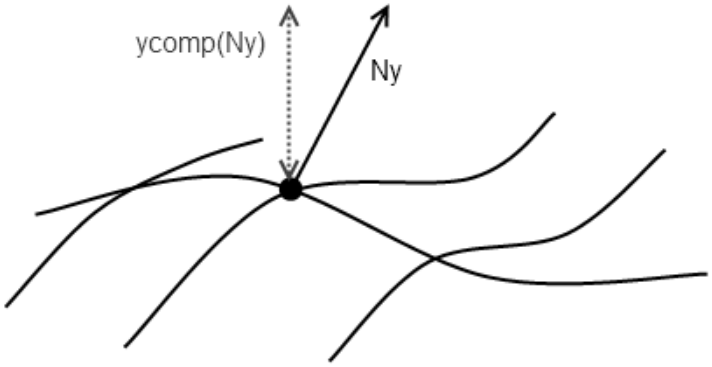
Pokud bude hranice minimální výšky aktivní, tak bude počítána proměnná `min_height_var`. Tato proměnná nabývá hodnot *(0.0f, 1.1f)* a určuje, zda je v daném místě podmínka splněna. Splnění podmínky jako takové se zjišťuje pomocí funkce *smoothstep()*, která je zobrazena na obrázku 3.12.

```
if (cond_max_height == 1)
max_height_var = 1 - smoothstep(max_height + terrain_min
   - crossover_height + random_number, max_height
   + terrain_min + random_number, ycomp(PP));
```

Stejným způsobem akorát s otočenou funkcí *smoothstep()* se vypočítá proměnná horní hranice `min_height_var`.

Po výpočtu výškových hranic terénu je potřeba zjistit, zda v daném místě odpovídá i minimální a maximální podmínka sklonu. Pro tento účel jsou použity následující výpočty, které jsou podobné výpočtům minimální a maximální výšky.

```
float min_angle = cos(radians(min_angle_param));
float max_angle = cos(radians(max_angle_param));
```



Obrázek 3.14: Princip počítání sklonu v bodě z vektoru normály.

Nejdříve je nutno přepočíst hodnoty ze vstupních hodnot minimálního a maximálního úhlu (ve stupních) na hodnoty cosinu (tedy že úhel 0° odpovídá hodnotě *1.0f* a 90° odpovídá *0.0f*).

```
if (cond_min_angle == 1)
min_angle_var = 1 - smoothstep(min_angle, min_angle
   + crossover_angle, abs(ycomp(Ny)+ random_number));

if (cond_max_angle == 1)
max_angle_var = smoothstep(max_angle - crossover_angle,
   max_angle, abs(ycomp(Ny)+ random_number));
```

Výpočet je obdobný s tím, že sklon v daném bodě se určuje pomocí funkce *ycomp(Ny)*, která je vidět na obrázku 3.14, kde *Ny* je normalizovaná normála v tomto bodě. V případě, že je v tomto bodě povrchu terénu sklon blížící se k 90° (tedy povrch je v tomto místě s velkým sklonem), tak je hodnota této funkce nulová a pro 0° (pro vodorovný povrch) nabývá jednotkové hodnoty. Což jsou přesně hodnoty potřebné pro porovnání ve funkci *smoothstep()* s krajními hodnotami sklonu.

```
Ct_mix = mix(Ct1, Ct2, max_height_var * min_height_var
   * max_angle_var * min_angle_var);
```

Pokud tedy budou splněny všechny podmínky (respektive *max_height_var*, *min_height_var*, *max_angle_var* a *min_angle_var* budou rovny hodnotě *1.0f*),

tak to znamená, že v daném místě má být terén umístěn. V tom případě se jako výsledná barva terénu v počítaném bodě zvolí *Ct2* - barva aktuálního terénu. Pokud jedna z podmínek splněna nebude, výsledek bude *0.0f* a výsledek funkce *smoothstep()* bude barva nadřazeného terénu.

3.4.5 Modul hlavního surface shaderu

Tento modul je hlavní a jde o surface shader, který je volán z RIB souboru a aplikován na povrch objektu (v tomto případě na povrch plochy představující terén). Podle parametrů nastavených přímo v tomto surface shaderu a podle parametrů v souboru `setting.h` se určuje obarvení každého bodu tohoto povrchu.

Každý parametr, který je vstupem tohoto surface shaderu, je předdefinován již z programu (jehož popis je v kapitole 3.2).

Uvnitř jsou volány dva druhy funkcí (modulů). Jako první je volána funkce *colour()* tolikrát, kolik bylo uživatelem v programu nastaveno terénů určených pro generování. Tato funkce zpět vrací obarvení bodu v daném místě. Avšak pro zjištění, kde má být ve výsledku toto obarvení použito, je potřeba použít tyto barvy pro další zpracování.

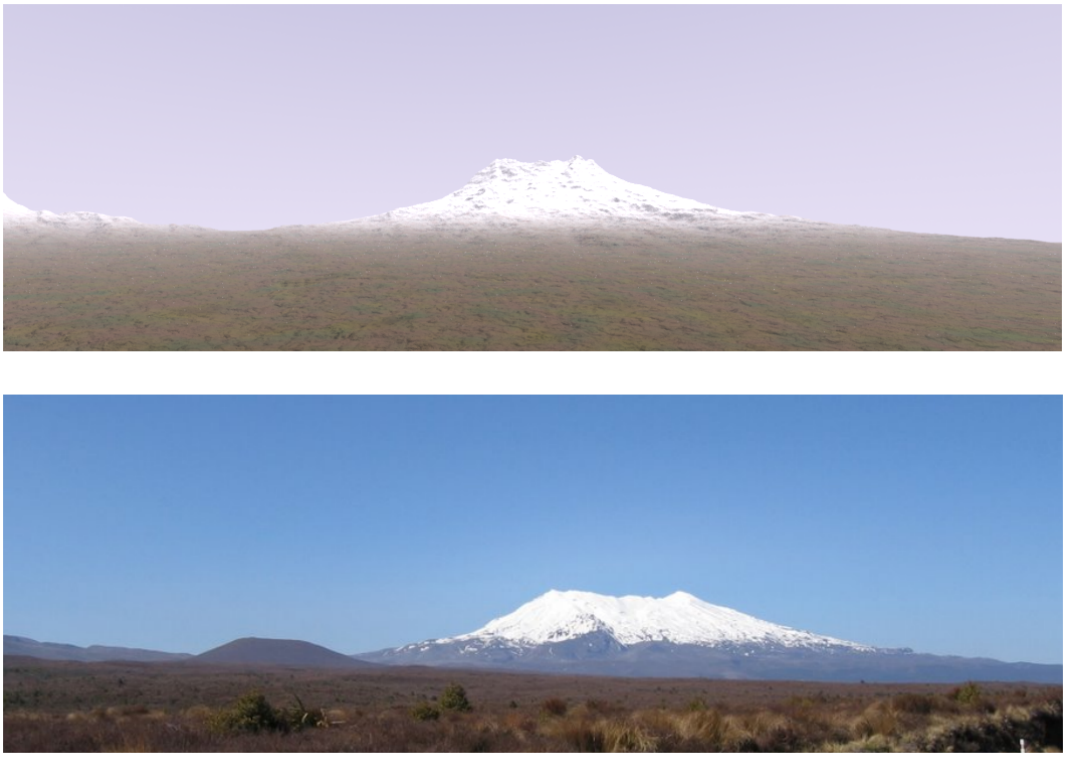
Po tom, co se vypočtou barvy jednotlivých povrchů, dojde k zpracování těchto barev funkcí *resolve_function()*, jejíž parametry i princip jsou popsány v kapitole 3.4.4.

Výstupem tohoto shaderu je barva pro každý bod povrchu.

3.5 Dosažené výsledky

3.5.1 Napodobení reálné hory Mount Ruapehu

K vytvoření subtextury terénu tohoto obrázku (jehož úkolem je napodobit reálnou fotku hory Mount Ruapehu) byly použity následující parametry.



Obrázek 3.15: Porovnání vytvořeného výsledku (*nahoře*) s reálnou fotografií hory Mount Ruapehu (*dole*) - autor fotografie: *M. Spadari - Panoramio.*

- **Podkladový povrch**
  - Parametry

	míchání barev	bump mapping
frekvence	0.09f	0.4f
amplituda	0.7f	0.7f
oktáva	5	7
persistence	0.5f	0.5f

  - Barvy subtextury

	Červená	Zelená	Modrá
1.	65	49	41
2.	39	53	36
3.	72	66	30
4.	38	29	26

- **Kamenitý povrch**
  - Podmínky výskytu subtextury

min. výška	13.38 metru
výška přechodu	25.78 metru
min. sklon	1°
max. sklon	5°
sklon přechodu	3°

  - Parametry

	míchání barev	bump mapping	přechod
frekvence	0.15f	0.8f	0.8f
amplituda	0.7f	0.7f	0.7f
oktáva	4	4	4
persistence	0.5f	0.5f	0.5f

  - Barvy subtextury

	Červená	Zelená	Modrá
1.	128	128	128
2.	192	192	192
3.	64	0	0
4.	128	128	64

- **Sníh**
  - Podmínky výskytu subtextury

min. výška	17.85 metru
výška přechodu	35.7 metru
max. sklon	50°
sklon přechodu	10°

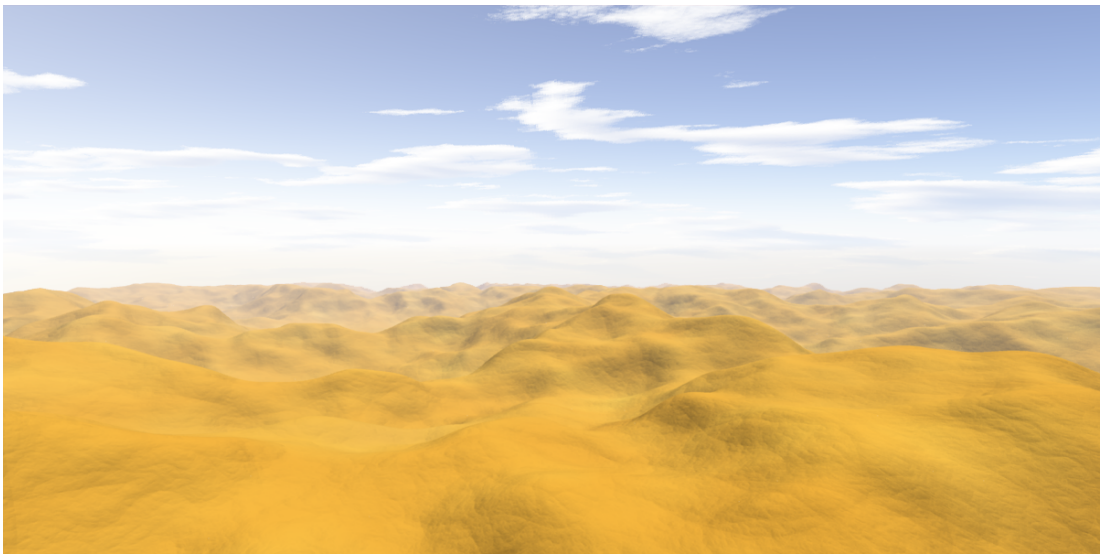
  - Parametry

	míchání barev	bump mapping	přechod
frekvence	0.2f	0.2f	0.8f
amplituda	0.7f	0.7f	0.7f
oktáva	4	4	4
persistence	0.5f	0.5f	0.5f

– Barvy subtextury

	Červená	Zelená	Modrá
1.	255	255	255
2.	252	252	252
3.	252	252	252
4.	248	248	248

3.5.2 Napodobení pouštní scenérie



Obrázek 3.16: Pouštní scenérie vygenerovaná pomocí RTC.

- **Písečné duny**

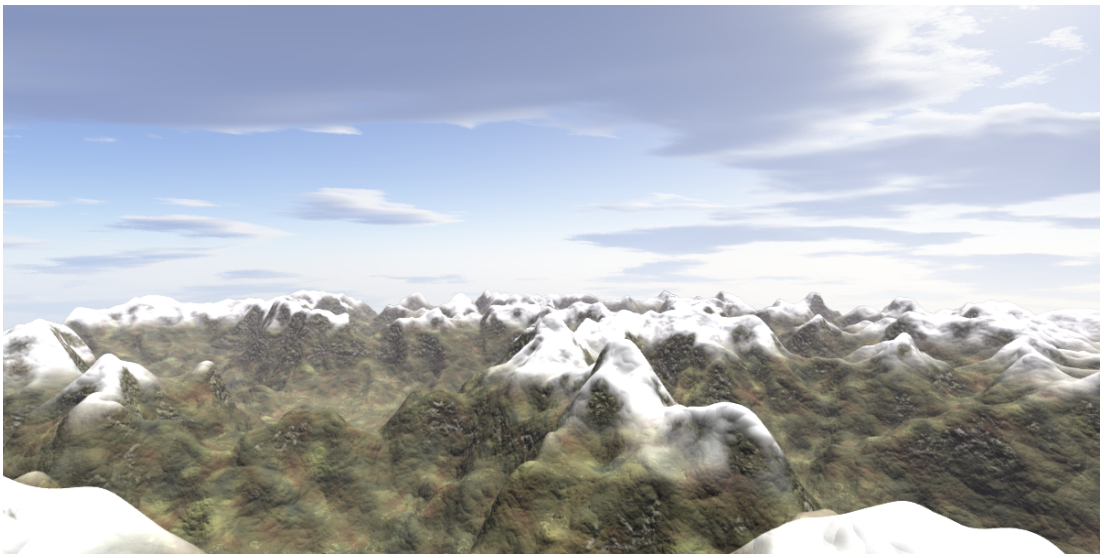
  – Parametry

	míchání barev	bump mapping
frekvence	0.1f	0.1f
amplituda	0.5f	0.4f
oktáva	5	9
persistence	0.5f	0.5f

– Barvy subtextury

	Červená	Zelená	Modrá
1.	197	141	37
2.	186	152	50
3.	215	185	64
4.	174	106	66

3.5.3 Napodobení horské scenérie



Obrázek 3.17: Horská scenérie vytvořená pomocí programu RTC.

- **Podkladový povrch**

  – Parametry

	míchání barev	bump mapping
frekvence	0.2f	0.3f
amplituda	1.0f	0.4f
oktáva	4	7
persistence	0.5f	0.5f

  - Barvy subtextury

	Červená	Zelená	Modrá
1.	78	79	49
2.	84	58	55
3.	55	57	51
4.	155	165	90

- **Kamenitý povrch**
  - Podmínky výskytu subtextury

min. výška	5.1 metru
výška přechodu	7.65 metru
min. sklon	50°
max. sklon	70°
sklon přechodu	15°

  - Parametry

	míchání barev	bump mapping	přechod
frekvence	0.2f	0.7f	0.8f
amplituda	0.7f	0.7f	1.7f
oktáva	4	4	4
persistence	0.5f	0.5f	0.5f

  - Barvy subtextury

	Červená	Zelená	Modrá
1.	184	180	133
2.	128	128	128
3.	192	192	192
4.	83	84	56

- **Travnatý povrch**
  - Podmínky výskytu subtextury

min. sklon	2°
max. sklon	15°
sklon přechodu	7°

  - Parametry

	míchání barev	bump mapping	přechod
frekvence	0.2f	0.8f	0.8f
amplituda	0.3 f	0.7f	0.7f
oktáva	4	4	4
persistence	0.5f	0.5f	0.5f

– Barvy subtextury

	Červená	Zelená	Modrá
1.	122	123	83
2.	60	60	43
3.	122	141	90
4.	154	170	136

- **Sníh**

– Podmínky výskytu subtextury

min. výška	15.3 metru
výška přechodu	5.1 metru
max. sklon	60°
sklon přechodu	10°

– Parametry

	míchání barev	bump mapping	přechod
frekvence	0.8f	0.1f	0.8f
amplituda	0.7f	0.7f	0.7f
oktáva	4	4	4
persistence	0.5f	0.5f	0.5f

– Barvy subtextury

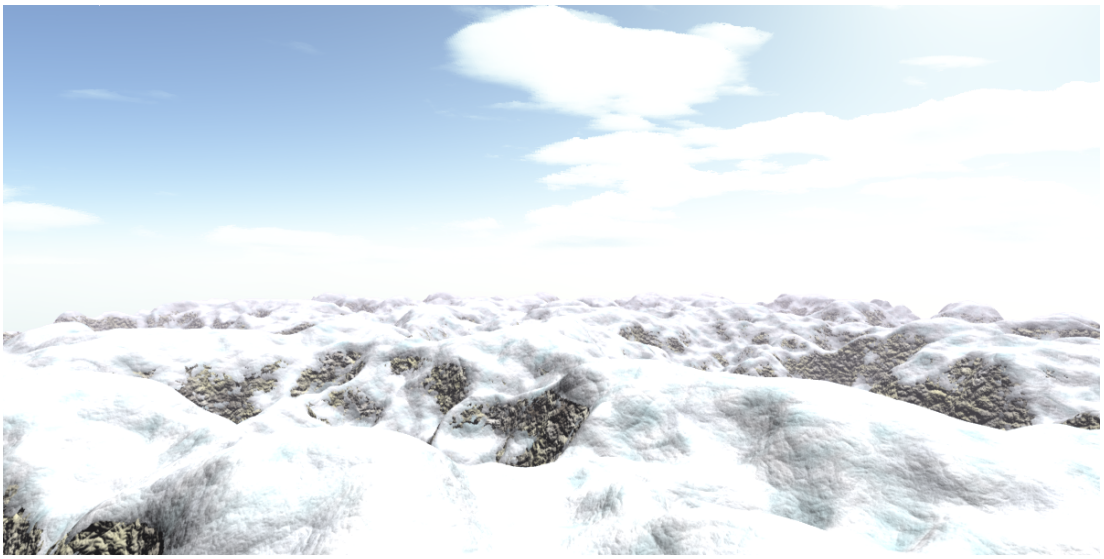
	Červená	Zelená	Modrá
1.	255	255	255
2.	247	247	247
3.	255	255	255
4.	248	248	248

3.5.4 Napodobení antarktické scenérie

- **Podkladový povrch - zledovatělý sníh**

– Parametry

	míchání barev	bump mapping
frekvence	0.3f	0.1f
amplituda	0.9f	0.7f
oktáva	4	8
persistence	0.5f	0.5f



Obrázek 3.18: Antarktická scenérie vytvořená pomocí programu RTC.

– Barvy subtextury

	Červená	Zelená	Modrá
1.	235	235	235
2.	208	214	221
3.	164	196	200
4.	232	238	238

- **Kamenitý povrch**

  – Podmínky výskytu subtextury

  min. výška 1.785 metru
  max. výška 14.28 metru
  min. sklon 45°

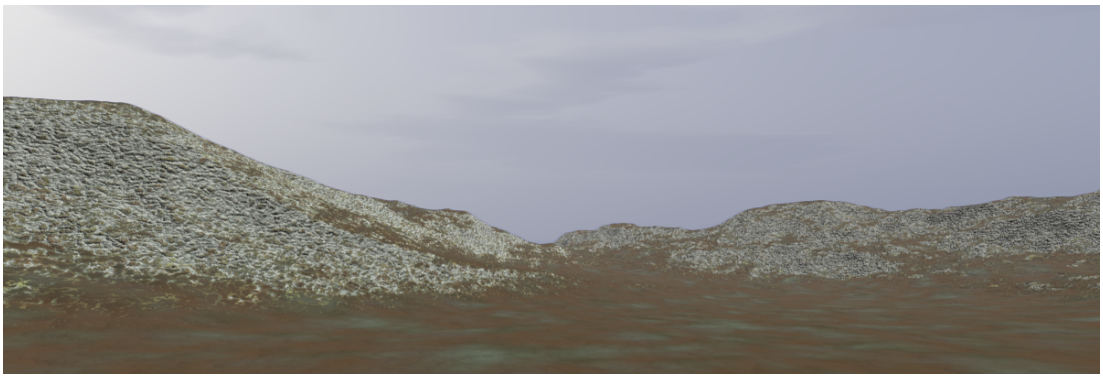
  – Parametry

	míchání barev	bump mapping	přechod
frekvence	0.2f	0.7f	0.8f
amplituda	1.0f	0.7f	1.7f
oktáva	4	4	4
persistence	0.5f	0.5f	0.5f

– Barvy subtextury

	Červená	Zelená	Modrá
1.	184	180	133
2.	128	128	128
3.	192	192	192
4.	83	84	56

3.5.5 Napodobení horské scenérie



Obrázek 3.19: Horská scenérie vytvořená pomocí programu RTC.

- **Podkladový povrch**

  – Parametry

	míchání barev	bump mapping
frekvence	0.1f	0.1f
amplituda	0.9f	0.7f
oktáva	6	8
persistence	0.5f	0.5f

  – Barvy subtextury

	Červená	Zelená	Modrá
1.	66	50	34
2.	50	48	35
3.	60	55	47
4.	69	77	64

- **Kamenitý povrch**
  - Podmínky výskytu subtextury

min. sklon	20°
max. sklon	90°
sklon přechodu	7°

  - Parametry

	míchání barev	bump mapping	přechod
frekvence	0.75f	0.8f	0.8f
amplituda	1.0f	0.7f	1.7f
oktáva	4	4	4
persistence	0.5f	0.5f	0.5f

  - Barvy subtextury

	Červená	Zelená	Modrá
1.	159	165	156
2.	80	69	48
3.	87	95	84
4.	73	78	67

- **Mech a bláto**
  - Podmínky výskytu subtextury

min. sklon	20°
sklon přechodu	7°

  - Parametry

	míchání barev	bump mapping	přechod
frekvence	0.2f	0.3f	0.8f
amplituda	1.0f	0.4f	1.7f
oktáva	4	4	4
persistence	0.5f	0.45f	0.5f

  - Barvy subtextury

	Červená	Zelená	Modrá
1.	78	79	49
2.	84	58	44
3.	55	57	51
4.	155	165	90

4 Závěr

Úkolem této bakalářské práce bylo přiblížení se realistickému vyobrazení terénu pomocí aplikace procedurálních textur na objekt reprezentující terén. K tomuto úkolu jsem využil znalosti specifikace RenderMan a programovacího jazyku RenderMan Shading Language.

Při práci jsem se naučil hlouběji používat jazyku RenderMan Shading Language pro tvorbu netriviálních procedurálních textur, které jsem aplikoval na objekty reprezentující geometrii povrchu terénu. K tvorbě procedurálních textur terénu jsem si vytvořil surface shader, který je složen z několika bloků (modulů). Každý z těchto modulů má za úkol jiný výpočetní úkon a pro vytvoření obarvovacího (surface) shaderu terénu jsou tyto bloky mezi sebou kombinovatelné.

Pro zpřehlednění tvorby a skládání těchto modulů jsem napsal interaktivní GUI program, pomocí něhož lze velmi jednoduše vygenerovat shader a RIB soubor s definicí výsledné scény. Tento program nabízí velkou škálu nastavitelných parametrů pro ovlivnění procedurální textury terénu. V programu je k dispozici i náhled toho, kde se v terénu změny projeví. Pro vytvoření programu jsem prostudoval různé programy zabývající se fotorealistickým renderingem terénu (viz příloha A) a vybral si program Terragen Classic jako inspiraci pro tvorbu mého programu RTC. Oproti programu RTC program Terragen Classic generuje trochu realističtější výsledky, avšak i pomocí programu RTC se při správném nastavení dají vytvořit velmi realistické obrázky.

Vytvořil jsem několik netriviálních realistických obrázků s terény a také porovnal výsledek RTC programu i s výsledky ostatních prozkoumaných programů (viz příloha B).

Literatura

[1] STEPHENSON, Ian. *Essential RenderMan*. 2nd ed. London : Springer, 2007. ISBN 978-1-84628-344-4.

[2] STEPHENSON, Ian. *Essential RenderMan fast*. London : Springer, 2003. ISBN 1-85233-608-0.

[3] APODACA, Anthony A. a GRITZ, Larry. *Advanced RenderMan : creating CGI for motion pictures*. San Francisco : Kaufmann, 2000. ISBN 1-55860-618-1.

[4] ŽÁRA, Jiří et al. *Moderní počítačová grafika*. 2., přeprac. a rozš. vyd. Brno : Computer Press, 2004. ISBN 80-251-0454-0.

[5] EBERT, David S. et al. *Texturing & modeling : a procedural approach*. 3rd ed. Amsterdam : Morgan Kaufmann Publishers, 2003. ISBN 1-55860-848-6.

[6] SHIRLEY, Peter. *Realistic ray tracing*. Natick : A K Peters, 2000. ISBN 1-56881-110-1.

[7] JEŽEK, František. *Geometrické a počítačové modelování : pomocný učební text*. 8. vydání. Plzeň : KMA ZČU, 2006.

[8] WIRTH, Niklaus. *Algoritmy a štruktúry údajov* [z angl.orig. přel. Pavol Fischer]. 1. vyd. Bratislava : Alfa, 1988.

[9] RAGHAVACHARY, Saty. *Rendering for beginners : image synthesis using RenderMan*. Oxford : Focal Press, 2005. ISBN 0-240-51935-3.

[10] TUREK, Milan. *Farao - perlin noise* [online]. 2002, poslední revize 14.8.2004 [citováno 13.3.2011].
Dostupné z <http://farao.czweb.org/perlin.htm>.

[11] *RenderMan - stručný popis* [online]. 1998, poslední revize 20.4.1998 [citováno 13.3.2011].
Dostupné z <http://herakles.zcu.cz/local/manuals/rman/popis.php>.

[12] ŠINDELÁŘ, Vladislav. *GRAFIKA - Digitální model terénu* [online]. 1999, poslední revize 10.8.1999 [citováno 13.3.2011].
Dostupné z <http://www.grafika.cz/art/3d/clanek1033198934.html>.

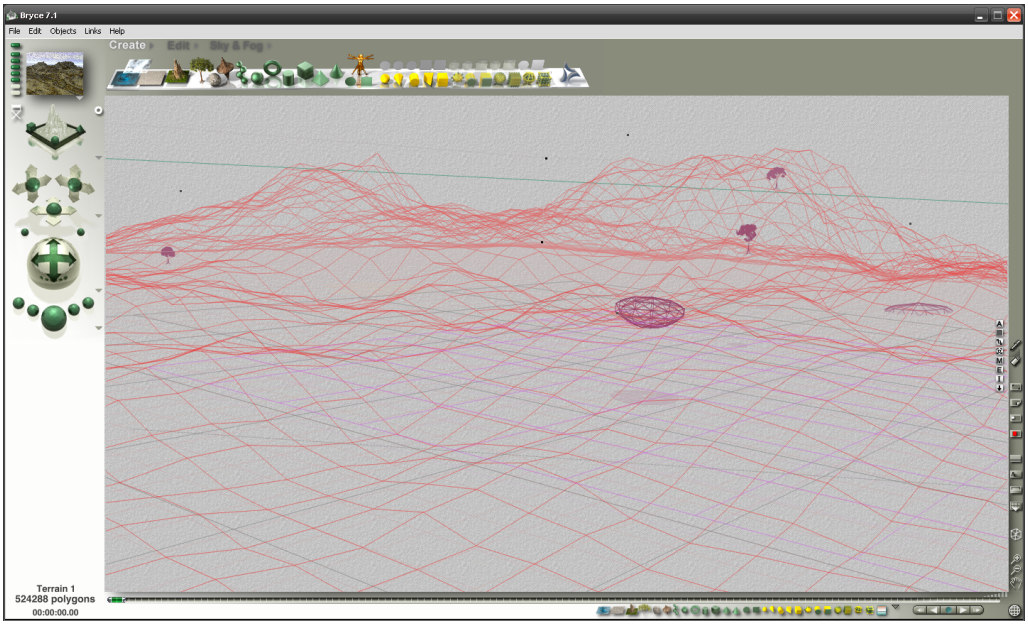
[13] *Bump mapping - Wikipedia* [online]. 2006, poslední revize 10.2.2010 [citováno 13.3.2011].
Dostupné z <http://cs.wikipedia.org/wiki/Bump_mapping>.

[14] *RenderMan Interface V3.1 Section 5* [online]. 1997, poslední revize 24.3.1997 [citováno 3.4.2011].
Dostupné z <http://graphics.stanford.edu/lab/soft/purgatory/prman/RISpec/section5.html>.

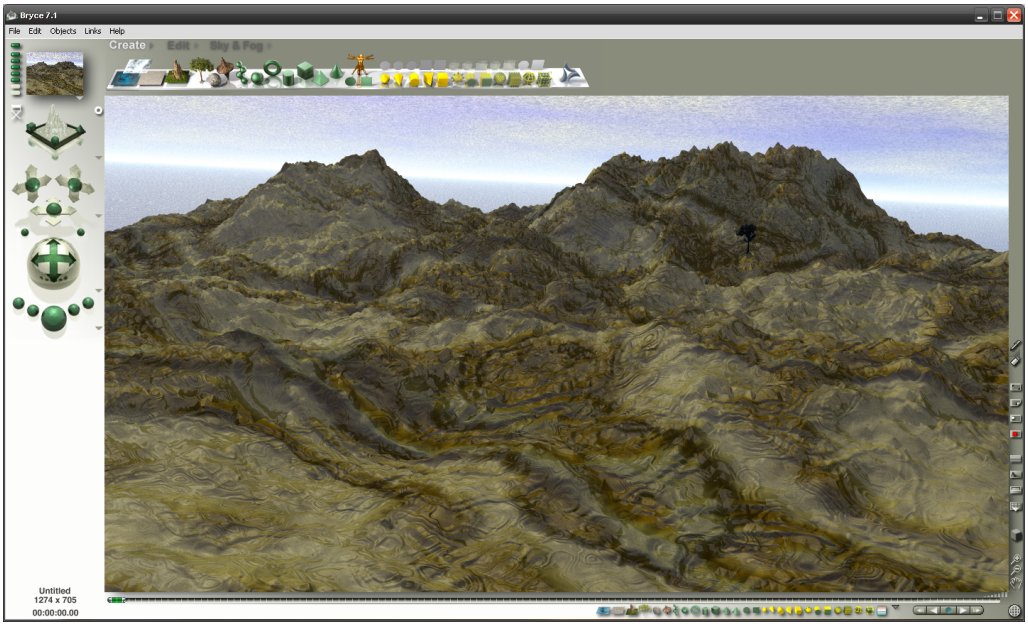
Seznam zkratek

DMT	Digitální model terénu
GUI	Graphical user interface
NASA	National Aeronautics and Space Administration
RIB	RenderMan interface bytestream
RSL	RenderMan shading language
RTC	RenderMan Terrain Creator
SL	Shading language
SRTM	Shuttle Radar Topography Mission

Příloha A



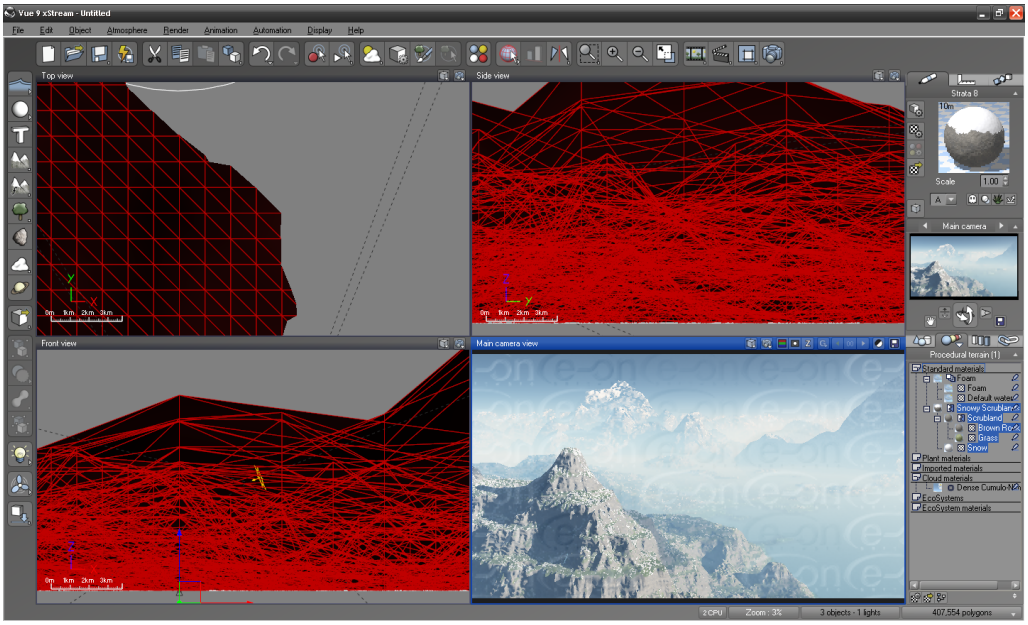
Obrázek 1: Ukázka prostředí programu Bryce 7.



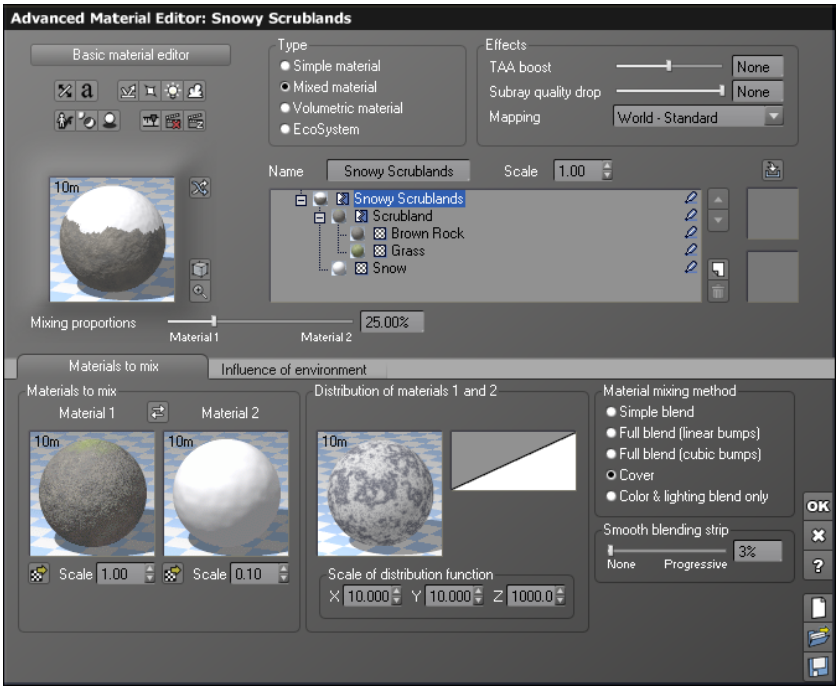
Obrázek 2: Ukázka renderingu v programu Bryce 7.

Obrázek 3: Ukázka prostředí programu Bryce 7 - nastavení materiálu povrchu terénu.

Obrázek 4: Ukázka výsledku generovaného programem Bryce 7 (*autor: David Brinnen*).



Obrázek 5: Ukázka prostředí programu E-ON Vue 9.

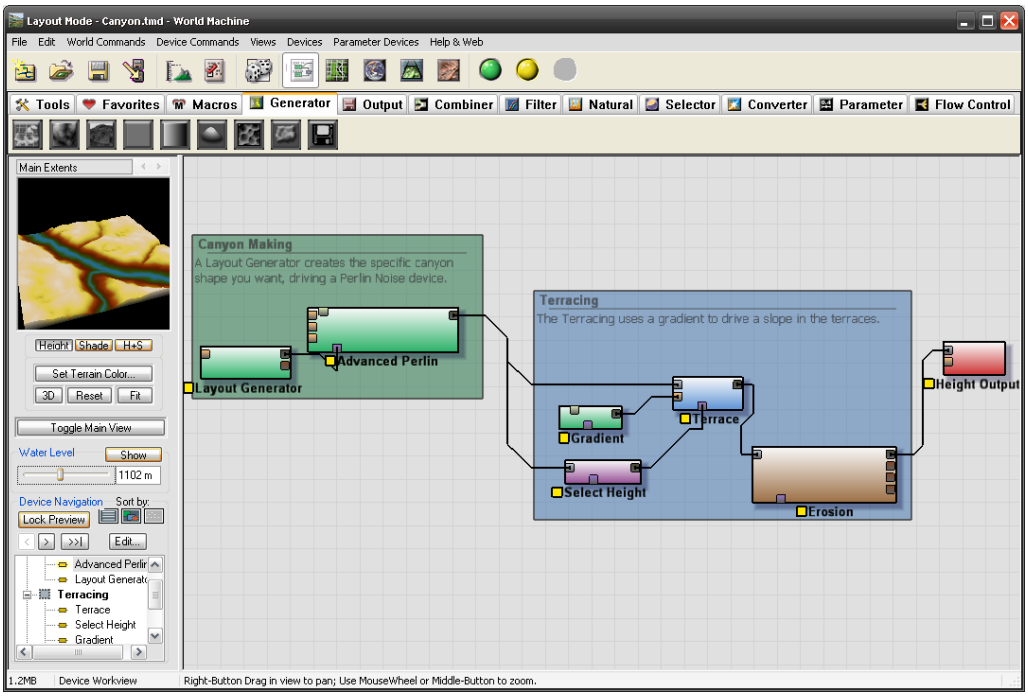


Obrázek 6: Nastavení materiálu terénu v programu E-ON Vue 9.

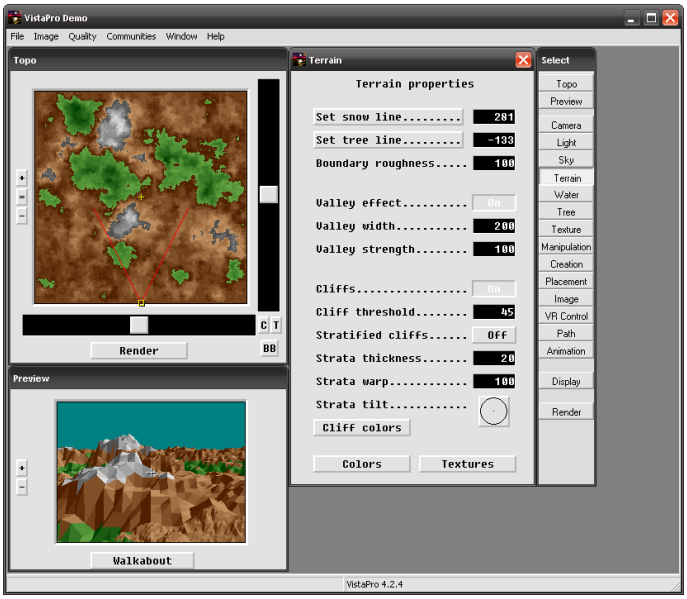
Obrázek 7: Ukázka výsledku generovaného programem E-ON Vue 9.

Obrázek 8: Ukázka výsledku generovaného programem E-ON Vue 9 (*autor: Oliver Regueiro*).

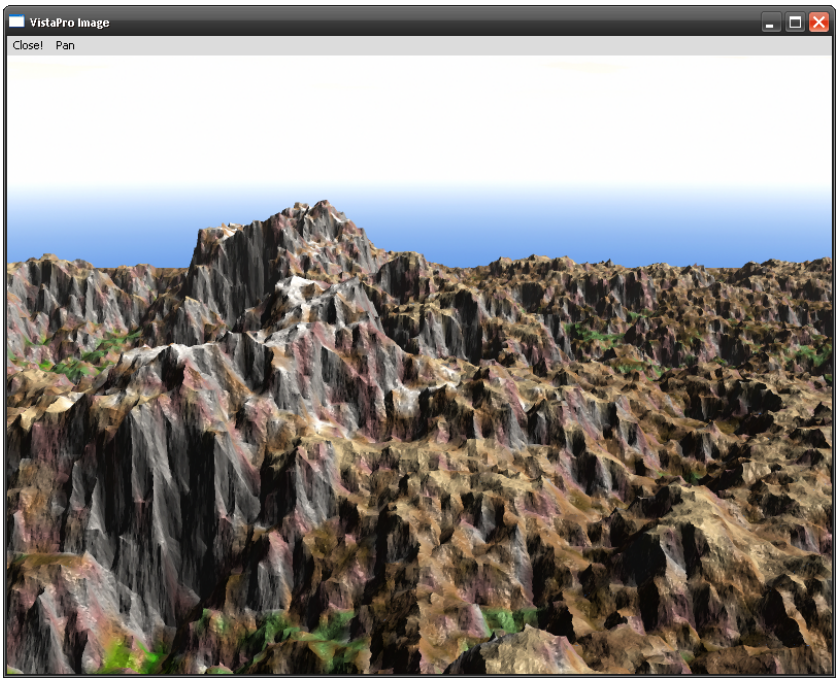
Obrázek 9: Ukázka výstupu programu E-ON Vue 9.



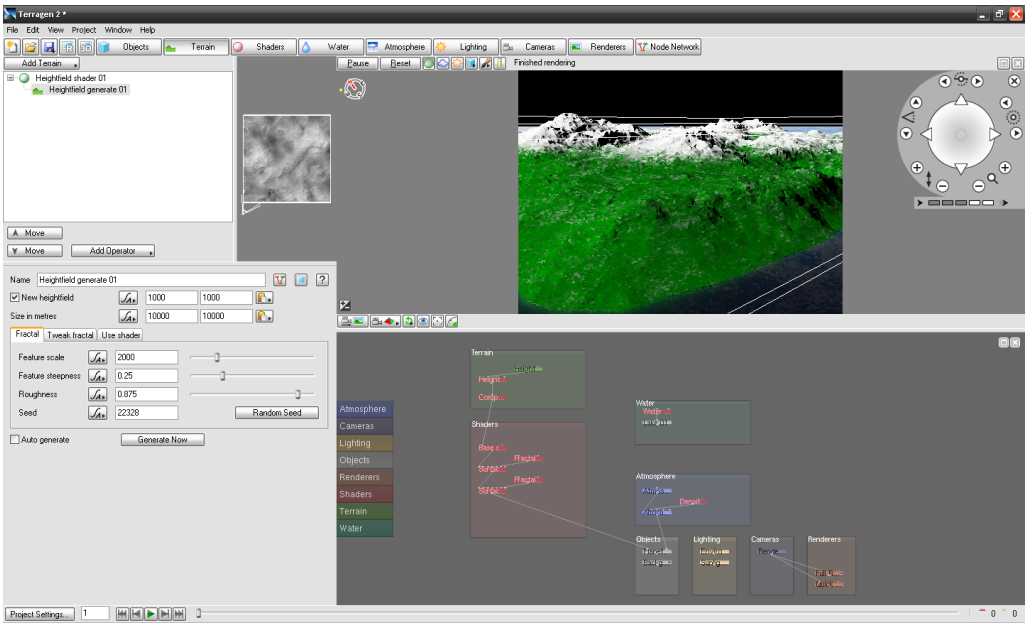
Obrázek 10: Ukázka prostředí programu World Machine.



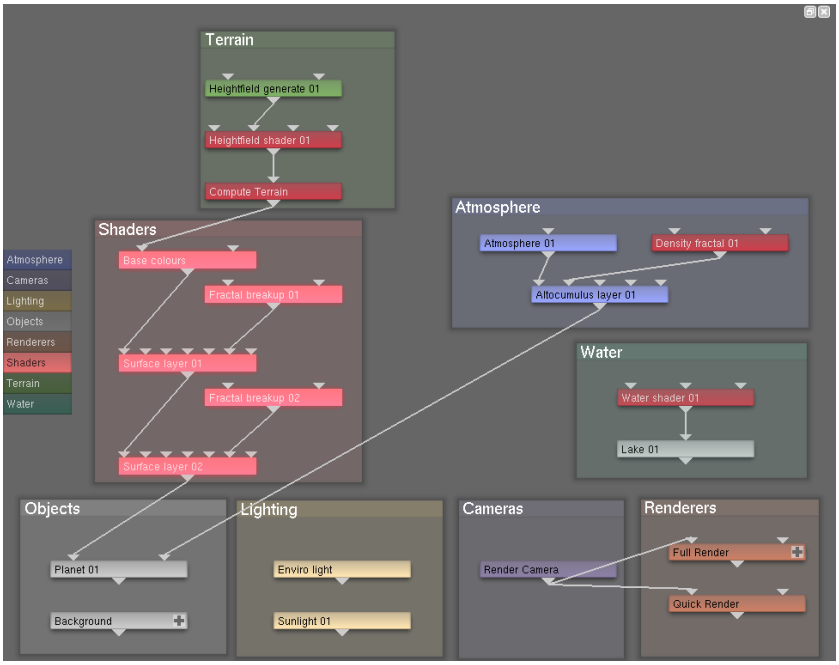
Obrázek 11: Ukázka prostředí poslední verze programu VistaPro 4.2.



Obrázek 12: Ukázka výsledku v programu VistaPro 4.2.

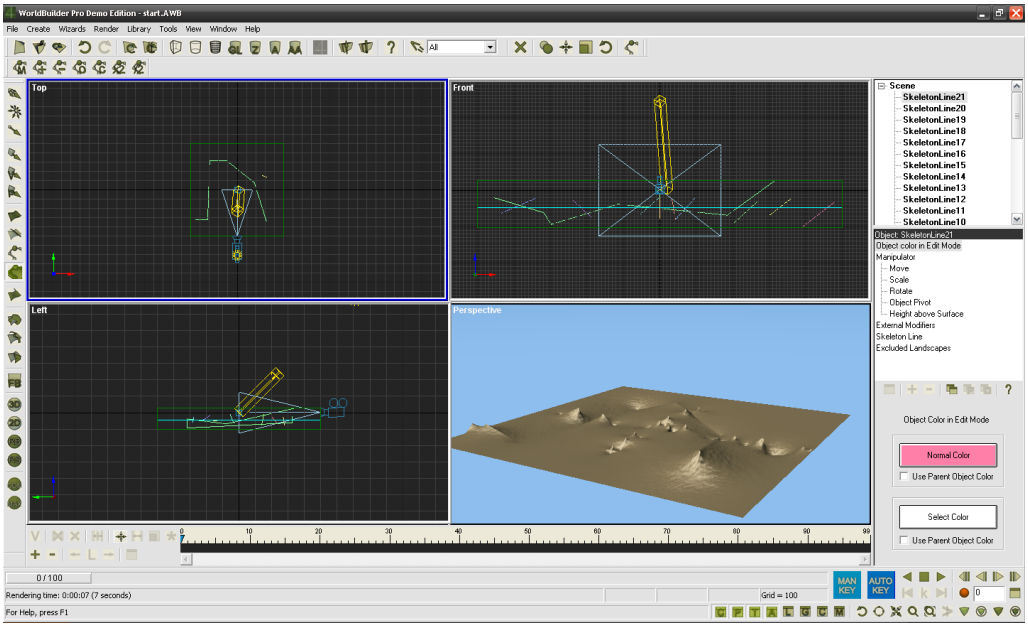


Obrázek 13: Ukázka programu Terragen 2.



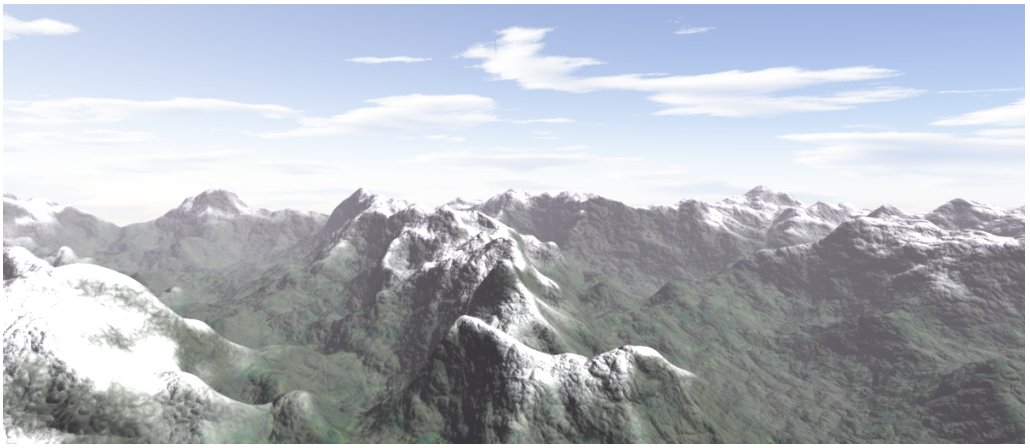
Obrázek 14: Ukázka „Node Network“ programu Terragen 2.

Obrázek 15: Ukázka výsledku z programu Terragen 2.



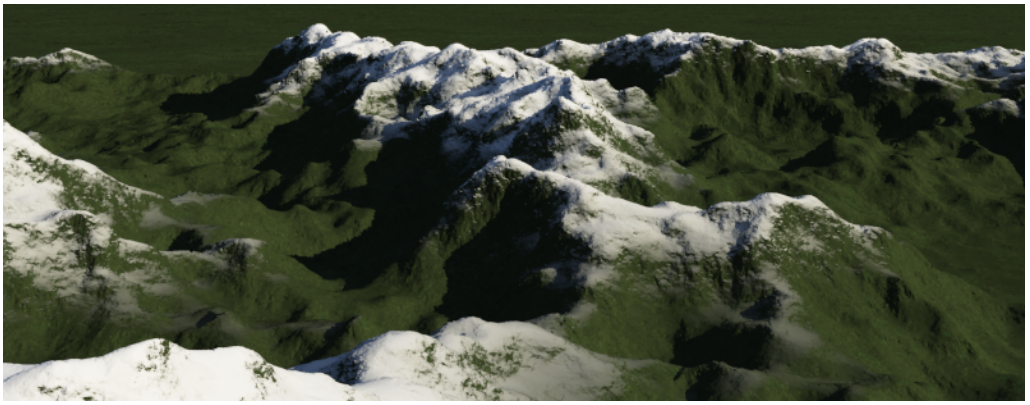
Obrázek 16: Ukázka výsledku v programu World Builder.

Příloha B

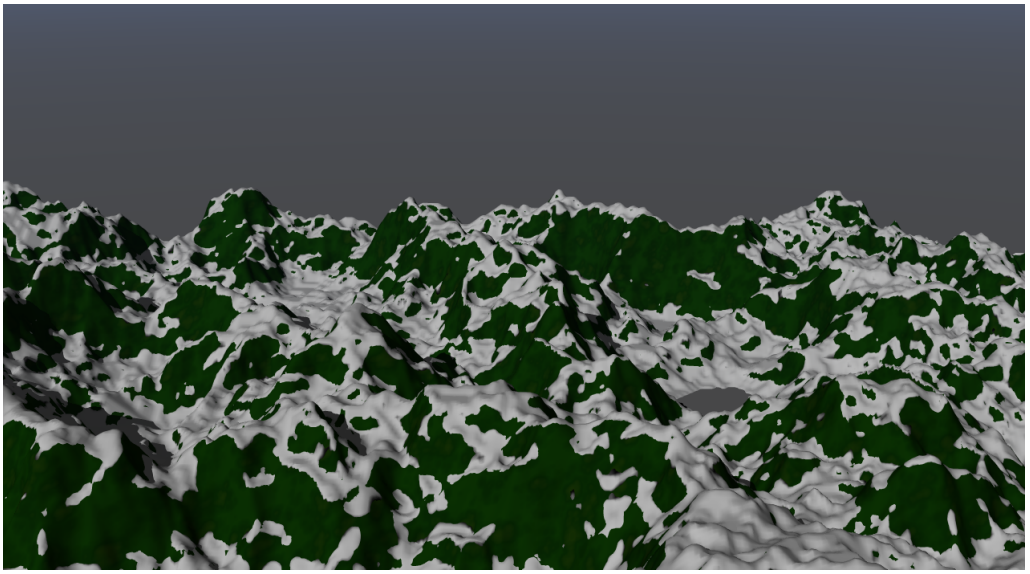


Obrázek 17: Ukázka výsledku programu RTC.

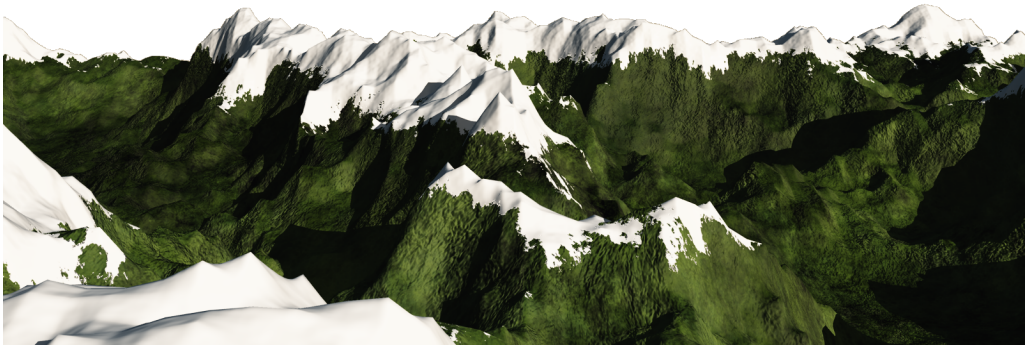
Obrázek 18: Ukázka výsledku programu Terragen Classic.



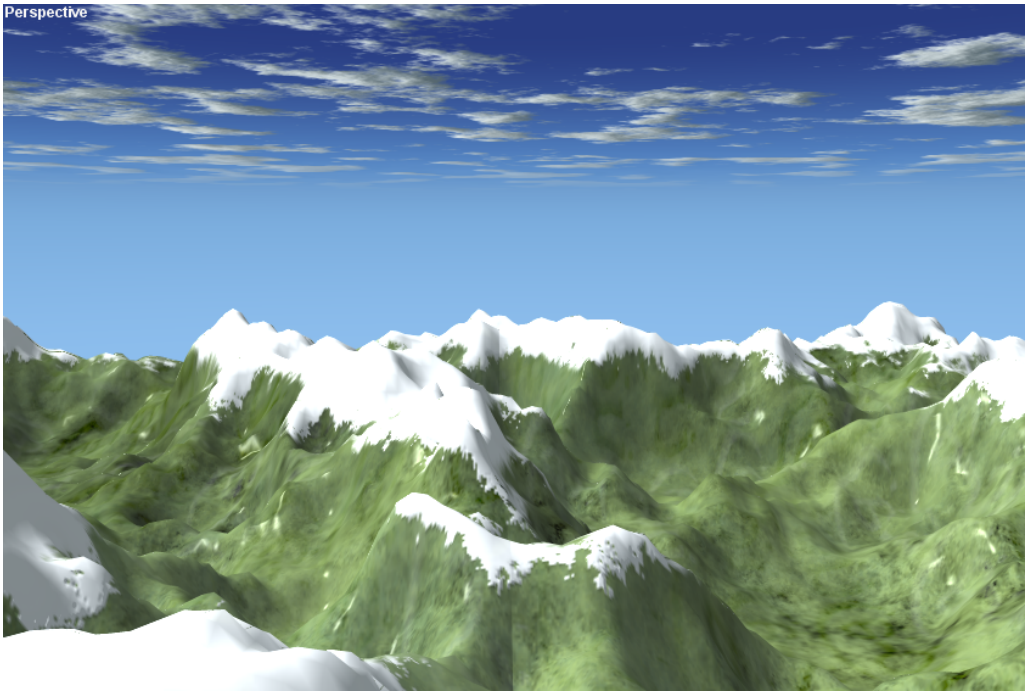
Obrázek 19: Ukázka výsledku programu Terragen 2.



Obrázek 20: Ukázka výsledku programu Bryce.



Obrázek 21: Ukázka výsledku programu Vue.



Obrázek 22: Ukázka výsledku programu WorldBuilder.