VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

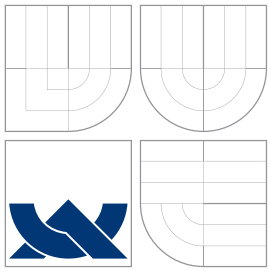
GENERÁTORY UMĚLÉHO ŠUMU V OBRAZE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

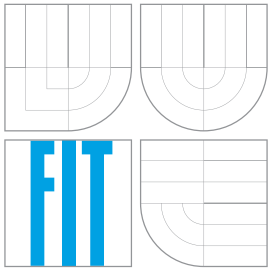
AUTOR PRÁCE
AUTHOR

PAVEL SIGMUND

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

GENERÁTORY UMĚLÉHO ŠUMU V OBRAZE

GENERATORS OF SYNTHETIC NOISE IN PICTURE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PAVEL SIGMUND
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL ŠPANĚL
SUPERVISOR

BRNO 2007

Abstrakt

V této práci implementuji knihovnu pro generovaní základních typů šumů, kterou dále využívám pro generování procedurálních textur a výškové mapy. Zaměřil jsem se na metodu K. Perlina, hojně využívanou pro tvorbu textur s přírodnímy motivy například dřevo, mramor, mraky. Tuto metodu jsem naimplementoval do knihovny pro tvorbu 2D a 3D textur. Knihovna pro generování šumu není díky své univerzálnosti předurčena pouze pro použití jako zdroj pseudonáhodný čísel pro metodu K. Perlina, ale dá se velmi dobře použít k přidání šumu do jakýkolich dat.

Klíčová slova

Pseudonáhodná čísla, šum, uměle generovaný šum, textury, výšková mapa, interpolace, Perlinův šum, OpenGL

Abstract

In this work I'm implementing library for generating basic kinds of noises which I'm also using for generating procedural textures and height maps . I have focused on K. Perlin method frequently used for making textures with nature-based motives, like wood, marble or clouds. I've implemented this method into the library to make 2D and 3D textures. Library used for generating noises, thanks to its variability, is not ment as a source of generated pseudo-random numbers for K. Perlin method only, but we can also use it as a tool to add a noise to any kind of data.

Keywords

Pseudo-random numbers, noise, artificially generadet noise, textures, height map, interpolation, Perlin noise, OpenGL

Citace

Pavel Sigmund: Generátory umělého šumu v obraze, bakalářská práce, Brno, FIT VUT v Brně, 2007

Generátory umělého šumu v obraze

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Španěla a uvedl v ní veškerou literaturu a zdroje, ze kterých jsem čerpal.

.....
Pavel Sigmund
15. května 2007

Poděkování

Na tomto místě bych rád poděkoval vedoucímu této bakalářské práce, panu Ing. Michalu Španělovi, za cenné rady a připomínky, které mi pomohly vytvořit tuto práci.

Zadání bakalářské práce/5224/2006/xsigmu01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Sigmund Pavel**
Obor: Informační technologie
Téma: **Generátory umělého šumu v obraze**
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte základy zpracování obrazu. Zaměřte se zejména na problematiku šumu v obraze, jeho typů a metod umělého generování.
2. Vyberte vhodné typy šumu a navrhněte algorithmy pro jejich umělé generování. Demonstrujte funkčnost algoritmů.
3. Zaměřte se na aplikace uměle generovaných šumů v počítačové grafice a experimentujte s vaší implementací.
4. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
5. Vytvořte stručný plakát prezentující vaši bakalářskou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních tří bodů zadaní.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Španěl Michal, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 L.S.

doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Pavel Sigmund**
Id studenta: 47888
Bytem: Nové Syrovice 38, 675 41 Nové Syrovice
Narozen: 28. 11. 1982, Třebíč
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Generátory umělého šumu v obraze
Vedoucí/školitel VŠKP: Španěl Michal, Ing.
Ústav: Ústav počítačové grafiky a multimédií
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Obsah

Seznam obrázků

Kapitola 1

Úvod

Většina lidí, kteří se přímo nezabývají touto problematikou, mají v podvědomí, že šum je vždy něco nažádoucího. Málokdo však ví, jak může být uměle generováný šum užitečný a že se s jeho aplikacemi setkává poměrně často. Uměle generovaný šum nachází využití například v biomedicíně či počítačové grafice.

Počítačová grafika je velmi rozšířený a rychle se vyvíjející obor. Lze se s ní setkat například ve filmovém průmyslu, počítačových hrách, Geografikých informačních systémech. Počítačová grafika nám umožňuje modelovat a zobazovat svět kolem nás. Je přirozenou snahou vytvořit co nejvěrnější model modelované skutečnosti. Z velké části záleží také na kvalitě textur objektů. Některé textury můžeme vytvořit buď pracně ručně nebo je procedurálně vygenerovat. Vytvoření textur procedurálně má tu výhodu, že nezabírají na pevném disku žádné místo a navíc je lze opětovně vygenerovat při použití stejných parametrů. Při vytváření textur přírodního charakteru se velmi často používá metoda Kena Perlina, které se v práci věnuji.

Metoda K. Perlina využívá pro svůj chod generátor pseudonáhodných čísel. Nahradíme-li tento generátor generátorem vracející pseudonáhodná čísla s určitým rozložením hodnot, dosáhneme možnosti lépe ovlivnit podobu výsledné textury. Cenou za to je nižší výkon generování, avšak v některých aplikacích to nevadí.

Perlinova šumová funkce, jak svou metodu K. Perlin nazval, lze využít k vytvoření mnoha různých přírodních vzorků jako například mramoru, textury dřeva, ohně, mraků, písku, vodní hladiny. Časté využití nachazí také při vytváření výškových map, používaných při generování terénu krajiny.

V této práci je popsáno několik algoritmů pro výtváření textur pomocí Perlinovi šumové funkce.

Kapitola 2

Generování Šumu

2.1 Šum v obraze

Šum je nežádoucí porucha obrazu, která vniká při snímání, přenosu a zpracování obrazu. Existuje poměrně mnoho metod filtrace šumu s tím že pro různé typy šumů jsou vhodné různé metody generování a filtrace. Uměle generovaný šum má však šikoké uplatnění. Například k testování vyvíjených metod filtrace obrazu, nebo se dá využít pro generování textur.

Šum můžeme rozdělit podle závislosti na velikosti obrazového signálu na aditivní a multiplikativní šum, jak je uvedeno v [3].

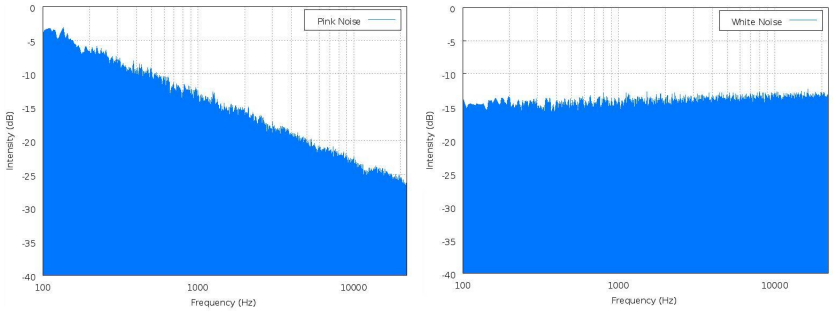
Obraz lze chápat jako funkci dvou proměných $f(x, y)$, kde x a y jsou souřadnice v obrázku a funkční hodnotou je hodnota pixelu na pozici x, y. Aditivní šum, kde obraz $f(x, y)$ a šum $v(x, y)$ jsou nezávislé

$$g(x, y) = f(x, y) + v(x, y)$$

Multiplikativní šum, kde velikost šumu závisí na velikosti obrazového signálu.

$$g = f + vf = f(1 + v) \approx fv$$

Šumy můžeme dále dělit podle závislosti intenzity šumu na frekvenci, tedy barvy šumu jak stojí v [2]. Rozeznáváme celou řadu barev šumu, pro názonost uvedu růžový šum a bílý šum, který ve své práci používám.



(a) Bílý šum (b) Růžový šum

Obrázek 2.1: Obrázky barav šumu převzaté z [3]

Na levém obrázku je vidět, že intenzita šumu s rostoucí frekvencí klesá, naproti tomu z pravém obrázku je patrné, že u bílého šumu je intenzita šumu nezávislá na frekveci.

2.2 Typy šumů

Při tvorbě této sekce jsem čerpal z [1].

Šum s rovnoměrně rozloženými hodnotami

Hodnoty tohoto šumu mají rovnoměrné rozložení, které se značí $R(a, b)$. Parametry a a b představují dolní a horní hranici oboru hodnot. Náhodná veličina nabývá hodnot rovnoměrně z intervalu $< a, b >$ tzn. že funkce hustoty pravděpodobnosti musí být na tomto intervalu konstantní nenulová a jinde nulová:

$$f(x) = \begin{cases} 0, & x < a \\ \frac{1}{b-a}, & a \leq x \leq b \\ 0, & x > b \end{cases}$$

Integrací této funkce získáme distribuční funkci rozložení, kterou využijeme při transformaci na požadovaný interval oboru hodnot:

$$F(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 0, & x > b \end{cases}$$

Charakteristiky rozložení:

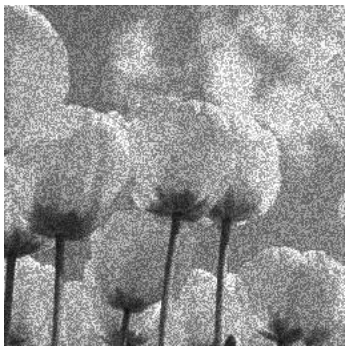
Střední hodnota:

$$E(x) = \frac{a+b}{2}$$

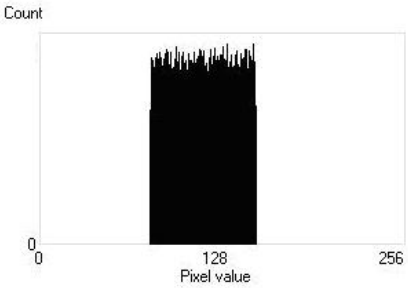
Rozptyl:

$$D(x) = \frac{(b-a)^2}{12}$$

Při použití tohoto šumu v obraze je postihnut každý pixel obrazu. Pro názornost je uveden obrázek 2.2.1 s přidáným šumem $R(a, b)$, kde $a = 109$ a $b = 183$. Vedle na obrázku 2.2.2 z [5] je jeho odpovídající historogram hodnot šumu.



(a) Obrázek s přidaným šumem $R(109, 183)$



(b) Historogram šumu

Obrázek 2.2: Šum s rovnoměrně rozloženými hodnotami

Šum s exponenciálně rozloženými hodnotami

Hodnoty tohoto šumu mají eponeciální rozložení, které se značí $Exp(\lambda)$. Je to nesymetrické rozložení.

Funkce hustoty rozložení pro $x \geq x_0$:

$$f(x) = \lambda e^{-\lambda x)}$$

Distribuční funkce:

$$F(x) = \begin{cases} 1 - e^{-\lambda x}, & x \geq x_0 \\ 0, & x \leq x_0 \end{cases}$$

Charakteristiky rozložení:

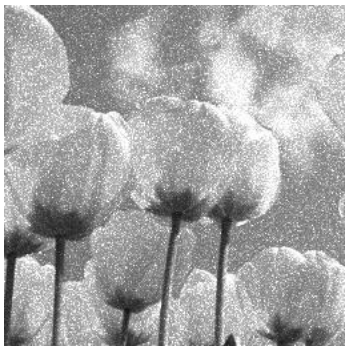
Střední hodnota:

$$E(X) = \frac{1}{\lambda}$$

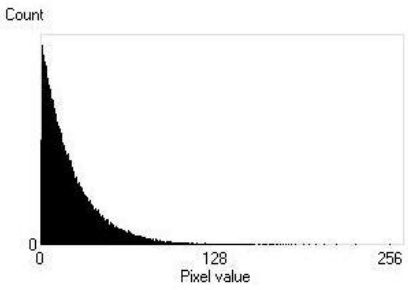
Rozptyl:

$$D(X) = \frac{1}{\lambda^2}$$

Při použití tohoto šumu v obraze je postihnut každý pixel obrazu. Pro názornost je uveden obrázek 2.3.1 s přidáným šumem $Exp(\lambda)$. Vedle na obrázku 2.3.2 z [5] je jeho odpovídající historogram hodnot šumu.



(a) Obrázek s přidaným šumem $Exp(\lambda)$



(b) Historogram šumu

Obrázek 2.3: Šum s exponenciálně rozloženými hodnotami

Gaussův šum

Hodnoty Gaussova šumu mají normální (někdy též Gaussovo) rozložení, které se značí $No(\mu, \sigma^2)$. Normální rozložení je vhodné tam, kde jsou změny náhodné veličiny způsobeny současným vlivem velkého počtu vzájemně nezávislých faktorů, které se uplatňují přibližně stejnou měrou. Bílý Gaussův šum v mnoha situacích velmi dobře aproximuje reálný šum.

Toto rozložení má funkci hustoty:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Charakteristiky rozložení:
Střední hodnota:

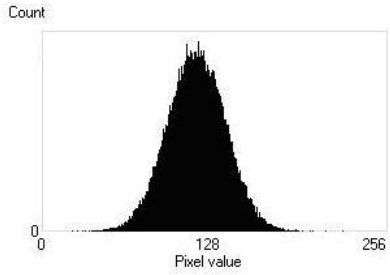
$$E(X) = \mu$$

Rozptyl:

$$D(X) = \sigma^2$$

Při použití tohoto šumu v obraze je postihnut každý pixel obrazu. Pro názornost je uveden obrázek 2.4.1 s přidáným šumem $No(\mu, \sigma^2)$. Kde $\mu = 0$ a $\sigma = 0.2$ .Vedle na obrázku 2.4.2 [5] je jeho odpovídající historogram hodnot šumu.

(a) Obrázek s přidaným šumem $No(\mu, \sigma^2)$



(b) Historogram šumu

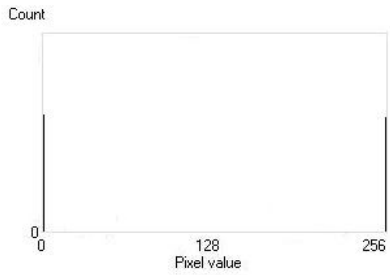
Obrázek 2.4: Gaussův šum

Šum typu pepř a sůl

Tento šum nevychází ze žádného pravděpodobnostního rozložení. Jedná se o takzvaný výstřelový šum, kdy jsou některé hodnoty nastaveny na maximální hodnotu, některé na minimální hodnotu a ostatní zůstavají beze změny.

Při použití tohoto šumu v obraze jsou postihnuty pouze některé pixely obrazu. Pro názornost je uveden obrázek 2.5.1 s přidáným šumem. Vedle na obrázku 2.5.2 [5] je jeho odpovídající historogram hodnot šumu.

(a) Obrázek s přidaným šumem typu pepř a sůl



(b) Historogram šumu

Obrázek 2.5: Šum typu pepř a sůl

2.3 Generátor pseudonáhodných čísel

Základem generátoru šumu je vhodný generátor čísel. Použití fyzikálních generátorů, které vytvářejí náhodná čísla na základě fyzikálních náhodných procesů v přídavném zařízení počítače, je nákladné a spojené s mnoha technickými obtížemi. Používáme proto generátorů pseudonáhodných čísel, které můžeme realizovat programově. Na posloupnost pseudonáhodných čísel klademe tyto základní požadavky: rovnoměrné rozložení, statistickou nezávislost, reprodukovatelnost, rychlé generování, minimální obsazení paměti počítače příslušným programem. Každá posloupnost pseudonáhodných čísel má vždy určitou konečnou periodu

a naší snahou je, aby tato perioda byla co nejdelší. Velkou výhodou tohoto způsobu je možnost reprodukování posloupnosti náhodných čísel, poněvadž je vytváříme podle jednoznačných aritmetických operací. Pseudonáhodná čísla můžeme vytvářet různými přemisťovacími metodami, které využívají speciálních instrukcí počítačů jako jsou posuny, binární součty bez přenosu a podobně. Statistické vlastnosti posloupnosti náhodných čísel se zpravidla zlepší, použijeme-li více nezávislých zdrojů náhodných čísel, které deterministicky nebo náhodně střídáme.

Při tvorbě této sekce jsem čerpal z [1].

2.4 Kongruentní metoda generování pseudonáhodných čísel

V lineárním kongruentním generátoru se posloupnost pseudonáhodných čísel získává na základě vztahu:

$$n + 1 = (aX_n + c)\ mod\ m$$

kde a, X_0, c, m jsou přirozená čísla.
X_0 . . . počáteční hodnota
a . . . multiplikativní konstanta
c . . . aditivní konstanta
m . . . modul

Tato čísla nejsou samozřejmě libovolná. Na jejich výběru závisí perioda i rozložení posloupnosti generovaných pseudonáhodných čísel.

Výběr modulu m

Při výběru modulu m se snažíme splnit tyto požadavky:

1. co nejdelší periodu posloupnosti
2. co největší rychlost generování pseudonáhodných čísel.

Prvnímu požadavku můžeme vyhovět výběrem co největšího modulu m. Položíme tedy m rovno maximálnímu číslu typu integer (používáme pro ně označení $maxint$). Zvolíme-li však $m = maxint$, pak se dá výraz $(aX_n + c)\ \ mod\ \ m$ nahradit výrazem $(aX_n + c) + m + 1$. Vyčíslování tohoto výrazu bude podstatně rychlejší.

Výběr konstant a a c

Předpokládáme, že modul m je již zvolen podle předešlého bodu. Konstanty a a c pak vybíráme tak, aby perioda byla co největší, tj. právě rovna modulu m. Tento požadavek lze splnit za podmínek:

1. c a m jsou navzájem nesoudělná čísla
2. $b = a - 1$ je dělitelné číslem p pro libovolné p, jež je dělitelem čísla m
3. b je dělitelné čtyřmi, jestliže m je dělitelné čtyřmi.

V zájmu urychlení generování lze volit $c = 0$, takovou volbou se však zpravidla zkrátí perioda. Nicméně i tak je možno dosáhnout vyhovujících výsledků

Návrh konstanty a se obvykle provádí spektrálním testem, jenž je popsán v literatuře. Osvědčená je multiplikativní konstanta získaná z firemní literatury IBM: $a = 1220703125$.

Výběr počáteční hodnoty X_o

Výběr počáteční hodnoty X_o je nelehkým úkolem, protože pro tento výběr známe nejméně pravidel, musíme jej provádět téměř empiricky. X_o by mělo být dostatečně velké liché číslo, nemělo by být soudělné s modulem m. V mnou použitém generátoru je zvoleno $X_o = 1537$.

2.5 Transformace rovnoměrného rozložení na požadovaný typ rozložení

Potřebujeme-li posloupnosti náhodných čísel s jiným, než rovnoměrným rozložením pravděpodobnosti, musíme použít transformační metodu, pomocí které dostaneme požadované rozložení transformací rovnoměrného rozložení. Transformačních metod existuje nekolik druhů. Při výběru vhodné metody transformace jde zpravidla o kompromis mezi její efektivností, požadavky na paměť a dosahovanými výsledky. Nejčastěji používanými metodami pro transformování náhodných čísel na dané rozložení pravděpodobnosti jsou:

- metoda inverzní transformace
- metoda vylučovací
- metoda kompoziční

Metoda vylučovací

Nechť $f : (x_1, x_2) \rightarrow (0, M)$ je požadovaná funkce hustoty generované náhodné veličiny, $X :\rightarrow (x1, x2)$ je náhodná veličina s rozložením $R(x1, x2)$ $Y :\rightarrow (0, M)$ je náhodná veličina s rozložením $R(0, M)$ Potom pro každé $t \in (x_1, x_2)$ platí:

$$p(X < t | Y < f(X)) = \int_{x_1}^{t} f(x)\, dx$$

za předpokladu, že náhodné veličiny X, Y jsou nezávislé.
Náhodnou veličinu ξ s funkcí hustoty $f : (x1, x2) \rightarrow (0, M)$ budeme generovat takto:

1. Generujeme číslo x z rozložení $R(x1, x2)$.
2. Generujeme číslo y z rozložení $R(O, M)$.
3. Jestliže $y < f(x)$, pak x prohlásíme za hodnotu náhodné veličiny ξ , jinak opakujeme celý postup znovu.

Efektivnost této metody je dána poměrem plochy S a plochy obdélníka $(x2-x1).M$, kde plocha S je plocha ohraničená osou x a funkcí hustoty $f(x)$. Jelikož většina funkcí hustoty, popisujících dané rozložení, nemá zprava, zleva nebo zprava i zleva ohraničený definiční obor. Efektivnost metody totiž se zmenšováním plochy S k ploše obdélníka klesá, proto zde volíme kompromis mezi časovou náročností transformace a šířkou intervalu $(x1, x2)$.

Metoda inverzní transformace

Nechť náhodná veličina R má rovnoměrné rozložení $R(0,1)$ a F je funkce, splňující podmínky pro distribuční funkci. Potom platí:

Náhodná veličina $F^{-1} \bullet R$ má rozložení s distribuční funkcí F, neboť

$$p((F^{-1} \bullet R) < x) = p(F \bullet (F^{-1} \bullet R) < F(x)) = p(R < F(x)) = F(x)$$

Takto můžeme přesně generovat náhodná čísla s daným rozložením. V případě, že distribuční funkce daného rozložení nemá inverzní funkci (např. když distribuční funkci nelze vyjádřit elementárními funkcemi), pak při nevhodnosti jiných metod můžeme použít aproximace této funkce jinou vhodnou funkcí, jejíž inverzní funkce je známá. Taková transformace již patří k přibližným metodám.

2.6 Perlinův šum

Perlin šumovou funkci navrhl už v roce 1983. V roce 1985 o její aplikaci vznikl článek prezentovaný na SIGGRAPHu (jedna z nejvýznamnějších konferencí počítačové grafiky) a v letech 1986 až 1988 tuto funkci s některými modifikacemi používaly takové firmy, jako Pixar, Alias, SoftImage apod. Pro vytváření textur s různými přírodními motivy jako jsou například dřevo, mraky či mramor nelze použít základní matematické funkce nebo generátory pseudonáhodných čísel. Ken Perlin navrhl výpočet pro vytváření přírodních textur, využívající generátor pseudonáhodných čísel tak, že jsou jednotlivé náhodné hodnoty interpolovány, čímž se výsledný průběh funkce vyhladí, tudíž se už nejedná o zcela náhodný šum. Tento vypočtený šum není příliš zájímavý, sečteme-li však více takovýchto šumů s různou frekvencí a amplitudou, dostaneme Perlinův šum [10, 8, 7, 6] jak jej Ken Perlin nazval. Přičemž pro amplitudu a frekvenci platí, že čím vyšší je frekvence šumové funkce, tím nižší je její aplituda.

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

Kde n je počet oktáv(příspěvků součtu) a $p \in (0,1)$ je tzv. persistence, která určuje rychlost klesání vlivu každé oktávy na výsledný součet. To zajišťuje amplituda a_i i-t0 oktávy:

$$a_i = p^i$$

Frekvence f_i i-té oktávy se vypočítá podle vztahu:

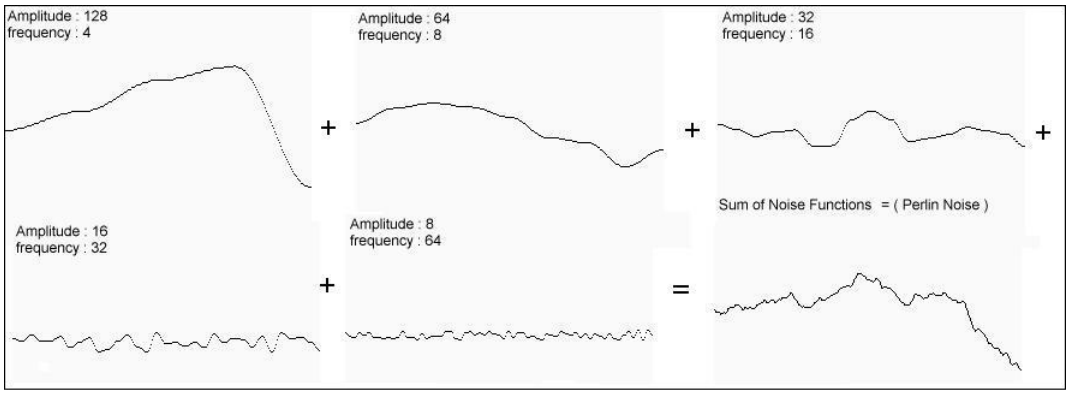
$$f_i = 2^i$$

K interpolaci hodnot je na výběr několik metod lišící se rychostí a kvalitou. Vybral jsem poměrně často používanou bilineární intepolaci, která dosahuje kvality výsledků téměř shodné s výsledky bikubické interpolace, je však rychlejší. Volba metody je tedy vždy věcí kompromisu.

Výslednou hodnotu získáme podle [2] pro systém čtyř bodů $(0,0), (0,1), (1,1), (1,0)$ takto:

$$f(x,y) \approx f(0,0)(1-x)(1-y) + f(1,0)x(1-y) + f(0,1)(1-x)y + f(1,1)xy$$

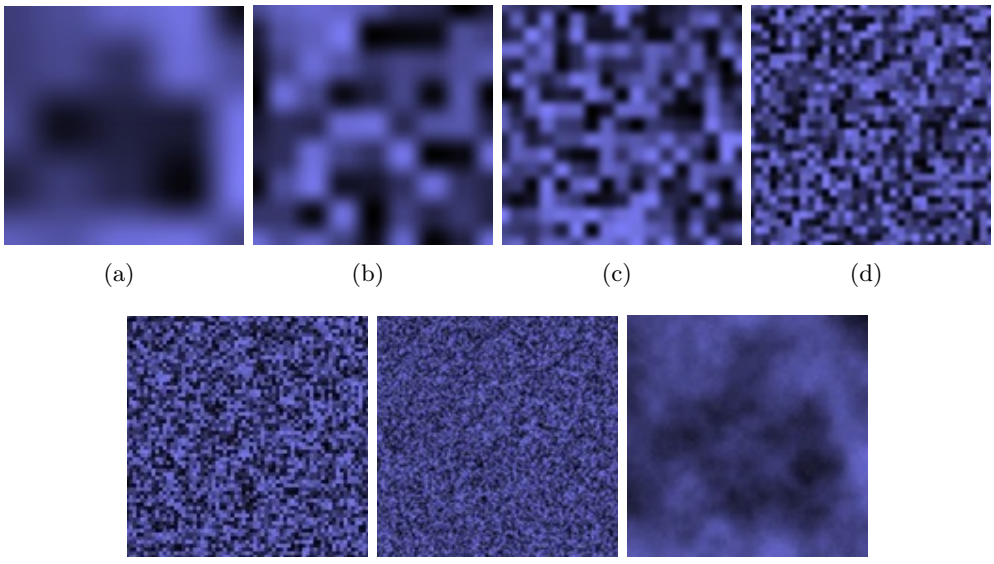
Součet jednodimenzionálních šumových průběhů s různou frekvencí a amplitudou:



(a)

Obrázek 2.6: Princip Perlinova šumu 1D

Součet dvoudimenzionálních šumových průběhů s frekvencí a amplitudou takovou, že na každém obrázku 2.7a až f má šumový průběh dvakrát větší frekvenci a do výsledného součtu přispívá s poloviční amplitudou, než je tomu u předešlého obrázku. Výsledný Perlinuv šum je na obrázku 2.7g. Obrázky jsou převzaty z [8]



(a) (b) (c) (d)

(e) (f) (g)

Obrázek 2.7: Princip Perlinova šumu 2D

Kapitola 3

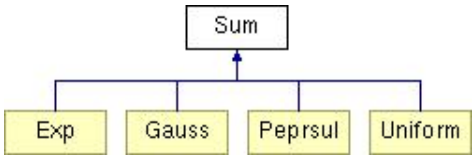
Návrh řešení

3.1 Knihovna pro generování šumu

Jedním z cílů této práce je implementovat knihovnu pro generování základních typů šumů, kterou budu dále v práci využívat. Tato knihovna by měla být univerzální, tedy schopná generovat náhodné hodnoty s různou maximální hodnotou. Dále je kladen důraz na efektivnost jednotlivých metod a v neposlední řadě jednotnost při práci s různými generátory.

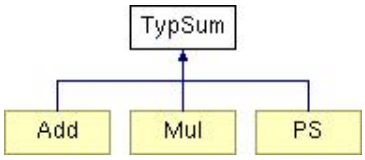
Rozhodl jsem se implementovat tyto čtyři druhy šumů: šum s rovnoměrně rozloženými hodnotami, šum s exponenciálně rozloženými hodnotami, Gaussův šum a šum typu Pepř a sůl, přičemž první tři druhy šumů mohou být přidávány aditivně nebo multiplikativně. Při výpočtu náhodných hodnot se vychází kongruentní generátoru pseudonáhodných čísel, který má rovnoměrné rozložení. To lze některou z transformačních metod změnit na požadované rozložení. Pro první dva druhy šumu je nejefektivnější metoda inverzní transformace. Z důvodu nemožnosti vytvoření inverzní funkce k funkci distribuční tohoto rozložení, je použita metoda vylučovací, jejíž efektivita však zavisí na parametrech rozložení. U typu šumu Pepř a sůl bylo více možností jak tohoto šumu dosáhnout, vybral jsem způsob, který zachovává jednotný přístup ke generátorům šumů na úkor výkonu generování.

Programovací jazyk pro tuto práci jsem si zvolil C++, tudíž se jako nejlepší řešení nabízí aplikovat jednotlivé typy šumů do malých tříd s využitím dědičnosti a polymorfizmu. Vztahy mezi třídami knihovny jsou zachyceny na následujících obrázcích.



Obrázek 3.1: Strom dědičnosti třídy Sum

Kde abstraktní třída Sum je děděná třídami různých druhů šumu, jejichž parametry jsou předány v konstruktoru. Třída Sum obsahuje pseudonahodný generátor čísel a čistě virtuální metodu pro generování daného typu šumu, implementovanou v jednotlivých třídách.



Obrázek 3.2: Strom dědičnosti třídy TypSum

Kde abstraktní třída TypSum, děděná třídami určujících způsob přidaní druhu šumu předaného v konstruktoru. Třída TypSumobsahuje čistě virtuální metodu pro přidání šumu do dané proměnné, implementovanou v jednotlivých třídách. Třída PS je určena pouze pro přídání šumu typu Pepř a sůl.

Knihovna dále obsahuje třídu pro práci se standartním dynamickým jednorozměrným polem s možností načtení či uložení obrázkového BMP souboru.

3.2 Knihovna pro generování Perlinova šumu

Tato knihovna umožňuje generovat jak dvoudimenzionální tak i třídimenzionální Perlinův šum. Je přímo zavislá na přítomnosti mnou vytvořené knihovny pro generování šumů, protože využívá její metody pro práci s dynamickým polem. Některé tyto metody, zvláště pak pro práci s trojrozměrným polem, byly přídány pouze za účelem spolupráce s knihovnou pro generování Perlinova šumu. Již zmíněné pole využivané při veškere práci s Perlinovým šumem si musí uživatel nadefinovat sám. Je tu však metoda Velikost, uvedená v programové dokumentaci, která vypočítá potřebnou velikost tohoto dynamického pole podle zadaných parametrů.

Kapitola 4

Implementace

Zdrojové soubory knihovny pro generování šumu se jmenují lsum.cpp a lsum.h. Zdrojové soubory knihovny pro generování Perlinova šumu se jmenují lperlin.cpp a lperlin.h.

Program demonstrující třídimenzionální Perlinův šum se jmenuje krychle.exe, jeho zdrojový soubor krychle.cpp.

Ovládání programu:

- klávesa f - program běží v celoobrazovkovém režimu
- klávesa w - program běží v okně
- klávesa a - zastavení procházení 3D texturou
- klávesa s - obnovení procházení 3D texturou
- myš - rotace kostky

Program demonstrující vytvoření terénu krajiny z vyškové mapy generované Perlinovým šumem se jmenuje teren.exe, jeho zdrojový soubor teren.cpp.

Ovládání programu:

- klávesa f - program běží v celoobrazovkovém režimu
- klávesa w - program běží v okně
- klávesa a - zvyšování parametru σ Gausova rozložení použitého při tvorbě výškové mapy
- klávesa s - snižování parametru σ Gausova rozložení použitého při tvorbě výškové mapy

Kapitola 5

Výsledky

5.1 Generátory šumů

Zde uvedu výsledky činností jednotlivých generátorů šumu způsob jak jich lze pomocí mnou vytvořené knihovny dosáhnout.

Generování Gaussova šumu

Gaussuv šum s parametry $\sigma = 0.5$, $\mu = 0$ se dá vytvořit zcela jednoduše, což je předvedeno v následujícím kódu. Přitom je na výběr jakým způsobem je vygenerovaný šum sloučen s původními daty, tedy additivně nebo multiplikativně.

Nejprve se tedy vybere druh šumu, zde Gaussuv šum.

```
Sum * GaussuvSum = new Gauss(0.5,0.0);
```

Následně se musí určit způsob přidání šumu tedy Addivně,

```
TypSum * AddivniSum = new Add(GaussuvSum, 255.0);
```

či multiplikativne.

```
TypSum * MultiplikativniSum = new Add(GaussuvSum, 255.0);
```

Pseudonáhodnou hodnotu z Gaussova rozložení získáme zavoláním metody GenSum() pro additvni šum takto:

```
float gs = AddivniSum->PridejSum(PuvodniHodnota);
```

nebo pro multiplkativní šum následovně:

```
float gs = MultiplikativniSum->PridejSum(PuvodniHodnota);
```

Originální obrázek 4.1.1 je nepostihnutý žádným šumem, případné zkreslení je způsobeno ztrátovou kompresí zvoleného formátů obrázků. Na obrázku 4.1.2 jsou ovlivněny aditivním Gaussovým šumem s parametry $\sigma = 0.3$, $\mu = 0$ všechny složky RGB barvy pixelu.

(a) Originální obrázek

(b) Obrázek postihnutý aditivním šumem

Na obrázku 4.1.3 jsou ovlivněny multiplikativním Gaussovým šumem s parametry $\sigma = 0.3$, $\mu = 0$ všechny složky RGB barvy pixelu. Zatímco na obrázku 4.1.4 je použit šum se stejnými parametry, ale pouze na B složky RGB barev pixelů.

(c) Obrázek postihnutý multiplikativním šumem

(d) Modrá složka barvy obrázku postihnutá aditivním šumem

Obrázek 5.1: Ukázka Gaussova šumu

Generování šumu s rovnoměrně rozloženými hodnotami

Šum s rovnoměrným rozložením $R(a, b)$, kde parametry $a = 0.3$ a $b = 0.8$ lze vytvořit pomocí následujícího kódu. Přitom je na výběr jakým způsobem je vygenerovaný šum sloučen s původními daty, tedy additivně nebo multiplikativně. Princip je stejný jako u předchozích typů šumů, pro zvýšení přehlednosti je však způsob generování zopakován.

Nejprve se tedy vybere druh šumu, zde šum s rovnoměrným rozložením.

```
Sum * UniformSum = new Uniform(0.3, 0.8);
```

Následně se musí určit způsob přidání šumu tedy Addivně,

```
TypSum * AddivniSum = new Add(UniformSum, 255.0);
```

či multiplikativne.

```
TypSum * MultiplikativniSum = new Add(UniformSum, 255.0);
```

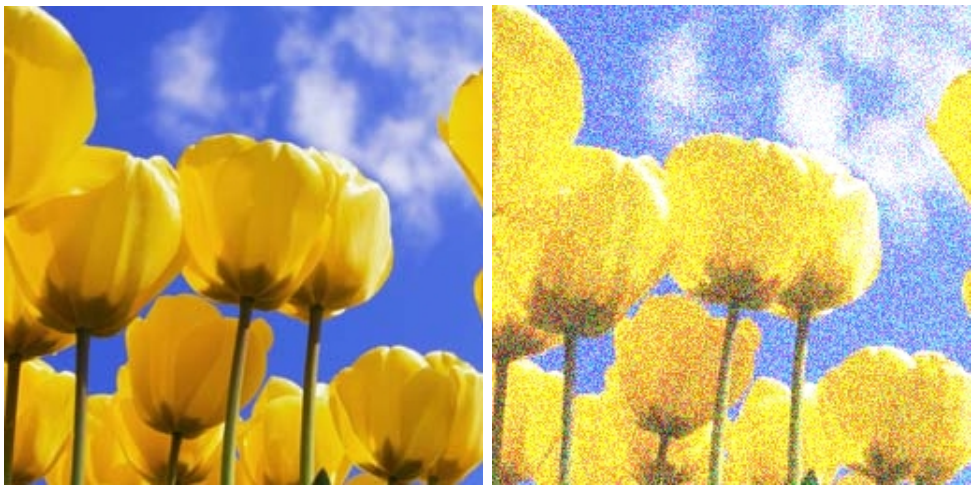
Pseudonáhodnou hodnotu z Gaussova rozložení získáme zavoláním metody GenSum() pro additvni šum takto:

```
float gs = AddivniSum->PridejSum(PuvodniHodnota);
```

nebo pro multiplkativní šum následovně:

```
float gs = MultiplikativniSum->PridejSum(PuvodniHodnota);
```

Originální obrázek 4.2.1 je nepostihnutý žádným šumem, případné zkreslení je způsobeno ztrátovou kompresí zvoleného formátů obrázků. Na obrázku 4.2.2 jsou ovlivněny s rovnoměrným rozložením šumem s parametry $a = 0.3$ a $b = 0.8$ všechny složky RGB barvy pixelu.



(a) Originální obrázek (b) Obrázek postihnutý aditivním šumem

Na obrázku 4.2.3 jsou ovlivněny multiplikativním šumem s rovnoměrným rozložením s parametry $\sigma = 0.3$, $\mu = 0$ všechny složky RGB barvy pixelu. Zatímco na obrázku 4.2.4 je použit šum se stejnými parametry, ale pouze na G složky RGB barev pixelů.

(c) Obrázek postihnutý multiplikativním šumem

(d) Zelená složka barvy obrázku postihnutá aditivním šumem

Obrázek 5.2: Ukázka šumu s rovnoměrným rozložením

Generování šumu s exponenciálně rozloženými hodnotami

Šum s exponenciálním rozložením $Exp(\lambda)$, kde parametrem $lambda = 5.0$ lze vytvořit pomocí následujícího kódu. Přitom je na výběr jakým způsobem je vygenerovaný šum sloučen s původními daty, tedy additivně či multiplikativně. Princip je stejný jako u předchozích typů šumů, pro zvýšení přehlednosti je však způsob generování zopakován.

Nejprve se tedy vybere druh šumu, zde šum s exponenciálním rozložením.

```
Sum * ExpSum = new Exp(5.0);
```

Následně se musí určit způsob přidání šumu tedy Addivně,

```
TypSum * AddivniSum = new Add(ExpSum, 255.0);
```

či multiplikativne.

```
TypSum * MultiplikativniSum = new Add(ExpSum, 255.0);
```

Pseudonáhodnou hodnotu z tohoto rozložení získáme zavoláním metody GenSum() pro additvni šum takto:

```
float gs = AddivniSum->PridejSum(PuvodniHodnota);
```

nebo pro multiplkativní šum následovně:

```
float gs = MultiplikativniSum->PridejSum(PuvodniHodnota);
```

Originální obrázek 4.3.1 je nepostihnutý žádným šumem, případné zkreslení je způsobeno ztrátovou kompresí zvoleného formátů obrázků. Obrázek 4.3.2 jsou ovlivněny s exponenciálním rozložením šumem s parametrem $\lambda = 5.0$ všchny složky RGB barvy pixelu.

(a) Originální obrázek

(b) Obrázek postihnutý aditivním šumem

Na obrázku 4.3.3 jsou ovlivněny multiplikativním šumem s exponenciálním rozložením s parametrem $\lambda = 5.0$ všechny složky RGB barvy pixelu. Zatím co v obrázku 4.3.4 je použit šum se stejnými parametry, ale pouze na R složky RGB barev pixelů.

(c) Obrázek postihnutý multiplikativním šumem

(d) Červená složka barvy obrázku postihnutá aditivním šumem

Obrázek 5.3: Ukázka šumu s exponenciálním rozložením

Generování šumu typu Pepř a sůl

Šum typu Pepř a sůl s parametrem udvající počet procent postihnutých šumem (v tomto případě 8procent) lze vytvořit pomocí následujícího kódu. Tento typ šumu nelze přidávat do dat aditivně ani multiplikativně.

Nejprve se tedy vybere druh šumu, zde šum typu pepř a sůl.

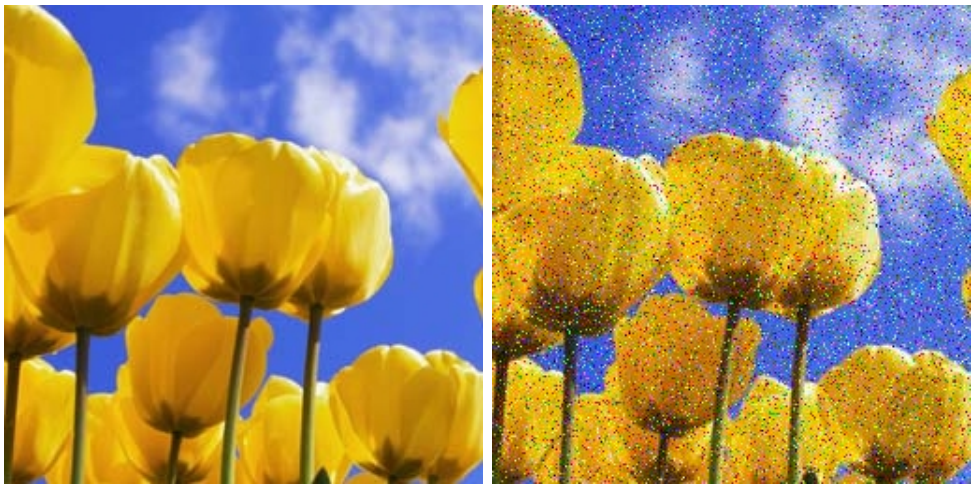
```
Sum * PeprSulSumGen = new Peprsul(8.0, 255.0);
```

```
TypSum * PeprSulSum = new PS(PeprSulSumGen, 255.0);
```

Výslednou hodnotu šumu typu Pepř a sůl získáme zavoláním metody GenSum() takto:

```
float gs = PeprSulSum->PridejSum(PuvodniHodnota);
```

Originální obrázek 4.4.1 je nepostihnutý žádným šumem, případné zkreslení je způsobeno ztrátovou kompresí zvoleného formátů obrázků. Obrázek 4.4.2 je postižen šumem typu Pepř a sůl s parametrem PocProc = 8.0 udávajícím počet procent postižených pixelů.



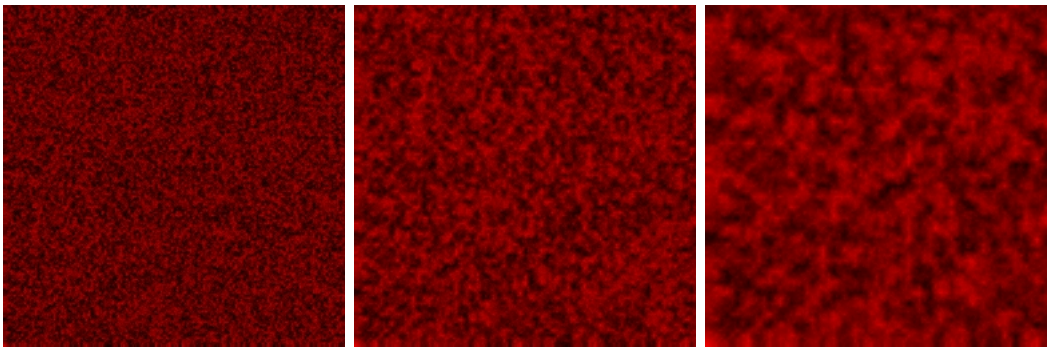
(a) Originální obrázek (b) Obrázek postihnutý šumem

Obrázek 5.4: Ukázka šumu typu Pepř a sůl

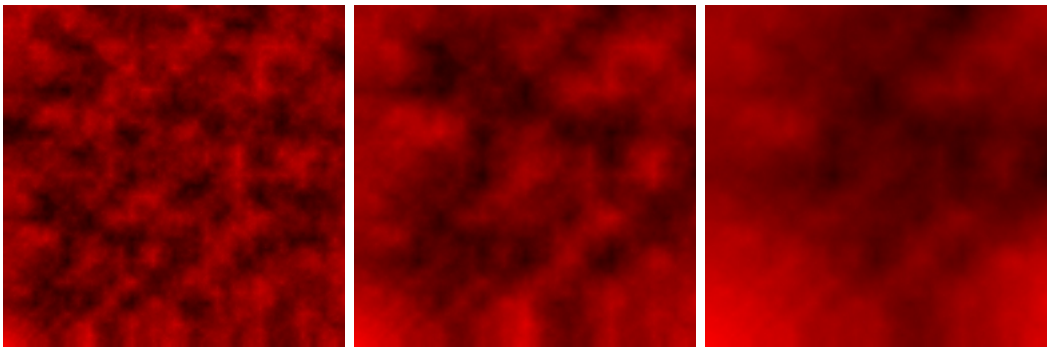
5.2 Generování Perlinova šumu

V následujících sekcích jsou uvedeny různé způsoby použití metod pro výpočet Perlinova šumu při tvorbě procedurálních textur a terénu krajiny. Nejsou uvedeny zcela jistě všechny, neboť využití Perlinova šumu pro generování textur je takřka neomezené.

Na následujících obrázcích je vidět výsledná činnost metody pro výpočet dvou dimenzionálního Perlinova šumu pro počet oktáv 2 až 7 včetně, přičemž je výsledný šum ukládán jen do R složek RGB barvy a ostatní složky jsou nulové. Použitý generátor $R(0, 1)$.



(a) Počet oktáv = 2 (b) Počet oktáv = 3 (c) Počet oktáv = 4



(d) Počet oktáv = 5 (e) Počet oktáv = 6 (f) Počet oktáv = 7

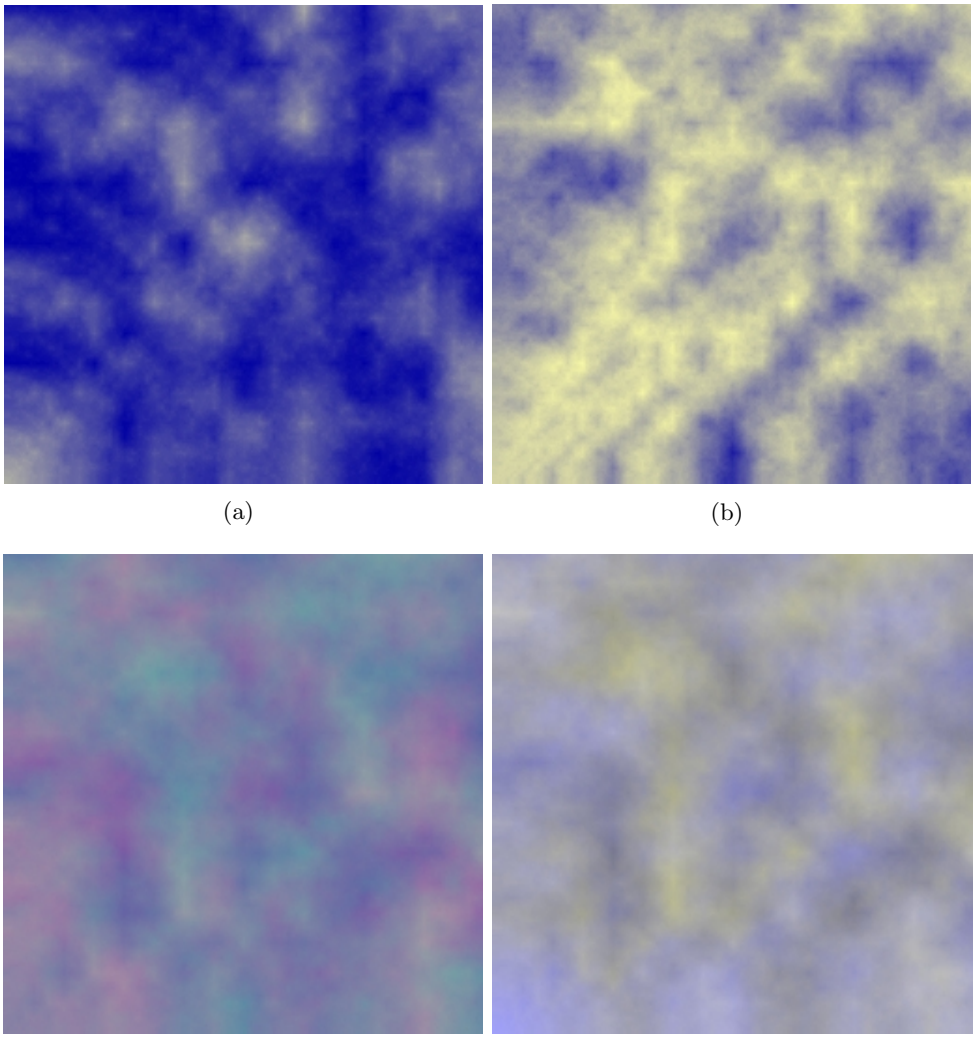
Obrázek 5.5: Perlinův šum s různým počtem oktáv

5.3 2D textura oblohy

Při tvorbě dvoudimenzionálních textur oblohy jsem použil konstantní nastavení parametrů metody generující Perlinuv šum. Změny vzhledu textůr oblohy, tedy množství mraků jsem docílil použitím různých generátorů šumu s různými parametry, jak je uvedeno dále. Tří a vícebarevné oblohy jsem dosáhl generováním Perlinova šumu pro každou složku RGB barvy zvlašť, případně jsem ponechal určité složce barvy konstantní hodnotu.

Pro vytvoření těchto obrázků jsem zvolil Perlinuv šum pro 6 oktáv a byly použity různé parametry uvedené v tabulce.

Obrázek	**Rozložení generátoru**	**Dosažená RGB barva**		
		R	**G**	**B**
a	$No(\sigma = 0.9, \mu = 0)$	P.šum č.1	P.šum č.1	165
b	$Exp(\lambda = 1)$	P.šum č.1	P.šum č.1	165
c	$R(0.3, 0.8)$	P.šum č.1	P.šum č.2	165
d	$R(0.2, 0.7)$	P.šum č.2 - 20	P.šum č.2 - 20	P.šum č.1



(a) (b)

(c) (d)

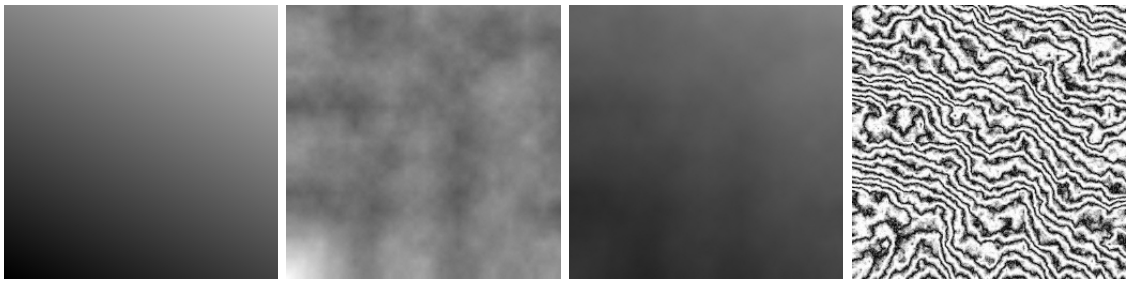
Obrázek 5.6: Vygenerované textury oblohy

5.4 2D a 3D textura mramoru

Velmi často se Perlinův šum používá v kombinaci s některými matematickými funkcemi, tak tomu je i u algoritmu pro generování textury mramoru uvedený na [7]. Jednotlivé kroky algoritmu jsou jsou znázorněny v kolekci obrázků 4.7. Vypočet textury mramoru se provádí následovně:

1. Vypočítame hodnoty jednotlivých pixelů, tak aby jejich hodnota byla lineárně závislá na součtu souřadnic právě počítaného pixelu. Odpovídající obrazek: 4.7a
2. Vygenerujeme bežným způsobem Perlinův šum. Odpovídající obrazek: 4.7b
3. Ve vhodném poměru sečteme obě hodnoty na odpovídajících si pozicích pro každý pixel. Odpovídající obrazek: 4.7c

4. Pro každou hodnotu provedeme funkci sinus a toto číslo v absolutní hodnotě poté vynásobíme maximální hodnotou pixelu. Odpovídající obrazek: 4.7d

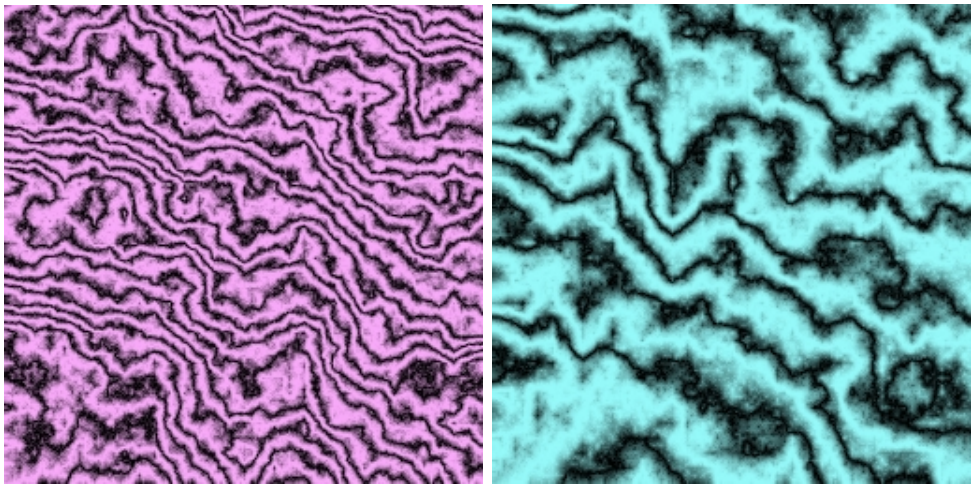


(a) (b) (c) (d)

Obrázek 5.7: Demonstrace vytvoření 2D texury mramoru

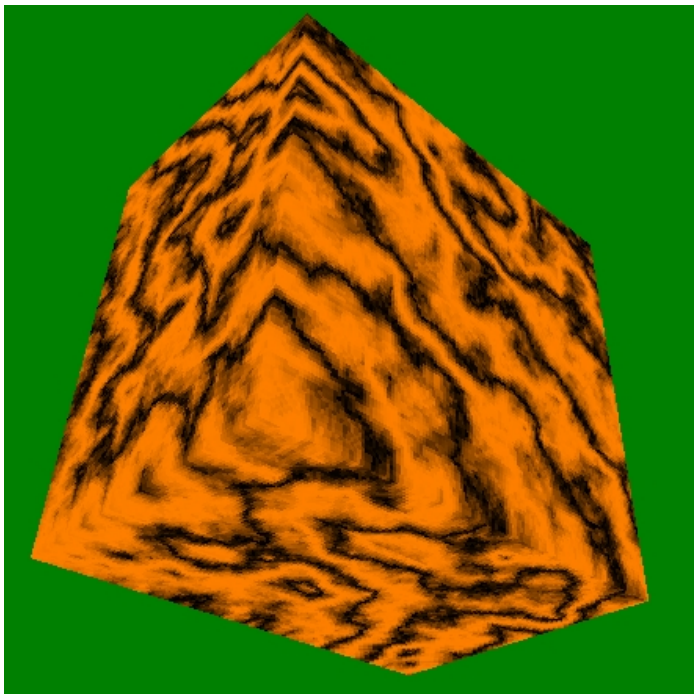
Tento algoritmus je implementován do metod Mramor a Mramor3D, které se nachází v knihovně pro výpočet Perlinova šumu. Výslednou texturu, vygenerovanou touto metodou, mužeme ovlivnit typem vybraného šumu a parametry udavající počet oktáv Perlinova šumu, barvu textury zadávanou ve tvaru R,G,B a příblížení.

Obrázek	**Rozložení generátoru**	**Přibližení**	**Počet oktáv**	**RGB barva**		
				R	**G**	**B**
a	$R(0.2, 0.9)$	6.0	6	250	170	250
b	$No(\sigma = 0.6, \mu = 0.0)$	2.0	6	150	250	250



(a) (b)

Obrázek 5.8: Ukázka vygenerovaných 2D textur mramoru

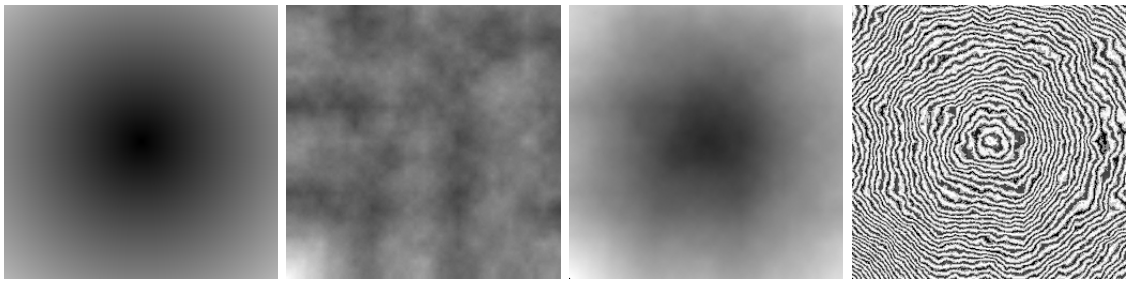


Obrázek 5.9: Ukázka vygenerované 3D textury mramoru

5.5 2D textura průřezu dřeva

Algoritmus pro generování textury dřeva, uvedený na [7], je podobný algoritmu pro generování textury mramoru, je však použita jiná matematická funkce při kombinaci s Perlinovým šumem. Jednotlivé kroky algoritmu jsou jsou znázorněny v kolekci obrázků 4.8. Vypočet textury dřeva se provádí následovně:

1. Vypočítame hodnoty jednotlivých pixelů, tak aby jejich hodnota byla lineárně závislá na součtu souřadnic právě počítaného pixelu. Odpovídající obrazek: 4.8a
2. Vygenerujeme bežným způsobem Perlinův šum. Odpovídající obrazek: 4.8b
3. Ve vhodném poměru sečteme obě hodnoty na odpovídajících si pozicích pro každý pixel. Odpovídající obrazek: 4.8c
4. Pro každou hodnotu provedeme funkci sinus a toto číslo v absolutní hodnotě poté vynásobíme maximální hodnotou pixelu. Odpovídající obrazek: 4.8d

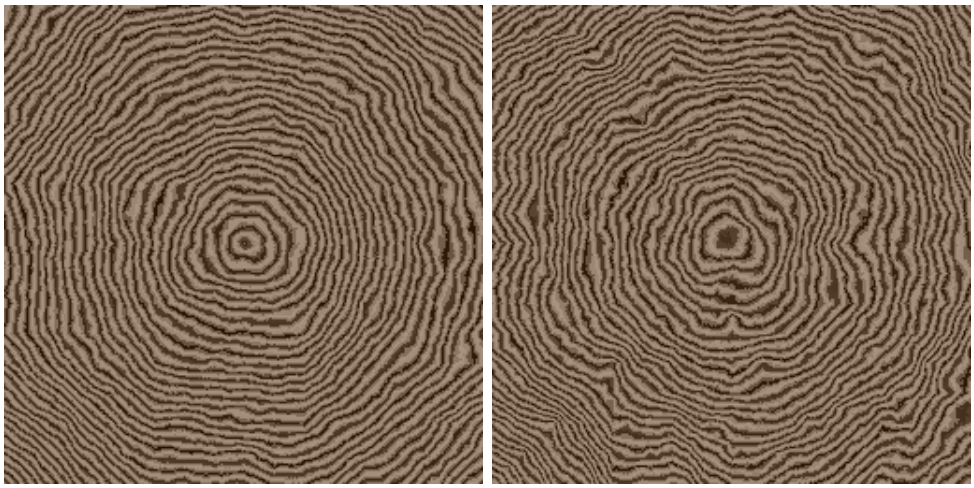


(a) (b) (c) (d)

Obrázek 5.10: Demonstrace vytvoření texury mramoru

Tento algoritmus je implementován do metody Drevo, která se nachází v knihovně pro výpočet Perlinova šumu. Výslednou texturu, vygenerovanou touto metodou, mužeme ovlivnit typem vybraného šumu, u této metody je však vhodný pouze generátor normálního a rovnoměrného rozložení. Vyslednou podobu určíme také parametry udavající počet oktáv Perlinova šumu a barvu textury zadávanou ve tvaru R,G,B.

Obrázek	**Rozložení generátoru**	**Počet oktáv**	**RGB barva**		
			R	**G**	**B**
a	$R(0.2, 0.7)$	6	140	120	100
b	$No(\sigma = 0.3, \mu = 0.0)$	6	140	120	100



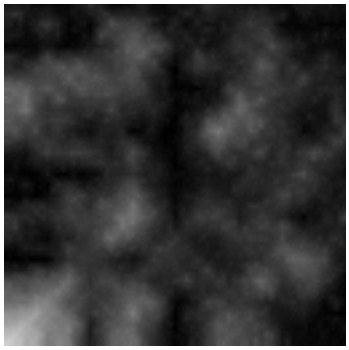
(a) (b)

Obrázek 5.11: Ukázka vygenerovaných texur průřezů dřeva

5.6 Generování výškové mapy pro tvorbu terénu krajiny

Výšková mapa slouží jako vstupní data pro většinu algoritmů určených pro zobrazení terénu. Hodnotami vyškové mapy jsou barvy šedi, kde černá barva jsou nejnižší položená místa terénu a bílá barva jsou nejvyšší místa terénu. Jedna z možností jak takovouto výškovou mapu vytvořit, je vygenerovat ji pomocí Perlinova šumu.

Zdrojem pseudonáhodných hodnot pro Perlinův šum jsem zvolil $No(\sigma = 0.3, \mu = 0.0)$ a poté běžným způsobem vygeneroval následující výškovou mapu zobrazenou na obrázku 4.11a.

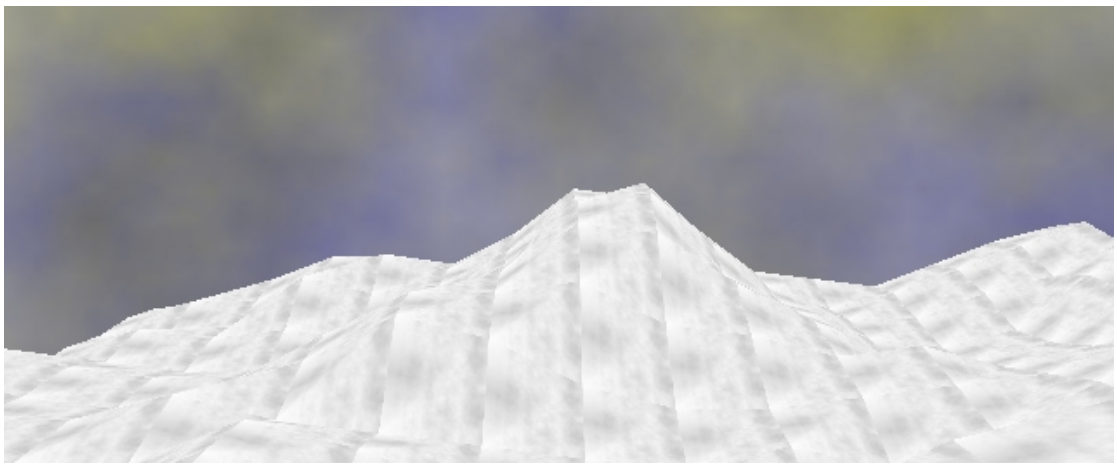


Obrázek 5.12: Ukázka vygenerované výškové mapy

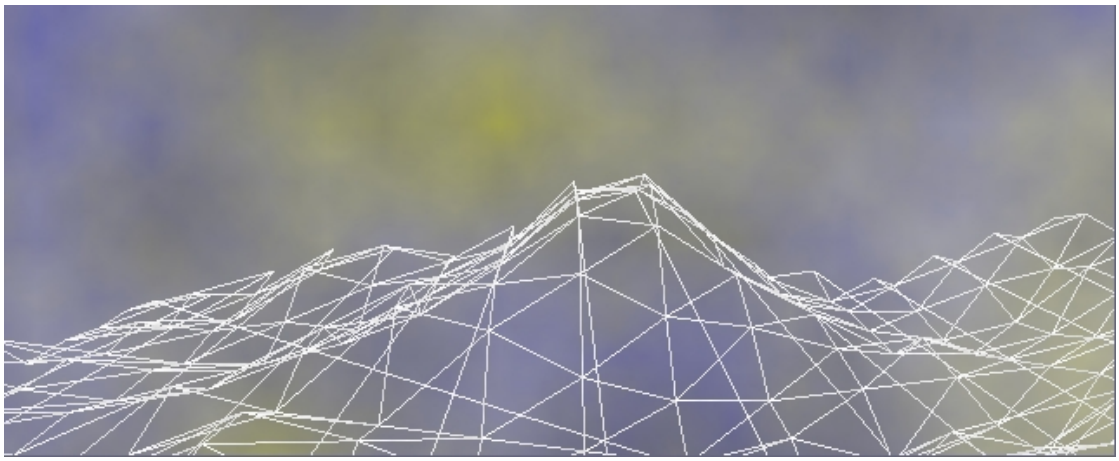
Vytvořil jsem jednoduchou aplikaci, která vytvoří terén krajiny z takto vygenerované výškové mapy pomocí dále uvedeného algoritmu z [9].

Hodnota na ose x odpovídá x-ové souřadnici výškové mapy a na ose z y-ové. Získali jsme umístění bodu na rovině, potřebujeme ho ještě vyzdvihnout do výšky, které v OpenGL odpovídá osa y. Tato výška je definována hodnotou uloženou na daném prvku pole. Přitom nevyužívám všechny hodnoty výškové mapy, ale prochazím ji po určitém kroku, sice pak nebude výsledný terén tak hladký a přesný, ale díky menšímu počtu polygonů se rendering urychlí.

Na obrázku 4.12a je síťový model terénu, vytvořený uvedeným způsobem a na obrázku 4.12b je tento síťový model potexturován.



Obrázek 5.13: Ukázka vygenerovaného terénu



Obrázek 5.14: Ukázka drátového modelu vygenerovaného terénu

Kapitola 6

Závěr

Tato bakalářská práce se zabývá generováním různých typů šumů a jejich aplikace v počítačové grafice. Generátory těchto šumů jsem aplikoval do knihovny, která nalezne díky své univerzálnosti mnoho využití. S ohledem na použití v počítačové grafice, která se vyznačuje velkými nároky na rychlost všech používaných algoritmů, byl kladen důraz především na efektivnost metod generování.

Dále jsem se věnoval metodě Kena Perlina pro niž jsem jako zdroj pseudonáhodných čísel použil mnou vytvořenou knihovnou pro umělé generování šumů. Metody pro generování Perlinova šumu jsem aplikoval do knihovny a experimentoval s ní při použití různých generátorů šumu.

Jelikož tato práce obsahuje poměrně velké množství obrázků, rozhodl jsem se v ukázkových příkladech ukázat příklady jejichž zobrazení na obrázcích není dostatečně efektivní. První příklad tedy obsahuje 3D texturu vytvořenou ponocí jedné z metod knihovny generování Perlinova šumu. Ve druhém příkladě je demonstrováno vytvoření terénu krajiny pro různé hodnoty generátoru Gaussova šumu.

Tato práce mi byla přínosem v tom, že jsem si prakticky například vyzkoušel práci s OpenGL a seznámil se s množstvím metod, používaných v počítačové grafice.

Další možný vývoj je ve specializaci použití knihovny pro výpočet Perlinova šumu, například procedurální generování 3D mraků, terénu krajiny či vodní hladiny.

Literatura

[1] RÁBOVÁ Z.; aj.: *Modelování a simulace*. Vysoké učení technické v Brně, Brno, 2005.

[2] Wikipedie: The free encyclopedia [online]. Dostupné na URL: <http://www.wikipedia.org>.

[3] ZEMČÍK, P.: *Zracování obrazu*. Vysoké učení technické v Brně

[4] FISCHER, R.; aj.: *Noise Generation* [online]. Poslední modifikace: 2004. [cit.2007-01-03].
Dostupné na URL: <http://homepages.inf.ed.ac.uk/rbf/HIPR2/noise.htm#1>.

[5] GERLA, V.; HONZMAN, J.; POP, M.: *Generování šumu* [online]. Poslední modifikace: 2004. [cit.2007-01-03]. Dostupné na URL: <http://mips.ic.cz>.

[6] BOURKE, P.: *Perlin Noise and Turbulence* [online]. Poslední modifikace: leden 2000. [cit.2007-04-02]. Dostupné na URL: <http://local.wasp.uwa.edu.au/~pbourke/texture_colour/perlin/>.

[7] VANDEVENNE, L.: *Perlin Noise* [online]. Poslední modifikace: 19.11.2006. [cit.2007-04-06]. Dostupné na URL: <http://student.kuleuven.be/~m0216922/CG/perlinnoise.html>.

[8] ELIAS, H.: *Perlin Noise* [online]. Poslední modifikace: 2001. [cit.2007-04-06]. Dostupné na URL: <http://freespace.virgin.net/hugo.elias/models/m_perlin.htm>.

[9] HUMPHREY, B.: *Generování terénů a krajin za použití výškového mapování textur* [online]. In Turek, M., překladatel. Dostupné na URL: <http://nehe.ceskehry.cz/tut_34.php>.

[10] TIŠŇOVSKÝ, P.: *Perlinova šumová funkce a její aplikace* [online]. Poslední modifikace: 20.3.2007. [cit.2007-04-06]. Dostupné na URL: <http://www.root.cz/clanky/perlinova-sumova-funkce-a-jeji-aplikace/>.