VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

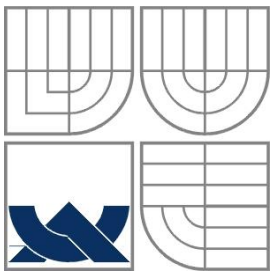
TVORBA 3D SCÉNY S VYUŽITÍM OPEN-SOURCE NÁSTROJŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

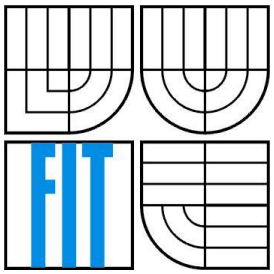
AUTOR PRÁCE
AUTHOR

PETR VAŠÁK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

TVORBA 3D SCÉNY S VYUŽITÍM OPEN-SOURCE NÁSTROJŮ

3D SCENE COMPOSITION WITH OPEN-SOURCE TOOLS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

PETR VAŠÁK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. VÁCLAV ŠIMEK

BRNO 2008

Abstrakt

Práce se zabývá vytvořením komplexní 3D scény pomocí open-source nástrojů. Je zde popsán postup vytváření 3D modelů, úprav scény a následný převod do renderovacího programu PovRay, kde je za pomoci úpravy kódu a přidáním vlastností vytvořen výsledný obraz. Cílem je vyrenderovaná scéna metodou Ray-tracing.

Klíčová slova

3D scéna, render, ray tracing, Rhino3d, Pov-ray

Abstract

The work deals with the creation of complex 3D scenes using open-source tools. It describes how to create 3D models, adjustments to the scene and subsequent transfer to renderer program PovRay, where with the help of regulation code and adding features created by the resulting image. The aim is to render scene with Ray-tracing method.

Keywords

3D scene, render, ray tracing, Rhino3d, Pov-ray

Citace

Vašák Petr: Tvorba 3D scény s využitím open-source nástrojů.
Brno, 2008, bakalářská práce, FIT VUT v Brně.

Tvorba 3D scény s využitím open-source nástrojů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Václava Šimka
Další informace mi poskytli
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Poděkování

Chtěl bych poděkovat vedoucímu této práce Ing. Václavu Šimkovi za pomoc při vedení a vytváření práce.

Obsah

Obsah 1
1 Úvod 2
1.1 Popis kapitol 3
2 Techniky modelování a zobrazování 4
2.1 Popis technik 4
2.1.1 Modelovací techniky 7
2.1.2 Zobrazovací techniky 8
2.2 Algoritmy a vzorce 16
2.2.1 Ray tracing 16
2.2.2 Anti-aliasing 18
2.2.3 Radiosita 19
3 Vytváření scény 20
3.1 Nástroje 20
3.1.1 Wings 3D 20
3.1.2 Rhinoceros 3D 21
3.1.3 PoseRay 22
3.1.4 Pov-Ray 23
3.2 Formáty 24
3.2.1 3DM 24
3.2.2 OBJ 24
3.2.3 POV 25
3.2.4 Prezentace dat 25
3.3 Postup a tvorba 26
3.3.1 Návrh 26
3.3.2 Modelování 26
3.3.3 Převod formátů a úpravy 29
3.3.4 Dokončení scény a render 30
3.4 Výsledek a zhodnocení 33
4 Možnosti dalšího vylepšení a rozšíření 34
5 Závěr 35
Literatura 36
Příloha A: Zdrojové kódy a popis 39
Příloha B: Modely a vyrenderované scény 42
Příloha C: DVD 46

1 Úvod

Počítačová grafika je samostatná kategorie grafiky. Jde o obor informatiky, jenž se používá na syntetické vytváření umělých snímků, a také pro úpravu zobrazitelných a prostorových informací nasnímaných z reálného světa.

Tento obor grafiky můžeme dělit a používat v mnoha oblastech jako například: 2D, 3D modelování, editace obrázků, počítačová animace, 3D renderování v reálném čase a spousta dalších. Dnes počítačová grafika zasahuje do mnoha jiných oborů, ať už jsou to například film, inženýrské účely nebo lékařské účely. Počítačovou grafiku můžeme z hlediska zobrazování dělit na dvě hlavní odvětví: 2D a 3D grafika *[2]*.

2D grafika je základem pro počítačovou grafiku a obecně pro její zobrazování. Můžeme jí dělit na *rastrovou* a *vektorovou*. Rastrová grafika se skládá z pravidelné sítě pixelů, organizovaná jako dvourozměrná matice bodů, která má omezené rozlišení. Vektorová ukládá přesná geometrická data, topologii, styl (souřadnice bodů, jejich propojení, barvu, šířku) a podle těchto parametrů zobrazuje výsledek.

3D grafika se používá pro modelování a prohlížení scény v prostoru. Trojrozměrné vyjádření dat je uloženo za účelem vykonání výpočtů, renderování a zobrazování 2D obrázků. 3D grafika se opírá o stejné algoritmy jako 2D grafika – 2D vektorová grafika v modelování objektu a 2D rastrová grafika v konečném vyrenderovaném zobrazení.

Nejvíce nás bude zajímat 3D grafika a v ní vytváření 3D modelů. 3D model je matematické vyjádření trojrozměrného objektu. Tento model je vytvořen za pomoci některého modelovacího nástroje a následně uložen do grafického souboru, ve kterém je parametricky popsán (síť bodů v prostoru). Modelování popisuje tvary vytvořených objektů. 3D model může být virtuálně zobrazen jako dvourozměrný obraz pomocí 3D renderování nebo použit v různých negrafických výpočtech a simulacích.

Po vytvoření 2D obrazu z 3D modelů se používá renderování. Jde o vizualizaci dat, tvorbu reálného obrazu (scény) na základě počítačového modelu, respektive přenesení modelu do 2D bitmapy. Je to odvětví počítačové grafiky, které se zabývá tvorbou obrazů napodobující reálný svět. Jedná se o způsob vizualizace dat, kde parametry popisují scénu (reálnou, imaginární) s jejími objekty a vlastnostmi. Konečný vzhled závisí na nastavení parametrů a vlastností renderovacího programu i vlastního modelu, a přidáním dalších technik renderování, kterými je např. Raytracing – vykreslení na základě sledování paprsků světla.

Renderování zajišťují programy obsažené přímo v modeláři nebo samostatné. Jeho úkolem je z počítačového modelu vytvořit obraz, který je pokud možno nerozeznatelný od definovaného objektu

v reálném světě. Realistické počítačové scény jsou uplatňovány v mnoha oblastech od architektury, tvorby filmů, efektů po vědecké simulace a práci s virtuální realitou (počítačové hry).

Mým úkolem v této práci bude vytvoření modelu a jeho zpracování v co nejdokonalejší a dostatečně náročnou trojrozměrnou scénu zachytávající reálné prostředí. Využiji při tom několik grafických nástrojů. Jednak grafické modeláře, ve kterých vytvořím samotné modely, které spojím v určitou scénu. Dále pak nástroj pro převody formátů mezi jednotlivými programy, úpravu vlastností a vzhledu scény. V závěrečné fázi vše převedu na kód ve specializovaném programovacím jazyku k popisu scén, kde přidám techniky renderu a upravím scénu do konečné podoby. Co se týče nástrojů, zaměřím se spíše na volný a open-source software.

1.1 Popis kapitol

Techniky modelování a zobrazování

V této kapitole jsou podrobně popisovány použité modelovací techniky od základů zobrazování po složitější techniky. Dále zobrazovací techniky, které jsem použil při renderování scény. U vybraných z nich jsem popsal algoritmy a vzorce, na základě kterých pracují.

Vytváření scény

Tato nejrozsáhlejší kapitola je zaměřena na přesný postup vytváření výsledné scény. Je tu obsažen každý krok vývoje. Nejdříve jsou zde popsány všechny nástroje, se kterými jsem scénu vytvářel (modelování, úprava, render) a formáty souborů se kterými jsem pracoval. Nakonec je rozepsán postup a tvorba v každém z nástrojů a postupná úprava až do finální scény.

Možnosti dalšího vylepšení a rozšíření

Diskuze a návrhy možných vylepšení, rozšíření a pokračování celé práce.

2 Techniky modelování a zobrazování

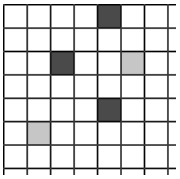
U vytváření 3D scény se uplatňují různé techniky a metody k jejímu zobrazení.

2.1 Popis technik

Úvodem popíši základy zobrazování počítačové grafiky. V ní existují dva hlavní způsoby zobrazování a ukládání obrázků. Každý z nich má využití v jiných případech.

Bitmapová (rastrová) grafika

Základní obrazový prvek je zde **pixel**. Nejmenší jednotka rastrové grafiky. Představuje jeden bod obrázku daný určitou barvou a souřadnicemi.



obr. 2.1 pixely v rastru.

Každý obraz je popsán pomocí jednotlivých bodů (pixelů) v mřížce (viz. *obr. 2.1*). Každý bod má učenou polohu souřadnicemi a barvu jako na *obr. 2.2*. Při určité velikosti a počtu bodů začnou pixely opticky splývat, a tak vzniká celý obraz. Kvalita obrazu je dána především celkovým rozlišením a barevnou hloubkou. Rozlišení udává velikost mřížky (šířku a výšku) v bodech a barevná hloubka určuje, kolik pixelů se vejde do délky jednoho palce (2.54cm) *[21]*.

Rastrová grafika má jisté nevýhody. Pokud chceme dosáhnout lepší kvality, potřebujeme vyšší rozlišení, čímž vzrůstá i náročnost. Když chceme změnit velikost obrazu, znamená to zhoršení kvality a ztrátu obrazových dat. Proto se tato grafika využívá zejména pro obraz a zobrazování, které je konečné a nebude se nadále měnit z hlediska určitých parametrů a tam kde by byla vektorová grafika příliš komplexní (fotografie, video).

Bitmapovou grafikou jsem při vývoji práce z praktického hlediska využil pro editaci textur modelů a zobrazením scény renderováním.

obr 2.2 jednotlivé pixely, ze kterých se celý obraz skládá.

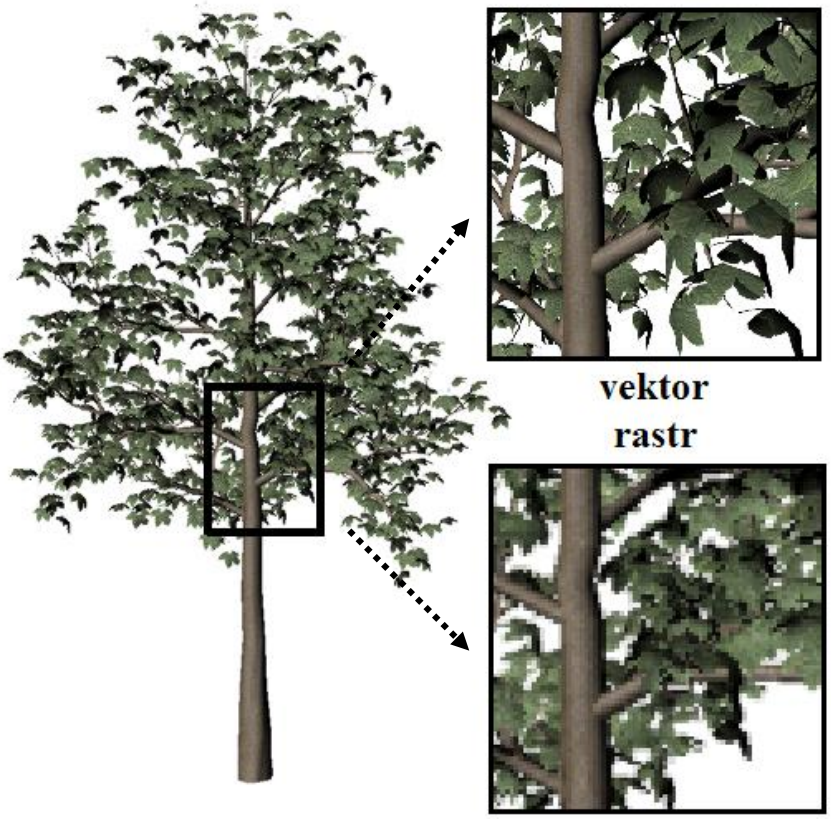
Vektorová grafika

Způsob zobrazování počítačové grafiky pomocí geometrických primitiv – bodů, čar, křivek a polygonů. Data obrazu obsahují pouze parametry (souřadnice, barvy) a nastavení těchto primitiv. Například k vykreslení kružnice jsou potřeba tyto parametry: poloměr a rovnice kružnice, pozice jejího středu, barva a styl čáry a barva a styl její výplně (stylem je myšlena např. šířka).

V této grafice je možné zmenšování a zvětšování obrazu bez jakékoliv ztráty kvality nebo dat. I při přiblížení neztrácí žádný objekt kvalitu, protože je obecně „nekonečně tenký“. Každý obraz je složen z daných primitiv a s každým z nich lze pracovat samostatně. Velikost výsledného obrazu je mnohem menší než u rastrové grafiky, protože se neukládají parametry každého bodu obrazu, ale pouze parametry jednotlivých primitiv, ze kterých je obraz složen. Vektorová grafika má své využití jak ve 2D tak ve 3D grafice. Nemůže však plně nahradit grafiku rastrovou. Například u fotografií těžko dosáhnout podrobností jako u rastru. Převodem na vektory se můžeme kvalitě pouze přiblížit.

Vektorového způsobu zobrazování se využívá hlavně ve 3D modelování, kde se model skládá z jednotlivých částí. Kdykoliv tak lze upravit parametry a rozměry objektů bez nějaké ztráty. Tuto grafiku jsem v práci využil ve 3D vývojových nástrojích při modelování a nastavení objektů pro celou scénu. Také při exportu modelů mezi jednotlivými programy, kde byly v souborech uloženy geometrické souřadnice základních prvků, které tvořily modely. Na obrázku *2.3* můžeme vidět rozdíl vektorové a rastrové grafiky v praktickém použití.

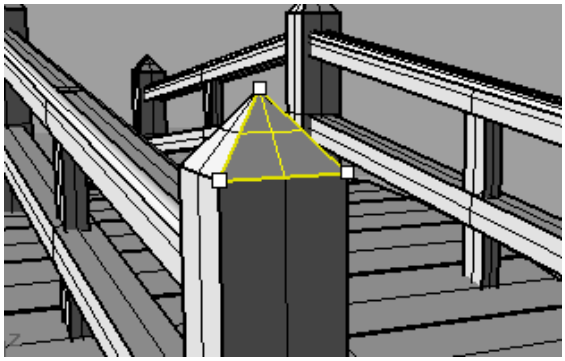
K převodu vektorové grafiky do rastrové se využívá tzv. rasterizace.



*obr. 2.3 na obrázku lze vidět přiblížení v podání **vektorové** a **rastrové** grafiky.*

Polygon – mnohoúhelník (obecně 2D), v 3d počítačové grafice představuje plochu (viz. *obr. 2.4*), z které se skládá model. Nejčastěji se používá trojúhelník, protože se nejsnadněji zpracovává.

Vertex – rohový bod polygonu (viz. bílé body na *obr. 2.4*). Bod je zadán souřadnicemi, následně z více bodů vznikají polygony a objekty.



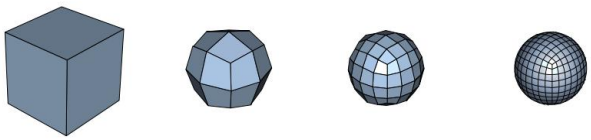
obr. 2.4 jeden polygon v modelu a jeho vertexy.

2.1.1 Modelovací techniky

Techniky, které jsem využil při vytváření modelů v nástrojích.

Subdivision

Metoda pro modelování objektů. Jde o postupnou úpravu těles. Reprezentuje hladký povrch modelu pomocí přesného popisu hrbolaté po částech rozdělené polygonové sítě. Povrch tělesa je spočítán z hrubé sítě opakováním procesů rozdělování každého polygonu na menší části, čímž se přibližujeme k hladkému povrchu (viz. *obr. 2.6*). Výsledný objekt zde vzniká úpravou, rozkládáním, rozšiřováním a připojováním dalších objektů *[20]*.



Obr. 2.6 postupná úprava rozdělováním polygonů [20].

NURBS (Non-uniform rational B-spline)

Matematický model používaný pro generování křivek a ploch. Křivky jsou definovány kontrolními body a uzlovými vektory (viz. *obr. 2.7*). NURBS křivky a povrchy jsou zobecněním B-spline a Bézierových křivek. Hlavní rozdíl spočívá v použití kontrolního bodu, který dělá křivky racionální.

Základní definice NURBS:

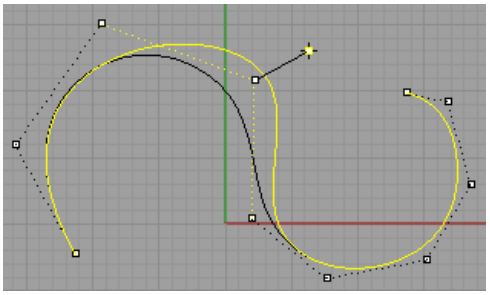
Je dáno m+1 kontrolních bodů P_i, m+1 kladných reálných čísel ω_i nazývaných váhový koeficient, stupeň křivky n, bázovou funkci N_i^n a uzlový vektor $t = (t_0, t_1, \ldots t_{\{n+m+1\}})$.

NURBS křivka je pak definována:

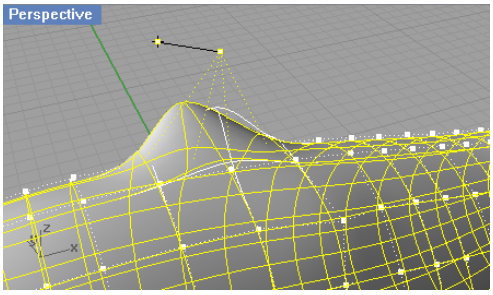
$$C(t) = \frac{\sum_{i=0}^{m} w_i P_i N_i^n(t)}{\sum_{i=0}^{m} w_i N_i^n(t)} \quad \text{, kde} \quad t \in \langle t_n, t_{m+1} \rangle$$

Každá NURBS křivka a plocha vzniká takto. NURBS plocha vznikne tenzorovým součinem dvou NURBS křivek *[25]*.

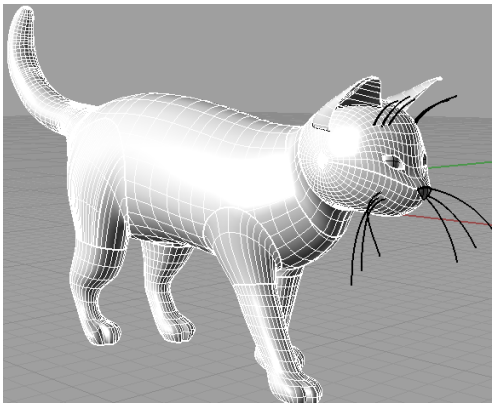
Ve 3D grafice reprezentují geometrii, která dokáže přesně popsat každý tvar od jednoduchých 2D obrazců po nejsložitější komplexní 3D povrchy a objekty (viz. *obr. 2.8*). Díky jejich přesnosti a flexibilitě mají NURBS modely široké využití ve vytváření obrazů, animací i ve výrobě. Jedná se o jednu z nejlepších modelovacích technologií. Modely navíc je lze kdykoliv v průběhu jednoduše upravit nebo změnit *[viz.26]*.



obr. 2.7 NURBS křivka.



obr. 2.8a použití NURBS ve 3D.



obr. 2.8b celý model z NURBS křivek.

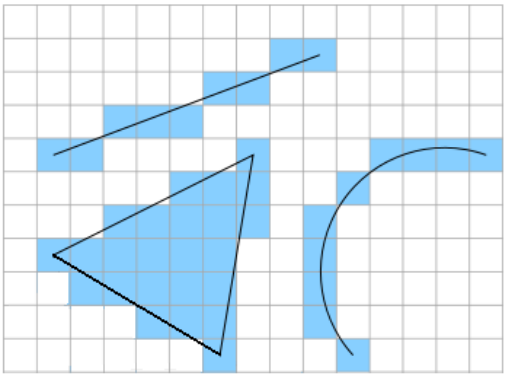
2.1.2 Zobrazovací techniky

2.1.2.1 Rasterizace

Obecně jde o převod vektorové grafiky na bitmapovou. Je to proces k určení souřadnic a barvy pixelu z vektorového popisu objektů a to ve 2D i 3D grafice. Vektorové objekty tak mohou být převedeny do rastrového formátu.

Všechny objekty (modely) ve 3D grafice mám popsány vektorově. Pro zobrazení (render) je potřeba je zobrazit rastrově (viz. *obr. 2.9*). Používá se vzorkování vektorových objektů. K rasterizaci slouží různé algoritmy na výpočet polohy bodů na rastru z vektorových prvků.

Nejzákladnější algoritmus rasterizace vezme 3D scénu popsanou polygony a renderuje jí na 2D plochu. Polygony jsou reprezentovány souborem trojúhelníků, které mají tři vrcholy ve třech rovinách. Tyto vrcholy se převedou na odpovídající body na 2D ploše a zobrazené trojúhelníky se zde vhodně vyplní.



obr. 2.9 příklad vyplňování bodů na základě vektorů.

2.1.2.2 Rendering

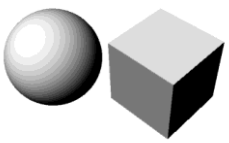
Renderování je proces generování obrazu z modelu pomocí nástrojů pro to určených. Model je zde popisován trojrozměrným objektem v přesně definovaném jazyku nebo datové struktuře. Může obsahovat informace o geometrii, úhlu pohledu, texturách, osvětlení. Vygenerovaný obraz představuje grafický rastr.

Je to jeden z hlavních oborů 3D počítačové grafiky prakticky související s ostatními. V grafice představuje poslední hlavní krok ke konečnému vzhledu modelů a animací. Zároveň se stoupající náročností na počítačovou grafiku se stává stále důležitějším. Renderování má široké spektrum využití, používá se v architektuře, designu, simulacích, počítačových hrách, filmech a pro různé speciální efekty. Každý z oborů využívá jiné prvky a techniky *[9]*.

K dispozici je mnoho produktů určených pro renderování. Některé jsou samostatné, open-source, jiné komerční a součástí komplexního modelovacího nástroje. V oborech 3D grafiky můžeme renderovat buď pomalu, nebo v reálném čase. Pomalý render se využívá k vytváření komplexních scén (statické obrázky, filmy), zatímco v reálném čase je využíván ve 3D počítačových hrách, kde se spoléhá na hardwarovou akceleraci grafických karet.

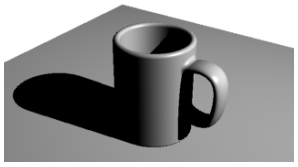
K renderování obrazu se používá mnoho prvků. Je mnoho způsobů jak vytvořit konečný vzhled co nejlépe a nejefektivněji *[viz. 7]*. Některé se přímo týkají algoritmů a technik. Popíši nejdůležitější a ty, co jsem použil ke tvorbě scény *[10], [18]*.

- **Shading** (stínování) – Určuje, jak se bude barva a jas povrchu měnit s osvětlením. Stínování objektů zobrazeno na *obr. 2.10.*



Obr. 2.10 stínování objektů

- **Shadows** (stíny) – Efekt při zamezení přístupu světla. Stíny modelu při osvětlení na *obr. 2.11*.



obr. 2.11 stín modelu.

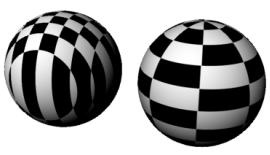
- **Soft shadows** (jemné stíny) – Proměnlivý stín způsobený částečným zamezením přístupu světla. Je vyhlazený, nemá ostré hrany a působí reálněji (viz. *obr. 2.12*).

obr. 2.12 jemné stíny modelu.

- **Texture mapping** (mapování textury) – Metoda aplikování textury (detailů) na povrch objektu (viz. *obr. 2.13*). Používá se přitom parametrů a různých způsobů mapování.

obr. 2.13 textury na modelu.

Nejvíce jsem používal metodu plošného mapování. Konkrétně tzv. UV mapování textur. V této metodě se textura mapuje podle U a V souřadnic, což jsou 2D souřadnice plošně rozloženého povrchu tělesa, na které texturu nanáším. Takto mohu texturu po povrchu tělesa posouvat nebo měnit její měřítko tak, aby co nejvhodněji na povrchu seděla. Dalším způsobem mapování je například kubické, sférické, válcové. Na obrázku *2.14* jde vidět rozdíl mezi UV plošným a rovinným mapováním.

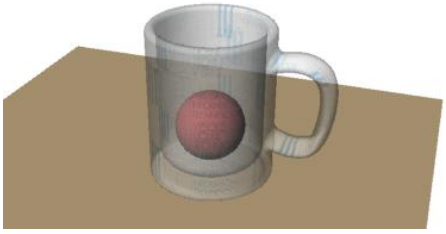


obr. 2.14 vlevo rovinné a vpravo plošné mapování.

- **Reflection** (zrcadlení) – Odraz okolních objektů na zrcadlících se nebo vysoce lesklých objektech (viz. *obr. 2.15*).

obr. 2.15 zrcadlící se okolí na modelu.

- **Transparency** (průhlednost) – Jasný prostup světla v určité míře skrz pevné objekty (viz. *obr. 2.16*).



obr. 2.16 model z průhledného materiálu.

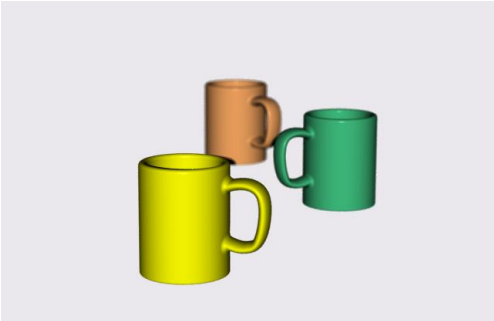
- **Fogging/participating medium** (mlha, efekty prostředí) – Tlumení světla při průchodu „nečistým" prostředím. Na *obr. 2.17* průchod světla v mlze.

obr. 2.17 průchod světla prostředím.

- **Refraction** (lámání) – Lom světla způsobený průchodem objektů spojený s průhledností.

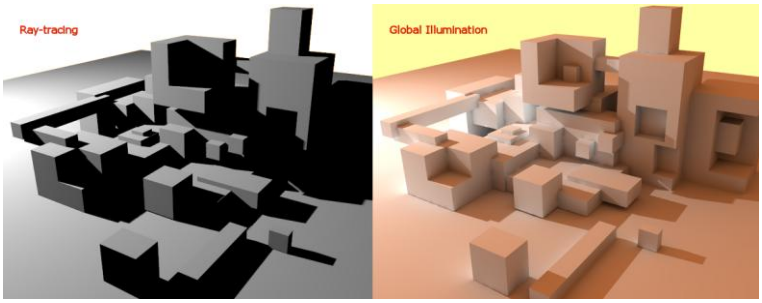
- **Diffraction** (ohyb) – Ohyb, rozptýlení a rušení paprsku světla průchodem objektu nebo otvorem, který rozloží paprsek.

- **Depth of field** (hloubka pole) – Objekty, které jsou nezaostřené nebo mimo zaostřený objekt vypadají rozmazaně. Využívá se zde nastavení ohniskové vzdálenosti, podle objektů, které chceme mít zaostřeny. Na *obr. 2.19* vzdálenější model rozmazán.



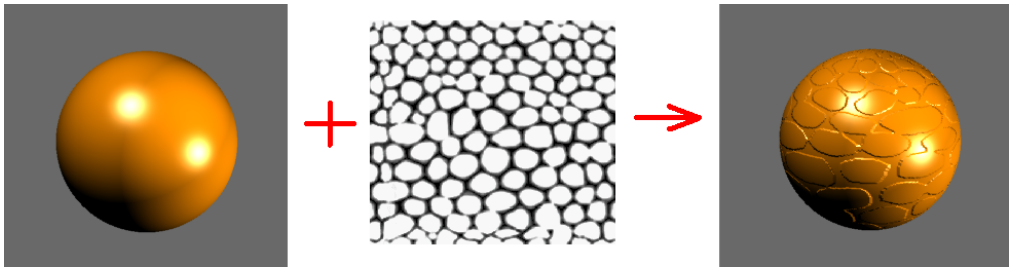
obr. 2.19 rozmazání nezaostřeného objektu.

- **Indirect/global illumination** (nepřímé/globální osvětlení) – Povrchy objektů jsou osvětlené nejen zdrojem světla, ale i odrazem paprsků od ostatních objektů ve scéně. Tuto metodu uplatňuje částečně radiosita. Vyrenderované obrázky jsou tak téměř nerozeznatelné od skutečnosti. Metoda je ovšem vysoce výpočetně i časově náročná *[11]*. Na následujícím *obr. 2.18* je vidět rozdíl mezi klasickým a globálním osvětlením.



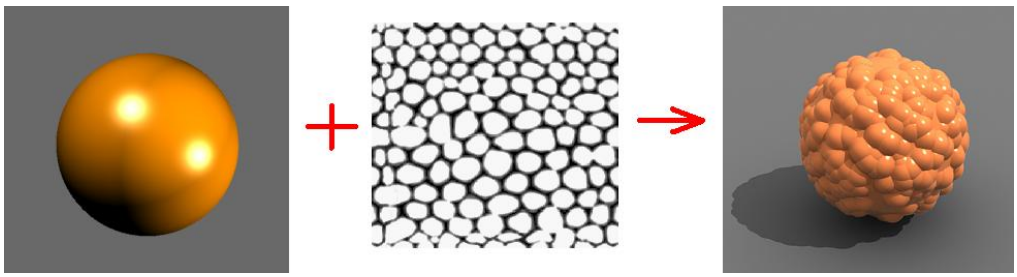
obr. 2.18 rozdíl mezi klasickým a globálním osvětlením [28].

- **Bump mapping** – metoda simulace hrbolatosti povrchu v malém měřítku. Simuluje náročnější povrchy bez zvýšení počtu polygonů. Opticky se zdá, že je povrch hrbolatý, přitom se nemění geometrie objektu. Hrbolatý povrch se mapuje podle zvolené textury (viz. *obr. 2.20*).



obr. 2.20 vzhled objektu bez/s metodou bump mapping.

- **Displacement mapping** – další metoda simulace hrubosti povrchu. Na rozdíl od bump mapping se geometrie objektu skutečně mění (objekt je deformován viz. *obr. 2.21*). Hrbolky např. vrhají stíny. Mapování povrchu probíhá také pomocí textury.

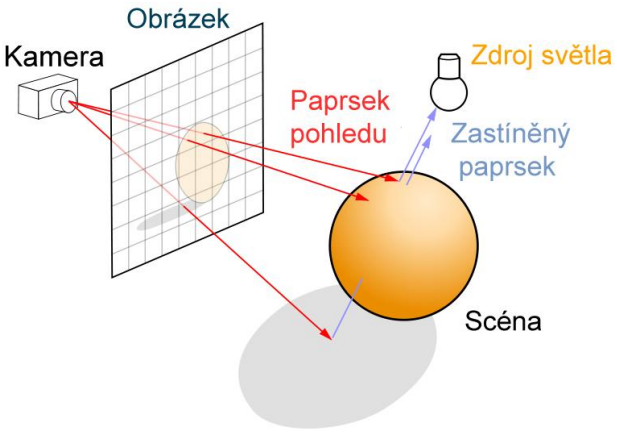


obr. 2.21 vzhled objektu bez/s metodou displacement mapping.

2.1.2.3 Ray tracing

Jedná se o techniku generování obrazu z určitého modelu a to pomocí sledování paprsku (cesty) světla skrze pixely v rovině obrazu. Pomocí této metody renderování jsme schopni vytvořit obraz s vysokou úrovni fotorealismu. To vše samozřejmě na úkor náročnějším výpočtům oproti běžnému renderu. Jedná se o vysoce výpočetně náročnou metodu. Je využívaná především tam, kde je dostatek času na renderování, například v nehybných obrazech, speciálních efektech filmů apod. Nevhodná je při použití v aplikacích běžících v reálném čase, jako jsou například počítačové hry, kde je rychlost kritickým faktorem a se současnou rychlostí PC hardware je to prakticky nemožné. Do budoucna se však s touto technikou počítá i v tomto oboru *[8]*.

Technika vychází z fyziky světla, kde se paprsky pohybují od zdroje, odráží se a lámou, až nakonec vejdou do oka pozorovatele. Zde však paprsky vycházejí z kamery (oka) a rozšiřují se směrem k pozorovaným objektům (viz. *obr. 2.22*). Objekty paprsek buď odrážejí (reflection) nebo ho propouští (transparency) a lámou (refraction). Ray tracing je schopen simulovat mnoho optických efektů a jevů jako odrazy, lomy, průhlednost, stíny, odchylky. Má ale také jisté nevýhody: ostré stíny, bodové zdroje světla, nemá sekundární zdroje světla (odrážející plochy se nechovají jako zdroje světla) *[13]*. Pro zdokonalení k němu můžeme používat další techniky jako je radiosita.



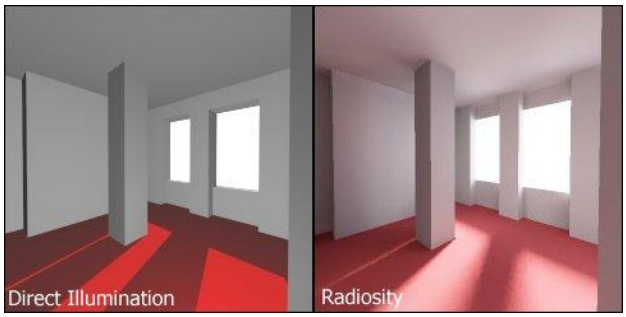
obr. 2.22 šíření paprsků do scény a její zobrazení [13].

Existují zde 3 typy paprsků:

Primární paprsek – je vyslaný od pozorovatele scény
Sekundární paprsek – vzniká odrazem nebo lomem paprsku od objektu ve scéně
Stínový paprsek – je vyslaný z místa dopadu paprsku na objekt ke zdrojům světla. Tak se zjistí, zda leží ve stínu. U stínových paprsků se zanedbává lom.

2.1.2.4 Radiosita

Je to metoda globální iluminace scény (šíření světelné energie) používaná k renderování 3D scény. Vychází ze zákona zachování energie, vyžaduje proto energeticky uzavřené scény (lze použít i v otevřených scénách, pokud ignoruji paprsky, které se po určité délce neodrazí). Sama nedokáže pracovat s průhlednými objekty, zrcadly a texturami, také proto je spojována s metodou Ray tracing. Všechny plochy zde odrážejí světlo a mohou mít i vlastní zářivost. Renderovaná scéna musí být prezentována polygonálním modelem, kde je u každé plošky modelu vypočítána její zářivost *[15]*. Na obrázku *2.23* je zřejmý rozdíl klasického šíření světla a radiosity.



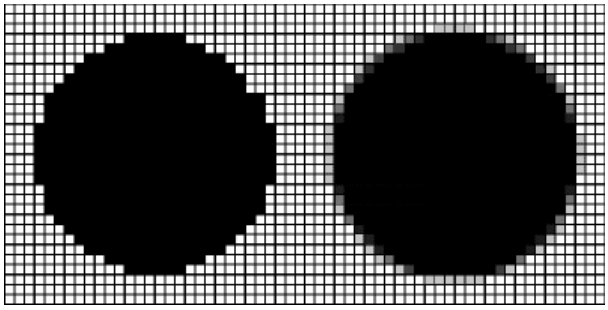
obr. 2.23 rozdíl mezi klasickým šířením světla a radiositou.

2.1.2.5 Anti-aliasing

Obecně se jedná o techniku minimalizování zkreslení. V počítačové grafice představuje tato technika zlepšující kvalitu obrazu odstraňováním zubatých okrajů. Tyto okraje vznikají při vzorkování vektorového objektu, metodě zvané Alias. Počítačový obraz je tvořen čtverci (pixely), které nejsou schopny adekvátně zobrazit čáry a křivky (vektorové objekty), které nejsou paralelní k pixelům.

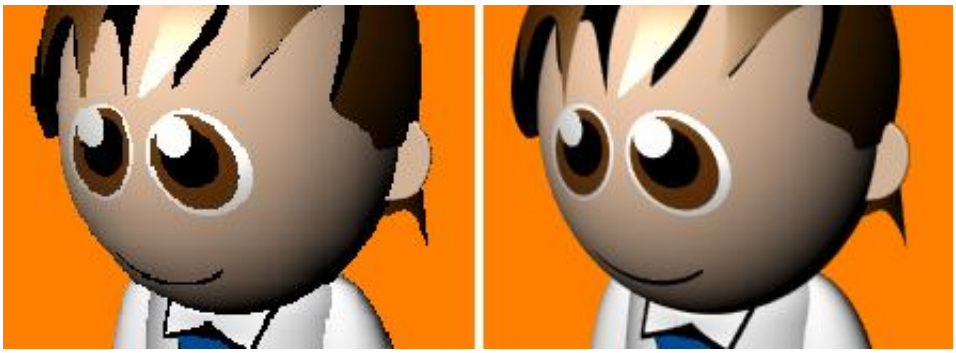
Tento problém můžeme částečně vyřešit zvýšením rozlišní obrazu. Tím se ale zvýší náročnost a problém navíc není uspokojivě vyřešen. Při zobrazování zkrátka není možné aliasing úplně eliminovat. Proto se používá anti-aliasing, který způsobí vyhlazení obrazu bez jakékoliv změny rozlišení nebo rozměrů (viz. *obr. 2.24*).

Anti-aliasing můžeme aplikovat až na výsledný obraz (což má za následek ztrátu detailů modelu a rozmazání) nebo přímo při renderování a výpočtu *[19]*.



obr. 2.24 kruh zobrazený v mřížce pixelů před/po anti-aliasingu [19].

Mezi jeho nevýhody patří zvýšení náročnosti výpočtu a rozmazávání obrazu, které je ale ve většině případů irelevantní vzhledem k původnímu k tomu bez anti-aliasingu.



Obr. 2.26 stejný model vyrenderovaný bez a s anti-aliasingem

Jak lze vidět na *obr. 2.26,* anti-aliasing pomáhá eliminovat ostré hrany, aby obraz vypadal více realisticky. Proto má široké využití v mnoha oborech počítačové grafiky a zpracování obrazu.

2.2 Algoritmy a vzorce

Podrobný popis algoritmů vybraných metod, které jsem použil při vytváření mojí scény.

2.2.1 Ray tracing

Všechny scény v ray tracingu jsou matematicky popsány. Typicky u každého paprsku světla musí být testováno protínání s každým objektem scény (a všemi polygony v každém objektu). Jakmile je nalezen nejbližší objekt, algoritmus odhadne vstupující světlo v bodě protínání, zkontroluje nastavení materiálu objektu a zkombinuje tyto informace k vypočtení konečné barvy pixelu. Každý průsečík paprsku s objektem generuje další dva parsky a stínový paprsek.

V každém průsečíku paprsku a objektu je potřeba provést ty stejné výpočty. Proto se při implementaci metody ray tracing používá rekurze, jejíž ukončení je možné zadáním kriterií:

- paprsek narazí na difúzní povrch
- je dosažena stanovená hloubka rekurze
- energie paprsku klesne pod určitý práh

V každém průsečíku **P** paprsku a objektu platí:

$$\mathcal{I}(P) = I_{local}(P) + I_{global}(P) = I_{local}(P) + k_{rg}I(P_r) + k_{tg}I(P_t)$$

P - průsečík
P_r - další průsečík odraženého paprsku
P_t - další průsečík propuštěného paprsku
k_{rg} - globální koeficient odrazivosti (reflexe)
k_{tg} - globální koeficient propouštění (lomu, transmise)

Rekurzivní charakter rovnice potvrzuje vhodnost tohoto přístupu pro implementaci raytracingu *[14]*.

Algoritmus:

funkce SledujPaprsek (R, H):

1. Najdi průsečík P mezi R a objektem.
2. Pokud P neexistuje (R jde mimo scénu) - pixel vyplň barvou pozadí.
3. Z P pošli ke zdrojům světla stínové paprsky.
4. Vyhodnoť součet osvětlovacích modelů v P pro nezakryté zdroje světla.
5. Pokud není překročeno H, vyšli z P:
    a. Odražený sekundární paprsek voláním SledujPaprsek (R_r, H + 1).
    b. Lomený paprsek SledujPaprsek (R_t, H + 1).
6. Pixel má barvu danou součtem barvy od osvětlení, odraženého paprsku R_r a lomeného paprsku R_t.

R – paprsek, H – hloubka rekurze

P_r - další průsečík odraženého paprsku

P_t - další průsečík propuštěného paprsku

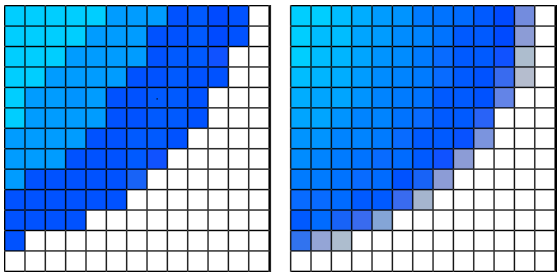
Počet rekurzivních volání procedury SledujPaprsek závisí na parametru H. Pokud se H rovná 1, jedná se o ray-casting, kde se vyhodnocují pouze primární paprsky a jejich dopady na nejbližší objekt. Při použití stínových paprsků jsou vyhodnoceny stíny. Pro zobrazení odrazů se musí procedura volat alespoň dvakrát a pro řešení průhledných těles alespoň třikrát.

Pseudokód: rekurzivní ray tracing *[13]*

```
For (každý pixel ve scéně) {
  Vytvoř paprsek z bodu pohledu procházející skrze tento pixel
  Nastav NejbližšíPrůseč na INFINITY a NejbližšíObjekt na NULL

  For (každý objekt ve scéně) {
     If (paprsek protíná tento objekt) {
        If (průsečík P je menší než NejbližšíPrůseč) {
           Nastav NejbližšíPrůseč na průsečík P
           Nastav NejbližšíObjekt na tento objekt
         }
      }
    }

  If (NejbližšíObjekt je NULL) {
    Vyplň tento pixel barvou pozadí
   }
   Else {
     Pošli paprsek ke každému zdroji světla – kontrola, zda není ve stínu
     Pokud je povrch reflexní, vygeneruj paprsek odrazu: rekurze
     Pokud je povrch průhledný, vygeneruj paprsek lomu: rekurze
     Použij NejbližšíObjekt a NejbližšíPrůseč k výpočtu stínovací funkce
     Vyplň tento pixel barvou vypočtenou stínovací funkcí
     }
}
```

2.2.2 Anti-aliasing

Princip této metody spočívá v eliminaci nebo spíše redukci nechtěných vlastností obrazu. Z pohledu rasterizace obrazu se vztahuje k redukci zubatých okrajů mezi barvami (viz. *obr. 2.27*). Když je použit anti-aliasing upravuje se každý pixel jako oblast, a jeho barva se vypočítává v závislosti na objektech scény v oblasti tohoto pixelu. Může se tak vylepšit vzhled a konečného obrazu, ale rapidně zvýší čas potřebný pro renderování, protože je prováděno mnohem více výpočtů. Existuje více metod anti-aliasingu *[17]*.



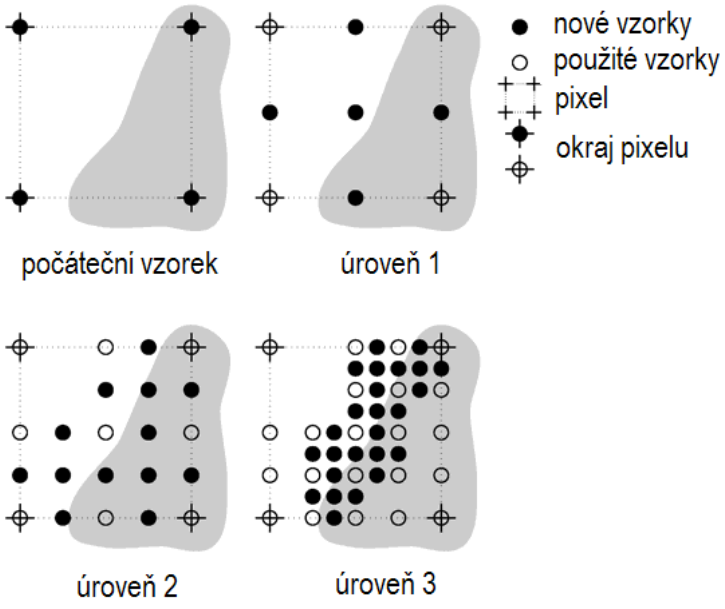
obr. 2.27 zobrazení před a po antialiasingu.

Mulsti-sampling

Detekuje a vyhlazuje pouze hrany objektů. Uvnitř se pixely vyhodnocují pouze pro jeden vzorek. Nemá tak vliv na zobrazování textur a náročnost je nižší než u následující metody.

Super-sampling (postfiltering)

Metoda obecně počítá scénu ve vyšším rozlišení a zobrazuje ji v menším přes interpolaci. Pracuje s celou scénou, je tedy pomalejší a náročnější.



obr. 2.28 postup metody anti-aliasingu [12].

Při renderování v programu PovRay jsem použil super-sampling adaptivní, rekurzivní. Tento způsob začíná posláním čtyř paprsků do rohů každého pixelu (viz. *obr. 2.28*). Pokud se od sebe výsledné barvy liší více než je nastaven povolený práh, je pixel rozdělen na další vzorky. Toto se děje rekurzivně, pixel je rozdělen do čtyř sub pixelů, které jsou odděleně zaměřeny a zkontrolovány pro další rozdělení. Maximální počet vzorků v pixelu si určím před začátkem. Výhodou je redukce počtu paprsků na množství, které bude opravdu potřeba. Vzorky, které jsou mezi sousedními pixely a sub pixely jsou uloženy a znovu použity, aby se zamezilo znovu posílání na stejné místo. Charakter této metody se soustřeďuje na ty části pixelu, které budou spíše použity. Výsledná hodnota pixelu je pak složena z průměru hodnot vzorků *[12]*.

Celé to proběhne renderováním obrázku v mnohem větším rozlišení, než v tom, ve kterém je pak zobrazen. Potom se obraz zmenší na požadovanou velikost. Počet vzorků určuje výslednou kvalitu výstupu. Obvykle jde nastavit rozmezí od 2x do 32x.

2.2.3 Radiosita

Vlastní výpočet může probíhat buď iteračně (progresivně) nebo řešením soustavy rovnic (maticové řešení). Před vlastním výpočtem je třeba polygony ve scéně rozdělit na malé plošky a spočítat konfigurační faktory (vliv každé plošky na každou jinou plošku ve scéně) *[1]*. Plošky, které na sebe nevidí, mají konfigurační faktor 0. Iterační výpočet má výhodu postupného zobrazení výsledku po každé iteraci.

Zářivost (radiosita) každé plošky je definována jako:

$$B_i = E_i + R_i \int_j B_j F_{ij}$$

B_i - radiosita plošky *i*.
E_i - vyzařovaná energie této plošky.
R_i - odrazivost plošky.
integrál reprezentuje součet energií přicházejících na plošku *i* ze všech ostatních plošek.
F_{ij} - konfigurační faktor mezi ploškami *i* a *j* (vliv plošky *j* na plošku *i*).

Konfigurační faktor určuje, kolik energie energie plošky i je přímo přijato ploškou i. Plošky s velkým rozdílem radiosity (ostrý světelný přechod) je vhodné rozdělit na menší plošky pro jemnější přechod osvětlení (adaptivní dělení ploch) *[16]*.

Výpočet radiosity je vysoce výpočetně i časově náročný, proto se nepoužívá pro výpočty v reálném čase. Výhodou této metody je, že se scéna nemusí přepočítávat při změně polohy kamery. Pro zobrazení výsledků radiosity může být použita metoda ray tracing. Ta navíc přidá zrcadlové odrazy, průhlednost, textury a vlastnosti povrchů, se kterými radiosita sama pracovat nedokáže.

3 Vytváření scény

Mým úkolem a cílem této práce bylo vytvořit 3D modely pro komplexní scénu a následně vše vyrenderovat. To zahrnovalo nejdříve výběr vhodných nástrojů a kontroly zda mezi nimi půjde bez problému přecházet. Dalším krokem bylo uvažovat vhodnou scénu a vytvořit pro ni všechny 3D modely. V poslední fázi bylo zapotřebí kompletní scénu co nejvhodněji zobrazit, to jsem provedl pomocí přidání vlastností (efektů) scény, jejího nastavení a konečného renderu. V této kapitole popíši vše, s čím jsem pracoval a jednotlivé kroky vývoje.

3.1 Nástroje

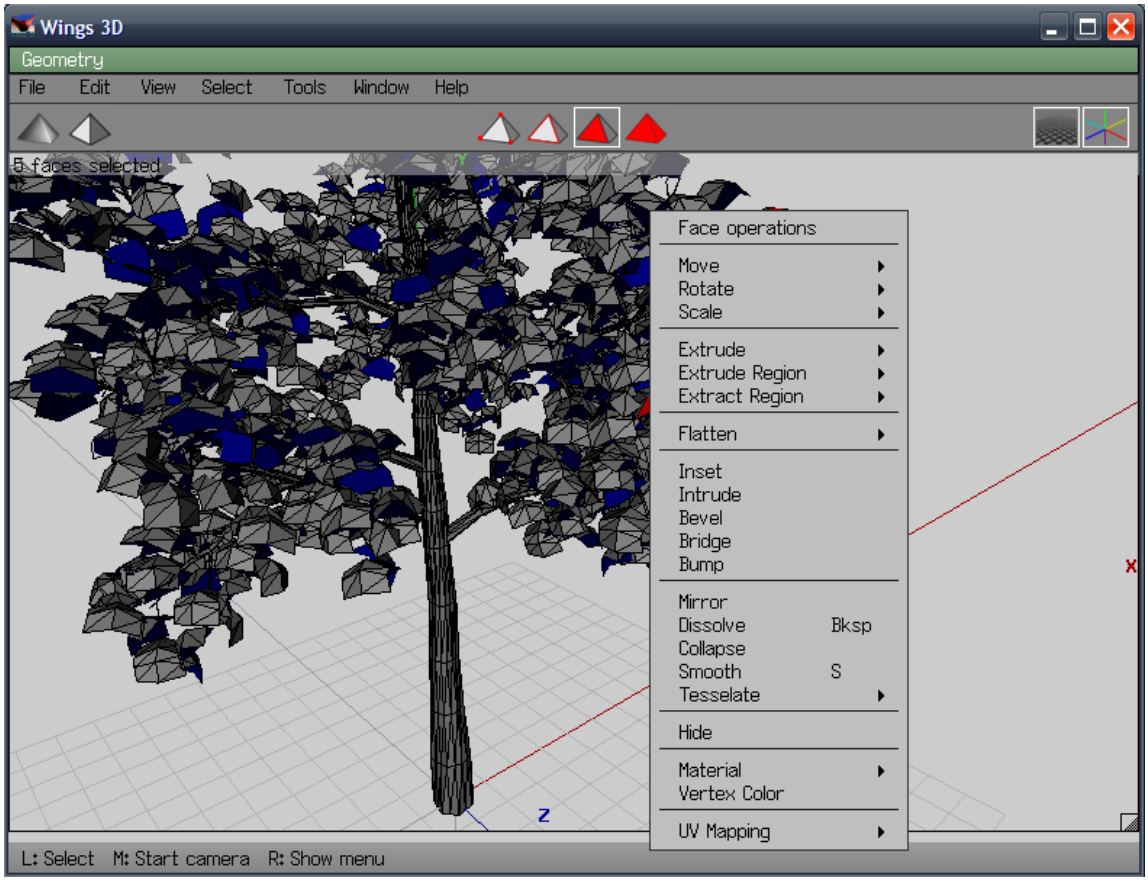
Pro vytvoření scény jsem si musel co nejvýhodněji zvolit nástroje, ve kterých budu práci vytvářet. Jednak jsem potřeboval modelář pro vytvoření objektů a modelů scény a jednak program, ve kterém scénu vyrenderuji a přivedu do konečného vzhledu. Navíc bylo třeba počítat s kompatibilitou vstupních a výstupních formátů mezi používanými programy.

Vyzkoušel jsem mnohé programy, pomocí kterých bych postupně mohl komplexní scénu vytvořit. Byly mezi nimi nástroje komerční, freeware i open-source, na které jsem kladl důraz především. Nejvíce se mi zamlouvala kombinace modelářů Rhino3d a Wings 3d, PoseRay pro převody a úpravu formátů a renderovacího programu Pov-Ray. Mezi těmito celkem snadno probíhal export a import vytvořených prvků.

V další části této kapitoly se budu věnovat právě nástrojům, ve kterých jsem scénu vytvářel.

3.1.1 Wings 3D

Tento freeware modelář je zaměřen na modelování pomocí subdivision. Wings3d je napsaný v interpretovaném jazyku Erlang. Je určen výhradně pro modelování a spíše pro modely s nižším počtem polygonů. Neobsahuje žádné pokročilejší nástroje a nastavení, proto je vhodné podrobnější věci dokončit v jiném programu *[5]*. Lze v něm však jednoduše a efektivně modelovat i složitější věci a na jeho poměry a velikost se jedná o vynikající nástroj. Pro vytváření byla použita verze 0.98.32a. (viz. *obr. 3.1*) pro OS Windows.



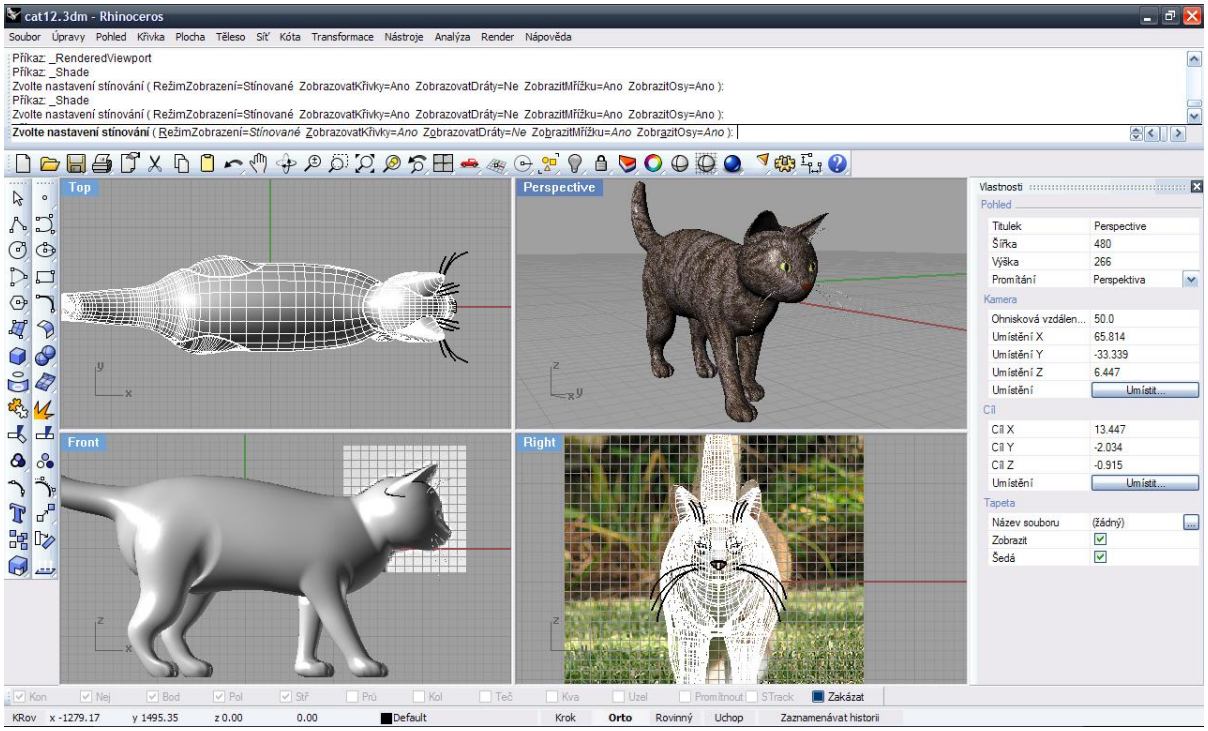
obr. *3.1 ukázka práce v prostředí modeláře Wings 3D.*

3.1.2 Rhinoceros 3D

Nebo také Rhino 3D je samostatný, komerční modelovací nástroj pro tvorbu 3d modelů. Je postaven na modelování pomocí NURBS křivek. Tento program je hojně využíván mnoha průmyslových oborech, architektuře, CAD/CAM systémech, při výrobě prototypů, ale i v multimédiích a grafickém designu. Program je specializován na volné modelování pomocí techniky NURBS. Lze do něj přidat mnoho komerčních i nekomerčních přídavků pro snadnější modelování a rozšíření možností editace. Rhino využívá, jako mnoho modelovacích nástrojů skriptovací jazyk založený na jazyku Visual Basic.

Modelovací nástroj Rhinoceros 3D byl původně distribuován zdarma jako verze open beta. Díky široké skupině jeho uživatelů byl však program odladěn. Bylo přidáno množství funkcí a nástrojů až nakonec vznikla verze, která se používá dnes.

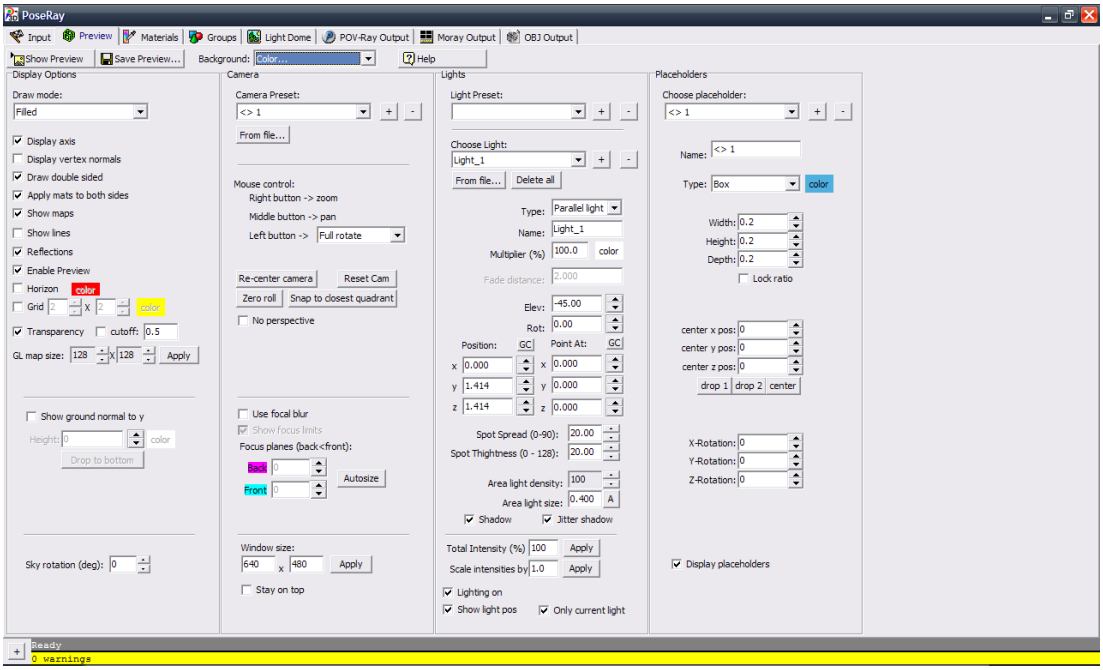
Rhinoceros jsem si vybral hlavně pro jeho různorodost, přehlednost, mnohé zjednodušené funkce a možnost exportu a importu více jak třiceti formátů souborů přímo v editoru (mezi nimiž je značná část open-source formátů), které umožňují také jednoduchý převod modelů mezi programy *[4]*. Rhino3D je podle mých zkušeností vynikající grafický nástroj a bez větších problému v něm jde vymodelovat opravdu cokoliv. Modelovat jsem se v něm naučil pomocí manuálu a různých tutoriálů *[viz. 27]*. Pro tvorbu scény jsem používal studentskou verzi Rhinoceros 4.0, SR1 (viz. *obr. 3.2*).



obr. 3.2 ukázka pracovního prostředí rhino3D 4.0.

3.1.3 PoseRay

PoseRay je freeware program určený pro převod sítí 3d modelu do formátu a scény programu Pov-Ray. Nástroj navíc dovede editovat materiály modelů a lze ho také použít pro jednodušší úpravu geometrie modelů a vlastností scény *[3]*. Tak lze jednoduše upravit vzhled 3d scény před editací a renderu v Pov-Rayi. Používal jsem verzi 3.11.0.436 pro OS Windows (viz. *obr. 3.3*).



obr. 3.3 prostředí programu PoseRay.

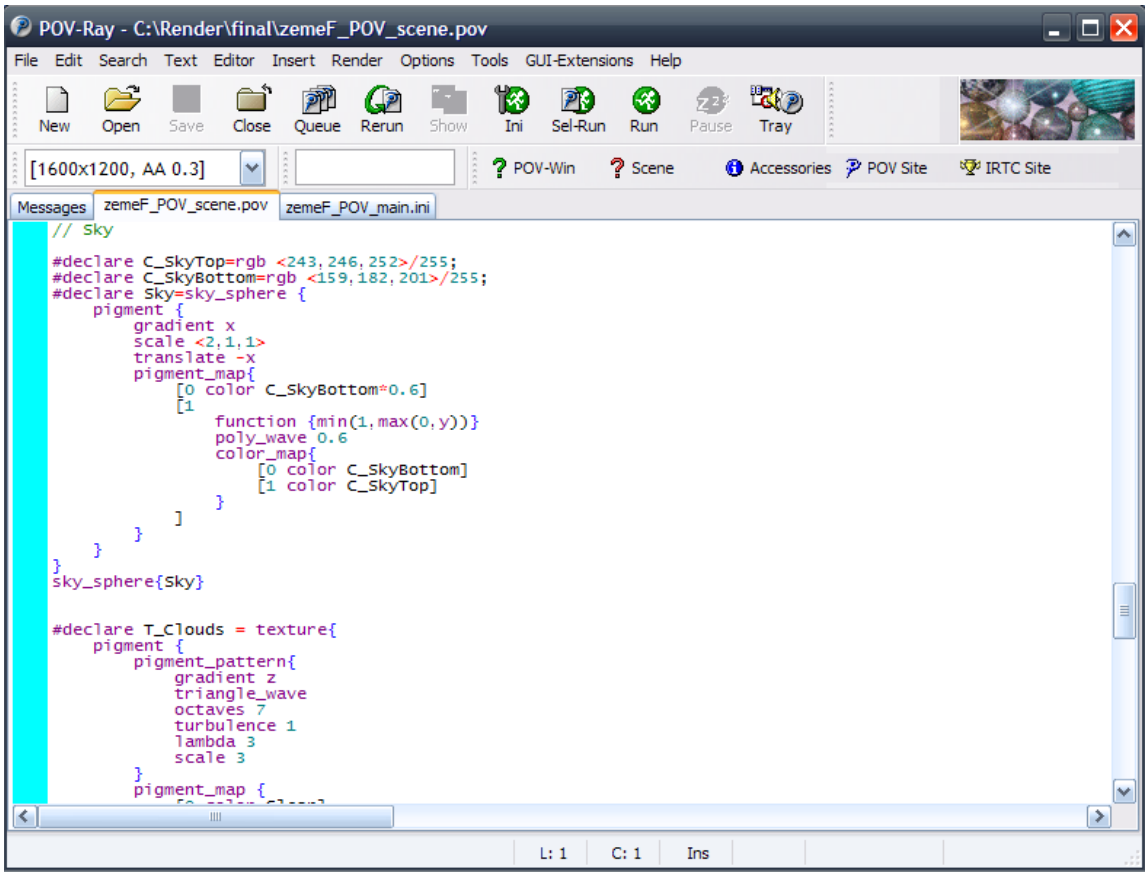
3.1.4 Pov-Ray

Neboli „Perstistence Of Vision Raytracer“ je renderovací freeware open-source program, který využívá ray tracing. Vznikl z DKBTrace - prvního programu využívajícího techniky sledování paprsku *[6]*.

Na rozdíl od jiných 3D programů tento pouze raytracer. Neobsahuje žádné modelovací či animační nástroje a fotorealistické obrázky počítá na základě textových souborů s kódem popisujícím objekty ve scéně, nastavení světel, kamery, atmosférických efektů a dalších. Vlastnosti scény jsou ve zdrojovém textu popsány pomocí jednoduchého jazyka. Vytváření a úprava modelů a scény tedy probíhá pouze v editaci textového (zdrojového) souboru obsahující příkazy.

Program využívá mnoho funkcí, které z něj dělají vynikající a efektivní nástroj, který se bez problému může rovnat i vyspělejším komerčním produktům. Mezi nejzajímavější patří velmi kvalitní výstup, světla, radiosita, photon mapping, UV mapping, particles, množství proceduláních textur, phong stínování, specular highlighting, množství primitiv, atmosférické efekty a mnoho dalších.

Scénu nebo modely zde tedy můžeme buď importovat z jiného programu, nebo přímo vytvářet a upravovat pomocí psaní zdrojového textu. V dnešní době existuje celá řada freeware i komerčních programů, které exportují modely do kódu pro Pov-ray. K editaci jsem používal verzi 3.6, a při renderování 3.7 beta pro OS Windows (s podporou vícejádrových procesorů, viz. *obr. 3.4*) a Linux.



obr. 3.4 ukázka prostředí programu; editace zdrojového kódu.

3.2 Formáty

Pro přenos modelů a parametrů mezi jednotlivými programy jsem používal jisté formáty souborů. Daný formát musel být podporován výstupem programu, z kterého jsem exportoval, tak vstupem dalšího pro import a musel přenášet požadované informace. Popíši zde všechny použité formáty.

3.2.1 3DM

Výchozí formát používaný modelářem Rhino3D, používal jsem ho výhradně pro ukládání modelů právě v tomto programu. Využívá techniku modelování pomocí NURBS křivek. Jde o tzv. OpenNURBS formát. Jedná se o open-source soubor nástrojů, který poskytl nástrojům CAD, CAM, CAE a ostatním vývojářům počítačové grafiky pro přesný převod 3D geometrie mezi aplikacemi.

OpenNURBS v každém nástroji obsahuje:

- specifikaci a dokumentaci formátu
- knihovny zdrojového kódu v C++ pro čtení a zápis do formátu s podporou Windows, Mac a Linux.
- rozhraní .NET 2.0 ke čtení a zápisu 3DM formátu.
- záruku kvality a technickou podporu
- podporu různých knihoven a utilit

Tento formát může obsahovat křivky, sítě, povrchy, obrázky. Tyto datové typy přesně popisují geometrii a podobu modelů *[22]*.

3.2.2 OBJ

Tento formát popisující geometrii objektů byl původně vyvinut firmou Wavefront Technologies pro jeden z jejich programů. Protože se jedná o formát open-source, začal být používaný i ostatními grafickými aplikacemi. V současné době je obecně uznávaný a používaný mnohými volnými i komerčními programy. Formát OBJ jsem použil pro export scény mezi modelovacím nástrojem Rhino3D a programem PoseRay.

Jedná se o jednoduchý formát popisující data reprezentující 3D geometrii. A to pozicí každého vertexu, jeho normály, souřadnice textur propojeny s vertexy a tak i vhled každého polygonu. Soubor OBJ podporuje také např. vyhlazování parametrů a možnost pojmenování skupiny polygonů. Dokáže také popsat materiál objektů a to vytvořením externího souboru MTL (material template library). Tento soubor obsahuje popis přiřazených materiálů jednotlivým objektům, které jsou popsány v souboru OBJ. Ukázka prezentace dat *[24]* a popis vzhledu kódu tohoto formátu viz. příloha *A*, část formát obj.

3.2.3 POV

POV je formát používaný k uložení programovacího jazyku programu POV-Ray popisující scénu. Programovací jazyk POV (Scene Description Language) je používaný k matematické prezentaci obrazových dat, které jsou následně vyrenderovány pomocí programu POV-Ray. Data v tomto jazyku představují sadu popisů objektů a celé scény. Jazyk popisující scénu může být chápán jako PostScript (programovací jazyk určený ke grafickému popisu dokumentů) pro renderované obrázky.

Formát POV jsem získal výstupem z programu PoseRay. Obsahuje jednak samotný soubor POV, ve kterém je hlavní popis scény a jednak externí soubory, ve kterých je popsána geometrie objektů (geom.inc), materiál objektů (mat.inc) a inicializace (nastavení) programu (main.ini). Soubor POV popisu scény obsahuje hlavičku s informacemi o všech ostatních souborech, globální nastavení a základní typy prvků – objekty, kameru a zdroje světla. Soubor geometrie objektů obsahuje souřadnice (sítě) všech objektů ve scéně. V souboru s materiálem jsou pojmenovány a definovány všechny druhy materiálů k objektům definovaným v souboru s geometrií. A nakonec v souboru s inicializací je základní nastavení vstupu, výstupu a základní parametry renderu *[12]*. Prezentace dat a popis zdrojového kódu programu je v části této kapitoly 3.3.4 a v příloze *2*, části scena.pov.

3.2.4 Prezentace dat

Ukázka prezentace dat jednotlivých formátů je popsána v dalších částech práce a v příloze 2. Modely jsou ve všech formátech uloženy pomocí sítě vertexů a jejich souřadnic. Textury jsou prezentovány bitmapami vyexportovanými s formáty nebo dodatečně přidanými.

3.3 Postup a tvorba

Popis jednotlivých kroků a rozvržení postupu vytváření scény.

3.3.1 Návrh

Prvním úkolem bylo navržení samotné scény, do které bych vhodně vytvořil jednotlivé modely. Byly přitom uvažovány vybrané nástroje a jejich možnosti. V první řadě jsem se musel naučit vytvářet ve vybraných programech, protože jsem s žádným neměl předchozí zkušenosti. Vytvořil jsem tedy jednoduchý zkušební model a prošel s ním všechny kroky, kterými bylo potřeba postupovat.

Nejdříve export vytvořeného modelu z programu Rhino3D/Wings3D. Načtení tohoto modelu do konvertoru PoseRay, kde se upravily některé z vlastností modelu a namapovaly na něj textury. Z něho následoval export modelu do renderovacího nástroje Pov-Ray. Zde probíhaly pomocí jazyku tohoto nástroje a úpravy zdrojového kódu závěrečné a asi nejdůležitější nastavení zahrnující korekci modelu, přidání mnoha efektů (světlo) a nastavení pohledu na scénu. Až na několik nepřesností a menších problému byl zkušební v pořádku vyrenderován, a tak jsem mohl pokračovat v návrhu scény.

Po odzkoušení postupu jsem se rozhodl vytvořit scénu, která bude zobrazovat část přírody. To zahrnovalo vytvoření modelů tohoto charakteru (rostliny, zvířata, živly), ze kterých bych scénu poskládal.

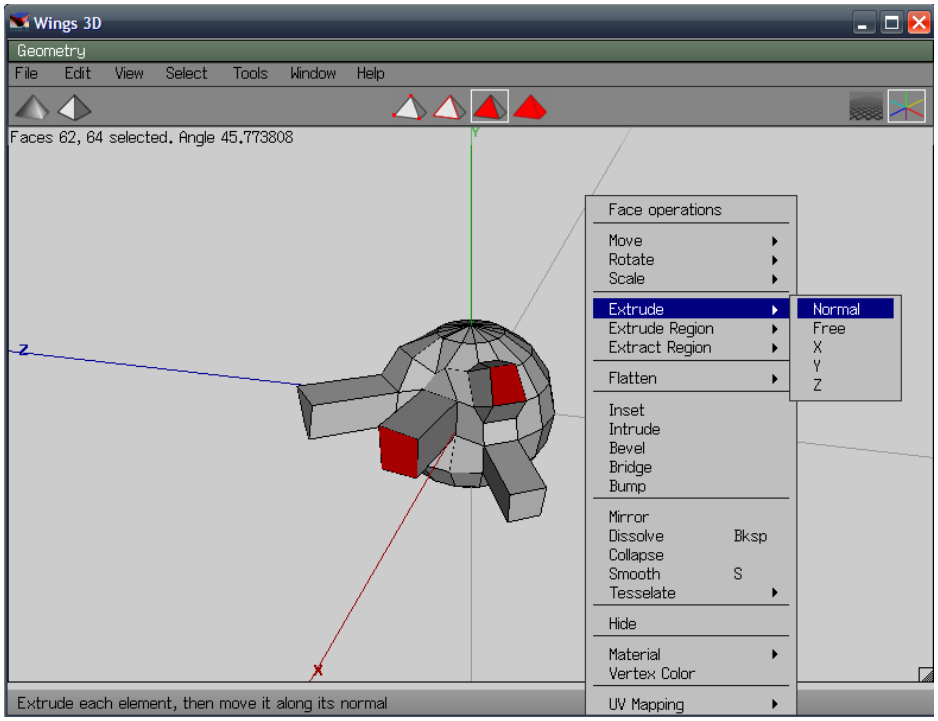
3.3.2 Modelování

Základním krokem práce bylo vytvoření 3D modelů a jejich složení ve scénu. Ty sem vytvářel za pomoci dvou modelářů Rhino3D a Wings3D s tím, že právě v Rhino3D jsem scénu kompletoval, protože nabízí širší možnosti nastavení a jednodušší manipulaci s komplexními modely. Ani s jedním s těchto modelářů jsem neměl předchozí zkušenosti. Celkově jsem pro scénu vytvořil asi 13 detailních modelů (ukázky viz. příloha *B a C*).

Wings 3D

Tento modelář pracuje se základními tělesy, kde pomocí jejich úprav a rozšíření získáme složitější modely (technika subdivision). Dokáže pracovat s více formáty souborů. Výchozí je *wings*, ale pro export a import mezi programy jsem použil formát *obj*.

Jak můžeme vidět na obrázku *3.5*, Wings 3D se skládá z jednoho základního okna s pohledem na model, kde také probíhají všechny operace. Samozřejmostí je přítomnost hlavní lišty s kompletní nabídkou. Nad ním je lišta pro nastavení způsobu zobrazení těles. Zde mimo jiné nastavíme, s čím chceme na modelu pracovat (body, hrany, plochy). Model se zde vytváří rozšiřováním a úpravou bodů, hran nebo polygonů základních těles (koule, hranol, jehlan,..).



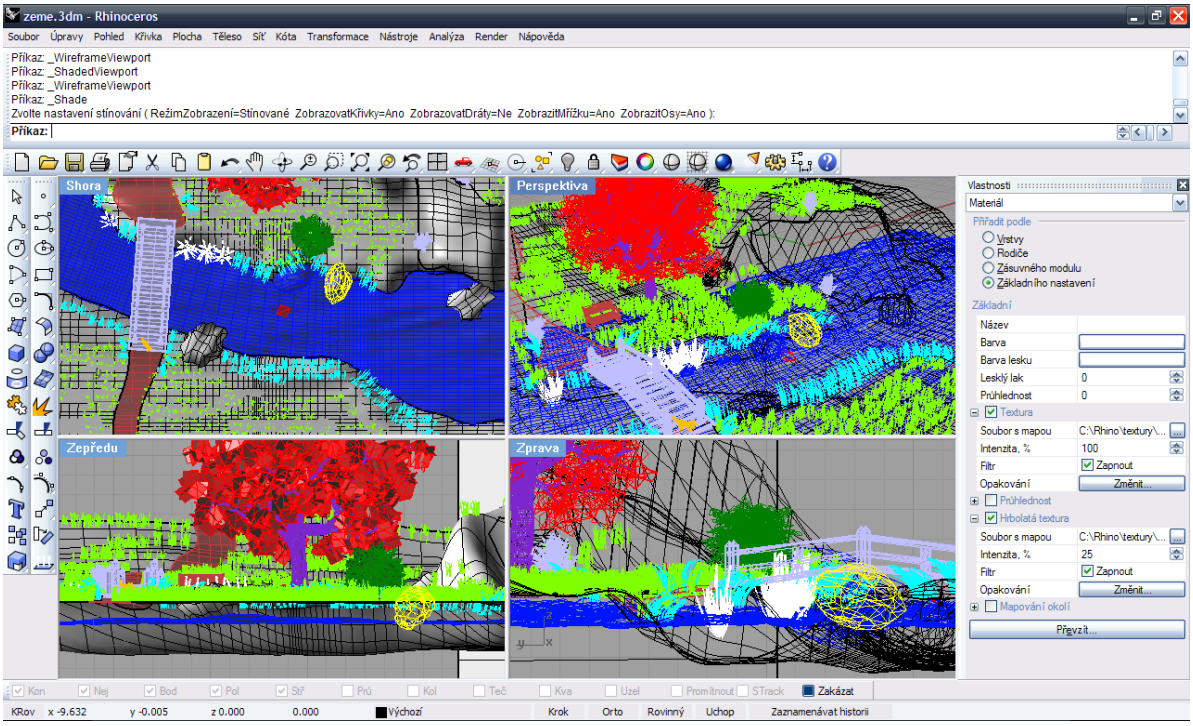
obr. 3.5 práce s programem Wings 3D.

Po zvyku na tuto techniku je práce v modeláři poměrně rychlá a bezproblémová. Program jsem použil pro modelování vybraných objektů, které jsem exportoval do programu Rhino3D pro další úpravy.

Rhino3D

V tomto nástroji jsem vytvořil řadu modelů a všechno spojil ve scénu, čímž byla dokončena modelová část práce. Jednotlivé modely jsem také pokryl pomocí rozšířeného nastavení programu texturami a přidal jim ostatními materiální vlastnosti (např. lesk, průhlednost). Tato nastavení byla prozatím pouze dodatečná pro náhled a přesněji dokončena později v dalších programech. Všechny použité textury jsem vytvořil pomocí bitmapových editorů nebo vyhledal v rozsáhlé databázi na webu *http://www.cgtextures.com/*.

Program se skládá z několika hlavních částí (viz. *obr. 3.6*). Hlavní částí je pohled na model, který představují v základním nastavení čtyři okna s různými úhly pohledu (zepředu, zezadu, shora a perspektiva), ve kterých probíhá modelování. V každém okně můžeme model zobrazit čistě jako síť, s vystínovanými polygony nebo rovnou s použitým materiálem. Napravo od oken je část zobrazující vlastnosti označeného modelu, to zahrnuje jeho vrstvu a materiální vlastnosti včetně nastavení textur. V horní části programu se nachází jeho nezbytná část - příkazový řádek. Do něj můžeme rovnou zadávat příkazy nebo slouží pro nastavení příkazu již provedeného. Je tak reprezentován každý úkon v programu, lze tak každý krok zadávat a upravovat. Ostatními prvky jsou hlavní panel a různé lišty s nástroji, které jdou individuálně nastavit.



obr. 3.6 vývojové prostředí programu Rhinoceros 3D.

Každý objekt je složen z NURBS křivek a celý nástroj je založen na práci s nimi. Při importu jiného formátu, v mém případě *obj*., je model reprezentován polygonovou (trojúhelníkovou) sítí, se kterou se pracuje odlišně než se sítí z NURBS křivek. Pro každý model jsem nastavil jinou vrstvu (barevné odlišení modelů na obr. 3.6). V průběhu vytváření jsem některé tyto vrstvy nezobrazoval, protože celá scéna se stávala značně náročná.

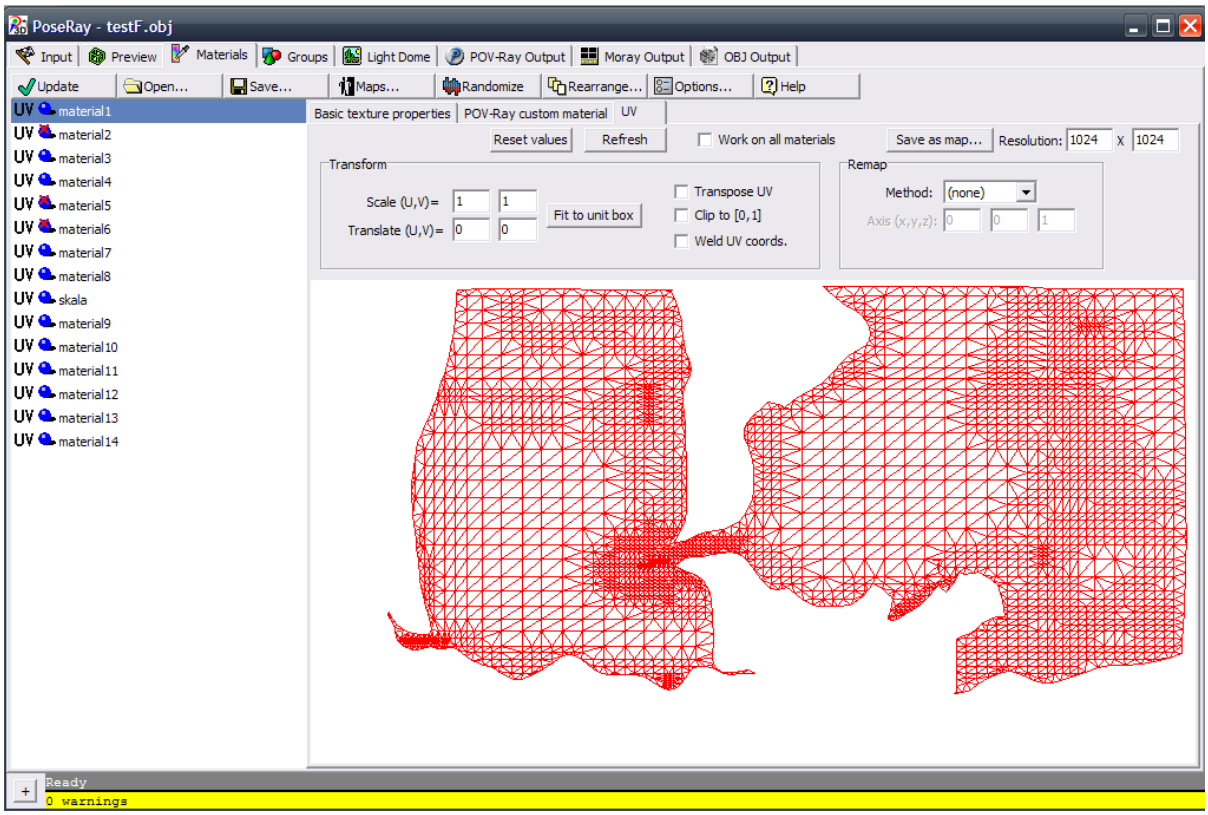
Následující obrázek 3.7 představuje přibližnou podobu scény při exportu z Rhina 3D. Jako výstupní formát scény jsem opět použil *obj*. a v nastavení exportu uložil objekty jako polygonovou síť a exportoval použité textury zároveň s nastavením materiálů v souboru *mtl*. Kompletní výstup z tohoto programu včetně všech souborů je v *příloze 3*.

obr. 3.7 podoba vymodelované scény v Rhino3D.

3.3.3 Převod formátů a úpravy

Pro převod scény z modeláře do renderovacího programu jsem použil program PoseRay. Jako jeho vstup jsem importoval scénu z formátu *obj*. včetně nastavení materiálu v souboru *mtl*.

PoseRay neslouží pouze pro převod formátů. Je to nástroj kde lze podrobně nastavit materiály objektů, kamery, zdroje světla, nastavení výstupu do programu Pov-Ray a spoustu dalšího. Osobně jsem z funkcí programu využil definování vlastností materiálů, přesné namapování textur a nastavení výstupu. PoseRay se skládá z několika záložek různých nastavení viz. *obr. 3.8.*



obr. 3.8 nastavení v programu PoseRay.

Záložka *Input* slouží k importování 3D modelů, to může trvat i několik minut pokud je model rozsáhlý. V záložce *Preview* jde model v náhledové kvalitě zobrazit a lze s ním i volně manipulovat. Je tu i nastavení předvoleb kamer a osvětlení pomocí souřadnic a parametrů. Do scény jsem zde přidal bodový zdroj světla. Asi nejdůležitější pro mne byla záložka *Materials*, kde jsem nastavil materiály všech objektů. Jsou zde vypsány všechny materiály použité v načtené scéně a po vybrání určitého se zobrazí jeho vlastnosti a nastavení.

U každého materiálu je základní nastavení textur (textura, textura pro bump mapping, textura pro průhlednost), vlastnosti objektu (průhlednost, zrcadlení, odraznost) a pokročilé nastavení mapování textur pomocí jejich měřítka (hodnoty UV). Je tu i místo pro deklarování materiálu přímo v jazyce Pov-Ray.

V posledním kroku jsem využil záložku *Pov-Ray Output*. Ta obsahuje nastavení přímo pro renderování v Pov-Ray – nastavení celkové kvality, kvality anti-aliasingu a radiosity, rozlišení výsledného obrázku a další podrobnější nastavování, které jsem nevyužil. Výstupem byl formát *pov* a s ním související soubory s materiály a nastavením včetně všech použitých textur. Výstup z tohoto programu je umístěn v *příloze 3*. Všechna tato nastavení byla předběžná a definitivně se upravila až v programu Pov-Ray.

3.3.4 Dokončení scény a render

Závěrečnou fázi vývoje představovala úprava zdrojového kódu scény v programu Pov-Ray. V něm byly upraveny vlastnosti jednotlivých modelů (materiály) a přidány efekty a vlastnosti celé 3D scény. Po veškerém nastavení byla scéna vyrenderována.

Pov-ray slouží jako editor kódu, ze kterého renderuje obraz. Základními formáty souborů v tomto programu jsou *pov*, *inc* a *ini*. Celou scénu jsem měl vygenerovanou do čtyř navzájem propojených souborů s kódem:

geom.inc

Soubor obsahující geometrii všech modelů ve scéně. Síť každého modelu je zde popsána souřadnicemi všech jeho vertexů (vrcholů). Každá síť má také přiřazeno jméno materiálu, který je pro ni definován. Každá deklarace sítě může obsahovat také ukazatele normál vektorů, vzhled povrchu, seznam textur a jejich UV souřadnice mapování.

Ukázka deklarace materiálu a jeho síť:

```
#declare material1=mesh2{                  //přiřazení materiálu síti
      vertex_vectors{                      //souřadnice všech vertexů
      2900, //celkový počet vertexů
      <-25.21512,-2.847649,4.407805>,
      <-25.23162,-2.694357,4.099485>,
      <-25.50373,-2.826643,4.082568>,..}
      normal_vectors{                      //souřadnice normál
      450,
      <-0.6820017,-7.439362E-15,0.7313506>,
      <-0.6820017,-7.214281E-15,0.7313506>,..}
      uv_vectors{                          //souřadnice mapování textur
      2900,
      <0.101581,0.3785394>,
      <0.101581,0.3850084>,..}
      }
```

mat.inc

Obsahuje definice a nastavení všech použitých materiálů. To zahrnuje textury a ostatní vlastnosti objektu, ke kterému bude toto nastavení přiřazeno.

Definice textur a textur pro bump mapping:

```
#declare p_map1=pigment {image_map{jpeg "SAMPLE.jpg" interpolate 2
transmit all 0 filter all 0} }

#declare p_map2=normal {bump_map{jpeg "SAMPLE_BM.jpg" interpolate 2 }}
//přiřazení a nastavení textur
```

Definice materiálu:

```
#declare material1=material{
            texture{ pigment{ p_map1}     //použití definované textury
            normal{ p_map2 bump_size 1}   //použití bump mapping textury
            finish{
                  specular 1                    //zrcadlení materiálu
                  roughness 0.09632328          //hrubost materiálu
                  ambient rgb <0,0,0>           //barva odrazu
                  diffuse 0.6                   //šíření světla
                  reflection{0 metallic}        //nastavení odrazu světla
                  conserve_energy} }
            }
```

Takto je definován každý materiál obsažený ve scéně.

main.ini

Tento inicializační soubor obsahuje podrobné nastavení renderu, vstupního souboru a formát výstupu renderovaného obrázku.

Vzorové nastavení:

```
//Vstupní Pov-Ray soubor
Input_File_Name="scene.pov"
//Výstupní renderovaný obrázek
Output_File_Name="renderVystup.png"
//Rozlišení renderu
+W1280
+H1024
//Výstupní formát obrázku N - PNG
+FN
//Zapnutí/vypnutí a kvalita Anti-aliasingu (+A, -A, +AM, +AM2)
+A
+A0.3
//Nastavení celkové kvality obrázku (Q0-Q11)
+Q9
//Alpha transparency – nastavení druhu průhlednosti
-UA
//Continuation option – pokračování renderu z místa předchozího ukončení
+C
```

scene.pov

Hlavní soubor celé scény. V první řadě obsahuje knihovny, které jsou zapotřebí, a zařazení ostatních předchozích souborů. Je to spouštěcí soubor a dává celou scénu dohromady. Dále obsahuje objekty jako je globální nastavení scény, kamera, zdroje světla a vše co lze v programu do scény vložit.

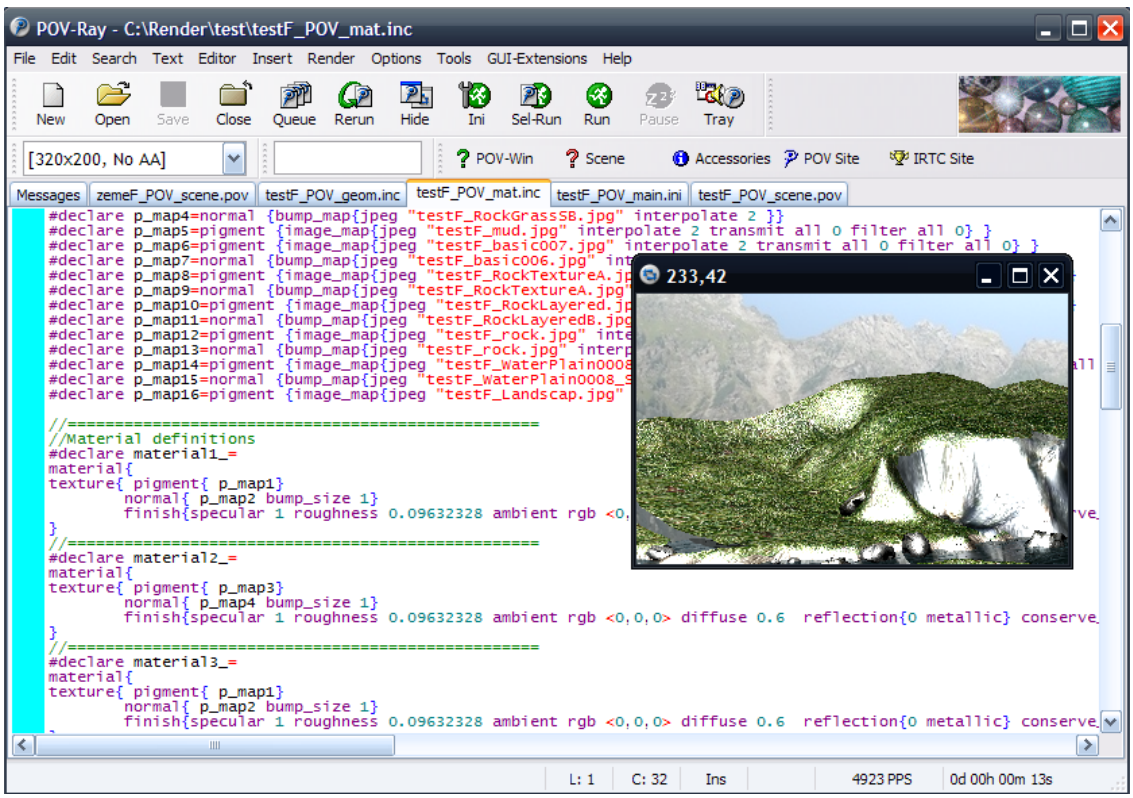
V tomto souboru jsem do scény přidal řadu prvků, kterými byly kamery (úhel pohledu na scénu, rozostření), zdroj světla (slunce), nebe (pozadí scény), mlhu. Popis tohoto souboru a všech objektů tak jak jsem je nastavil je v *příloze 2*, části scene.pov.

V Pov-Rayi jsem musel nastavit a přidat řadu věcí:

- nastavit materiály a mapovat textury ke všem modelům.
- nastavit kameru tak, aby byl pohled vyhovující
- přidat zdroj světla pro co správné osvětlení a vrhání stínů
- přidat pozadí (nebe) scény a ostatní efekty
- správně a vhodně nastavit parametry renderu

Po veškerých úpravách jsem začal renderovat výsledný obrázek (viz. *obr. 3.9*).

Podrobný popis všech použitých příkazů se nachází v manuálu programu Pov-Ray *[12]*. Všechny soubory a zdrojové kódy Pov-raye z renderované scény jsou v příloze *3*.



obr. 3.9 renderovací okno.

3.4 Výsledek a zhodnocení

Výsledkem této práce jsou vytvořené modely (ukázky viz. příloha *B* a *C*), zdrojové kódy na vytvoření scény pro Pov-Ray (obsaženy v příloze *A*) a vyrenderované obrázky ve formátu PNG (viz. příloha *B* a *C*). Renderování konečné scény ve vyšším rozlišení trvalo poměrně dlouho (několik hodin). Celý výpočet byl pomalý, a proto z něj bylo nutné vypustit většinu náročnějších prvků, které výpočet zpomalovaly. To zahrnovalo vypuštění radiosity, atmosférických prvků a jemných stínů. Práci jsem zkoušel renderovat i na školním výpočetním serveru edesign2 ale neúspěšně (z neznámého důvodu nešel spustit výpočet scény). Výsledné obrázky jsou proto v odpovídající kvalitě nárokům.

Během vytváření jsem se setkal i s několika problémy. Asi největší komplikací při tvorbě bylo, že jsem nemohl vidět definitivní vzhled scény před finálním výstupem. Z časových důvodů jsem si nemohl pokaždé úpravě a doplnění dovolit scénu vyrenderovat v plné kvalitě a rozlišení. Proto jsem renderoval pouze po částech a v nízké kvalitě, z čehož nelze dostatečně posoudit kvalitu výslednou.

Dalším problémem bylo například odlišné zobrazení perspektivy v modeláři a v programu Pov-Ray. Kompletace scény probíhala v programu Rhino3D, kde jsem uvažoval model v dané perspektivě. Po náhledu v PoseRay nebo renderu Pov-Ray jsem zjistil, že perspektiva je zde „méně protáhlá“ a do záběru se všechno nevejde tak jak má. Proto jsem musel radikálně změnit nastavení pohledu a přizpůsobit kameru scéně. Ve výsledku tak výsledná scéna vypadá jinak, než jsem předpokládal.

Celou práci jsem vytvářel pod OS Windows XP (s výjimkou testování na serveru edesign - Linux). Kromě závěrečného renderování byla scéna vyvíjena na starším notebooku, proto bylo například složitější modelování nebo převod mezi jednotlivými programy pomalejší a potýkal jsem se tak s řadou komplikací.

Pokud bych měl zhodnotit programy, se kterými jsem pracoval, jedná se o opravdu kvalitní nástroje a lze pomocí nich vytvořit prvotřídní scény srovnatelné s výtvory v profesionálním komerčním softwaru. Vývoj pomocí open-source programů je ovšem celkově náročnější, než by se mohlo zdát. Všechna data, se kterými jsem pracoval a výsledná scéna včetně programů, dokumentace a popisu jsou obsaženy v příloze *3*.

4 Možnosti dalšího vylepšení a rozšíření

V poslední kapitole uvedu, jak by mohla jít moje práce ještě o něco vylepšit. V základu lze říci, že vyhotovení scény je poslední krok, ale možností pro rozšíření 3D modelů nebo renderu scény po dokončení této práce je mnoho. Uvedu některé z nich.

Výpočty byly díky složitějším modelům a celkově vysokému počtu polygonů dosti časově náročné. I při renderu na více jádrových procesorech zabralo minimálně několik hodin a v nejvyšší kvalitě by to mohlo trvat i několik dní. Přesto, že se jedná o vytváření grafiky, pracuje výpočet scény pouze s procesorem. A tak i výkonná grafická karta zůstává v tomto procesu nevyužita. Vzhledem ke stále vzrůstajícím výkonům grafických karet by bylo vhodné do tohoto procesu zapojit i samotné GPU (Graphic processing unit). Doba potřebná pro renderování obrázků by se tak mohla při této akceleraci rapidně snížit. Bohužel Pov-Ray ani jiné renderovací programy podporu grafického procesoru nemají a tak zůstává nevyužit. Zajímavým řešením by byla možnost vypracovat skript pro Pov-Ray, který by umožnil souběžné využití procesoru grafické karty a CPU právě v renderu. Protože je Pov-Ray open-source nástroj a jsou přístupné jeho zdrojové kódy, jistě by šla podpora GPU doplnit.

Další variantou rozšíření práce v Pov-Rayi je vytvořit skripty, které by generovali dané modely v určitém zadaném rozsahu nebo poli. Tato možnost by se využila například pro generování trávy na určenou plochu nebo srsti zvířat. Odpadla by tak práce s těmito objekty přímo v modeláři.

Poslední možnost, kterou jsem uvažoval jako rozšíření by bylo obohacení jednotlivých modelů a tak i celé scény o fyziku (fyzikální vlastnosti). Všechny objekty by získaly svoje těžiště, váhu a pevnost, a podle toho by ve scéně reagovali. V modeláři by tak například stačilo chaoticky umístit modely do scény a po renderu by všechny byly na místě, které by určil jejich fyzikální model. Ten by získaly buď jistým exportem přes vytvořený program nebo opět pomocí skriptu v Pov-Rayi.

5 Závěr

Toto téma jsem si vybral z důvodu osobního zaujetí ve výtvarném umění a zájmu o modelování, vytváření realistických scén a počítačovou grafiku vůbec. Před vypracováním práce jsem měl v tomto oboru jen menší praktické zkušenosti v komerčních modelářích jako je například AutoCAD z hlediska spíše strojařského, technického. Vždy jsem chtěl vymodelovat a vyrenderovat komplexní scénu ale neměl jsem dostatečné znalosti. V průběhu vytváření této bakalářské práce jsem získal mnoho zkušeností právě v této technice, a to od samotného modelování jednotlivých modelů po nastavení způsobu renderování a výstup kompletní scény.

Všechny kroky vývoje probíhaly celkem bez problémů, až na závěrečné kompletování a přiřazení materiálních vlastností tak, aby objekty vypadaly tak, jak požaduji. Největší komplikací byl závěrečný render. Výpočet probíhal velice pomalu, a to i po odstranění náročnějších vlastností scény. Ani na výkonném procesoru se výpočet výrazněji neurychlil. Proto by bylo vhodné rozšířit právě tyto výpočty o akceleraci grafickou kartou, ve které je jistě velký potenciál z hlediska tohoto zobrazování.

Tato bakalářská práce mi však dala mnoho zkušeností jak s 3D modelováním, tak se samotným výpočtem scény právě pomocí open-source nástrojů. A to i přes výstup, který jsem musel omezit a nebyl tak zcela podle mých představ. Získal jsem tak celkovou orientaci v tomto oboru grafiky a zjistil, že vytváření 3D scény není problém, byť pomocí open-source nástrojů, se kterými byl vývoj zřejmě komplikovanější.

Literatura

[1] C.Goral, K.E.Torrance, D.P.Greenberg and B.Battaile. *Modeling the interaction of light between diffuse surfaces.* Computer Graphics, Vol.18, No.3.

[2] *3D Computer graghic* [online]. Wikipedia, poslední modifikace 11.5.2008. [cit. 2008-02-10]. URL: <*http://en.wikipedia.org/wiki/3D_computer_graphics*>.

[3] *PoseRay homepage* [online]. PoseRay, poslední modifikace 11.6.2007. [cit. 2007-11-25]. Dostupné na URL: <*http://mysite.verizon.net/sfg0000*>.

[4] *Rhinoceros modeling* [online]. McNeel products, poslední modifikace 2007. [cit. 2007-11-20]. Dostupné na URL: <*http://www.rhino3d.com*>.

[5] *Wings 3D* [online]. Wings 3D team, poslední modifikace 9.3.2008. [cit. 2007-11-18]. Dostupné na URL: <*http://www.wings3d.com*>.

[6] *POV-Ray raytracer* [online]. Persistence of Vision Raytracer Pty. Ltd., poslední modifikace 1.5.2008 [cit. 2007-11-15]. Dostupné na URL: <*http://www.povray.org*>.

[7] Bill Fleming. *Advanced 3d Photorealism Techniques*. Canada, Wiley Computer Publishing, 1999.

[8] Eric Lengyel. *Mathematics for 3D Game Programming and Computer Graphics, Second edition*. Charles River Media, 2004.

[9] Jenemy Birn. *Digital Lighting and Rendering.* New Riders, 2006.

[10] *Grafické enginy a reálný svět* [online]. Pavel Kovač, Svět Hardware, poslední modifikace 13.2.2008. [cit. 2008-03-20]. URL: <*http://www.svethardware.cz/art_doc-2C877F3E7A25842DC125735B00430161.html*>.

[11] *Global Illumination* [online]. Ambient software, poslední modifikace 2007. [cit. 2008-03-25]. URL: <*http://www.dotray.org/intro_sk.htm*>.

[12] *Dokumentace POV-Ray* [online]. Persistence of Vision Raytracer Pty. Ltd., poslední modifikace 1.5.2008. [cit. 2007-11-18]. Dostupné na URL: <*http://www.povray.org/documentation*>.

[13] *Ray-tracing* [online]. Wikipedia, poslední modifikace 7.5.2008. [cit. 2008-03-10]. URL: <*http://en.wikipedia.org/wiki/Ray_tracing*>.

[14] *Ray-tracing algoritmy* [online]. Wikipedia, poslední modifikace 4.12.2006. [cit. 2008-04-10]. URL: <*http://cs.wikibooks.org/wiki/Raytracing*>.

[15] *Radiosita* [online]. Hugo Elias, poslední modifikace 2000. [cit. 2008-04-17]. URL: <*http://freespace.virgin.net/hugo.elias/radiosity/radiosity.htm*>.

[16] *Radiosita – vzorec* [online]. Wikipedia, poslední modifikace 31.3.2008. [cit. 2008-04-18]. URL: <*http://cs.wikipedia.org/wiki/Radiozita*>.

[17] *Antialiasing methods* [online]. Sudhir R Kaushik. [cit. 2008-04-13] URL: <*http://web.cs.wpi.edu/~matt/courses/cs563/talks/antialiasing/methods.html*>.

[18] *Rendering* [online]. Wikipedia, poslední modifikace 3.5.2008. [cit. 2008-03-28]. URL: <*http://en.wikipedia.org/wiki/Rendering_%28computer_graphics%29*>.

[19] *Raytracing algorithm* [online]. Jamis Buck, poslední modifikace 16.12.1999. [cit. 2008-04-05]. URL: <*http://www.geocities.com/jamisbuck/raytracing.html*>.

[20] *Subdivision surface* [online]. Wikipedia, poslední modifikace 22.2.2008. [cit. 2008-04-16] URL: <*http://en.wikipedia.org/wiki/Subdivision_surface*>.

[21] *Bitmap* [online]. Wikipedia, poslední modifikace 29.4.2008. [cit. 2008-04-02] URL: <*http://en.wikipedia.org/wiki/Bitmap*>.

[22] *Open Nurbs* [online]. OpenNURBS Initiative, Robert McNeel & Associates, poslední modifikace 24.8.2007. [cit. 2008-04-25]. Dostupné na URL*:* <*http://www.opennurbs.org*>.

[23] *POV file format* [online]. Chris Young, POV-Ray Team Coordinator. [cit. 2008-04-01]. Dostupné na URL: <*http://www.fileformat.info/format/pov*>.

[24] *OBJ file format* [online]. Wikiped*ia*, poslední modifikace 8.5.2008. [cit. 2008-04-02] URL: <*http://en.wikipedia.org/wiki/Obj*>.

[25] *Křivky NURBS* [online]. Jana Procházková, Root.cz, poslední modifikace 3.3.2006. [cit. 2008-04-24]. URL: <*http://www.root.cz/clanky/krivky-nurbs-1*>.

[26] *Formát NURBS* [online]. Pavel Tišnovský, Root.cz, poslední modifikace 21.2.2008. [cit. 2008-04-22]. URL: <*http://www.root.cz/clanky/format-x3d-a-zahadna-zkratka-nurbs*>.

[27] *Rhino totoriály* [online], 3D scéna,Grafika publishing, poslední modifikace 20.9.2005. [cit. 2008-01-10]. URL: <*http://www.3dscena.cz/3dshowks.php?xuid=252*>.

Seznam příloh:

Příloha A: Zdrojové kódy a popis

Příloha B: Obrázky a vyrenderované scény

Příloha C: DVD

Příloha A: Zdrojové kódy a popis

Formát obj.

```
# toto je komentář
# popis vertexu se souřadnicemi (x,y,z)
v 0.123 0.234 0.345
v ...
...

#souřadnice textur
vt 0.3 0.2 0.5
...

#souřadnice normál
vn ...
..

#Každý povrch je dán sadou ukazatelů na vertexy/textury/normály
#pomocí pole souřadnic
#Toto je trojúhelník se souřadnicemi textur a normál pro jeho 3 vrcholy
#mající vertex 1 ze seznamu vertexů v, souřadnice textur 2 ze seznamu
#souřadnic textur vt a normálu 3 ze seznamu normál vn.
f 1/1/1 2/2/2 3/3/3
#např.
f v0/vt0/vn0 v1/vt1/vn1 ...
f ...
...
```

Scene.pov

```
//vložení potřebných knihoven a ostatních souborů scény
#include "stdinc.inc"
#include "skies.inc"
#include "stars.inc"
#include "textures.inc"
#include "rad_def.inc"
#include "zemeF_POV_geom.inc" //Geometrie scény

//globální nastavení scény
global_settings {
  assumed_gamma 2         //nastavení jasu
  max_trace_level 100     //maximální počet vysílání paprsků
  //nastaveni radiosity
  radiosity {
    count 10            //počet vysílaných paprsků
    recursion_limit 1   //počet použitých rekurzí šíření
    low_error_factor .5 //kontrola překročení chyb při procházení obrázku
    gray_threshold 0.0  //úprava odstínů pro lepší vzhled
    minimum_reuse 0.01  //nastaví minimalní radius pro znovupoužití vzorku
    brightness 1        //jas
    adc_bailout 0.01/2  //zastavení výpočtu odražených paprsků,které jsou
                        //již bezvýznamné
    brightness 0.5
    normal on           //zapne normálový vektor
    media on            //zapne výpočet atmosferických médií
        }
}
```

```
//nastaveni kamery pohledu na scenu
camera {
        perspective                   //nastavení způsobu promítání
        up <0,1,0>                    //nastavení pohledu
        right -x*image_width/image_height
        location <-10.78332,7.401802,286.0937>
        look_at <-10.78332,7.401802,285.0937>
        angle 7.375931 // horizontal FOV angle
        aperture 3.790801             //rozmazání nezaostřených objektů
        blur_samples 100
        focal_point<0,0,31>
        rotate <0,0,0.5486721> //roll //rotace a posun pohledu kamery
        rotate <-10.36886,0,0> //pitch
        rotate <0,-30.52492,0> //yaw
        translate <14.48085,-8.329885,-21.24378>
        }

//svetelny zdroj
light_source {
      <-55,80.81,-45>           //pozice zdroje světla
      color rgb <1,1,1>*1.6     //barva světla
      area_light                //plošný zdroj světla
      <28.609,0,0> <0,0,28.609>
      10,10
      Jitter    //zdroj světla "roztřepán" pro zamezení ostrých hran stínů
      circular  //zdroj světla upraven na kruhový-zlepšení kvality stínů
      adaptive 1     //adaptivní vzorkování zdrojů světla
      orient         //lepší vytváření "měkkých" stínů
      fade_power 2   //slábnutí světla ve větší vzdálenosti od zdroje
      fade_distance 143.047
      }

//mlha
fog {                         //přidání atmosferického efektu - mlhy
  fog_type   2                //typ
  distance   10               //vzdálenost
  color      rgb 0.6 // gray  //barva mlhy
  fog_offset 0.1              //hustota
  fog_alt    0.2              //výška od počátku
  turbulence 0.8              //vířivost efektu
}

// Sky-přidání oblohy
#declare C_SkyTop=rgb <243,246,252>/255;  //deklarace barev
#declare C_SkyBottom=rgb <159,182,201>/255;
#declare Sky=sky_sphere {                 //deklarace pozadí
    pigment {                             //nastavení vzoru a barvy
        gradient x                        //měřítko a orientace
        scale <2,1,1>
        translate -x
        pigment_map{                    //nastavení definovaných barev
            [0 color C_SkyBottom*0.6]
            [1
                function {min(1,max(0,y))}
                poly_wave 0.6
                color_map{
                    [0 color C_SkyBottom]
                    [1 color C_SkyTop]
                }
            ]
        }}}
```

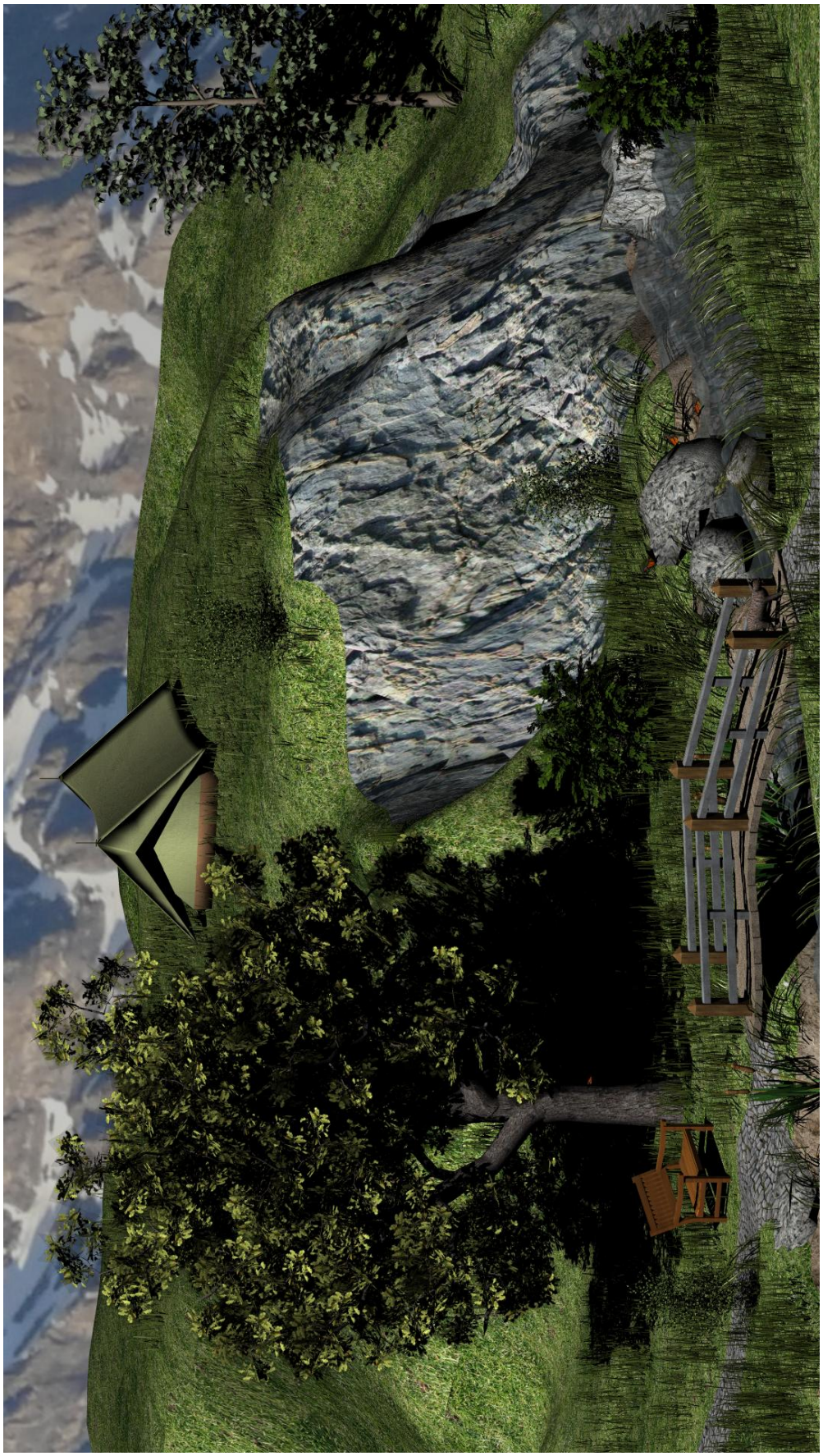
```
sky_sphere{Sky}
//definování kruhové oblasti zobrazení oblohy
#declare T_Clouds = texture{   //deklarace mraků na obloze
    pigment {
        pigment_pattern{       //vzor barvy a jeho nastavení
            gradient z
            triangle_wave
            octaves 7
            turbulence 1
            lambda 3
            scale 3
        }
        pigment_map {           //nastavení barev a vzhledu
            [0 color Clear]
            [0.6 bozo
                color_map {
                    [0.1 White]
                    [0.8 Clear]
                }
                scale 0.5
                turbulence 1
                octaves 6
                omega 0.7
                lambda 2
            ]
            [1 bozo
                color_map {
                    [0 White]
                    [0.4 Clear]
                }
                scale 0.5
                turbulence 1
                octaves 6
                omega 0.7
                lambda 2
            ]

        }
        octaves 6
        omega 0.7
        lambda 3
    }
    finish{ambient 1.1 diffuse 0} //konečné vlastnosti oblohy
}
sphere{0,1                          //nastavení měřítka zobrazení oblohy
    scale <5,6,6>*100000
    texture{T_Clouds scale 30000}
    hollow
}


//Pozadí
background { color rgb<0.9411765,0.9411765,0.9411765>  }
//barva pozadi renderované scény

//Celá scéna jako pojmenovaný objekt v geom.inc
object{
      zemeF_
      }
```

Příloha B: Modely a vyrenderované scény

Ukázky některých modelů

Více ukázek modelů v příloze *C*.

Vyrenderované scény



Více vyrenderovaných scén v příloze *C*.

Příloha C: DVD

Příloha obsahuje tyto složky:

Dokumentace

Obsahuje tuto dokumentaci ve formátu pdf.

Nástroje

Obsahuje instalace programů PoseRay, POV-Ray, Wings 3D a Rhino 3D pro OS Windows.

Obrázky

Vzorové ukázky modelů a obrázky vyrenderované scény.

Soubory a zdrojové kódy

Všechny soubory a zdrojové kódy, se kterými jsem pracoval při vývoji. Je zde scéna uložena ve formátu 3DM pro Rhino3D, výstup z něj ve formátu OBJ a výstup z PoseRaye ve formátu POV do POV-Raye. Všechny formáty uloženy včetně textur a potřebných souborů.

Podrobný popis je v souboru *readme.txt*.