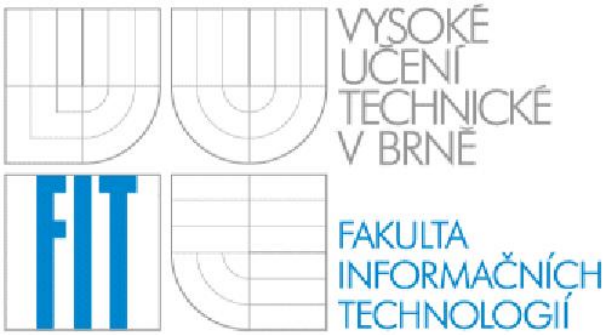


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Ročníkový projekt

2005 **Petr Bulíček**

Poděkování

Chci poděkovat svému vedoucímu Ing. Janu Pečivovi za přátelský přístup a pomoc při řešení obtížných problémů.

Prohlášení

Prohlašuji, že jsem tento ročníkový projekt vypracoval samostatně pod vedením Ing. Jana Pečivy. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....

Petr Bulíček

Abstrakt

Projekt se zabývá návrhem a tvorbou systému generujícího vesmírné těleso požadovaného tvaru, u kterého se podle potřeby mění úroveň detailů. Popisuje použité metody a navrhuje možné rozšíření systému. Jedná se o sytém, který vytváří vesmírné těleso náhodných tvarů, požadované velikosti a členitosti. Tento system byl vytvořen, aby byl použit jako součást akademického projeku SpaceGame. Proto je navržen tak, aby jeho uživatelsý vstup byl co nejjednodušší a rychlost generování těles byla co možná největší.

Program je vytvořen v jazyce C++ s využitím grafického rozhraní OpenGL a knihovny Open Inventor.

Klíčová slova

Vesmírné těleso, úroveň detailů, OpenGL, Open Inventor

Obsah

1 Úvod

Problematika snižování úrovně detailů je v realtime grafice nutností pokud chceme docílit atraktivního vzhledu. Vývoj a výroba výkonnějšího hardwaru jde sice dopredu obrovskou rychlostí, ale ruku v ruce s ním jde dopředu i vývoj vizuálně lepších a hardwarově náročnejších aplikací.

Chceme-li u aplikace, která se renderuje v reálném čase, dosáhnout plynulého pohybu, je zapotřebí minimalizovat počet zpracovávaných primitiv tak, aby výsledek byl z vizuálního hlediska přijatelný. V tomto případě jsou to trojúhelníky a hlavním principem pro změnu úrovně detailů je algormitmus, který tyto trojúhelníky dělí.

V následujících kapitolách se pokusím objasnit, jak tento program pracuje.

V kapitole č.2 uvádím soubory z nichž se projekt skládá a uvádím zde implementační jazyk.

V kapitole č. 3 najdeme informace o rozhraní OpenGL a Open Inventor a o jejich vzájemném postavení.

V kapitole č. 4 je řečeno, co je to vesmírné těleso a cím je tvořeno.

5. kapitola vysvětluje algoritmus na delělení trojúhelníků a jeho implementace.

V kapitole č. 6 najdeme informace důvodu zvyšování detailu, jak k tomu dochází a jak je zvýšení detailu v tomto projektu implementováno.

7. kapitola nám říká, co je to úroveň detailu, proč je používáme a jakým způsobem je přepínáme.

8. kapitola ukazuje použití tohoto programu.

V 9. kapitole jsou stručně a pomocí obrázků zhodnoceny výsledky mojí práce.

10. kapitola popisuje, jak by bylo možno projekt rozšířit a vylepšit.

2 Struktura programu

Program je vytvořen v jazyce C++ s využitím grafického rozhraní OpenGL. Nad rozhraním OpenGL je použita ještě knihovna Open Inventor, sloužící pro tvorbu realtimové grafiky. Samotný systém se skláda ze dvou souborů, create.cpp a create.h.

Jelikož tento systém byl vytvořen aby byl součástí vetšího projektu, nejsou v něm implemetovány žádné možnosti interakce s uživatelem zvenčí. To znamená, že to jak bude generované těleso vypadat, lze ovlivnit pouze změnou parametrů uvnitř programu. Nikoli pomocí nějakého uživatelského rozhraní, nebo zadáním parametrů při spouštění programu.

3 OpenGL v programu

3.1 Co je OpenGL?

OpenGL je softwarové rozhraní ke grafické kartě. Příkazy OpenGL se používají ke specifikaci objektů a operací potřebných k vytvoření trojrozměrné aplikace.

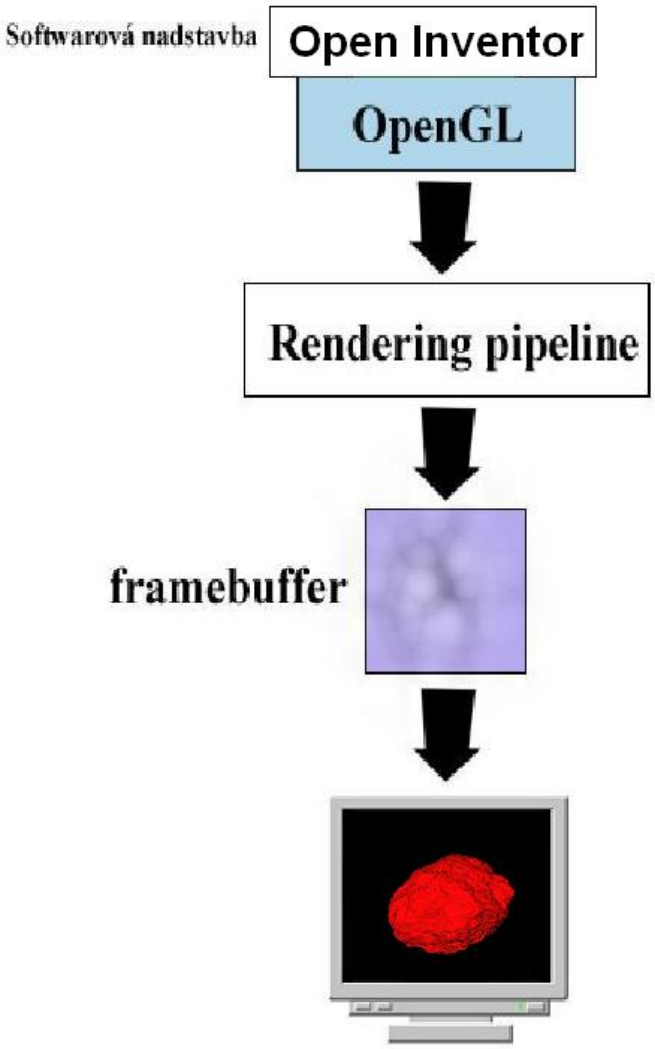
OpenGL je hardwarově nezávislé rozhraní. Dnes je dostupné ve většině operačních systémů (Windows, Unix, Linux, Sun, Irix). Tato hardwarová a platformová nezávislost je umožněna díky nepoužívání příkazů pro práci s okny nebo pro zpracování uživatelského vstupu.

Pomocí OpenGL lze vykreslit model pouze použitím základních příkazů vykreslujícím body, čáry a polygony.

Při zobrazení pomocí této metody jsou na vstupu objekty 3D scény, které jsou převedeny na 2D obrázek pomocí vykreslovacího řetězce (tzv. rendering pipeline viz Obrázek 3). Tento obrázek je uložen v tzv. paměti snímku (framebufferu viz Obrázek 3), odkud se již zobrazuje na monitor. Zobrazení snímku viz Obrázek 3.

OpenGL je tzv. stavový stroj. Pokud nastavíme v programu nějakou vlastnost, pak platí do té doby, dokud není nastavena vlastnost jiná.

Výhodou OpenGL je, že vývojář na její použití nepotřebuje žádnou licenci.



Obrázek 1: Vykreslení pomocí OpenGL a knihovny Open Inventor

3.2 Co je Open Inventor?

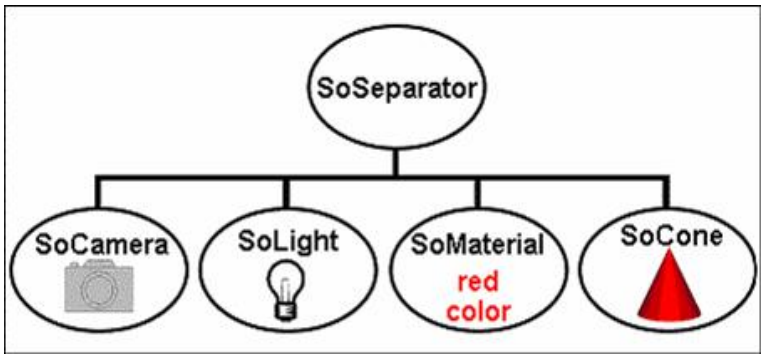
Open Inventor je velmi populární knihovna pro tvorbu realtimové 3D grafiky, tedy i her.

Programátorovi poskytuje rozsáhlou množinu C++ tříd, které skrývají před programátorem vlastní OpenGL API a posunují ho na mnohem vyšší úroveň. Tak může programátor mnohem rychleji vyvinout to, co potřebuje.

Navíc, aplikace napsané v Open Inventoru jsou obyčejně rychlejší než ty přímo psané v OpenGL.

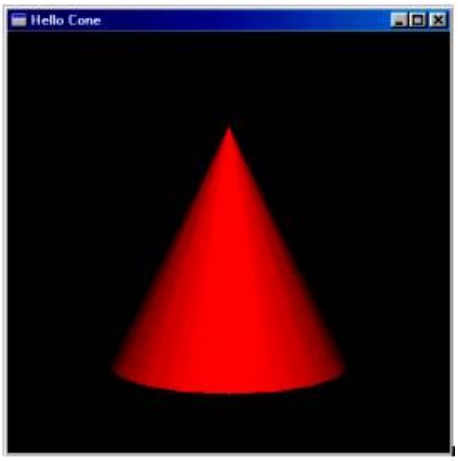
Je rovněž platformově nezávislé, stejně jako OpenGL.

V tomto příkladu vytvoříme minimální aplikaci zobrazující červený kužel osvětlený jedním světlem. Graf scény, zobrazující červený kužel osvětlený jedním světlem, vidíme na obrázku 2.



Obrázek 2: Graf jedoduché scény

Kořen grafu tvoří objekt typu SoSeparator. Když se podíváme pod separátor, zjistíme, že má čtyři syny: kamera, světlo, materiál a kužel. Kamera je nod, který určuje umístění pozorovatele a některé další atributy pohledu do scény. Světlo (SoLight) osvěcuje scénu bílým světlem. Materiál, udává optické vlastnosti kužele, jednoduše řečeno - udává jeho barvu. Posledním nodem je pak vlastní kužel, což je nod specifikující geometrii tělesa. Výsledkem takového grafu je pak scéna znázorněná na obrázku 3.



Obrázek 3: Jednoduchá scéna

3.3 Využití technik OpenGL

Procesu vytváření obrazu scény ve framebufferu říkáme rendering. Data scény jsou pak často výsledkem simulace nějakého modelu viz obrázek 1.

Aby na scéně bylo vůbec něco vidět, je scéna osvícena. Světlo přidá tělesu lesk a osvítí celou scénu. Je použito materiálové nastavení, to definuje vlastnosti povrchu tělesa.

Hlavní stvební jednotkou tělesa je trojúhelník.

Jak jsem se jid dríve zmínil, k těmto možnostem OpenGL, přistupuji skrze knihovnu Open Inventor.

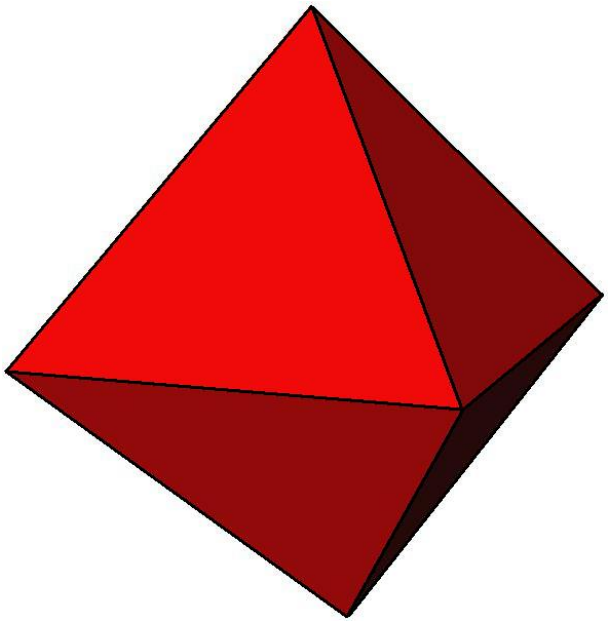
4 Vesmírné těleso

4.1 Co to je vesmírné těleso?

Vtomot projektu se pod pojmem vesmírné těleso myslí jakykoliv organicky objekt, ktery se může ve vesmíru vyskytovat. Takze je to vlastně planeta nebo nejaký úlomek kamene.

4.2 Čím je tvořeno vesmírné těleso

Základ tělesa tvoří osm trojúhelníků (viz obrázek 4), které jsou sestaveny do dvou jehlanů bez základen. Spojením techto dvou jehanů vnikne ojekt pripomínající diamant. Na tyto trojúhelníky je pak aplikován algoritmus, který pro kazdý další vyšší detail rozdelí každý zatím nedrozdelený trojúhelník na ctyři nové a je jimi nahrazen. Toto bude podrobně vysvetleno dále viz. strana 5.

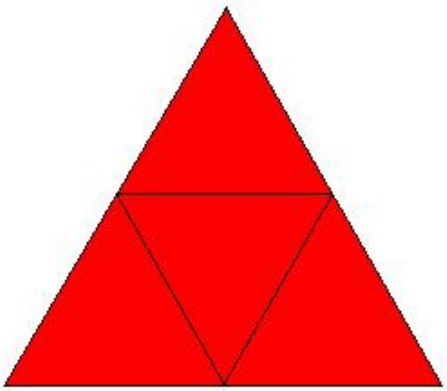


Obrázek 4: Základ vesmírného tělesa

5 Algoritmus na dělení trojúhelníků

5.1 Způsob rozdělení

Základem tělesa je 8 trojúhelníků, jak už jsme si řekli. Základním stavbním prvkem je tedy trojúhelník. Jedinou cestou pro deailnější zobrazení, je zmenšení základního stavebního prvku. Toho lze docílit tak, že jeden trojúhelník nahradíme nekolika menšími. Já jsem zvolil, podle mého názoru, ideální a jednoduché řešení. Spojení středů stran, nám vzniknou z jednoho velkeho 4 menší trojúhelníky, jak je videt na obrázku 5.



Obrázek 5: Rozdělení trojúhelníku

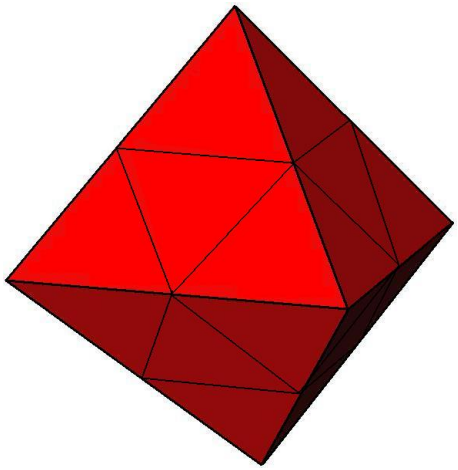
5.2 Implementace rozdělení

Souřadnice vrcholů jednotlivých trojúhelníků jsou uloženy v dynamickém poli. V jiném poli jsou uloženy trojice čísel. Tyto čísla jsou indexy pozic konkrétních bodů. Pro kazdý trojúhelník je v tomto poli trojice indexů. Algoritmus si z pole vyjímá tyto troice, spočítá středy stran podle konktétních bodů na které indexy odkazují a z těchto šesti bodů (3 body jsou vrcholy původního trojúhelníku a 3 body jsou spočtené středy stran tohoto trojúhelníku) sestaví 4 nové trojúhelníky. Do pole vrcholů se přidají 3 nově vzniklé body a do nového dynamického pole se ukládají nově vzniklé trojice indexů, které představují odkazy na vrcholy nově vznikých trojúhelníků. Toto je provedeno pro všechny trojice z původního pole indexů a původní pole je tím nově vzniklým nahrazeno.

6 Zvýšení detailu

6.1 Docílení vyššího detailu

Rozdělením vetšího trojúhelníku na několik menších jsme sice zvýšili počet stavebních jednotek, ze kterých se těleso skládá a můžeme docílit toho, aby bylo zobrazeno detailněji, ale zatím jsme toho nedocílili. V tomto okamžiku máme sice více trojúhelníku, ale těleso vypadá stále naprosto stejně jak před aplikací dělícího algoritmu. Každé 4 nové trojůhelníky totiž tvoří jednu plochu jak je vidět na orázku 6. Je tedy potřeba nově vzniklé vrcholy posunout tak, aby neleželi na hraně původního trojúhelniku. V tomto projektu se posunutí provede po přímce spojující tento bod se středem tělesa smerem ke středu nebo od něj.



Obrázek 6: Základní těleso po dělení trojúhelníku

6.2 Kam posunout nový bod?

Způsob, jakým budeme body posouvat, nám určí výsledný vzhled tělesa. Vzdálenost všech vrcholů základního tělesa od středu je stejná. Pokud budeme všechny body posouvat na stejnou vzdálenost od středu tělesa a pokud je tato vzdálenost rovna vzdálenosti vrcholů základního tělesa, pomom výsledným tělesem bude koule. Toto je nejjednodušší případ. Chceme-li vytvořit těleso, které vypadá jako úlomek kamene, nebo jako planeta(koule se zvlneným povrchem), je situace složitejší. Nové body musíme posouvat o pseudonáhodné vzdálenosti.

To znamená, že bude posunut o vzdálenost, která bude náhodně vybrána z určitého intervalu. Pro jednu úroven detailů bude interval stejný.

Čím vyšší uroveň detailů, tím bude interval menší, z toho vyplývá, že pro vetší detail bude dochazet k menším změnám ve tvaru tělesa. Tím docílíme vizuálně atraktivnějšího vyhlazeného vzhledu tělesa.

6.3 Implementace posunu bodu

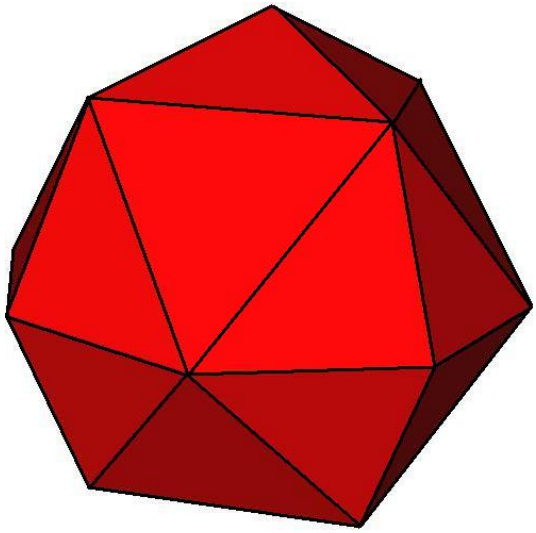
Již při dělení se kazdý nový bod odsnune od středu na vzdálenost, která je rovna průměrné vzdálenosti od středu bodů, jejichž spojnici jsme rozdelili pro vznik tohoto nového bodu.

Budeme procházet pole, ve kterém jsou uloženy všechny vrcholy. Prochazíme jej od místa, kde začínají dosud neposunuté body do konce.

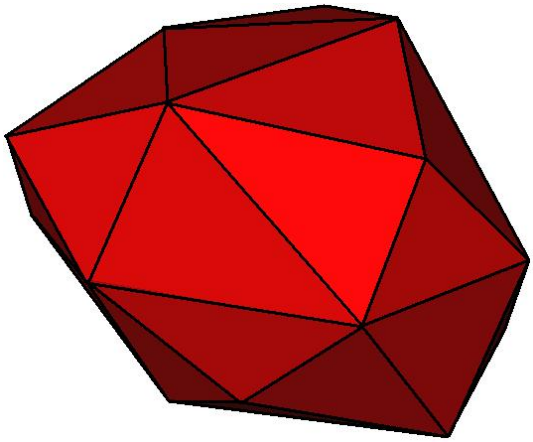
Rozptyl nám udává horní hranici intervalu, ze kterého se bude náhodně vybírat hodnota, o kterou se bod bude posouvat. Spodní hranice se vypocítá z horní hranice tak, aby horní i spodní hranice byla od současné polohy bodu stejně vzdálené, ale každá v opačném směru.

Pokud ma být výsledným tělesem koule, tj. pokud je parametr rozptylu roven nule, nebude se bod posouvat vubec. Jelikož body, z jeichž průměrné vzdálenosti se počítá vzdálenost tohoto bodu je stejná, bude nový bod stejně daleko od středu jak tyto 2 body a proto výsledné těleso nemuže být nic jiného než koule. Jak těleso vypádá po prvním dělení, je-li rozptyl roven nule vidíme na obrázku 7.

Pokud je rozptyl větší než nula, vybere se pro kazdý nový bod náhodně hodnota vzdálenosti z intervalu, se z hodnoty rozptylu vypočítá. A na tuto vzdálenost od středu tělesa je nový bod posunut. Pro kazdou následujuící úroveň detailu je hodnota rozptylu dělena dvěma z důvodu uvedeného v kapitole 6.2. Jak těleso vypádá po prvním dělení, je-li rozptyl roven 750 vidíme na obrázku 7.



Obrázek 7: Základní těleso po dělení trojúhelníku a odsunutí bodů, je-li rozptyl roven 0



Obrázek 8: Základní těleso po dělení trojúhelníku a odsunutí bodů, je-li rozptyl roven 750

7 Úrovně detailů

7.1 Co to je úroveň detailů

Výše jsem popsal, jak ze základního tělesa vytvářím další tělesa tak, že dělíme trojúhelníky a posouváme nově vzniklé body. Takže nám vnikně nekolik těles.

Tato telesa mají základní tvarové rysy shodné. Kazdé takto vniklé těleso je úrovní detailu jednoho jediného tělesa. Takže všechny takto vzniklá tělesa slouží k vyobrazení jednoho tělesa. Zobrazeno je vždy pouze jedno z nich o tom, které z nich to bude bude popsáno dále.

7.2 Proč používat úrovně detailů

Otázkou zůstává, proč vlastně vytvářet několik těles, když bude zobrazeno pouze jedno z nich. Proč si nevybrat pouze jedno z nich a ostatní nevypustit. Odpověď je jednoduchá.

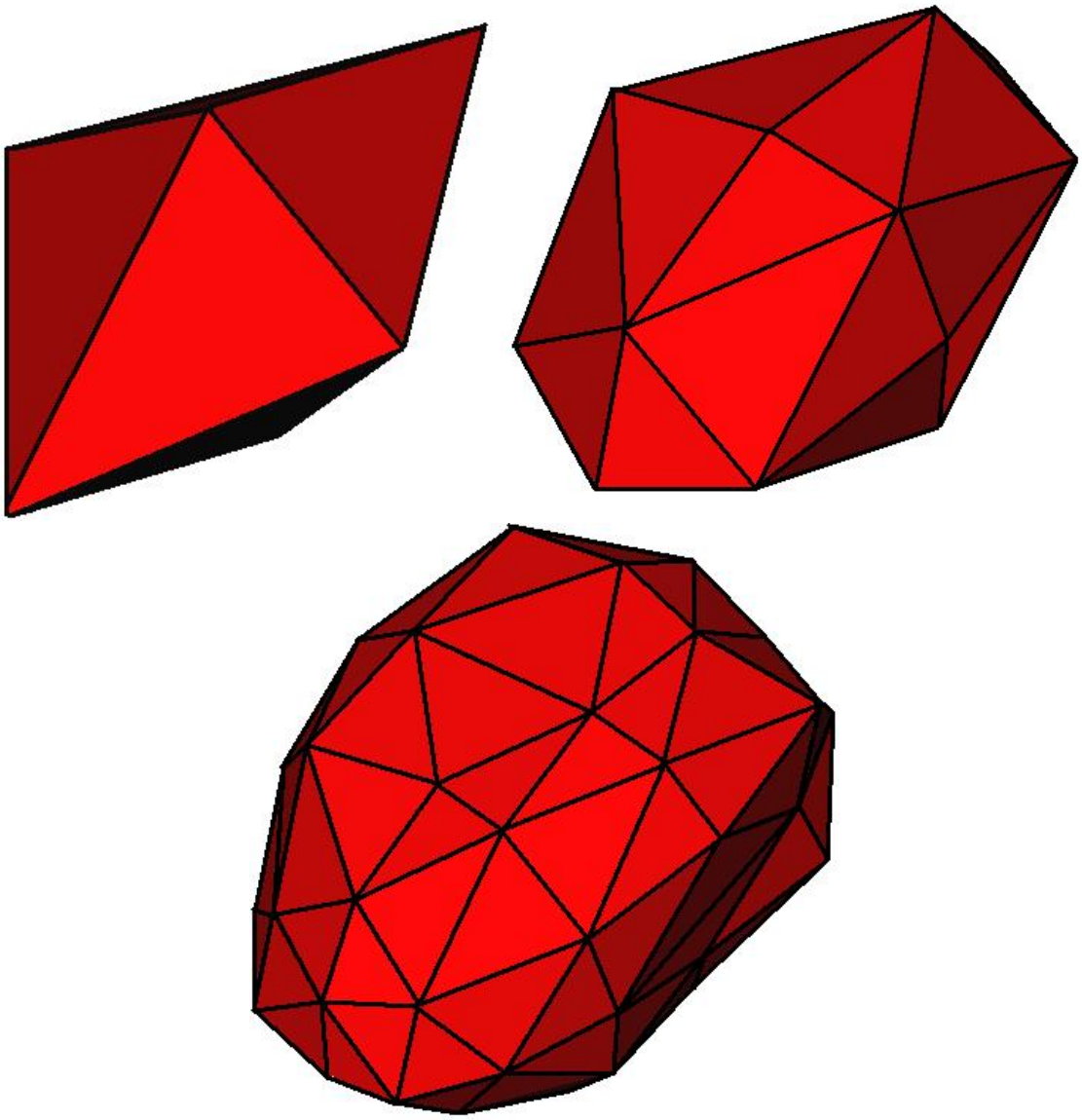
Chceme-li dosáhnout vizualně atraktivního zobrazení tělesa a zároven plynulého chodu aplikace, nelze to dělat jinak než používat více těles a vhodně tato tělesa měnit.

Kdybychom totiž meli pouze těleso s nevyšším detailem, tj. největším počtem trojúhelníků, tak by aplikace při větším počtu takovýchto těles na scéně rozhodně nebyla plynylá. Pokud bychom meli pouze těleso s nízkým detailem, tak by aplikace byla plynulá i při velkém počtu těchto těles na scéně, ale o vizuální atraktivnosti by nemohla být řeč. Proto je nutný tento kompromis.

Pokud bude těleso hodně vzdálené od kamery, bude zobrazeno těleso s nejmensím detailem. Kdyz se bude přibližovat, bude postupne nahrazováno tělesy s detailem vyšším. Tím si zajistíme, že se těleso bude vypadat stále atraktivně a zároven se bude zobrazovat pouze tolik trojúhelníků, kolik bude nezbytně nutné.

Já sem implementoval 7 úrovní detailů.

Na obrázku 9 jsou ukázány 3 úrovně detailů, vidime na něm, jak velký přírustek trojůhelníků v každém dalším detailu je.



Obrázek 9: Ukázka tří úrovní detailů

7.3 Metody přepínání úrovní detailů

Knihovna Open Inventor ma implementovány dvě třídy pro přepínání mezi tělesy.

První z nich je *SoLOD*. Tato třída přepíná tělesa, na základe jejich vzdálenosti od kamery. Intervaly vzdáleností ve kterých budou jednotlivá tělesa zobrazena se musi ručně nastavit, tak aby to bylo přijatelné.

Toto není pro náš případ příliš vhodné, protože tento způsob je závislý na velikosti okna, ve kterém je scéna zobrazena. Zmeníme-li velikost okna, budou se sice modely porád přepínat ve vzdálenostech, které jsme si nastavili, ale vizuální dojem stejný nebude.

Druhou trídou je *SoLevelOfDetail*. Ta přepína tělesa podle počtu pixelů, které na obrazovce zaujímá pomyslný kvádr, který obaluje těleso. Rovněž zde je potřeba ručně zadat intervaly poctu pixelů, ve kterých budou jednotlivá tělesa zobrazena. Ale je zřejmé, že tento způsob je na velikosti okna ve kterém bude scena zobrazena naprosto nezavislý.

Přestože je tento způsob výpočetně náročnejší než ten předešlý, tak jsem ho použil, protože se pro tento projekt výborně hodí.

V projektu jsou ze zkušebních důvodů implementovány způsoby oba, ale primárně se pouzívá třída *SoLevelOfDetail*.

8 Používání programu

Když chceme tento program použít, musíme si vytvorit instanci třídy *create*. A přidat do kořene scény přidat potomka, kterého vrací metoda *createObject*.

Tato metoda má dva parametry. Prvím určíme, jaký poloměr bude výsledné těleso mít. Druhým parametrem nastavíme hodnotu proměnné rozptyl. Pokud bude tento parametr roven nule, bude výsledné těleso koule. Čím vetsí hodnoty bude nabývat, tím vetší bude členitost výsledného tělesa.

Zdrojový kód, který zajistí zobrazení tělesa, které má tvar koule o polomeru 2 by mohl vypadat asi takto:

```
#ifdef _WIN32
    HWND window = SoWin::init(argv[0]);
#else
    QWidget window = SoQt::init(argv[0]);
    if (window == NULL) exit(1);
#endif
// koren sceny
SoSeparator *root = new SoSeparator;
root->ref();
//Material
SoMaterial *material = new SoMaterial;
material->diffuseColor.setValue(1.0, 0.0, 0.0);
root->addChild(material);
create teleso;
root->addChild(teleso.createObject(2,0));
```

```
// okno prohlizece
#ifdef _WIN32
    SoWinExaminerViewer viewer = new SoWinExaminerViewer(window);
#else
    SoQtExaminerViewer viewer = new SoQtExaminerViewer(window);
#endif
viewer->setSceneGraph(root);
viewer->setTitle("Deleni trojuhelniku");
//viewer->setHeadlight(TRUE);
viewer->show();
// renderovaci smycka
#ifdef _WIN32
    SoWin::show(window);
    SoWin::mainLoop();
#else
    SoQt::show(window);
    SoQt::mainLoop();
#endif
// uvolneni pameti
delete viewer;
root->unref();
```

9 Vyhodnocení výsledků

Pokoušel jsem se vytvořit systém, který generuje tělesa, která pripomínají reálná vesmírná tělesa, která mění svůj detail tak, aby vypadala porád pokud možno reálně a aby byla zachována plynulost aplikace.

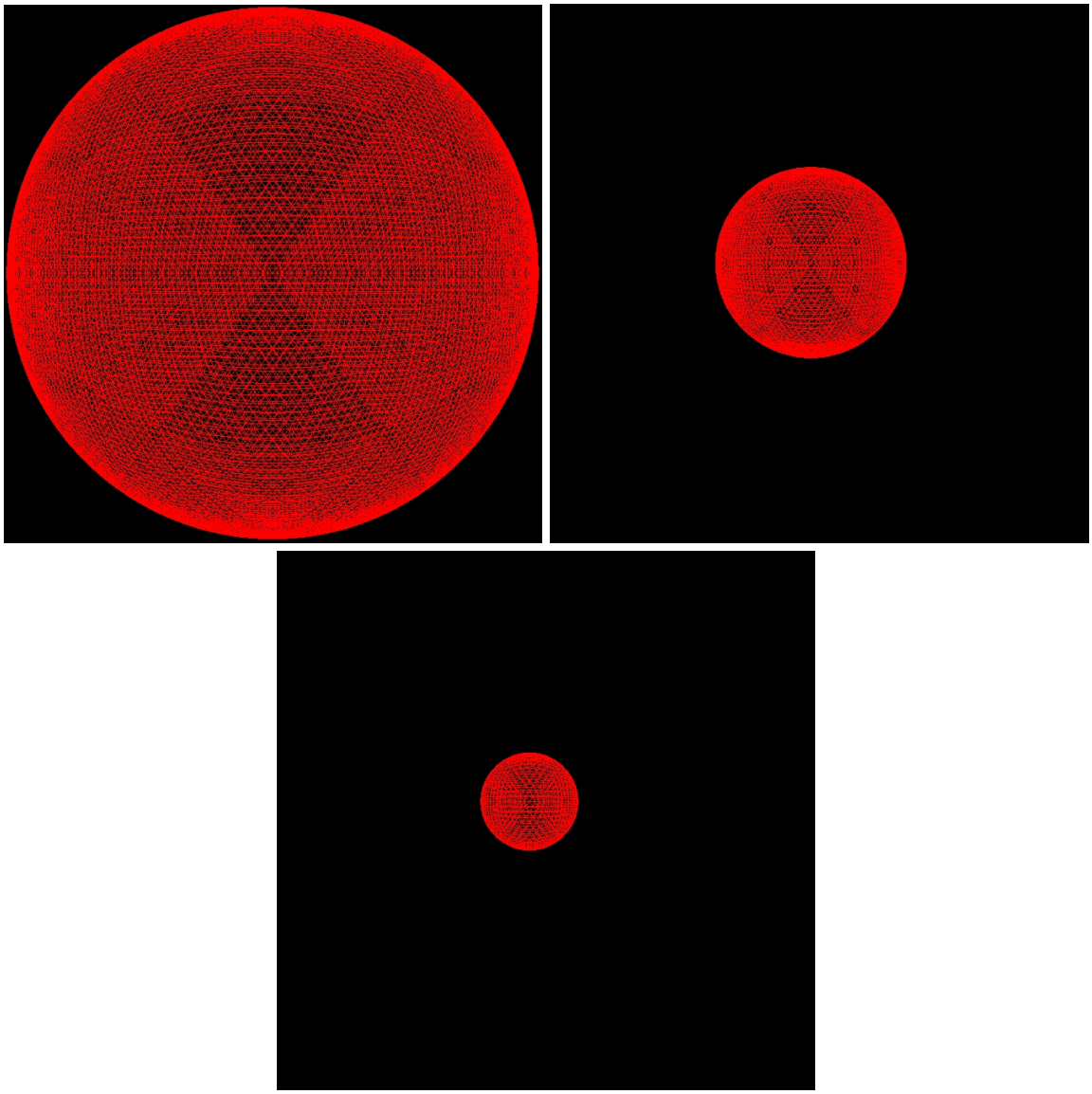
Jsem si vědom,že je na projektu ještě zapracovat, aby generoval tělesa, která se realitě alespoň blíží. Na čem se dá jestě pravovat je nastíněno v následující kapitole.

Nejhez4í tělesa jsou generována, je-li prametr rozptyl z intervalu <0-1000>.

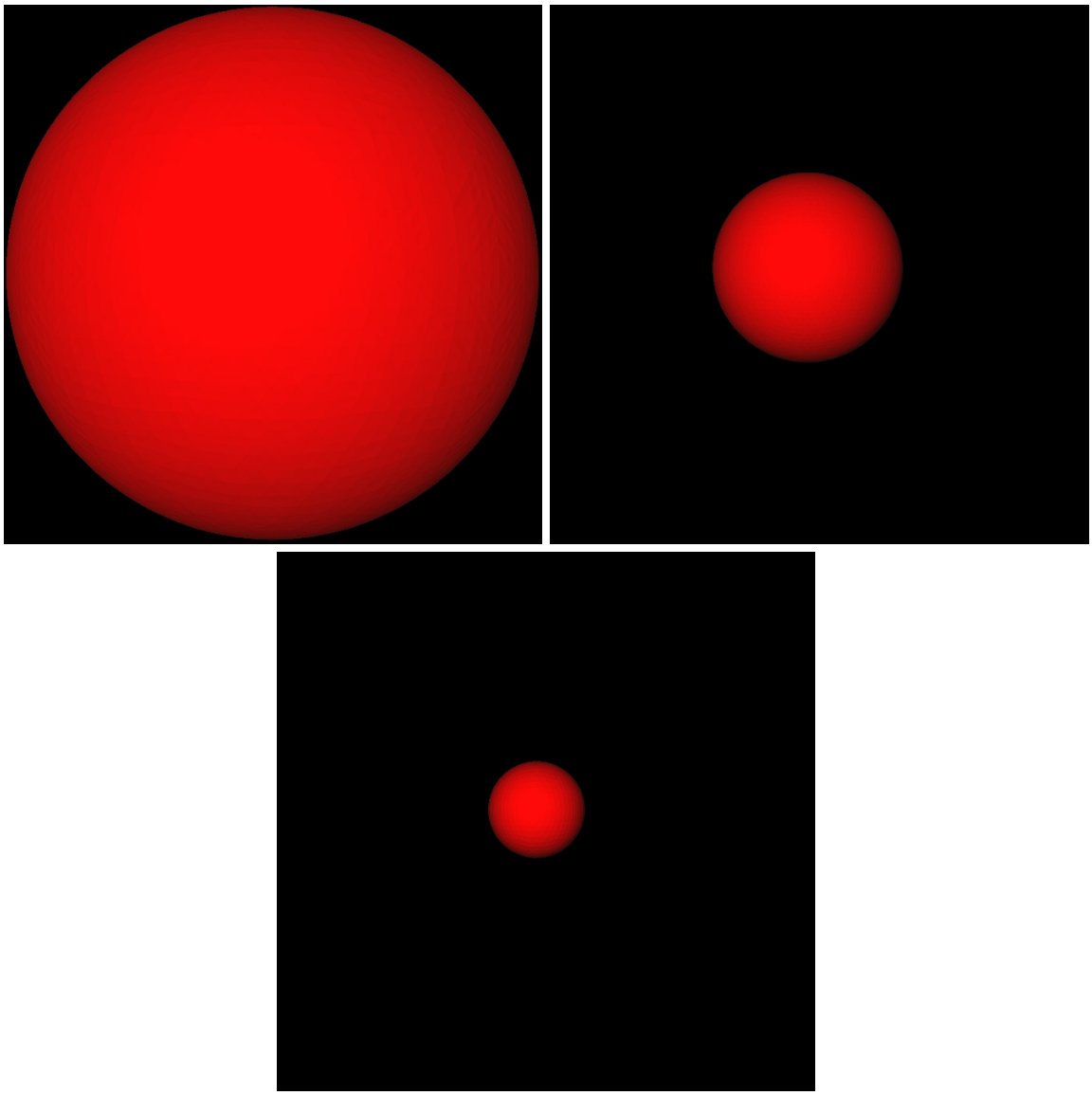
V tomto odstavci se na obrazcích (obrázky 10 a 11) pokusím demonstrovat výsledky mojí práce. Je na nich zachyceno těleso, které se vzdaluje od kamery.

Z drátěných modelů na obrázku 10 vidíme, že hustota drátěné sítě je u všech približně stejná, což dokazuje, že pocet trojúhelníků ze kterých se teleso skládá se zmenšuje se zvětšující se vzdáleností od kamery.

Na obrázku 11 je vidět, že přesto, že je počet trojúhelníků menší, těleso je stále stejně kvalitně vyobrazeno.



Obrázek 10: Drátěné modely tělesa, které se vzdaluje od kamery



Obrázek 11: Modely tělesa, které se vzdaluje od kamery

10 Plány do budoucna

I když generovaná tělesa jsou tvarově vcelku uspokojivá, musí se udělat ještě spousta práce proto, aby vypadala alespoň částečně reálně.

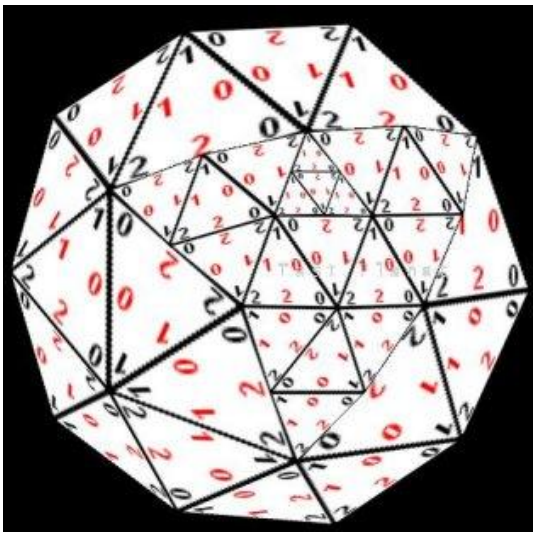
Prvním takovým krokem k reálně vyhlížejícím vesmírným tělesům jsou textury.

Dalším možným krokem by mohlo být upravení generování pseudonáhodných čísel, o jejichž hodnoty se posouvají nove vniklé body tak, aby jsme mohli přesneji určit výsledný tvar generovaného tělesa.

Mohlo být implementováno, aby uživatel používající tento program mohl parametrem nastavit, kolik úrovní detailů požaduje.

Pro zvýšení rychlosti behu aplikace by mohl být implementován lepší algoritmus na dělení trojúhelníků a to takový, který by nedělil vsechny trojúhlníky tělesa, ale jen trojúhelníky, které dosáhli urcité velikost viz obrazek 12.

A v neposlední řadě by mohl být optimalizován a urychlen algoritmus, který ze základního tělesa vypočítá úrovně jeho detailů.



Obrázek 12: Lepší algoritmus na dělení trojúhelníků

11 Závěr

Na základě požadavků byl vypracován funkční generátor vesmírných těles, která podle pořeby mění počet trojúhelníků, ze kterých jsou slozena.

Použití projektu je popsáno v kapitole 8. Lze jej rošírit na diplomový, například implementací rozšíření uvedených v kapitole 10.

Projekt jsem si vybral, protože počítačová grafika mě zajímá a ta realtimová obzvlášť.

Tento projekt je přínosem, zejmená pro vývojáře her, protože snižování detailů je nutností pro zachování plynulosti pohybu v realtimových aplikacích.

Při vytváření dokumentace jsem se inspiroval z prací mých starších kolegů.

Zadání projektu bylo z mého pohledu úspěšně splneno.

12 Literatura

[1] Hlavenka, J. a kol.: Výkladový slovník výpočetní techniky a komunikací.
Computer Press 1997
ISBN 80-7226-023-5

[2] Virius, M.: Od C k C++.
Nakladatelství Kopp 2000
ISBN 80-7232-110-2

[3] Přednášky a cvičení z předmětu Počítačová grafika.
http://www.fit.vutbr.cz/study/courses/index.php?id=366

[4] Open Inventor - tutoriály
http://www.root.cz/clanky/open-inventor

[5] Algoritmus na dělení trojúhelníků
http://www.gamedev.net/reference/articles/article2074.asp

[6] Ivo Chvojka - Ročníkový projekt

13 Přílohy

CD-ROM se zdrojovými kódy.

14 Seznam obrázků