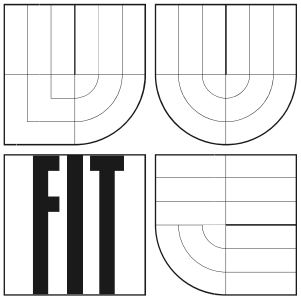
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Modelovanie 3D scény a jej vizualizácia

Bakalárska práca

2005 Miloš Palguta

Modelovanie 3D scény a jej vizualizácia

Odovzdané na Fakultě informačních technologií Vysokého učení technického v Brně, dňa 1. mája 2005.

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Jána Pečivu.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Miloš Palguta
30.4.2005

Abstrakt

Tento dokument pojednáva o vypracovaní bakalárskej práce. Je tu opísaný program, v ktorom boli vytvorené modely, zároveň je tu vysvetlené, čo to je model v 3D virtualnom svete. Zaoberá sa teda tvorbou objektov do ešte stále vyvíjajucej sa vesmírnej hry Space Game. Sú tu podrobnejšie prezkúmané rôzne techniky modelovania, mapovania a tvorba textúr. Zároveň je tu preskúmaná problematika exportu z modelovacieho softwaru a importu do softwaru, ktorý bude scénu zobrazovať. Druhá časť dokumentu sa zaoberá hierarchickou animáciou modelov, teda viacnásobnými transformáciami. Konkrétne sa bude jednať o rotácie, pretože na nich je názorne vidieť tieto transformácie.

Kľúčové slová

Modelovanie, mapovanie, textúrovanie, transformácia, animácia, model, shrink wrap, polygón, model, hierarchická sústava, rotácia, translácia, zmena mierky, zkosenie

Poďakovanie

Tu sa chcem sa poďakovať všetkým, ktorí mi na vypracovaní bakalárskej práce akokoľvek pomohli. Obzvlášť ďakujem môjmu vedúcemu Jánovi Pečivovi za jeho ochotu pomôcť mi so všetkými problémami, s ktorými som sa na neho obrátil.

Abstract

This document concerns about elaboration bachelor’s work. A program in which were created models, is described here. 3D model in virtual world in association with this program is here explained too. So this document deals with creation of models, that will be used in actually developing space simulation game Space Game. There are intimately examined different techniques of modeling and mapping objects. Creation of textures is described here too. The problems of export in association with modeling program and import in association with display program are here descrived too. Second part of this document is dedicated to hierarchical animation of models. It means multiple transformations. Rotation transformations are here described specifically, because we can see it clearly on this type of transformations.

Keywords

Modeling, Mapping, Texturing, Transformation, Animation, Model, Shrink Wrap, Polygon, Model, Hierarchic System, Rotation, Translation, Scale, Skew

Obsah

1 Úvod

1.1 Uvedenie do problematiky

Grafika je jednou z najrýchlejšie rozvíjajúcich sa odvetví v obore informatiky. Zaoberá sa spracovaním rôznych informácii a ich reprezentáciou pomocou obrazu. Grafické podanie informácie je, narozdiel od textovej formy informácie, pre užívateľa omnoho prijateľnejšie a príjemejšie. Napríklad ak by si mal užívateľ vybrať, či preskúma rôzne štatistiky, ktoré sú reprezentované číslami, alebo pozrieť si veľmi prehľadné grafy, tak je takmer isté, že užívateľ si zvolí grafickú podobu informácie. Grafický obraz ako taký je pre ľudské oko prirodzenejší a preto si z neho dokáže užívateľ vytiahnuť potrebné informácie oveľa skôr, ako by to mal nájsť v texte.

Aj z podobných dôvodov sa prešlo z operačných systémov, ktoré boli reprezentované textovou formou (MS-DOS), na operačné systémy, ktoré reprezentujú mnoho informácii pomocou grafiky (MS-WINDOWS).

S veľmi rýchlo rozvíjajucou sa grafikou, sa o počítačovú sféru začal zaujímať aj zábavný priemysel. Filmový priemysel začal využívať počítačovú grafiku na rôzne efekty, ktoré by pomocou normálnych efektov neboli dosiahnuteľné, alebo dostatočne realistické. Neskôr sa začali vytvárať filmy, ktoré sú kompletne vytvorené pomocou počítačovej grafiky. Dnes je už filmová grafika natoľko rozvinutá, že bežný divák má veľmi malú šancu určiť, čo bolo vytvorené počítačovou grafikou a čo nie.

S grafikou úzko súvisia aj počítačové hry. Majú taký vplyv na vývoj grafiky, že dnešný hardware (grafické karty atď.) sa z veľkej časti prispôsobuje vyvíjaným hrám. So zvýšujucou sa kvalitou grafického zobrazovania, sa stáva zážitok z hrania čoraz viac realistickejší a príťažlivejší.

Táto bakalárska práca sa zaoberá práve vývojom hry. Jej cieľom bolo vytvorenie objektov použiteľných v projekte Space game. Tento projekt smeruje spočiatku k veľmi jednoduchému leteckému simulátoru vo vesmíre, ale s ďalším vývojom sa bude komplexita tohoto projektu zvyšovať. Účel vypracovania tejto práce je pomôcť pokročiť vo vývoji projektu Space game. Druhú polovicu projektu tvorí vizualizácia a animácia vytvorených objektov. Táto práca sa zamerala hlavne na rotácie, pretože je tam názorne vidieť, ako funguje hierarchická animácia.

Vznikol na základe požiadavok ukončenia trojročného bakalárskeho štúdijneho programu Informačné technológie v akademickom roku 2004/2005. Tento projek vznikol a je ďalej vyvíjaný na pôde Vysokého učení technického v Brne.

1.2 Prehľad dokumentu

1. **Základné pojmy**

   Táto časť je venovaná vysvetleniu dôležitých pojmov, ktorých nevedomosť by mohla spôsobiť ťažkosti pri čítaní tejto práce.

2. **Modelovací software**

   V tejto kapitole sa zoznámime s použitým softwarom na modelovanie objektov, konkrétne 3D studio Max verzia 7. Je tu vysvetlené, prečo sme zvolili práve tento program z vcelku širokej ponuky iných podobných produktov.

3. **Transformácie**

   Teoretické vysvetlenie pojmu transformácie a rôzne metódy, ktoré sa používaju na transformácie. Sú tu vypísané zároveň niektoré dôležité algoritmy. Podrobnejšie tu bude rozpísaná teória rotačných transformácii.

4. **Tvorba objektov**

   Je tu popísaný postup pri tvorení modelov. Pri tvorbe rôznych modelov boli použité rôzne modelovacie techniky, takže budú všetký podrobnejšie popísané. V tejto kapitole bude tiež zahrnuté mapovanie objektov a tvorba textúr pre dané objekty.

5. **Implementácia algoritmov**

   V tejto časti je popísane, ako som postupoval pri tvorbe implementácii algoritmov, ktoré predstavujú rotácie.

6. **Exportovanie** a **inportovanie**

   Je tu popisané ako boli objekty exportované z modelovacieho softwaru a ako boli importované do použitého zobrazovacieho softwaru. Zároveň su tu popísané problémy, ktoré sa vyskytli pri tejto fáze projektu.

7. **Záver**

   Záverečna kapitola je venovaná zhodnoteniu dosiahnutých výsledkov a mojim prínosom touto prácou. Zhodnotil som tu možnosť ďaľšieho vývoja tohto projektu. Zároveň som tu uviedol nové skúsenosti, ktoré som pri riešení tejto bakalárskej práce nadobudol. Sú tu tiež uvedené súvislosti s inými projektami.

2 Základné pojmy

Základné pojmy sú veľmi dôležité pre jednoduché a jasné porozumenie zvyšku textu. Vysvetlením tychto pojmov tak môžeme zabrániť neskorším nedorozumeniam a chybám v chápani textu a jednotlivých súvislostiach.

2.1 Polygón

Polygón (doslovne „monohouholník“) je uzatvorená plocha poskladaná z konečného počtu za sebou nasledujúcich segmentov čiar. Čiary, ktoré obkresľujú polygón sa nazývajú hrany. Body, v ktorých sa jednotlive hrany stretávajú sa nazývajú vrcholy. Ak je polygón jednoduchý, tak obvykle všetky čiary a vrcholy tvoria hranicu pre plochu polygónu. Niekedy sa za polygón považuje oblasť vo vnútri, teda plocha, ktorú uzatvoria hrany. Alebo sa za polygón považuje zjednotenie oblasti aj hranice. Táto plocha však nemusí byť zákonite úplne rovná. V priestore nastávajú situácie, keď sa plocha musí na nejakej vnutornej priamke zlomiť, aby sa rozprestrela medzi všetky vrcholy.

2.2 Polygónová sieť

Polygónová sieť je súbor vrcholov a polygónov, ktorý definuje tvar objektu v 3D grafike. Sieť obvykle pri exporte obsahuje len trojuholníkove polygóny, ktoré sú ľahké na matematické prepočty, no nie je to podmienka. Sieť može obsahovať prakticky polygón s mnoho stranami, no užívateľ tak stráca kontrolu nad jeho deformáciou a vnútorným lámaním. Najčastejšie používane polygóny pri tvorbe objektu sú štvoruholníky, pretože znázorňujú obrys tak, že je pre užívateľa najľahšie pochopiteľný. Zároveň sa pri zhusťovaní polygonálnej siete nevytvárajú tzv. artefakty (sú to úkazy, ktoré narušujú plynulý prechod medzi polygónmi z globálneho, nie lokálneho, hľadiska)

2.3 High/Low-poly modely

V dnešnej dobe sa grafika delí na dve odvetvia z pohľadu procesu a výsledku tvorby modelov. Rozdiel medzi nimi je napríklad v tom, ako sa tvoria dané modely, aká je výsledná kvalita modelu, aký majú dopad na spotrebu výkonu, ako pôsobia na užívateľa z pohľadu reálnosti atď.

2.3.1 High-poly modely

Prvé odvetvie sa zaoberá tzv. High-poly modelmi (mnoho polygonálne modely). Modely takto vzniknuté maju veľký počet polygónov. Napríklad pri tvorbe postavy sa za high-poly model považuje objekt s minimálnym počtom polygónov okolo 8.000. No v poslednej dobe sa s rozvojom nových

technológii a zrýchľovaním hardwaru tato hranica posúva do 500.000 až 5.000.000 polygónov. Tieto modely sa používaju hlavne vo filmovom priemysle, vedeckom sektore a na tvorbu vysokokvalitných obrazov. V hernom priemysle sa tieto modely používaju na produkciu tzv. normálových máp. Sú to v podstate mapy, ktoré dokážu na rovnej ploche pomocou prepočtu dopadajúceho svetla vtvoriť ilúziu hrbatosti povrchu. Ušetrí sa tak obrovská časť výkonu a strata kvality modelu je pre užívateľa minimálna.

2.3.2 Low-poly modely

Druhé odvetvie sa zaoberá tzv. Low-poly modelmi (málo polygonálne modely). Sú to modely, ktoré sa vyznačujú veľmi nízkym počtom polygónov. Kedysi bol štandard od 300 do 800 polygónov na jednu postavu, no dnes sa to pohybuje okolo 2.000 až 3.000 polygónov. Programy, ktoré sú postavené na najmodernejších enginoch a dokážu tak využiť výkon dnešného hardwaru na 100%, si môžu dovoliť aj modely okolo 5.000 polygónov. Tieto modely sa používaju prevažne v hernom priemysle, pretože tu musí byť dodržaný real-time rendering. To znamená, že scéna by mala byť zobrazovaná minimálne 15 krát za sekundu, ale optimálne by bolo 30 krát a viac.

2.4 Textúra

Je to v podstate akýkoľvek obrázok, kotrý sa aplikuje na povrch 3D objektov. Obvykle sa ale textúry vytváraju špeciálne pre jeden druh modelu. Keďže obrázok nemôže byť v počítači reprezentovaný inak, ako štvorcom, alebo obdĺžnikom, tak sa nevyužitá časť obrázku necháva jednofarebná, aby sa ušetrilo miesto v pamäti. No pri profesionalnej tvorbe textúry sa tento nevyužitý priestor pohybuje pod 10%, pretože pri mapovaní textúr sa dajú koordináty rôzne posúvať, otáčať a meniť ich mierka.

S narastajúcim výkonom grafického hardwaru je možné vytvárať čoraz väčšie a detailnejšie textúry. Zároveň sú tieto grafické karty schopné navrstviť na seba rôzny počet textúr, pričom každá môže mať iný efekt pri spracovaní obrazu. Takéto navrstvenie textúr má obrovský dopad na realistickosť zobrazovanáho povrchu.

2.5 Mapovanie

Mapovanie textúr je metóda, ktorá pridáva realizmus do počítačovej grafiiky. Textúra je pridaná na jednoduchý povrch vygenerovaný v scéne, ako tapeta prilepená na jednoduchú rovnú plochu. To samozrejme redukuje množstvo výpočtov, ktoré sú potrebné pri tvorbe povrchov a textúr v scéne. Napríklad rovnú plochu môžeme namapovať textúrou zeme a tak sa môžeme vyhnúť zdĺhavému modelovaniu kamenia. Pričom výkon bude potrebný len na nanesenie textúry na jeden polygón, narozdiel od tisícok potrebných polygónov potrebných na dosiahnutie podobného efektu realizmu.

3 Modelovací software

3.1 Voľba programu

Pred začiatkom tvorby objektov do scény pre Space Game, vznikla vcelku závažná otázka. Bolo totiž potrebné rozhodnúť, ktorý modelovací software použijem. Na výber som mal niekoľko produktov, ako napríklad: Maya, Xgi, 3D studio Max, Cinema 4D, Rhyno caros.

Maya je program, ktorý sa od svojho počiatku venoval hlavne filmovej tvorbe. Z toho vyplýva veľka prepracovanosť tvorby materiálov a jednoduchá práca s high-poly modelmi.

Xgi sa vyznačuje svojou prepracovanosťou vo všetkých oblastiach grafiky. No v Európe je veľmi nedostupný a je veľmi drahý, preto jeho použiteľnosť s ohľadom na tento projekt nie je prípustná.

3D studio Max ma obrovskú podporu v hernom priemysle a je preto špecializovaný hlavne na prácu s low-poly modelmi. V nových verziách má už taktiež prepracované mapovacie nástroje, čo ušetrí veľa času. Tento program má však určité nedostatky pri tvorbe materiálov.

Cinema 4D ako je to už z názvu vidieť, je zameraná hlavne na filmovú tvorbu. Dajú sa v nej samozrejme modelovať aj low-poly modely, ale ich tvorba by bola obtiažnejšia.

Rhyno caros je založený hlavne na modelovaní pomocou NURBS (sú to krivky, pomocou ktorých sa vytvorí povrch medzi nimi), takže všetky nástroje sú optimalizované hlavne pre tieto krivky. Podpora ľahkého a efektívneho modelovania pomocou polygonálnej siete je tu takmer vylúčena.

3.2 3D studio Max

Po zhodnotení všetkých faktov som zvolil software 3D studio Max, konkrétne verziu 7. Je to program, ktorý je špecializovaný aj na tvorbu low-poly modelov a to je to, čomu som sa venoval v tejto bakalárskej práci. Má veľmi jednoduché uživateľské rozhranie, ktoré je zároveň veľmi ľahko modifikovateľné. Teda prácu mi uľahčil rýchly prístup ku všetkým nástrojom, ktoré som potreboval.

Tento program má veľmi dlhú históriu. Ak vezmem do úvahy prvé 4 verzie programu 3D studio a za nimi nasledujúcich 7 verzií 3D studio Max, tak je toto už 11. verzia. Vývoj tohoto softwaru teda dospel už do fázy, keď má zabezpečenú popularitu a stálych klientov. Využívaju ho aj spoločnosti, ktoré majú na hernom trhu veľký podiel, ako napríklad: Electronic Arts, Ubisoft, Blizzard Entertainment.

Popularita tohoto programu je aj v jeho rozšíriteľnosti o rôzne pluginy (malé programy, ktore pridajú nejaké funkcie, alebo vylepšia súčasné). Jeden takýto voľne dostupný plugin (Greeble) som použil aj na vytvorenie základu pre textúry použité na vesmírnych lodiach. 3d studio max zároveň

podporuje veľké množstvo formátov súborov, s ktorými dokáže pracovať. Ja som v tejto bakalárskej práci využil formáty max, 3ds a wrl

3.2.1 Formát max

Formát max je štandardný formát s ktorým pracuje 3d studio. Tento formát dokáže niesť všetky informácie, ktoré sú potrebné pri práci s modelmi, ich animácii atď. Ukladá v sebe aj úrovňovú štruktúru modifikátorov, čo umožnuje zjednodušený prístup k rôznym fázam a rôznym detailom modelovania.

3.2.2 Formát 3ds

Tento formát je značne zjednodušená forma formátu max. Uchováva v sebe informácie o modeloch. Sú v ňom vypísané všetky vrcholy a ich zoskupenia, kvôli tvorbe povrchu. Dôležitá časť tohoto formátu je, že obsahuje v sebe mapovacie informácie. Teda tento formát som využil pri tvorbe textúr v špeciálnom maľovacom programe Deep Paint 3D.

3.2.3 Formát wrl

Formát wrl je odvodený od VRML (Virtual Reality Modeling Language). Je to štandardný formát súboru na reprezentáciu 3D vektorovej grafiky. Na rozdiel od max a 3ds sa vyznačuje univerzálnosťou a ľahkou prenositeľnosťou, teda dokážu s ním pracovať aj iné modelovacie programy (napr.: Maya, Xgi atd.). Je to textový súbor, kde môžu byť napríklad súbory vrcholov a hrán, ktoré definujú farbu povrchu, mapovacie koordináty pre textúry, lesklosť, priehľadnosť atď. Zároveň sa môžu v tomto súbore vyskytnúť aj URL odkazy na iné VRML na internete. Veľkou výhodou tohoto súboru je, že užívateľ si môže pri exporte do tohoto formátu, zvoliť ako budú reprezentované polygóny. Tak pri výbere štvorcových polygónov nebude problém s parametrickým zvyšovaním hustoty polygonálnej siete (nebudú vznikať artefakty). Do tohoto súboru je zároveň možné uložiť aj osvetlenie a animácie. Toto som ale ja už nepotreboval, lebo osvetlenie a animácia sa dá jednoducho nastaviť pri implementácii do programu.

VRML súbory sa bežne nazývaju svety (angl. worlds) a teda majú príponu .wrl (napríklad base.wrl). Tieto súbory sú v textovej forme a dajú sa veľmi dobre komprimovať a tak posielať cez internet oveľa rýchlejšie.

Prvá verzia VRML bola špecifikovaná v novembri 1994. Táto verzia bola špecifikovaná z a veľmi sa podobá súborovemu formátu vyvynutého firmou SGI (Silicon Graphics) pre Open Inventor. Momentálne je kompletne funkčná verzia z roku 1997 (ISO/IEC 14772-1:1997). Nastávajúca verzia tohoto formátu sa volá X3D.

4 Transformácie

Geometrické transformácie zvyknú byť jedny z najrozšírenejších a najčastejšie používaných operácii v počítačovej grafike. Môžeme ich rozdeliť na lineárne a nelineárne. Medzi lineárne transformácie patria napríklad posunutie, otočenie, zkosenie, zväčšenie, zmenšenie a ich rôzne kombinácie. Nelineárne sú rôzne deformácie telies v priestore, warpovanie textúr na plochu objektu. Špeciálnou transformáciou je projekcia. Je to prevod viacrozmerných objektov na menejrozmerné objekty. Typický príklad je prevod trojrozmerného objektu na dvojrozmerný, aby ho bolo možné zobraziť na konvenčných periférnych zariadeniach (tlačiareň, monitor atď.). Transformáciou objektu rozumieme aplikáciu transformácie na všetky body, z ktorých je poskladaný objekt.

Objekty v 3D (samozrejme aj v 2D) sú reprezentované vrcholmi. Vrcholy sú vlastne súradnice, ktoré sa vzťahujú ku zvolenému súradnicovému systému. Geometické transformácie sa môžu aplikovať na jednotlivé súradnice, ale aj na všetky naraz. Pri nelineárnych transformáciach dokážeme deformovať aj len jeden bod nezávisle na ostatných. Lineárne transformácie sú charakteristické tým, ze ovplyvňujú zoskupenia vrcholov. Pri posunutí, sú to úplne všetky vrcholy a sú transformované s rovnakou váhou. Otočenie ovplyvňuje vrcholy s váhou v pomere k vzdialenosti od stredu otáčania. Pri zošikmení sa váha zvyšuje so zvýšujucou sa vzdialenosťou od osi zošikmenia. Zväčšenie a zmenšenie je na tom tak ako otáčanie. Bežný užívateľ sa s týmito transformáciami stretne napríklad pri animácii nejakého objektu. Modelár sa s transformáciami stretáva pri modelovaní pomocou polygónovej siete takmer pri každom kliknutí myšou.

Transfomovať môžeme aj súradnicový systém. Táto transformácia môže nastať napríklad, keď do scény vložíme nový objekt. Daný objekt má svoj vlastný (lokálny) súradnicový systém. No pre ďalšie použitie môže vzniknúť potreba pretransformovať systém do svetového súradnicového systému. Môže sa tak napríklad zistiť vzájomná poloha jednotlivých objektov v danom svete.

Táto kapitola vychádza z knihy Moderní počítačová grafika [5], kde je tiež možné nájsť viac informácii k danej téme.

4.1 Homogénne súradnice

Na reprezentáciu bodov môžeme použiť homogénne súradnice. Pomocou nich môžeme značne zjednodušiť a urýchliť výpočet transformácií. Homogénne súradnice nám umožňujú vyjadrenie používaných transformácii pomocou jednej matice. Tento spôsob vyjadrenia transformácií je obzvlášť výhodný práve preto, že na ich implementáciu môžeme použiť existujúce knižnice pre prácu s maticami. Skladanie transformácií sa tu realizuje úplne jednoducho pomocou vynásobenia matíc. Inverzná transformácia sa realizuje pomocou inverznej matice. Dnešné grafické karty majú všetky tieto transformačné výpočty plne hardwarovo ošetrené a preto sú veľmi efektívne a rýchlo

realizovateľné. Homogénne súradnice bodu *P* s kartézskymi súradnicami [*X, Y, Z*] predstavuje v troch rozmeroch usporiadaná štvorica čísel [*x, y, z, w*], ak platí:

$$X = \frac{x}{w},\ Y = \frac{y}{w},\ Z = \frac{z}{w},\ w \neq 0$$

Homogénny bod je svojimi súradnicami určený jednoznačne. Súradnicu *w* môžeme nazvať váhou bodu. Homogénne súradnice transformovaného bodu *P'* s kartézskymi súradnicami [*X, Y, Z*] budeme označovať [*x', y', z', w'*]. Obvykle sa volí *w* = 1, môžeme za homogénne súradnice považovať [*X, Y, Z,* 1]. Rozdiel dvoch bodov A = [a_0, a_1, a_2, 1] a B = [b_0, b_1, b_2, 1] určí vektor $\vec{p} = (a_0 - b_0, a_1 - b_1, a_2 - b_2, 0)$. Sčítaním bodu a vektoru dostaneme bod.

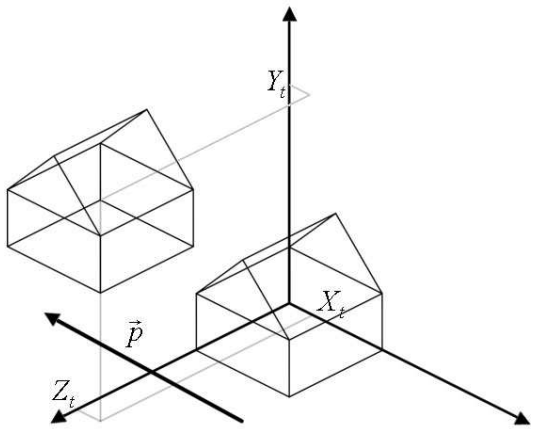
Všeobecná matica 4×4 reprezentujúca lineárnu transformáciu bodu *P* = [*x, y, z, w*] na bod *P'* = [*x', y', z', w'*] budeme označovať *A*. Jej špeciálne prípady potom podľa druhu transformácie, napríklad *T* (translácia), *R* (rotácia). Transformáciu súradnic môžeme zapísať nasledovne:

$$P' = \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = A_{4\times4}.P = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

4.2 Posunutie

Posunutie v 3D je určené vektorom posunutia $\vec{p} = (X_t, Y_t, Z_t)$. Transformačná matica posunutia *T* a inverzná matica T^{-1} sú:

$$T = T(X_t, Y_t, Z_t) = \begin{bmatrix} 1 & 0 & 0 & X_t \\ 0 & 1 & 0 & Y_t \\ 0 & 0 & 1 & Z_t \\ 0 & 0 & 0 & 1 \end{bmatrix},\ T^{-1} = T(-X_t, -Y_t, -Z_t) = \begin{bmatrix} 1 & 0 & 0 & -X_t \\ 0 & 1 & 0 & -Y_t \\ 0 & 0 & 1 & -Z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Obr 4.1 Translácia o daný vektor posunutia

4.3 Otáčanie

4.3.1 Otáčanie okolo súradnicových osí

Otáčanie v troch rozmeroch môže byť jedným z podprípadov otáčania okolo jednotlivých súradnicových osí. Matice R_x reprezentujúca otáčanie okolo osi x o uhol α a zodpovedajúca inverzná matica R_x^{-1} sú:

$$R_x(\alpha)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha & -\sin\alpha & 0\\ 0 & \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 0 & 1\end{bmatrix},\ R_x^{-1}(\alpha)=R_x(-\alpha)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha & \sin\alpha & 0\\ 0 & -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Podobne vyzerajú matice pre otáčanie okolo osi *y* alebo *z*:

$$R_y(\beta)=\begin{bmatrix}\cos\beta & 0 & \sin\beta & 0\\ 0 & 1 & 0 & 0\\ -\sin\beta & 0 & \cos\beta & 0\\ 0 & 0 & 0 & 1\end{bmatrix},\ R_z(\beta)=\begin{bmatrix}\cos\gamma & -\sin\gamma & 0 & 0\\ \sin\gamma & \cos\gamma & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

4.3.2 Otáčanie okolo všeobecnej osi

Otáčanie okolo všeobecnej osi v priestore môžeme realizovať zložením niekoľkých transformácii okolo osi x, y, z, nájdenie príslušných transformácii (uhlov otočenia) nie je jednoduché. Môžeme na to však použiť Rodriguesovú formulu, ktorá prevádza rotačnú úlohu na premietanie a skladanie niekoľkých vektorov.

Predpokladajme, že os otáčania je určená počiatkom súradnicového systému *O* a jednotkovým vektorom $\vec{a}$. Polohový vektor $\vec{x}$ transformovaného bodu *X* je okolo tejto osi otočený o uhol α a výsledný polohový vektor je označený $\vec{x}'$. Za týchto predpokladov môžeme rotáciu popísať vzťahom

$$\vec{x}'=\cos\alpha.\vec{x}+(1-\cos\alpha)(\vec{a}.\vec{x})\vec{a}+\sin\alpha(\vec{a}\times\vec{x})$$

Rotáciu bodu môžeme prepísať do maticového tvaru s využitím definícii vektorového a skalárneho súčinu, matica *I* ja jednotková matica.

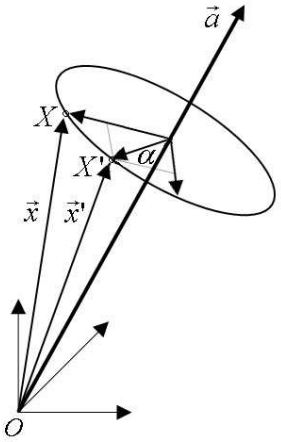
$$X'=R(\vec{a},\alpha).X$$

$$R(\vec{a},\alpha)=\cos\alpha.I+(1-\cos\alpha).\begin{bmatrix}a_x^2 & a_xa_y & a_xa_z & 0\\ a_xa_y & a_y^2 & a_ya_z & 0\\ a_xa_z & a_ya_z & a_z^2 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}+\sin\alpha.\begin{bmatrix}0 & -a_z & a_y & 0\\ a_z & 0 & -a_x & 0\\ -a_y & a_x & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}$$

Ak zvolíme napríklad os rotácie zhodnú s osou x, tj. $\vec{a}=[1,0,0]$, po dosadení do vzťahu uvedeného vyššie, dostanemem maticu zhodnú s $R_x(\alpha)$. Všeobecnú rotáciu riešime tak, že na osi

rotácie zvolíme bod $P=[P_x,P_y,P_z,1]$, vypočítame jednotkový vektor v smere osi $\vec{a}$, zvolený bod posunieme do počiatku súradnicového systému, otočíme transformované objekty o daný uhol a zpätným posunutím vrátime výsledok do východiskovej pozície. Transformačná matica *A* bude zložená z troch základných transformácii.

$$A=T(P_x,P_y,P_z).R(\vec{a},\alpha).T(-P_x,-P_y,-P_z)$$



Obr 4.2 Rotácia okolo všeobecnej osi

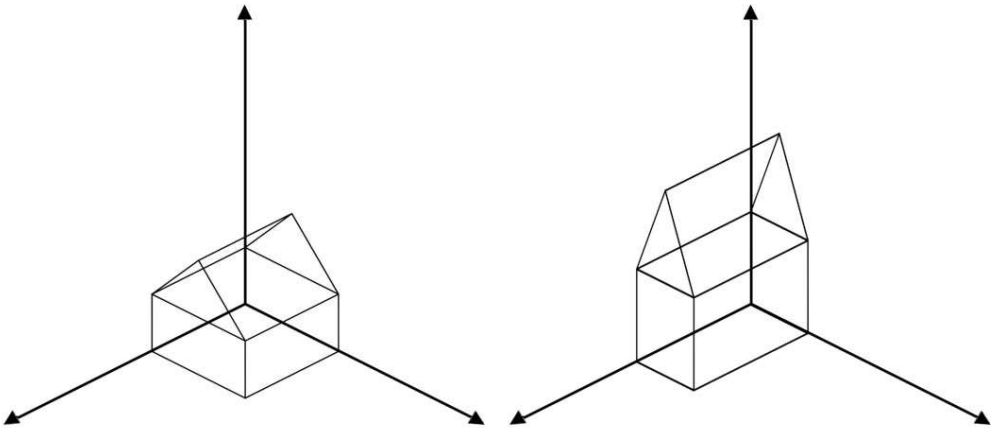
4.4 Zmena mierky

Zmenu mierky (scale) v priestore popisujú transformačné matice:

$$S(s_x,s_y,s_z)=\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ S^{-1}(s_x,s_y,s_z)=S(1/s_x,1/s_y,1/s_z),$$

v ktorých koeficienty $s_x\neq 0, s_y\neq 0, s_z\neq 0$ určujú zmenu v smere príslušnej súradnicovej osi. Pomocou mierkových koeficientov môžeme realizovať niektorú z transformácii súmernosti v priestore. Súmernosť podľa roviny *yz* môžeme realizovať zmenou koeficientu s_x:

$$s_x\neq 0, s_y\neq 0, s_z\neq 0$$

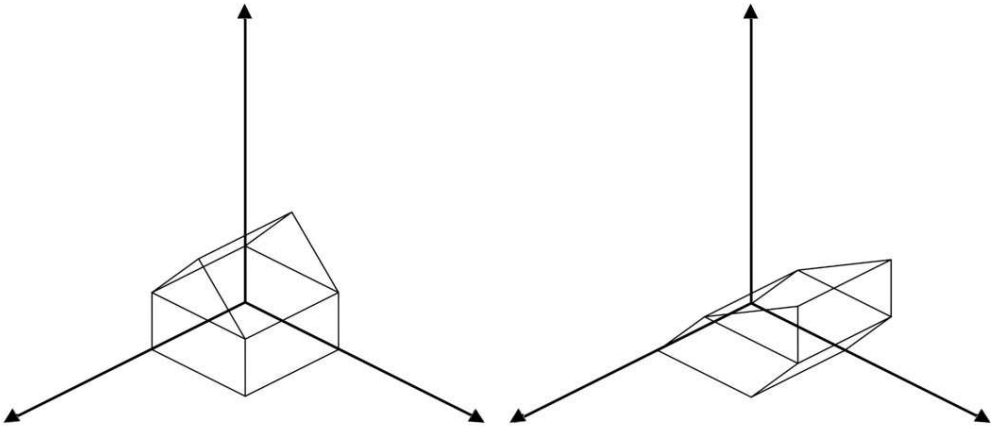


Obr 4.3 Zmena mierky objektu v rôznych osiach

4.5 Zkosenie

Operáciu zkosenia v troch rozmeroch môžeme rozdeliť na tri prípady zkosenia v smere jednotlivých rovín *xy*, *xz* a *yz*. Vo všetkých troch prípadoch určujú koeficienty sh_x, sh_y a sh_z mieru zkosenia v danom smere. Matice transformácie majú tvar:

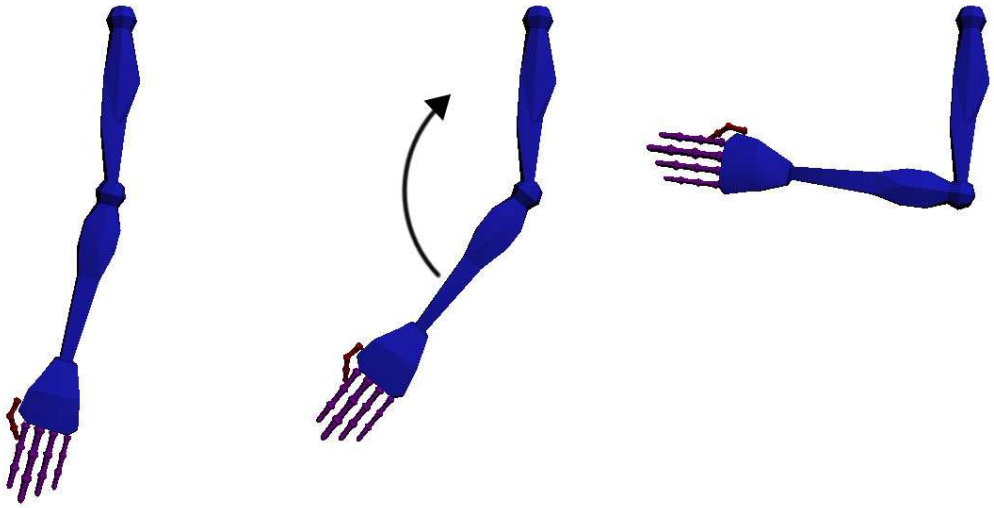
$$Sh_{yz} = \begin{bmatrix} 1 & sh_y & sh_z & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; Sh_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ sh_x & 1 & sh_z & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; Sh_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ sh_x & sh_y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Obr 4.4 Zkosenie objektu v smere jednej osi

4.6 Animácia hierarchických sústav

Hierarchická sústava je súbor objektov na sebe nejako závislých. Viackĺbové rameno je typický príklad takejto sústavy. Každá časť ramena je závislá na predchádzajúcej. Prvá časť takejto sústavy sa nazýva *root* (koreň) a sú na nej závisle všetky ďalšie časti. Posledná časť sústavy sa nazýva *leaf* (list) a je to časť, ktorá už nijako neovplyvňuje ostatné. Teda ak sa pohne root nejakým všeobecným smerom, tak aj všetky časti na ňom závislé sa pohnu rovnakým smerom. Ak by sa pohla iná časť ako root, tak by sa pohli len časti, ktoré sú na tejto závislé. Všetky časti sústavy v hierarchii vyššie, alebo v inej vetve by neboli nijako ovplyvnené.



Obr 4.5 Hierarchická animácia na sebe závislých objektov

5 Tvorba objektov

Projekt Space Game nemal doteraz veľké množstvo objektov, s ktorými by sa dalo dobre pracovať. Ak nejaké modely boli, tak sa využívali iba na testovanie, pretože nedosahovali kvalitu potrebnú na prezentovanie, alebo neboli vhodné na komplexnú implementáciu s výhľadom na budúci vývoj projektu Space Game. Na to aby vesmír vyzeral aspoň z časti realistický bolo potrebné vytvoriť niekoľko rôznych objektov s rôznym pôvodom. Objetky, ktoré som vytvoril s ohľadom na ich potrebu pre túto bakalársku prácu sú:

- Planéta
- Malý asteroid
- Stredne veľký asteroid
- Veľký asteroid
- Transportér
- Spojenecký stíhač
- Nepriateľský stíhač
- Spojenecká základňa

Takmer každý z týchto objektov je svojou štruktúrou úplne odlišný od ostatných. Preto som musel na rôzne objekty použiť rôzne metódy modelovania, mapovania textúr a tvorby textúr.

5.1 Planéta

5.1.1 Modelovanie

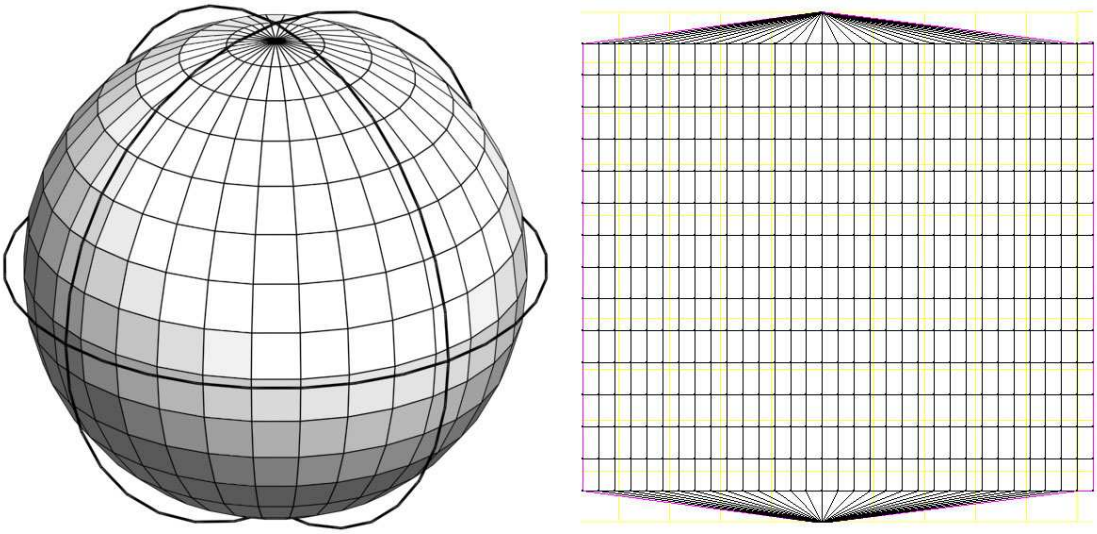
Planétu som vytvoril ako úplne prvý objekt pre vyvíjanú hru. Je to jednoduchá guľa. Vytvoril som ju ako primitívny objekt v 3D studiu Max. Jej základný rys je, že sa parametricky skladá z 32 segmentov, čo má za následok, že táto guľa je pre ľudské oko takmer úplne oblá, ale zároveň nie je zložená z veľkého počtu polygónov. Polygónová sieť je zložená presne z 960 prevažne štvoruholníkových polygónov. Trojuholníkove polygóny sú len na póloch planéty, pretože keby sa nespájali všetky zvislé hrany v póloch, tak by tam vznikli diery, alebo artefakty.

5.1.2 Mapovanie

Najprv som si musel planétu namapovať, aby som videl, kde na textúre sa budu nachádzat jednotlivé sektory planéty. Do úvahy prichádzali tri druhy mapovania: guľovité (sphere), plošné (flatten) a valcovité (cilindrical). Každé mapovanie má svoje výhody, ale aj nevýhody.

5.1.2.1 Guľovité mapovanie

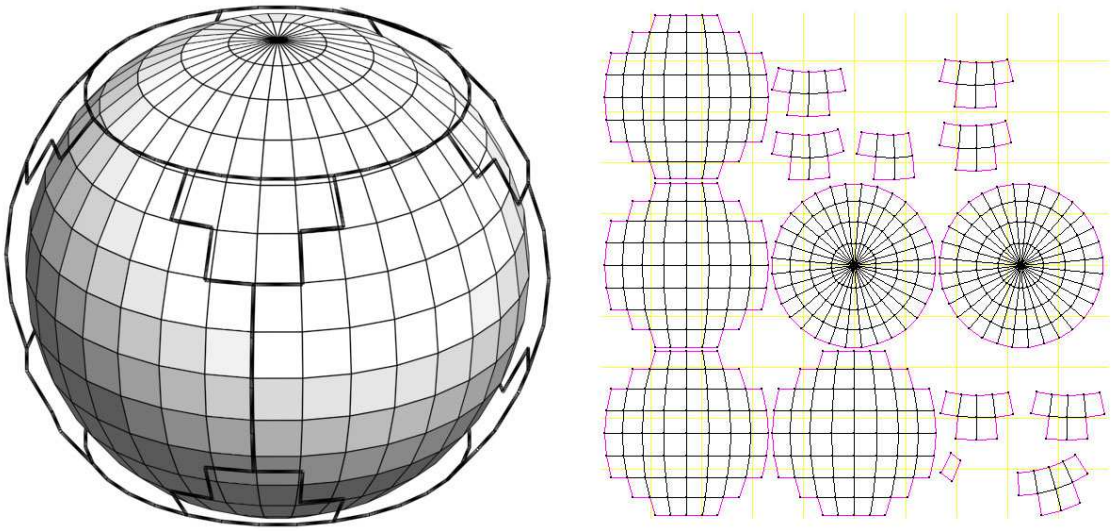
Guľovité mapovanie by sa mohlo zdať ako tá správna voľba, pretože už pri namapovaní vidí užívateľ, že mapovacie koordináty obliehajú guľu úplne presne. K tomu sa pridáva ešte aj fakt, že týmto mapovaním sa dosiahne viac ako 95% využitie obrázku použitého ako textúra. Problém však nastáva pri zobrazovaní takto namapovanej planéty. Okolo jej rovníka sa totiž nachádza najmenej pixelov, teda musia byť roztiahnuté na väčšiu plochu a stráca sa tak detail. Pri póloch sú pixely také malé a nahustené, že dosiahnutý detail už užívateľ často nemusí ani zpozorovať. Ďaľšia strata detailu by bola pri filtrácii pólu grafickou kartou (grafické karty uberajú z detailov textúr so zvýšujúcou sa vzdialenosťou).



Obr: 5.1 Guľové mapovanie

5.1.2.2 Plošné mapovanie

Plošné mapovanie ma oproti ostaným mapovaniam výhodu v tom, že takmer v každom bode celej planéty je rozloženie pixelov z textúry rovnomerné. Odchýlky v hustote si bežný užívateľ nevšimne. Rozloženie mapovania v obrázku texúry má výrazne menšie pokrytie, pretože medzery medzi jednotlivými sektormi sú veľké. Tieto medzery sa zväčšujú podľa členitosti segmentov. Problém však nastáva, keď je potrebné danú textúru nejako upraviť. Obrázok je totiž rozdelený na rône segmenty, ktoré sú rôzne pootáčané. Pri manuálnej úprave takejto textúry by bolo pre mňa takmer nemožné dosiahnúť kvalitné prechody medzi jednotlivými segmentami. Jednotlivé segmenty sa totiž poskládajú na planétu a zaplnia tak celý jej povrch. V prípade, že by som modifikoval jednu hranicu nejakého segmentu, tak by som musel nájsť a modifikovať aj hranicu druhého segmetnu, ktorý susedí s prvým segmentom presne na danej hrane. Túto hranu by som musel na dvoch miestach obrázku dolaďovať a to by mi zabralo veľké množstvo času. Textúra by bola rovnomerne rozložená, ale jej kvalita by sa výrazne znížila, čo by mohlo byť užívateľom vnímané ako chyba.

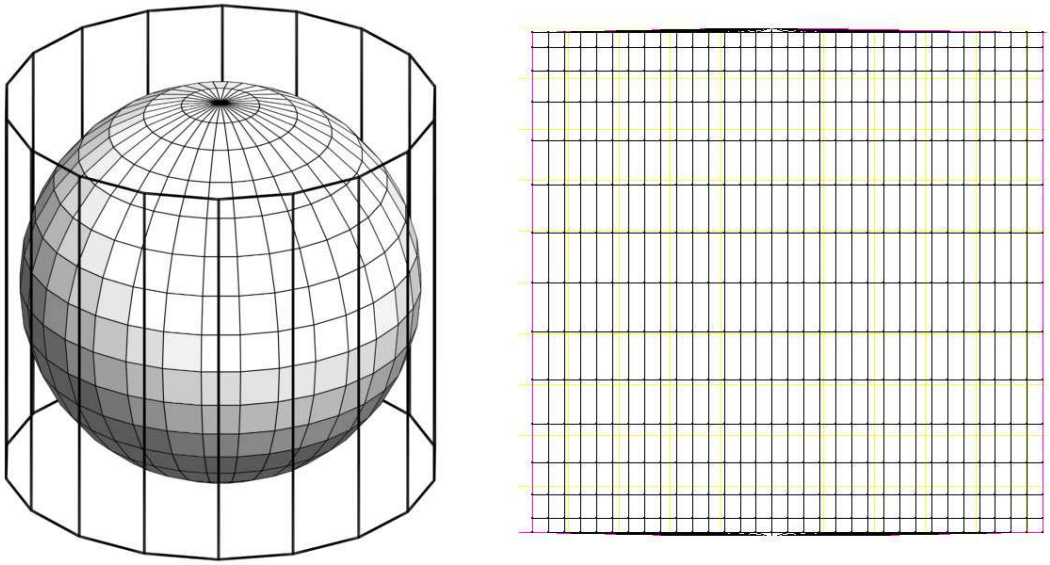


Obr: 5.2 Shrink wrap mapovanie

5.1.2.3 Valcové rozloženie

Valcové rozloženie má podobné vlastnosti ako guľové. Narozdiel od neho, však mapovacie koordináty neobliehajú dokonale tvar planéty. Planéta sa tak musí projekčne zobraziť na plochu, kde sú mapovacie koordináty. Takto sa pixely na planéte zobrazia takmer rovnomerne. Na rovníku sa nachádza viac pixelov, ako pri póle. Je to spôsobené práve projekciou guľatej planéty na valec. Pokrytie textúry mapovacími koordinátmi je tu ešte väčšie ako v guľovom rozložení.

Zvolil som teda válcové mapovanie, pretože ho považujem za najlepšiu voľbu vzhľadom k požiadavkám kladeným na kvalitu planéty.



Obr: 5.3 Válcové mapovanie

5.1.3 Tvorba textúry

Pri tvorbe textúry planéty som bol inšpirovaný našou susednou planétou Mars. Aj napriek tomu, že som mal k dispozícii voľne dostupné textúry Marsu, som sa rozhodol vytvoriť vlastnú textúru. Hra tak bude pôsobiť omnoho originálnejšie a užívateľ sa tak dokáže vžiť aj do iného miesta ako je naša slnečná sústava. Kedže na tvorbu textúry som nepoužil žiaden dostupný obrázok, tak som musel celý povrch planéty vytvoriť sám.

Pri tvorbe tejto textúry som strávil najviac času zo všetkých potrebných textúr. Na výber som mal niekoľko spôsobov, ako vytvoriť túto textúru.Rozhodoval som sa medzi vygenerovaním povrchu pomocou nejakého voľne dostupného pluginu, kompletným namaľovaním vo Photoshope, alebo parametrickým vytvorením v 3D studio Max. Po zvážení všetkých kladov a záporov som zvolil kombináciu parametrického vytvorenia a dokončenia detailov vo Photoshope.

5.1.3.1 Parametrická tvorba textúry

Pri tejto tvorbe som postupoval takzvane z dola nahor. Najprv som si vytvoril najjemnejšie materiály a potom som ich začal spájať dokopy. Celkový výsledok danej práce sa skladá z 12 materiálov a vrstiev.

Prvý materiál, ktorý som vytvoril, je typu *cellular* (bunky). Týmto materiálom som dosiahol to, že povrch planéty vyzerá, akoby mal v sebe veľké kaňony a trhliny. Takto však tento material nevyzerá veľmi realistický, pretože jeho výsledkom je len vyobrazenie trhlín v planéte. Zvyšná, omnoho väčšia časť je jednofarebná a planéta tak vyzera umelo. Preto som na tento zvyšný povrch aplikoval parametrickú mapu *noise* (šum). Takto som dosiahol určitého znerovnomernenia neporušeného povrchu planéty.

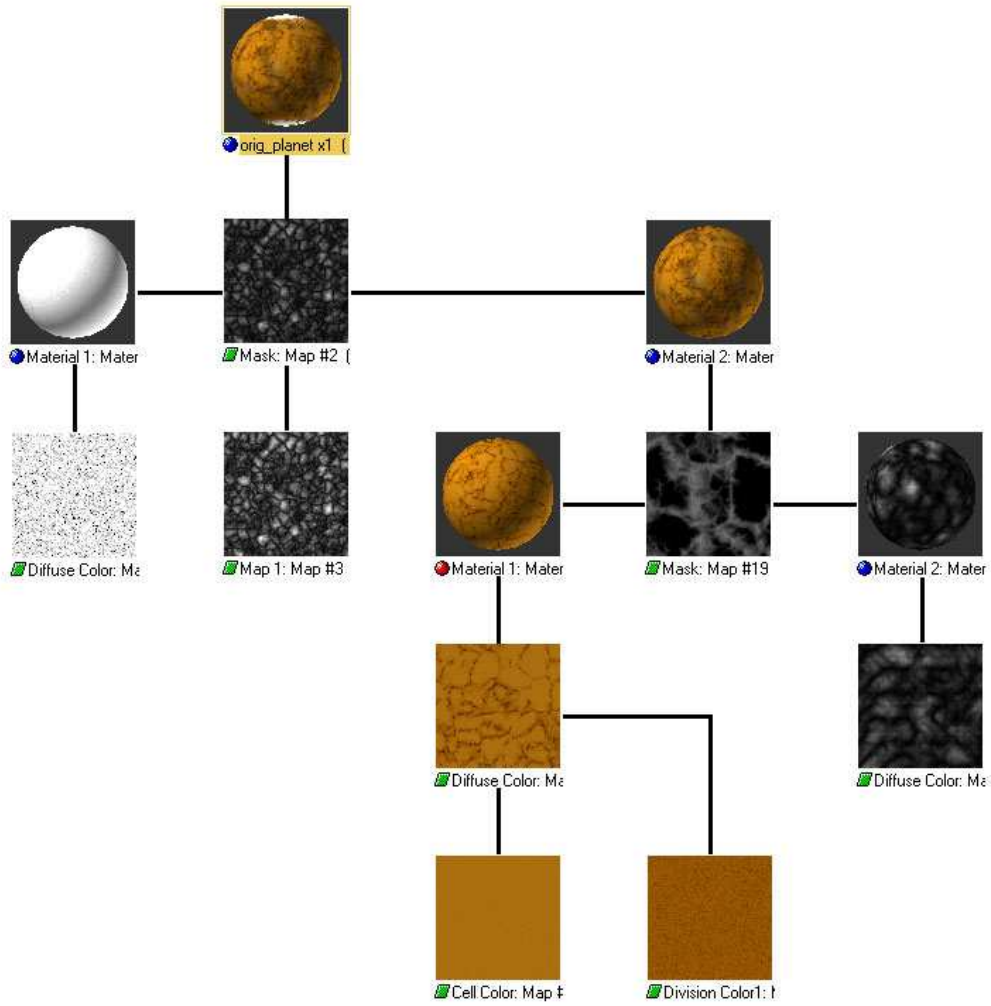
Ďalší materiál, obsahuje difúznu mapu (základná textúra, ktorá netvorí žiaden efekt), ktorá je zložená z parametrickej mapy typu *smoke* (dym). Táto mapa je použitá na vytvorenie tmavých a svetlých škvŕn na planéte.

Tieto dva materiály som spojil pomocou materiálu *blend* (zmiešanie). Tento materiál zmiešava dva materiály podľa presne určených podmienok. V tomto prípade som za rozlišovací parameter použil *cellular* mapu. Tu som však už ale nastavil na najvyšší detail a zjemnil som ju tak, aby sa tmavé a svetlé škvrny z druhého materiálu vzniknuté na prvom materiále, zmiešali jemne s povrchom planéty. Takto som dostal nezamrznutú žasť planéty.

Zamrznutú časť tvorí len jeden materiál, ktorý ma v difúznej zložke *cellular* mapu. Táto mapa už nemá žiadne submapy, ale rozlišuje sa len pomocou farieb.

Tieto dva materiály som zlúčil pomocou materiálu blend. Rozlišovací parameter je tu mapa typu *falloff* (odklon). Táto mapa je nastavená na rolišovací zlom pozdĺž osi *z*. Tu som však ešte

aplikoval do tmavej zložky (výsledok je zobrazenie ľadu) vložil *cellular* mapu, ktorá vytvorila efekt zamrznutých riečných korýt v okolí pólov.



Obr. 5.4 Kompozícia máp a materiálov.

5.1.3.2 Export textúry

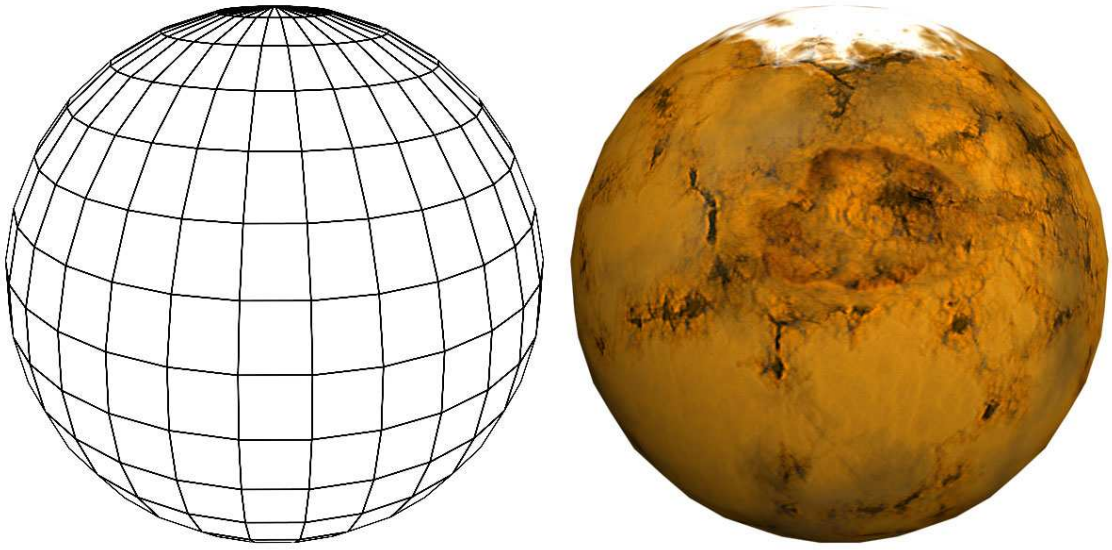
Po vytvorení textúry pomocou parametrických materiálov a máp, som musel tento výsledný materiál vyexportovať do texúry, aby som ho mohol ďaľej spracovať a pridať detaily.

Toto som dosiahol pomocou štandardne dodávaného pluginu Render to texture. Tento plugin pomocou informácii z mapovacích koordinátov nastaví renderovaciu kameru na každý pixel tak, aby bola v každom bode rovnobežne s normálou na povrch objektu v tom mieste, kde sa momentálne renderuje. Takto sa dajú vytvoriť jednoducho textúry, ktoré by v sebe uchovávali aj tiene a podobné efekty. Na tejto textúre by však takéto efekty prekážali a tak som nepridal žiadne lokálne svetlo, ale

globálne osvetlenie som nastavil na maximálnu hodnotu. Tak bola planéta osvetlená z každej strany rovnako a export dosiahol požadovanú kvalitu.

5.1.3.3 Dokončenie textúry

Na dokončenie textúry som použil Photoshop. Najprv som textúru zaostril pomocou filtru *unsharp mask*. Tento filter mi umožnil zaostriť textúru presne podľa mojich potrieb. Jemne som pridal kontrast a tak som zvýraznil trhliny a zlomy. V záverečnej fáze tvorby textúry som namaľoval na povrch veľký kráter, ktorý sa nedal pomocou parametrov vytvoriť. Použil som na to rôzne pomôcky na maľovanie a prekladanie obrazu. Potom som na to aplikoval pomôcku *healing brush*, ktorá farebne doladila textúru a uložil som to do súboru typu jpg, pretože dovtedy som pracoval s nekomprimovanými dátami.



Obr: 5.5 Aplikovanie texúry na objekt planéty

5.2 Veľký a stredne veľký asteroid

5.2.1 Modelovanie

Veľký a stredný asteroid som vytvoril identickým postupom. Rozdiel bol iba v nastavení niekoľkých parametrov. Model vychádza z primitívu Geosphere. Táto guľa je špecifická tým, že sa skladá len z trojuholníkov. Pridal som na detailnosti a vytvoril tak asi 100.000 polygónový objekt. Potom som na neho aplikoval niekoľko *noise* (zrnitosť) modifikátorov, aby som z takmer dokonalej guľe dostal nepravidelný tvar. Tak som dostal základ pre asteroid.

Ďalej som potreboval vymodelovať krátery, ktoré ostali po zrážkach s menšími asteroidmi. Na toto modelovanie som použil špeciálnu techniku modelovania pomocou *displacement* (vytláčanie).

Princíp tejto techniky spočíva v tom, že povrch sa deformuje v závislosti na intezite obrazu, ktorý je nad ním. Na túto techniku som však nepotreboval mať namapovaný objekt, pretože každý takýto modifikátor nesie so sebou vlastné koordináty. Ak by som mal objekt namapovaný, tak sa nič nedeje, pretože tieto koordináty sú iba lokálne a nededia sa do ďalších vrstiev. Vytvoril som si teda niekoľko obrázkov, ktoré mali v strede kráteru čiernu farbu, teda najnižšiu intenzitu a tá stúpala až do bielej. Z bielej sa potom jemne s narastajúcim polomerom prechádza do stredne sivej. Stredne sivá farba má nulový účinok na eleváciu voči pôvodnemu povrchu. Takto som deformoval povrch asteroidu až do prijateľného vzhľadu.

Po vytvorení dostatočne kvalitného asteroidu som tento objekt optimalizoval znížením počtu polygónov. 100.000 polygónov je totiž natoľko hustá polygonálna sieť, že by s takýmito asteroidmi mali problémy aj tie najvyspelejšie grafické karty. Preto som počet zredukoval pod 6.000 polygónov. Túto prijateľnú hodnotu som dosiahol použitím modifikátoru *multires*, ktorý v sebe uchováva celú polygonálnu sieť a dokáže tak redukovať množstvo polygónov pri zachovaní najdôležitejších tvarov.

5.2.2 Mapovanie

Na mapovanie som použil mapovaciu techniku *Shrink wrap*. Toto mapovanie si môžeme ľahko predstaviť na nejakej šatôčke a guli. Ak pustíme na guľu šatôčku, tak sa horná časť gule obalí šatôčkou takmer rovnomerne. Dolná časť nás nezaujíma, ale pre úplnosť to uvediem. V dolnej polovici gule sa všetky hrany šatky stretnú v jednom bode, teda na spodnej časti gule je deformácia mapovania veľmi veľká.

Aby som mohol dobre využiť túto techniku, tak som najprv urobil z asteroidu guľu. Urobil som to tak, že som si najprv urobil kópiu ateroidu, aby som mal jeho tvar niekde uložený. Potom som na jeden z týchto asteroidov použil modifikátor *spherify* (zaguľatenie). Tak som dostal požadovanú guľu. Hornej polovicu gule som zmenil *material ID* (identifikátor materiálu), aby som mohol aplikovať mapovanie na dve polovice zvlašť. Týmto som docielil takmer rovnomerné namapovanie na guľu. Ostala mi tam len jedna chyba a to že v strede je tento asteroid rozdelený a sú tam mierne šikmo namapované pixely. To je už ale nedostatok, ktorý som odstránil neskôr pri textúrovaní.

Takto namapovanú guľu som potom musel premeniť naspäť na asteroid. Použil som na to modifikátor *morph* (formovanie). Ako referncia mi poslúžil skôr okopírovaný asteroid.

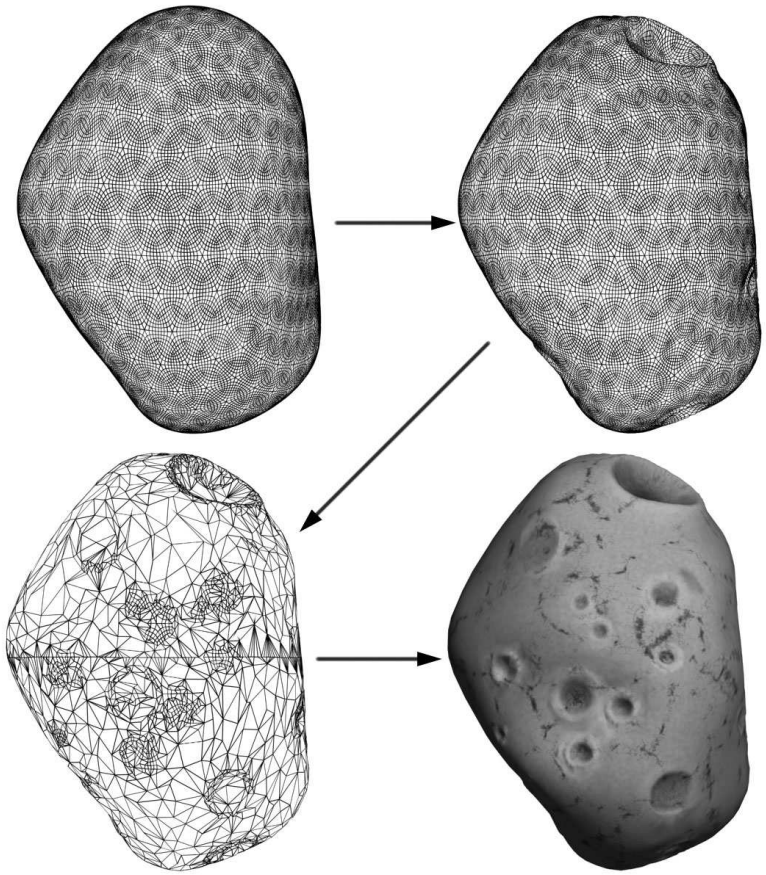
5.2.3 Textúrovanie

Na textúrovanie som si vytvoril parametrický materiál. Obsahoval podobné vrstvy ako materiál pri tvorbe planéty, ale nemal až toľko vrstiev.

Aby som odstránil už uvedený spoj v strede asteroidu, potreboval som domaľovať textúru tak, aby som videl spoje vedľa seba. Na to som použil program *Deep Paint 3D*, ktorý dokáže otvoriť 3D objekt, namapovať na neho textúru a užívateľ potom môže na tento do tejto textôry maľovať. Na

import do tohoto programu som použil formát .3ds, pretože dokáže uchovávať v sebe aj mapovacie koordináty a program ho dokáže otvoriť.

Po premaľovaní potrebných častí okolo spojov som v tomto programe zároveň namaľoval na miesta, kde má asteroid krátery, tmavšie a popraskané miesta, aby sa tak zýšila realistickosť asteroidov. Potom som už len uložil textúru do formátu jpg, ktorý je dobrý kôli svojej veľkosti a rýchlemu spracovaniu v grafických kartách.



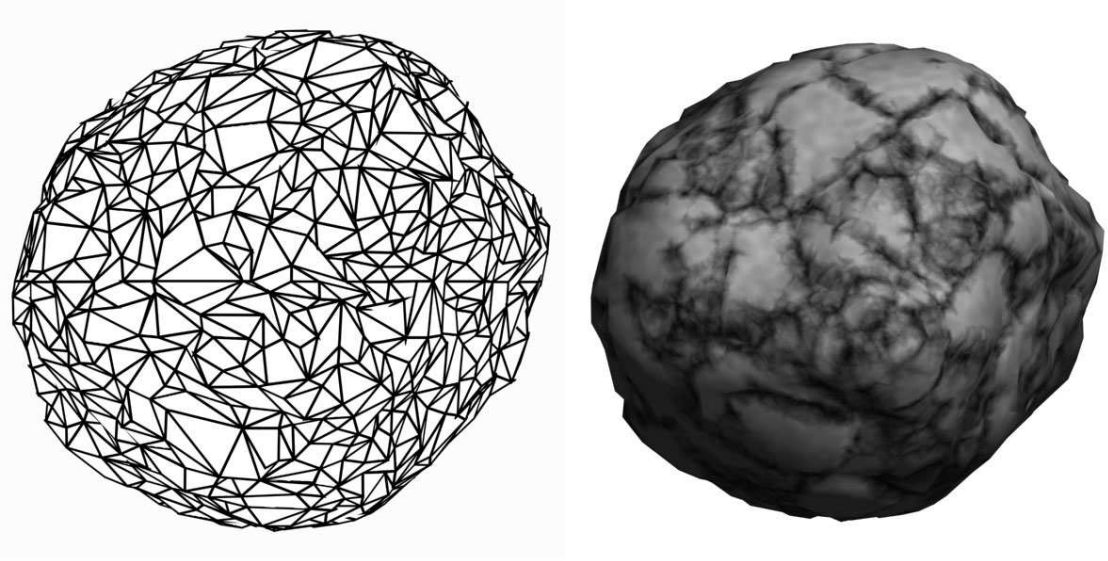
Obr: 5.6 Tvorba asteroidu

5.3 Malý asteroid

Malý asteroid bol z hľadiska doby tvorby najjednoduchší model zo všetkých. Modeloval som ho tak ako veľké asteroidy, teda veľka hustota polygonálnej siete, *noise* a *displacement*. Rozdiel bol ale v tom, že som nepoužil pripravené obrázky s krátermi, ale vytvoril som si najprv materiál. Tento materiál som vytvoril podobne ako pri tvorbe planéty. Ešte pred vytvorením materiálu som si tento asteroid namapoval. Na *displacement* som aplikoval tento materiál, a keďže som ho nastavil tak, aby prevzal koordináty z mapovania, tak sa v asteroide vytvorili praskliny presne na tmavých miestach.

Potom som takto vymodelovaný asteroid optimalizoval pomocou *multires*. Veľka výhoda je, že tento modifikátor zachováva aj mapovacie koordináty. Tak som ušetril veľa času.

Textúru som exportoval pomcou *Render to texture.*



Obr. 5.7 Malý asteroid

5.4 Ďalšie objekty

5.4.1 Modelovanie

U ďalších modelov som postupoval prevažne štandardným spôsobom. Primitívny kváder som premenil na *editable poly*. Je to modifikovateľná polygonálna sieť, ktorá obsahuje mnoho priamych modifikátorov a poskytuje prostriedky pre prácu s jednotlivými vrcholmi, hranami, polygónmi, dierami a celými zoskupeniami polygónov.

Z kvádra som postupným vyťahovaním, rezaním a inými rôznymi modifikáciami postupne modeloval low-poly modely. U týchto objektov som musel šetriť každým polygónom, pretože som predpokladal, že sa týchto objektov bude v scéne nachádzať veľký počet.

Rozdiely pri modelovaní boli len minimálne. Napríklad pri modelovaní stíhačov som od počiatku mal polovicu kvádra odstránenú a miesto nej som mal o vrstvu vyššie symetrický modifikátor, ktorý mi chábajúcu polovicu nahradil už modifikovanou. Pri tvorbe základne som vymodeloval jeden segment a ten som potom pomocou poľa nakopíroval 5-krát a dosiahol som tak spojitý kruh.

5.4.2 Mapovanie

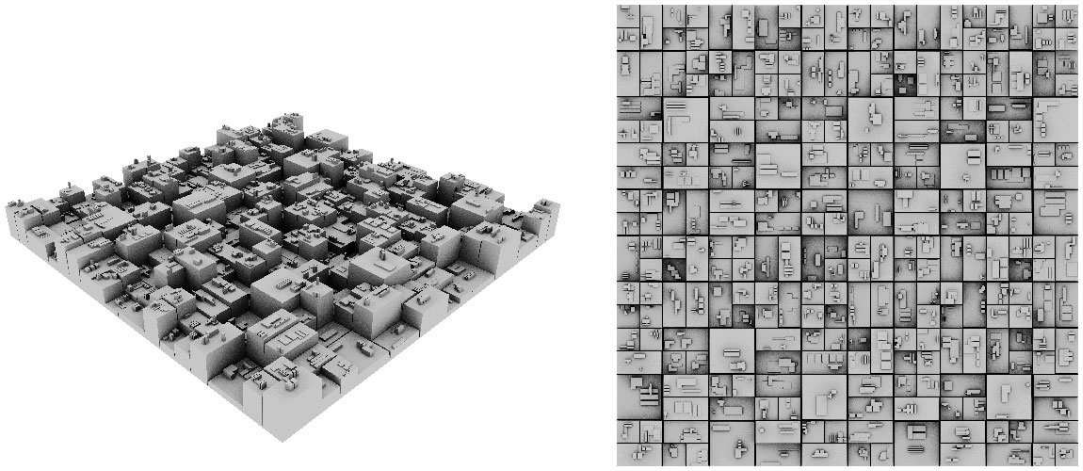
Na mapovanie som použil u všetkých modelov najpoužívanejší mapovací modifikátor v dnešnej grafike. *UVW Unwrap* je nástroj, ktorý umožnuje modifikovať mapovacie koordináty priamo. Je tu možné nastaviť mapovacie koordináty pre skupinu, ale aj pre jednotlivé vrcholy. Vo verzii 7 pribudla

tomuto modifikátoru možnosť vidieť hranice mapovania priamo na modele, čo veľmi urýchli a sprehľadní prácu.

5.4.3 Textúrovanie

Textúry som vytvoril tak, že som urobil screen-shot s mapovacími koordinátmi. Na tieto koordináty som vo Photoshope postupe kreslil detaily do rôznych vrstiev.

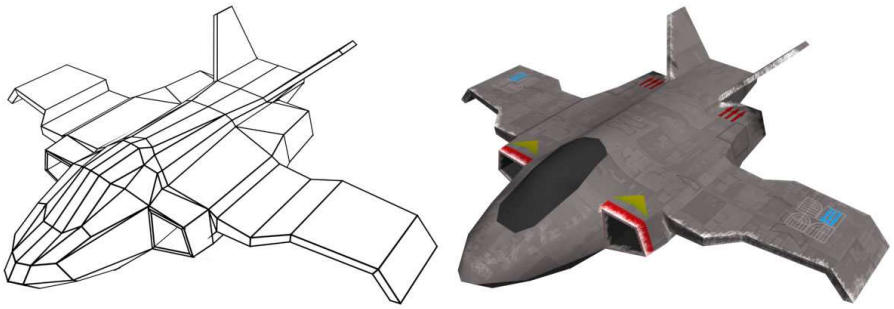
Ako podklad pod detaily nemohla ostať mriežka s koordinátami, tak som vytvoril v 3D studio Max pomocou pluginu *Greeble* bitmapu, ktorá znázorňuje plechový trup lode. Vytvorí rôzne vysoké kvádre na ploche a dokáž ich rôzne náhodne deliť na menšie. Pri priamom pohľade zhora a použití globálneho osvetlenia s jemnými tieňmi vznikne požadovaný obraz. Takto skombinované textúry som uložil do formátu jpg.



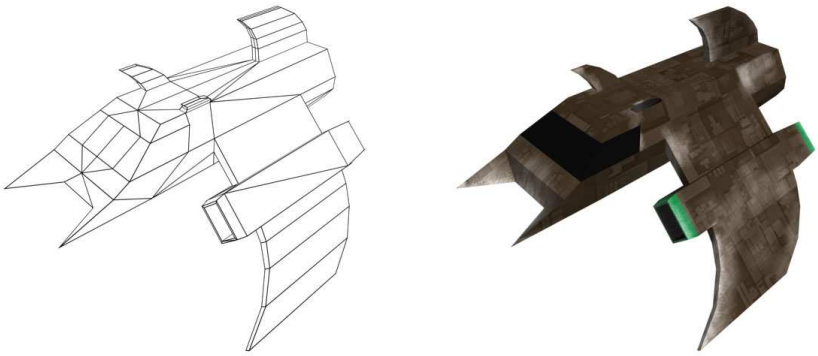
Obr: 5.8 Tvorba plechových plátov pomocou Greeble pluginu

5.5 Finalizácia tvorby objektov

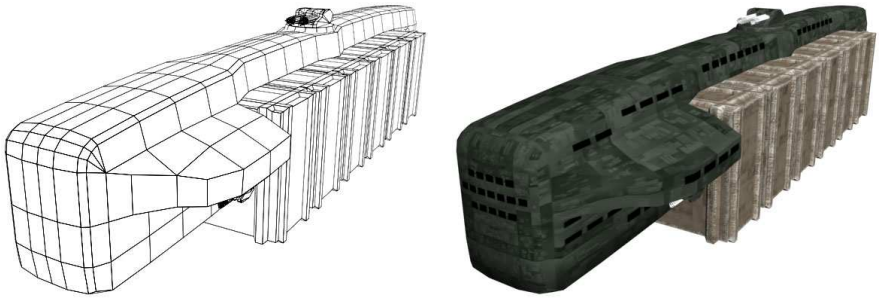
Po namodelovaní, namapovaní a vytvorení textúr som musel tieto tri fázy tvorby modelu spojiť do jedného celku. Preto som exportoval model s mapovacími koordinátmi do formátu wrl. Tento formát nezahrňuje textúry. Má v sebe ale odkazy na všetky potrebné texúry.



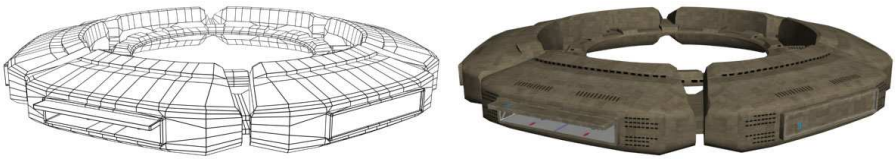
Obr: 5.9 Spojenecký stíhač



Obr: 5.10 Nepriateľský stíhač



Obr. 5.11 Spojenecký transpotér



Obr. 5.12 Spojenecká základňa

6 Implementácia algoritmov

Na implementáciu všetkých algoritmov som použil Open Inventor. Je to objektovo orientovaný 3D nástroj postavený na knižnici OpenGL. Bol vyvinutý firmou SGI, ktorá je na poli grafiky jednou z najpokrokovejších. Je optimalizovaný tak, že je pre užívateľa veľmi praktický a efektívny.

Na implementáciu animácie hierarchických sústav som využil rotácie a translácie, pretože je na nich jasne vidieť, ako ovplyvňujú otcovia svoje deti a tie svoje deti. Implementoval som tieto algoritmy na model transportéru, pretože má najviac na sebe závislých objektov. Trup lode je otec, podstavce palebných veží sú jeho deti a hlavne veže sú deti podstavcov. Podobne som implementoval algoritmy na základňu. Má totiž dvere, ktoré sa môžu otvárať a zatvárať.

Keďže som nadviazal na už existujúci projekt, tak som použil niektoré zdrojové kódy, ktoré by bolo zbytočne vytvárať znova. Použil som moduly SgObject, SgManager a časť kódu je aj v main.cpp. Na tieto zdrojové kódy som doplnil moduly SgBase a SgTransport a dokončil main.cpp.

6.1 Príprava scény a užívateľského rozhrania

6.1.1 Uzly grafu scény

Celá scéna sa načítava do grafu scény. Graf scény obsahuje uzly, ktoré môžu obsahovať ďalšie uzly, alebo listy. Listy grafu obsahujú napríklad reprezentáciu modelu, translácie a iné.

Na začiatku sa vytvorí koreň (root). Je to otec všetkých ostatných potomkov. Prvý potomok, ktorého pridáme hneď po štarte aplikácie je objekt typu *SoEventCallback*. Tento stále visí a odchytáva klávesy, ktoré previedli nejakú zmenu svojho stavu (stlačenie, alebo uvoľnenie). Ak nastane nejaká zmena, tak sa zavolá funkcia *keyboard_cb,* ktorá zabezpečuje nastavovanie premenných podľa toho, ktorá klávesa zmenila svoj stav. Ďalší dôležitý objekt je typu *SoOneShotSensor*, ktorý riadi ukladanie udalostí do kalendára.

Vytvoríme si kameru, ktorú nastavíme do pozície, z ktorej bude dobre vidieť potrebné objekty v scéne. Najlepší spôsob ako to dosiahnuť, je umiestniť ju do bodu, kde ani jedna z osí súradnicového systému nie je rovnobežná s osou pohľadu kamery. Kameru potom pridáme do grafu scény.

```
SoPerspectiveCamera *camera = new SoPerspectiveCamera;
camera->position.setValue(10.f, 5.f, 15.f);
camera->orientation.setValue(SbRotation(SbVec3f(0.f,1.f,0.f),
     float(M_PI_4)) * SbRotation(SbVec3f(0.f,0.f,1.f), float(M_PI_4/3)) *
     SbRotation(SbVec3f(1.f,0.f,0.f), -float(M_PI_4/8)));
root->addChild(camera);
```

Pridáme si tzv. *skybox*. Je to kocka, na ktorú sú z vnútorných hrán nanesené textúry, ktoré predstavujú prostredie. V našom prípade to je vesmír. Pridáme ho do grafu scény.

Ďalší potomok koreňa, tzv. *sgoManager*, bude združovať vytvorené objekty v scéne. Teda všetky modely sa pripoja na tento uzol v grafe scény. Potom začneme vytvárať jednotlivé objekty, ktoré budú v scéne zobrazené. Muíme načítať model, nastaviť jeho pozíciu, aby sa všetky neobjavili v počiatku súradnicového systému. Jeho rýchlosť nastavíme na nulu v každom smere.

```
SgTransport *tr = new SgTransport;
  transporter = tr;
  tr->loadModel();
  tr->setModel("Transport-C.wrl", SgObject::COLLISION);
  tr->setPos(0.f, 0.f, 0.f);
  tr->setSpeed(0.f, 0.f, 0.f);
  sgoManager->appendSgo(tr);
```

Posledné čo vytvoríme, je renderovacie okno. V ňom sa bude všetko zobrazovať. Toto okno má už predvolené klávesové skratky, preto ich vypneme, aby sme mohli ovládať modely v scéne. Vypneme ich metódou *setViewing* s parametrom FALSE. Okno tak presne odchytávať události z periférii a teda objekty v programe tak budú môcť reagovať na zmenu stavu periférii.

```
  viewer->setSceneGraph(root);
  viewer->setViewing(FALSE);
```

6.1.2 Klávesové skratky

Aby užívateľ mohol ovládať modely v scéne, je potrebné vytvoriť funkciu, ktorá nastaví rôzne premenné a atribúty. Tieto premenné sa nastavia podľa toho, ktorá klávesa bola stlačená. Niektoré reagujú iba na stalčenie a niektoré menia svoj stav pri stlačení a nezmenia ho pokiaľ je daná klávesa stlačená.

Na rozlíšenie kláves si vytvoríme štruktúru, ktorá obsahuje zoznam stlačiteľných kláves, stav stlačenia a inštanciu triedy *key*. V zozname kláves sa budú nachádzať klávesy pre otéčanie transportéru v jednej jeho osi, otáčanie podstavca palebnej veže, otáčanie hlavne palebnej veže, prepínače medzi jednotlivými vežami. Ďalšie klávesy budú ovládať prepínanie medzi jednotlivými bránami základne a klávesy na otáčanie bránou.

6.2 Štruktúra vrml formátu a načítanie modelu

```
Separator {
          Transform {
            translation -3482.093 357.20001 2009.8311
            rotation 0 1 0  2.0945928
          }
          RotationXYZ { axis X angle 0.5 }
          DEF BaseDoor File { name "Base-Door.iv" }
      }
```

Vrml má jednoduchú štruktúru založenú na vnorovaní sa od rodičov k potomkov. Prvá položka udáva verziu vrml formátu. My pracujeme s verziou 2.1. Položka *File*{ name "Base-Body.iv" } vytvorí objekt typu *SoFile* a nastaví jeho dátovú položku *name* na Base-Body.iv. Hneď potom sa tento model načíta z disku a bude zobrazený na aktuálnych súradniciach. Potom nasleduje položka *Separator*. Je to uzol, ktorý zoskupuje jednotlivých potomkov. Prvá položka je *Transformation.* Je zložená z niekoľkých dátových položiek, ktoré umožňujú transláciu, rotáciu, zmenu mierky atď. Tieto dáta ovplyvňujú všetky položky, ktoré sú ďalej v danom uzle uvedené. *RotationXYZ* zrotuje priestor okolo osi *x*, ale zároveň musíme počítať s tým, že sa tu už skladajú transformácie s predchádzajúcimi. *File* načíta a zobrazí na už transformovaných súradniciach. Špeciálna položka je *DEF meno,* ktorá definuje „meno“ nasledujúcemu *File*. Ak sa niekde v súbore bude nachádzať rovnaký model, tak sa to jednoducho načíta pomocou *USE meno*. Nevytvára sa tak nový *File*, ale vytvorí sa len ukazateľ na pôvodný *File*. Zároveň sa tým odstráni aj potreba opätovného načítania z disku.

Modely načítavame pomocou metódy *loadModel*(). Model sa pridá do uzlu grafu scény

```
void SgTransport::loadModel(){
  setModel("Transport.iv", SgObject::RENDER);
  SoSeparator *model = (SoSeparator*)this->model;
```

Model sa načíta celý, ale stále môžeme pracovať s jeho hierarchickým usporiadaním. Do premenných si môžeme načítavať rôzne dáta postupným zanorovaním sa k nim.

```
SoSeparator *child1 = (SoSeparator*)model->getChild(6);
  turretYaw[1] = (SoRotationXYZ*)child1->getChild(1);
  turretPitch[1] = (SoRotationXYZ*)child1->getChild(3);
```

6.3 Transformačné metódy

Na transformáciu som plne využil knižnicu open inventor. Nemusel som tak vytvárať rôzne zložité výpočty, ale využil som už optimalizované kódy. Vytvoril som dve triedy, v ktorých som aplikoval transformácie.

V *SgBase* som vytvoril metódy na otváranie, respektíve zatváranie brány. Metóda *setKeyDoorRotation* je volaná po stlačení určitej klávesy vyvolávajúcej rotáciu brány. Ako parameter dostane číslo brány a uhol, o ktorý sa má daná brána otočiť. Táto metóda si zavolá metódu *getDoorRotation*, ktorá dostane ako parameter číslo brány a uhol otočenia. Táto metóda si zistí pôvodné natočenie brány a pripočíta si k nemu uhol otočenia. Hneď si porovná nový uhol s maximálnymi hodnotami pre otočenie brány a ak táto hodnota presahuje povolené natočenie, tak metóda vráti maximálne možné natočenie. V opačnom prípade metóda vráti novo vypočítané natočenie. *setKeyDoorRotation* potom naství uhol, ktorý dostala od *getDoorRotation*.

```
baseDoorRotation[name]->angle.setValue(getDoorRotation(name, value));
```

V *SgTransport* som vytvoril dve metódy *setKeyTurretPitch* a *getTurretPitch*, ktoré majú rovnaké algoritmy, ako *setKeyDoorRotation* a *getDoorRotation*. Rozdiel je v tom, že tieto metódy zapríčiňujú otáčanie hlavne na podstavci palebnej veže. Prvý parameter označuje číslo hlavne. Ďalej som vytvoril dve metódy na rotáciu podstavca pre veže. Metóda *setKeyTurretYaw* si zavolá metódu *getKeyTurretYaw*, ktorá vráti natočenie podstavca. Potom *setKeyTurretYaw* pripočíta uhol otočenia a nastaví natočenie.

```
turretYaw[name]->angle.setValue(getTurretYaw(name)+value);
```

Ďalej som pre *SgTransport* vytvoril dve metódy pre transláciu *setSpeed* a *setPosit*. Metóda *setSpeed* sa zavolá, ak sa stlačí na klávesnici nejaká klávesa na zmenu rýchlosti. Sú tu implementované tri zmeny rýchlosti. Mení sa tu jedna hodnota vektoru translácie. Teda Transportér tak získa možnosť pohybovať sa dopredu, dozadu a zastať. Pri stlačení dostane táto metóda parametre zmeny pohybu. Táto metóda nastaví tri premenné, ktoré sa potom použiju na nastavenie pozície. *SetPosit* je metóda, ktorá nastavuje pozíciu transportéru. Na toto použije premenné, v ktorých je uložená rýchlosť a pozíciu, na ktorej sa momentálne nachádza transportér. Momentálnu pozíciu dostane zavolaním metódy *getPos*. Novú pozíciu nataví pomocou *setPos*. Tejto metóde sa predajú ako parametre sčítane hodnoty momentálnej polohy a rýchlosti.

```
setPos(position[0]+speedX, position[1]+speedY, (position[2]+speedZ*dt));
```

7 Exportovanie a importovanie

Exportovanie a importovanie mi robilo občas nečakané problémy, ktoré som musel preklenúť rôznymi spôsobmi.

7.1 Prenos pomocou 3ds formátu

Pri prenose súboru do programu Deep Paint 3D som sa stretol s problémom mapovania. 3D studio pri exportovaní jedného objektu nedokáže zapísať rôzne ID materiálu a teda nie je možné oddeliť mapovacie koordináty. Preotže pri Shrink wrap mapovaní som dostal na jednom obrázku dve pologule a 3ds formát nevie rozdeliť mapovacie koordináty pomocou ID mateiálu, tak sa tieto koordináty pospájali. Tak som stratil presné mapovanie krajných hrán pretože sa roztiahli medzi obe hranice pologulí. Tento problém som vyriešil tak, že som daný objekt rozdelil v mieste, kde sa oddeľujú materiály a exportol som to ako dva nezávislé objekty. V Deep Paint 3D som to importol ako nezávislé objekty. Keďže v 3ds formáte je zapísaná pôvodná translácia voči súradnicovému systému, tak sa tieto dva objekty presne posunuli tak, že som mohol na ne maľovať bez toho, aby som videl nejakú medzeru.

7.2 Prenos pomocou wrl formátu

Prenos pomocou wrl formátu mi už činil značne menšie problémy. Spočiatku sa zdalo, že wrl formát má problémy s presným umiestením textúr, no prekompilovanie Modelvieweru to vyriešilo. Neskôr som narazil na problém načítania z tohoto formátu do uzlov stromu. Tak som to s výdatnou pomocou môjho vedúceho preniesol do formátu iv. Tento formát používa Open Inventor. Má takmer zhodnú štruktúru s wrl, ale je o mnoho prehľadnejší a teda sa s ním ľahšie pracovalo. Ďalší problém nastal, priamo pri exporte do súboru. Niektoré modely exportér zmenšil tak, že otcovi pridal transformáciu zmeny mierky. Takto zmenené objekty potom nezodpovedaly veľkostným pomerom pri tvorbe modelov. Tento problém bolo potrebné vyriešiť dopísaním týchto zmien mierky vo všetkých súborov. S inými problémami som sa pri prenose objektov nestretol.

8 Záver

V tejto bakalárskej práci som vymodeloval celkom 8 modelov. Takmer každý z nich som vymodeloval inou technikou. Využil som niekoľko techník mapovania koordinátov. Na tvorbu textúr som použil tri odlišné programy (3D studio Max 7, Deep Paint 3D, Photoshop 8). Na niektoré z modelov som aplikoval transformácie.

Táto bakalárska práca bude mať hlavný prínos v podobe kompletne namodelovaných, namapovaných a otextúrovaných modelov. Pomocou týchto modelov sa môže vývoj projektu Space Game posunúť o niečo ďalej, pretože sa na nich môžu skúšať rôzne veci, ako napríklad kolízie objektov, zároveň bude možné rozvinúť moje transformácie na úroveň, ktorá by nekončila len pri reprezentácii, ale v nejakom hernom engine. Objekty, ktoré som namodeloval, som sa snažil vytvoriť s veľkou rozdielnosťou v architktúre. Tak bude v budúcnosti možné nadviazať na najvýraznejšie rysy jednotlivých modelov a tak vytvoriť napríklad aj súbor vesmírnych plavidiel. Bakalárska práca naväzuje na projekt p. Masteru (xmaste00). Zdrojové kódy som rozvinul na jeho metódach, ktoré mi uľahčili prácu pri implementácii.

Počas tvorby modelov som sa naučil veľa nových techník, ktoré uľahčia a hlavne urýchlia proces modelovania low-poly modelov. Zo získaných skúsenosti môžem vyvodiť, že modelovacích techník je o mnoho viac, ako som pred touto bakalárskou prácou predpokladal. Pri mapovaní som sa naučil používať dovtedy mnou nepoužívané metódy. A pri tvorbe textúr som sa naučil skombinovať rôzne druhy práce v rôznych programoch. Výsledkom týchto všetkých nových zkúsenosti a vedomostí je zdokonalenie mojích schopnosti pri tvorbe reálnejších a opimalizovanejších objektov. Na druhej strane som sa ponoril hlbšie aj do procesov na pozadí, ktoré sú takmer pri každom klinutí myšou v 3D studiu Max.

Literatura

[1] Bell, J.A. *Efekty a design*. 1.vyd. Brno, Softpress 2000.

[2] Foley, J.D., van Dam, A. *Computer graphics: principles and practice*. 2.vyd. Boston Addison-Wesley Publishing Company 2003

[3] Murdock, K.L., *3ds max 4 Bible*. 1.vyd. New York, Hungry Minds 2001

[4] Murdock, K.L., *3ds max 6 Bible*. 1.vyd. Indianapolis, Wiley Publishing 2004

[5] Žára, J., Beneš,B. *Moderní počítačová grafika*. 1.vyd. Computer Press 2004.

[6] Pečiva, J., Open Inventor: Jednoduché příklady. *root.cz*. dostupné z www: http://www.root.cz/clanky/open-inventor-jednoduche-priklady/

[7] Pečiva, J., Open Inventor: Jednoduchý model. *root.cz*. dostupné z www: http://www.root.cz/clanky/open-inventor-jednoduchy-model/

Príloha

Užívateľská príručka

Tento program zobrazuje trojrozmerné objekty a umožňuje ich ovládať. Umožnuje jednoduchú navigáciu v priestore načítanej scény. Tieto funkcie boli však znefunkčnené z dôvodu potreby prevodu kontroly z okna do vnútra okna. Klávesnicou tak nebude možná ovládať navigáciu, ale bude možné aplikovať transformácie na modeloch pomocou klávesnice. Pre aktiváciu okna je potrebné kliknúť na akúkoľvek ikonu v pravom panely daného okna. Znemožní to ale návrat k ovládaniu objektov. Program sa spúšta pomocou súboru *space.exe*, nie sú potrebné žiadne parametre.

Použiteľné klávesy

Klávesa	**Popis**
1	Zaostrenie ovládania na centrálnu palebnú vežu na transportéri
2	Zaostrenie ovládania na prednú pravú palebnú vežu na transportéri
3	Zaostrenie ovládania na prednú ľavú palebnú vežu na transportéri
4	Zaostrenie ovládania na zadnú pravú palebnú vežu na transportéri
5	Zaostrenie ovládania na zadnú ľavú palebnú vežu na transportéri
W	Pohyb transportéru dopredu
X	Pohyb transportéru dozadu
S	Zastavenie transportéru
A	Rotácia transportéru doľava
D	Rotácia transportéru doprava
R	Otáčanie základne palebnej veže doľava
T	Otáčanie základne palebnej veže doprava
F	Zdvihnutie hlavne palebnej veže
G	Sklopenie hlavne palebnej veže
Z	Otáčanie základne doľava
C	Otáčanie základne doprava
B	Otváranie brány základne
V	Zatváranie brány základne
6	Zaostrenie ovládania na 1. bránu základne
7	Zaostrenie ovládania na 2. bránu základne
8	Zaostrenie ovládania na 3. bránu základne
9	Zaostrenie ovládania na 4. bránu základne
0	Zaostrenie ovládania na 5. bránu základne
-	Zaostrenie ovládania na 6. bránu základne