Dynamics of Image-based Clouds

ŠTUDENTSKÁ VEDECKÁ KONFERENCIA

Bc. Pavol BELUŠKO

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Doc. RNDr. Andrej Ferko, PhD.

BRATISLAVA 2009

Abstrakt

Bibliografická identifikácia	BELUŠKO, Pavol. *Dynamics of Image-based Clouds* [študentská vedecká činnosť]. Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky; Katedra algebry, geometrie a didaktiky matematiky. Školiteľ: Doc. RNDr. Andrej Ferko, PhD. Bratislava : FMFI UK, 2009. 23 s.
Text abstraktu	Cieľom prace je podrobná špecifikácia a implementácia grafického enginu schopného renderovať oblaky v reálnom čase. K dispozícii je možnosť modelovať oblaky. Dôraz je kladený na rýchlosť a použitie optimalizačných techník, ale aj možnosť dynamizácie, rozšíriteľnosť a ľahkú aplikovateľnosť technológie v ďalších aplikáciách.
Kľúčové slová	real-time dynamic image-based cloud

Obsah

Abstrakt i
Obsah ii
1. Úvod 1
2. Softwarové dielo 1
4.1. Technické požiadavky 1
4.2. Použité technológie 2
2.3. Špecifikácia a implementácia softwarového diela 4
2.3.1. CommonStructs 4
2.3.2. Cloud Engine 6
2.3.3. Cloud Modeler 17
2.3.4. Scene Modeler 19
4.4. Špecifikácia transferu dát (Xsd schémy) 20
3. Test výkonu 23
4. Záver 24
5. Zoznam použitého softwaru 24
6. Použitá literatúra 24

1. Úvod

Metódy Image-based renderovania a modelovania majú v počítačovej grafike nezastupiteľné miesto a vo viacerých oblastiach nahrádzajú pôvodné, na geometrii založené prístupy. Sú schopné realisticky a výpočtovo jednoducho zachytiť javy, ktoré je takmer nemožné modelovať geometricky.

Medzi dôležité aplikácie počítačovej grafiky patrí vizualizácia oblakov. Využitie rýchlo vykresľovaných volumetrických oblakov je napríklad v leteckých simulátoroch, kde používateľ trávi väčšinu času vo vzduchu. Väčšina súčasných image-based riešení však dokáže pracovať len so statickými oblakmi.

Cieľom tejto práce je špecifikovať a implementovať systém, ktorý by dokázal v reálnom čase vykresľovať veľké scény plné volumetrických a dynamických oblakov.

Úvodom by som chcel poďakovať spoločnosti Microsoft za zapožičanie textúry za účelom vizuálneho obohatenia diela.

Za východiskovú prácu pre toto dielo bol zvolený článok Niniane Wang – Realistic and fast Cloud rendering ([01]). Tento článok bol citovaný napríklad aj Carstenom Wenzelom v jeho článku Real-time atmospheric effects in games ([04]), kde popisuje použitie tejto modifikovanej technológie v modifikovanej forme v CryEngine 2, v súčasnosti považovanom za jeden z najlepších herných enginov. Toto dokazuje úspech pôvodného konceptu.

2. Softwarové dielo

Nosnou časťou tejto práce je jej praktická časť – softwarové dielo. V nasledujúcich kapitolách sú popísané technické a návrhové aspekty vytvoreného programu.

4.1. Technické požiadavky

Priorita pri písaní programu bola kladená na rýchlosť, cieľom bolo dosiahnuť real-time beh pre veľké scény s hustou oblačnosťou, s čím súvisí nasadenie optimalizačných techník. Nasleduje realistickosť riešenia. Vzhľadom k tomu, že sa pri tejto metóde vizualizácie nejedná o fyzikálne presný model oblakov, realistickosť sa meria vizuálnou presvedčivosťou.

Dôležitými sú aj technické aspekty. Dielo by malo byť modulárne, platformovo nezávislé, škálovateľné, rozšíriteľné na plnohodnotný engine o prepracovanejší osvetľovací model, schopnosť vrhať tiene, nefotorealistické osvetlenie apod. Z hľadiska predpokladaného budúceho nasadenia klient-server architektúry je potrebná schopnosť prenosu oblaku po sieti.

Dielo by malo demonštrovať dynamiku oblakov.

4.2. Použité technológie

Platforma: za účelom zachovania multiplatformnosti boli zvažované dva prístupy: programovanie v nízkom jazyku (C++) kompilovanom pre konkrétny systém alebo programovanie vo vysokom jazyku, pre ktorý existuje virtuálne prostredie, ktoré je portované pre rôzne operačné systémy.

Prvý prístup ma výhodu v tom, že strojový kód, ktorý získame kompiláciou pre konkrétny systém, je optimalizovaný a teda je predpoklad, že bude bežať rýchlejšie než kód interpretovaný. Nevýhodou však zostava fakt, že tato práca je akademická a na jej vývoji sa podieľal jeden človek po dobu približne dvoch rokov, vzhľadom na rýchlosť vývoja hardwaru teda hľadisko optimalizácie technológie v tomto prípade nie je kritické. Programovanie na nízkej úrovni je taktiež časovo náročné, programátor je kvôli vyššej komplexnosti technických detailov náchylnejší na chyby technického i návrhového charakteru. Z hľadiska softwarového manažmentu je teda tento prístup pre túto prácu nevhodný.

Zo spomínaných dôvodov bol na implementáciu ako platforma zvolený MS .NET Framework v najnovšej verzii 3.5. Z hľadiska výkonu je podľa viacerých testov na internete ([06], [07]) pomalší ako natívne kompilované C++ avšak vo väčšine prípadov rýchlejší než Java. Je nutné podotknúť, že cena za interpretovaný jazyk je takmer nemožná časová odhadnuteľnosť behu programu (v exaktnom technickom, nie algoritmickom zmysle. napr. nepredvídateľný moment volania Garbage Collectora), a teda aj jeho odozvy, čo môže byt problém pre real-time aplikácie. Preto bude zaujímavým zistením tejto prace aj závažnosť tejto a podobných skutočností vyplývajúcich z behu interpretovaného programu.

Do úvahy treba vziať obmedzenie týkajúce sa použitých externých knižníc. Oba prístupy vyžadujú ich použitie, je teda dôležité, aby aj tieto boli prístupné v rôznych verziách pre rôzne operačné systémy.

Potenciálnym problémom pri zvolenom .NET Frameworku bol nedostatok knižníc, ktoré by v súčasnosti mali obálku (wrapper) pre tento Framework. V prípade potreby programovania

takéhoto rozhrania by to znamenalo programátorskú záťaž degradujúcu použiteľnosť .NET Framework ako vhodnej platformy. Preto bolo nutne zvážiť tento aspekt. Kvôli vyššej kontrole nad výkonom činnosti programu spojených s grafickými operáciami nebolo použité grafické API vyššej úrovne, akými sú napr. XNA ([08]) alebo Ogre ([09]), ale priamo API OpenGl, pre ktoré existuje wrapper pre .NET - Tao Framework. Súčasťou tohto balíka je aj v práci použitá knižnica utilít pre OpenGl – FreeGlut a knižnica na spracovanie obrazu DevIl. Aplikácia bola vyvíjaná v jazyku C# a použité bolo voľne dostupné prostredie MS Visual C# Express 2008.

.NET Framework ([10])

Microsoft .NET Framework je softwarový systém v súčasnosti dostupný pre rôzne verzie operačného systému MS Windows. Jeho najnovšia verzia v súčasnosti je 3.5. Obsahuje knižnicu s veľkým množstvom naprogramovaných riešení bežných programátorských problémov. Takisto obsahuje virtuálny stroj zodpovedný za kompiláciu a spúšťanie aplikácii, ktoré sú písané priamo pre .NET Framework. S určitými koncepčnými obmedzeniami možno pre .NET kompilovať programy písané v ľubovoľnom programátorskom jazyku.

Visual C# Express 2008 ([11])

Voľne dostupné vývojové prostredie spoločnosti Microsoft podporujúce vývoj aplikácii pre .NET 3.5.

Tao Framework ([12])

Balík multimediálnych knižníc portovaných pre .NET Framework. Je voľne stiahnuteľný v jeho súčasnej verzii 2.1.0 z mája 2008. Obsahuje: Tao.OpenAl (Open Audio Library), Tao.OpenGl (Open Graphics Library), Tao.Sdl (Simple DirectMedia Layer), Tao.Platform.Windows (podporné nástroje pre MS Windows), Tao.PhysFs (platformová abstrakcia súborového systému), Tao.FreeGlut (OpenGL Utility Toolkit), Tao.Ode (Open Dynamics Engine), Tao.Glfw (abstrakcia pre platformovo závislé prvky využívané v OpenGl, ako napr. okná), Tao.DevIl (Developer's Image Library, známa tiež ako OpenIL), Tao.Cg (Cg Shading Language), Tao.Lua (skriptovací jazyk).

OpenGl technológie:

Vertex Buffer Object (VBO, [13])

Vertex Buffer Object je rozšírenie OpenGl poskytujúce metódy na upload dát (vrcholy, normály, atribúty, ...) na grafickú kartu za účelom buffrovaného renderovania (non-immediate-mode rendering). VBO ponuka značné urýchlenie oproti nebuffrovanému renderovaniu, hlavne vďaka tomu, že dáta sú uskladnené v rýchlo dostupnej grafickej pamäti a môžu tak byť použité na vykreslenie bez potreby I/O operácii medzi systémovou pamäťou a grafickou kartou. Technológia VBO bola uvedená v OpenGl 1.4.

Frame Buffer Object (FBO, [14])

Frame Buffer Object je rozšírenie OpenGl pre potreby flexibilného renderovania mimo obrazovky (off-screen), vrátane renderovania do textúry. Pomocou tejto technológie možno implementovať rôzne druhy filtrov a postprocessing efektov (viacprechodové renderovanie). Rozšírenie FBO bolo uvedené v OpenGl 1.1.

Shader ([15])

Moderné grafické karty umožňujú nahradiť fixnú renderovaciu pipeline vlastnými programami. Programátor môže vytvoriť vlastný program na manipuláciu vrcholov (Vertex Shader), fragmentov prichádzajúcich z rasterizera (Fragment Shader) a najnovšie grafické karty počnúc Nvidia GeForce série 8 umožňujú geometriu aj generovať (Geometry Shader).

2.3. Špecifikácia a implementácia softwarového diela

Softwarová časť pozostáva z troch hlavných časti. Prvou časťou je Cloud Engine, grafický engine schopný optimálne renderovať scény oblakov. Obsahuje definície tried pre jednotlivé prvky scény. Vytvorené boli tiež dva nástroje na modelovanie oblakov a scén: Cloud Modeler a Scene Modeler. V úvode je popísaný balík pomocných tried v balíku CommonStructs.

2.3.1. CommonStructs

Funkcionalita:

Balík obsahuje pomocné triedy používané v celom programe.

Špecifikácia tried:

PathStructs.PathFinder

Statická trieda, ktorá umožňuje vyhľadať korektnú cestu k požadovanému súboru. Všetky I/O operácie v programe využívajú túto triedu na zrobustnenie svojej funkcionality. Vstupom metódy Find, ktorá zabezpečuje vyhľadávanie, je cesta k súboru (resp. jeho názov). Ak nie je korektná, metóda sa pokúsi súbor lokalizovať v niekoľkých ďalších preddefinovaných adresároch (napr. koreňový adresár aplikácie, adresár Data apod.). Výstupom je korektná cesta k súboru, ak taká existuje.

UVStructs.UVcoordinate

Obálka pre dvojicu UV textúrových súradníc.

UVStructs.UVquad

Trieda reprezentujúca štvoricu UVcoordinate.

UVStructs.UVtable

Statická trieda obsahujúca 16 inštancií UVquad inicializovaných pre potreby textúry pre sprity. Jednotlivé primitívy len referencujú tieto inštancie za účelom minimalizácie redundancie dát.

MathStructs.Point3D

Point3D reprezentuje bod v trojrozmernom afinnom priestore. Implementuje operácie s bodmi.

MathStructs.Vector3D

Vector3D reprezentuje vektor v trojrozmernom vektorovom priestore. Implementuje operácie s vektormi.

MathStructs.Quaternion

Trieda reprezentuje kvaternión a implementuje operácie na kvaterniónoch.

2.3.2. Cloud Engine

Logický celok Cloud Engine pozostáva z troch balíkov (assemblies): Engine, EngineShared, Cloud. Tieto sú popísané v príslušných podkapitolách.

Funkcionalita:

- Renderovanie grafu scény (objekty, kamera, okolie)
- Voľný pohyb pozorovateľa po scéne
- Optimalizácia (implementácia optimalizačných technológii)
- Import/export oblakov (Xml)
- Modifikátory pohybu objektov a kamery (rôzne módy pohybu oblakov a kamery).

Výkon:

- Beh v reálnom čase (30 a viac FPS)
- Plynulosť (zamedziť veľké dávkové spracovania v rámci počítania jedného framu)

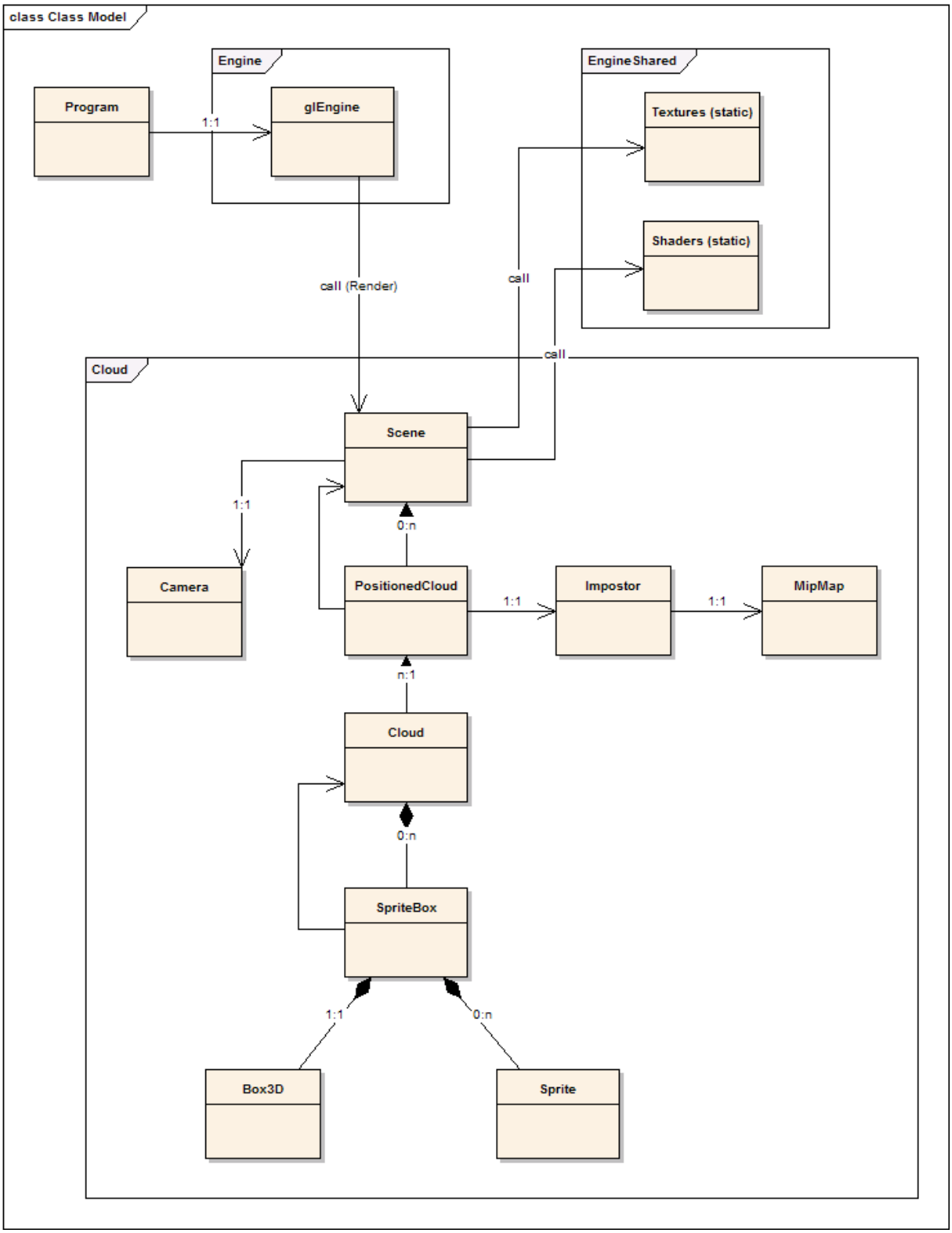
Externé rozhrania:

- Inicializácia, start/stop
- Rozhrania na manipuláciu so scénou
- Rozhrania na ovplyvňovanie atribútov scény a enginu

Atribúty

- Aktuálne vykresľovaná scéna
- Letové módy (free flight, airplane)
- Módy na ovplyvnenie farebnosti oblakov (enhance by distance, enhance by height)
- Editovaci mód (zvýraznenie aktuálne označeného oblaku zapnuté/vypnuté)
- Vykresľovanie Sky Boxu (zapnuté/vypnuté)
- Fullscreen/Window

Špecifikácia tried:



UML model tried zabezpečujúcich hlavnú funkcionalitu

Engine.glEngine

Táto trieda je umiestnená v samostatnom balíku (assembly) Engine. Inicializuje prostriedky OpenGl, vytvorí renderovací kontext (okno), registruje callback funkcie (vstup (klávesové skratky), renderovací cyklus, zmena veľkosti okna, ...), zabezpečuje vykresľovanie vonkajšieho prostredia. Sem patrí vykreslenie súradnicových osí či skyboxu. Taktiež prepínanie nasledujúcich atribútov scény.

K dispozícii sú dva módy letu v scéne:
MovementType.FREE_FLIGHT – kamera sa posunie o fixný skok v smere danom stlačenou klavesou (použiteľný hlavne pri modelovaní oblaku/scény kvôli efektivite pohybu)
MovementType.AIRCRAFT – aktivuje sa modifikátor pohybu kamery, ktorý pri každom vykresľovacom cykle posunie kameru v smere jej normálového vektora. Možno určiť rýchlosť pohybu.
V práci sú implementované dva druhy nefotorealistického ofarbovania, ktoré možno zapnúť a vypnúť. Tieto módy slúžia na zlepšenie vizuálnej stránky.
No Enhancement – oblaky sú jednej farby (základná textúra zblendovaná s bielou farbou)
Enhance by Distance – Vzdialenejšie oblaky sú tmavšie, čo napomáha dojmu hĺbky scény. Toto nastavenie je pri štarte zapnuté.
Enhance by Height – Oblaky sú svetlejšie, čím vyššie sa nachádzajú v scéne. Toto nastavenie je pri štarte vypnuté.
Taktiež možno zapnúť/vypnúť Editovací mód. Toto nastavenie sa týka len použitia v programe Scene Modeler, kde používateľ volí aktuálne editovaný oblak. Keď je Editovací mód zapnutý, aktuálny oblak je zvýraznený červenou farbou. Toto zvýraznenie je možné vypnúť po skončení editovania. Toto nastavenie je pri štarte zapnuté.
Trieda glEngine umožňuje zapnúť/vypnúť vykresľovanie skyboxu okolo scény. Toto nastavenie je pri štarte zapnuté.
Renderovacíi kontext možno prepnúť do Fullscreen módu.

Všetky nastavenia zvyšujúce vizuálnu kvalitu sú vypnuteľné za účelom merania výkonu pri vykresľovaní oblakov.

Trieda glEngine zabezpečuje volanie vykresľovania scény. Aktuálne vykresľovanú scénu možno kedykoľvek nastaviť na ľubovoľnú inú inštanciu tejto triedy. glEngine taktiež

podporuje zmenu modifikátora pohybu kamery v prípade potreby implementácie ďalšieho typu.

EngineShared.GlEnvironments

Statická trieda určená na prepínanie sady stavov OpenGl. Prepína nasledujúce stavy: MAIN_RENDER (hlavný renderovací cyklus v triede glEngine), SCENE_RENDER (hlavný renderovací cyklus triedy Scene), CLOUD_RENDER (vykreslenie oblaku v Geometry móde), IMPOSTOR_RENDER (vykreslenie impostora oblaku), IMPOSTOR_REDRAW (vykreslenie oblaku do impostora). Účelom triedy je sprehľadniť nastavovanie stavov OpenGl.

EngineShared.Shaders

Statická trieda určená na natiahnutie, kompiláciu a linkovanie shadrovych programov použitých v programe. Taktiež zabezpečuje ich prepínanie. Pri neúspešnom pokuse o natiahnutie alebo kompiláciu shadera vyvolá výnimku (jedna sa o kritickú funkcionalitu). Trieda definuje dva shadrove programy pre potreby tohto enginu: CloudShaderProgram (shader na geometrickú manipuláciu so spritmi), FixedPipelineProgram (program fixed pipeline = vypnutie vlastného shadera).

EngineShared.Textures

Statická trieda určená na natiahnutie v programe používaných textúr a nastavenie ich parametrov a ich prepínanie. V prípade problému vyvolá výnimku. Trieda definuje textúru pre vykreslenie oblakov a sky boxu.

V predošlých dvoch triedach bol zvolený prístup zamedzenia zbytočných Gl volaní. Pri programovaní boli na nastavenia shadera/textúry použité výhradne volania pomocou týchto dvoch tried, čo umožnilo udržiavanie Id čísla aktuálneho shadera/textúry a tým diagnostiku, či je potrebné prepnutie.
Samy osebe nie sú neúnosne drahými operáciami, avšak sú volané často, pričom zmena shadera či textúry nie je v programe potrebná často. Pritom z hľadiska objektového prístupu by mal každý objekt (v našom prípade oblak) zabezpečiť korektnosť všetkých stavov OpenGl, čo však nie je optimálne. Na druhej strane prísna optimalizácia a redukcia Gl volaní na základe predpokladov o následnosti vo vykresľovaní je veľmi nebezpečný prístup vedúci

k neprehľadnému kódu a je v rozpore s paradigmou objektového programovania. Toto riešenie je teda vhodným kompromisom.

Cloud.Box3D

Trieda implementujúca kváder. Táto trieda bola použitá na definíciu obaľujúcich objemov, z ktorých sa buduje model oblaku. Definuje metódy na manipuláciu s kvádrom a má schopnosť vykresliť svoje priemety do rovín xy, yz, xz v rámci metódy kresliacej do objektu Graphics (.NET). Box3D nemá schopnosť vykresliť sa v OpenGl.

Cloud.Sprite

Základná primitiva každého oblaku implementovaná ako GL_QUADS. Drží informáciu o pozícii, rozmeroch, UV súradniciach použitých pri textúrovaní, id číslo a náhodnú rotáciu okolo vlastnej normály (tretí rozmer rotácie. Zvyšné dva sú počítané automaticky. UV súradnice sú predrátané v statickej triede UVtable (pozri špecifikáciu týchto tried), Sprite drží len referenciu do tejto tabuľky. Trieda Sprite mala v pôvodnej verzii schopnosť vykreslenia v OpenGl, táto funkcionalita však bola zablokovaná kvôli použitiu optimálnejšej technológie (pozri triedu PositionedCloud). Sprite má schopnosť vykresliť svoje priemety do rovín xy, yz, xz v rámci metódy kresliacej do objektu Graphics (.NET).

Cloud.SpriteBox

Trieda SpriteBox je obálkou pre inštanciu triedy Box3D a zoznam inštancií Sprite. Hlavnou funkciou je zjednotiť manipuláciu s obaľujúcim objemom daným inštanciou Box3D a spritmi umiestnenými do tohto objemu. Ma tiež schopnosť vyplniť tento objem náhodne rozmiestnenými spritmi.

Cloud.Cloud

Trieda Cloud je hlavnou triedou Cloud Enginu. Hlavnou funkciou je reprezentácia modelu oblaku. Obsahuje zoznam SpriteBoxov. Určuje spôsob vyplnenia objemu daného SpritBoxmi na pseudonáhodnom princípe. Používateľ môže zmeniť typ oblaku (rozsah UV hodnôt pre textúrovanie spritov), veľkost spritov, prípustnú odchýlku veľkosti, počet spritov, ich minimálnu vzdialenosť. Na základe týchto parametrov náhodne vyplní svoj objem.
Cloud následne udržuje zoznam všetkých Spritov v ňom za účelom ich usporiadania. Nemá schopnosť Gl vykresľovania, je však najvyššou triedou podporujúcou kreslenie do Graphics (.NET). Nástroj Cloud Modeler vytvára model oblaku na tejto úrovni abstrakcie.

Cloud.MipMap

Trieda MipMap je vlastnou modifikovanou implementáciou technológie MipMappingu. MipMapping je technológiou rozširujúcou textúry. K hlavnej textúre sa predpočítajú textúry vo viacerých úrovniach tak, že každá ďalšia úroveň je štvrtinového rozlíšenia (každý rozmer sa predelí dvoma) od predošlej. Podľa potreby sa potom vyberie vhodná textúra, čím sa predíde spracovaniu zbytočne veľkého množstva dát a takisto sa pomôže zamedziť vzorkovacím problémom. Použitie MipMappingu zvyšuje pamäťovú náročnosť programu. Vyžaduje približne 1.3 násobné viac miesta v pamäti, čo je však prijateľný pomer vzhľadom na to, aké urýchlenie táto technológia ponúka. V tejto práci boli MipMapy použité pri textúrovaní spritov.

Pri renderovaní impostorov bolo použitie MipMáp modifikované pre túto technológiu. Pri prekreslení impostora by sa museli prekresliť všetky úrovne MipMapy. Toto je zbytočná záťaž vzhľadom na fakt, že počas životnosti impostora sa jeho priemet čo do veľkosti veľmi nezmení a teda je málo pravdepodobné použitie viacerých úrovni MipMapy. Preto je použitie MipMappingu ako konceptu vyslovene nevhodné. Naša trieda MipMap je preto skôr kolekciou textúr viacerých rozlíšení. Renderuje sa vždy len do jednej úrovne. Pri prechode do ďalšej je nutné obraz vykresliť znovu.

Ďalším rozdielom je adaptívne rozlíšenie a škálovací faktor medzi úrovňami MipMap. Maximálne rozlíšenie možno stanoviť veľkosťou priemetu oblaku v mieste, kde je najbližšie ku kamere a ešte sa nekresli v plnom geometrickom móde. Keďže táto vzdialenosť je previazaná s polomerom obaľujúceho objemu oblaku, tento priemet je konštanta. Do úvahy treba vziať aj rozlíšenie okna. V programe práve toto určí maximálnu veľkosť MipMap. Optimalizácia teda spočíva v tom, že pri zmene rozlíšenia celého renderovacieho kontextu sa realokuju aj všetky MipMapy v scéne. Spodné ohraničenie rozlíšenia možno však len odhadnúť a to podľa veľkosti priemetu oblaku vo vzdialenosti zadnej orezávacej roviny. Rozdiel medzi maximálnym a minimálnym priemetom je však ca 3-násobný (pri fixne zvolených parametroch kamery). Preto by použitie MipMap so škálovacím faktorom 1/2 neprinieslo optimálny efekt; reálne by sa použilo málo úrovní. Bol preto zvolený škálovací faktor 2/3 a počet úrovní 4.

Škálovací faktor nie je vhodné mať príliš vysoký. Spolu s nárastom počtu úrovní by to prinieslo neprimerané zvýšenie pamäťových nárokov a viedlo by to tiež k zbytočne častému prepínaniu úrovní, čím by sa vyvolalo prekreslenie impostora.

Podľa Nyquist–Shannonovej vety o vzorkovaní je vhodné použiť textúru minimálne dvojnásobného rozlíšenia ako je minimálne nutné. Vzhľadom na značne amorfnú povahu oblakov a snahu dosiahnuť čo najvyššiu rýchlosť programu však toto nebolo zohľadnené a použitá je textúra minimálneho možného rozlíšenia. Z výsledkov je vidieť, že zhoršenie kvality je nepatrné.

Z technického hľadiska má trieda MipMap schopnosť alokovať/uvoľniť textúry v nastaviteľnom maximálnom rozlíšení, škálovacom faktore a počte úrovní a obsahuje podporné funkcie na určenie rozlíšenia na príslušnej úrovni a na určenie vhodnej úrovne pre renderovanie podľa veľkosti priemetu objektu.

Cloud.Impostor

Impostor je definíciou zástupcu pre oblak. Implementačne sa jedná o jeden veľký sprite, ktorý si okrem štyroch bodov definujúcich polygón ukladá aj pozíciu kamery, z ktorej bol zástupca zhotovený (je to nutné z dôvodu použitia Shader programu pre Sprity, ktorý ich rotuje smerom ku kamere). Stará sa o alokáciu a uvoľňovanie VBO pre vrcholy a FBO ako renderovacieho kontextu, do ktorého sa priraďuje potrebná textúra z objektu MipMap, ktorý Impostor vlastní. Impostor sa vie vykresliť v OpenGl renderovanom kontexte, avšak o samotné snímanie obrazu oblaku sa nestará kvôli optimalizácii volaní. Pri vykreslení do Impostora sa totiž používa tá istá metóda na vykreslenie oblaku ako v plnom geometrickom móde (renderovanie oblaku je definované v triede PositionedCloud). Spôsob zachytenia obrazu do Impostora je popísaný v príslušnej kapitole.

Cloud.PositionedCloud

PositionedCloud je nadstavbou na Cloud. Obsahuje polohu oblaku v scéne, údaje o jeho náhodnom natočení a referenciu na inštanciu triedy Cloud (škálovanie nie je implementované, implementácia však nie je náročná). Obsahuje tiež inštanciu Impostora. PositionedCloud má navyše len minimum informácie oproti Cloud a aj napriek tomu, že sa jedná o jeho špeciálny prípad, nie je odvodený od Cloud z technických dôvodov. Je predpoklad, že v scéne sa bude vyskytovať viacero inštancii jedného typu oblaku. Kvôli minimalizácii pamäťových nárokov a inicializačných činností (medzi ktoré patrí aj natiahnutie zo súboru, čiže náročná I/O operácia) je teda vhodné udržovať zoznam vytvorených inštancií a len určovať, kam a ako sa má konkrétny oblak vykresliť. Oblaky možno natiahnuť dopredu pri známej scéne, spomínaný prístup je však užitočný pre

dynamické scény (aj v zmysle zamýšľanej klient-server funkcionality, kde bude vzhľad scény určovať server).

PositionedCloud riadi vykreslenie v rámci Gl kontextu. Inicializuje Vertex Buffer Object (VBO) a vypĺňa ho dátami z priradeného Cloud. Jedna sa o dôležitú optimalizáciu, pretože v plnom móde sa množstvo I/O operácii medzi operačnou pamäťou a grafickou kartou nahradilo jedným volaním- vykreslením VBO nášho oblaku.

Pri inicializácii sa tiež vytvorí inštancia triedy Impostora, ktorá je implementáciou modifikovanej technológie zástupcov. Pri vzdialení sa od kamery sa prepne vykresľovanie plnej geometrie na vykresľovanie Impostora. PositionedCloud vtedy kontroluje nutnosť prekreslenia impostora. Táto technika je tiež významnou optimalizáciou, pretože nielen znižuje komplexnosť scény, ale znižuje aj tzv. overdraw, číže množstvo dát akumulovaných v Stencil Buffri.

Ukázalo sa, že počet I/O operácii a Alpha blending sú kritickými miestami v tomto systéme. Spomínané technológie pomohli limitovať oba problémy.

PositionedCloud vie kresliť do inštancie Graphics (.NET), pričom vykreslí len kružnicový priemet obaľujúcej gule. Je možné nastaviť škálovanie zobrazenia, čo je výhodné pri modelovaní veľkej scény.

Dôležitou funkcionalitou je taktiež kontrola nutnosti preusporiadať Sprity podľa vzdialenosti a prekresliť Impostora (delta uhol, delta vzdialenosť, preusporiadanie, moment zapnutia Impostora). Prekreslenie a preusporiadanie sa nedeje pri každom pohybe, experimentálne boli stanovené intervaly tolerancie, pričom bolo snahou dosiahnuť synchronizáciu oboch činností.

Cloud.Camera

Trieda kamera reprezentuje pozorovateľa. Obsahuje informáciu o pozícii, vektor smerovania (facing - normála priemetne), up-vektor a right-vektor kamery. Vektor facing možno nastaviť, pomocou neho sa následne dopočíta right-vektor tak, že je rovnobežný s rovinou xy (kamera je tým fixovaná proti otočeniu okolo vektora facing). Up-vektor je vypočítaný ako vektorový súčin vektora facing a right-vektora. Všetky tieto vektory sa udržujú normalizované, pretože dôležitý je ich smer.

Možné je nastaviť parametre premietania, a to rozlíšenie obrazovky, vertikálny zorný uhol (horizontálny sa dopočítava na základe pomeru strán z rozlíšenia), vzdialenosť prednej a zadnej orezavacej roviny. Predná orezávacia rovina je takisto rovinou priemetne, čiže jej vzdialenosť od priemetne definuje ohniskovú vzdialenosť.

Trieda Camera tiež definuje metódy na výpočet veľkosti priemetu objektov na obrazovku a metódy na pohyb kamery.

Inštancii typu Camera môže byť vytvorených ľubovoľne veľa (aktívna len jedna), všetky môžu byť animované (rotácia kamery je implementovaná pomocou kvaterniónov, čo je vhodné pre animácie, pretože kvaternióny sa jednoducho interpolujú). Grafický engine tak dostáva možnosti réžie snímania.

Cloud.Scene

Trieda Scene reprezentuje graf scény. Pozostáva z kolekcie oblakov a referencie na aktívnu kameru. Obsahuje metódy na manipuláciu oblakov, dôkaze sa vykresliť v OpenGl renderovanom kontexte a v kontexte Graphics, takisto kontroluje a nastavuje globálne nastavenia OpenGl ako je pozícia kamery pre shader. Stará sa tiež o usporiadanie oblakov podľa vzdialenosti od kamery. Prístup separovaného usporadúvania (usporiadanie oblakov a nasledne spritov v rámci nich) bol uprednostnený pred usporadúvaním všetkých spritov v scéne kvôli výpočtovej náročnosti. Platí, že usporiadanie viacerých menších úsekov je asymptoticky rýchlejšie ako usporiadanie jedného veľkého. Taktiež v rámci oblakov, ktoré sú blízko pozorovateľa, treba preusporadúvať sprity častejšie ako pri vzdialených oblakoch. Pri zjednotenom prístupe by sa medzi týmito situáciami nedalo rozlišovať. Nevýhodou separovaného prístupu je podmienka, že obaľujúce gule oblakov musia byť disjunktné, inak by mohlo dôjsť k artefaktom v dôsledku zlého usporiadania.

Nasledujúca hierarchia tried popisuje funkcionalitu importu a exportu prvkov scény. Bola navrhovaná tak, aby bolo prvky možné ukladať do súboru alebo do inštancie XmlDocument. Druhá spomínaná metóda je prípravou na prenos oblakov po sieti. Triedy implementujú návrhový vzor Visitor. Je tak možné definovať ľubovoľné ďalšie spôsoby importu/exportu.

Cloud.Loaders.AbstractLoader

Základná abstraktná definícia triedy na import a export objektov scény. Definuje metódy Load a Save do súboru.

Cloud.Loaders.XmlLoader

Interface, ktorý definuje metódy Load a Save do Xml dokumentu.

Cloud.Loaders.FileXmlLoader

Trieda FileXmlLoader špecializuje triedu AbstractLoader a implementuje interface XmlLoader. Zabezpečuje ukladanie objektu uloženého do inštancie XmlDocument do súboru. Tento XmlDocument získa využívajúc virtuálne volania XmlLoadera. Implementáciu metód XmlLoadera však prenecháva ďalším špecializáciám. Účelom tejto triedy je zabezpečiť ukladanie na I/O úrovni.

Nasledujúce Loader triedy okrem PositionedCloudXmlLoader sú špecializáciou triedy FileXmlLoader.

Cloud.Loaders.CloudXmlLoader

Definuje ukladanie a natiahnutie objektu Cloud z inštancie XmlDocument implementovaním abstraktných metód vyžadovaných z FileXmlLoader.

Cloud.Loaders.PositionedCloudXmlLoader

Definuje ukladanie a natiahnutie objektu PositionedCloud z inštancie XmlDocument. Využíva CloudXmlLoader ako prostriedok na manipuláciu s informáciami o objekte Cloud, ktorý trieda PositionedCloud referencuje. Podporuje ukladanie v plnom móde (ukladá sa celý Xml strom objektu Cloud) a v referenčnom móde (ukladá len názov súboru, z ktorého sa Cloud natiahne). Nie je špecializáciou FileXmlLoader, implementuje len rozhranie XmlLoader. Nepodporuje tak ukladanie do súboru. Táto funkcionalita nie je reálne využiteľná. Oblak totiž do súboru ukladá CloudXmlLoader a oblak s pozíciou má význam len v rámci konkrétnej scény.

Cloud.Loaders.SceneXmlLoader

Definuje ukladanie a natiahnutie objektu PositionedCloud z inštancie XmlDocument. Využíva PositionedCloudXmlLoader ako prostriedok na manipuláciu s oblakmi v scéne.

Modifikátory pohybu slúžia na animáciu pohybu. Implementujú návrhový vzor Visitor. Možno tak definovať rôzne spôsoby pohybu pre objekty bez zásahu do tried samotných.

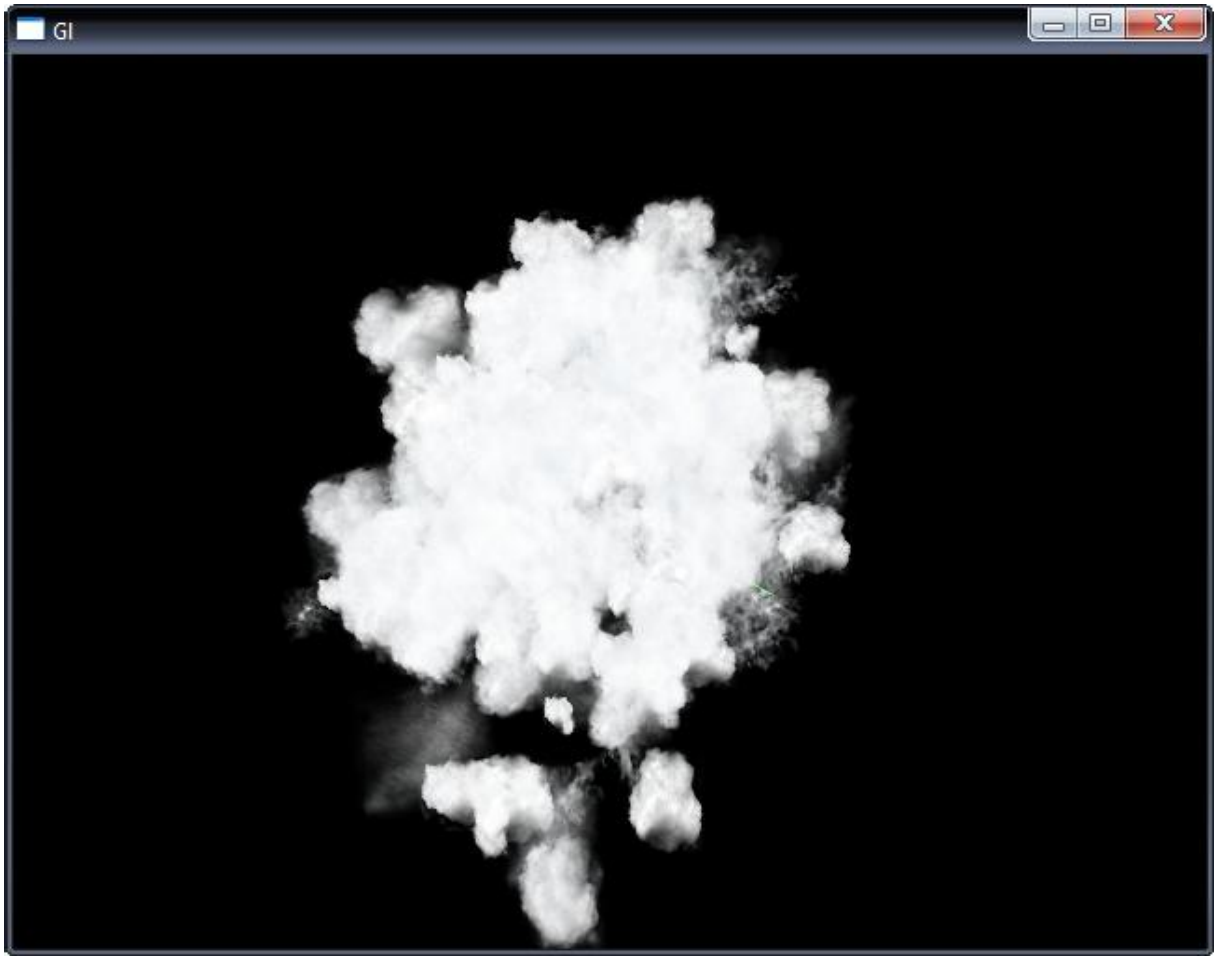
Cloud.Modifiers.AbstractModifier

Abstraktná trieda, z ktorej sú odvodene všetky modifikátory pohybu prvkov v scéne. Definuje metódu Step, ktorá reprezentuje jeden krok v animácii.

Cloud.Modifiers.CameraModifier

Modifikátor pohybu kamery. Možno nastaviť rýchlosť pohybu. Kamera sa pohybuje v smere pohľadu.

Grafické rozhranie (GUI)



Screenshot GUI Enginu

Podporovane klávesové skratky:

'Esc': ukončenie programu

'w': posun vpred (Free Flight) / akcelerácia (Aircraft)

's': posun vzad (Free Flight) / spomalenie (Aircraft)

'a': posun vľavo (Free Flight) pozdĺž right-vektora kamery

'd': posun vpravo (Free Flight) pozdĺž right-vektora kamery

'r': posun nahor (Free Flight) pozdĺž up-vektora kamery

'f': posun nadol (Free Flight) pozdĺž up-vektora kamery

'x': zapni/vypni mód Enhance by Distance

'z': zapni/vypni mód Enhance by Height

'c': zapni/vypni mód Editovací mód

'v': prepína medzi pohybovými módmi

'b': zapni/vypni vykresľovanie Sky boxu

'm': zapni/vypni Fullscreen mód

Orientácia kamery sa nastavuje stlačením ľavého tlačidla myši a ťahaním.

2.3.3. Cloud Modeler

Nástroj Cloud Modeler bol vyvinutý ako prostredie na modelovanie jednotlivých oblakov. Modelovanie spočíva v rozmiestňovaní obaľujúcich objemov v scéne, ktoré budú následne vyplnené spritmi.

Funkcionalita:

- Rozmiestňovanie a manipulácia s obaľujúcimi boxmi v trojpohľadovom nákrese
- Nastavenie atribútov oblaku
- Nastavenia vizualizácie (po optimalizácii technológie renderovania oblaku neboli reimplementované)
- Import/export

Výkon:

- Náhľad na modelovaný oblak v reálnom čase

Externé rozhrania:

- Grafické rozhranie
- Konzolové ovládanie programu nie je podporované

Atribúty (atribúty modelovaného oblaku)

- Aktuálne modelovaný oblak
- Názov oblaku (používateľom zvolené pomenovanie)
- Typ oblaku (kumulus, status, masívny kumulus, cirrus)
- Veľkosť spritu (základná veľkosť primitívy)
- Interval variácie veľkosti spritu. Definuje povolenú odchýlku od veľkosti spritu. Tá je následne zvolená náhodne.
- Počet spritov
- Minimálna vzdialenosť spritov

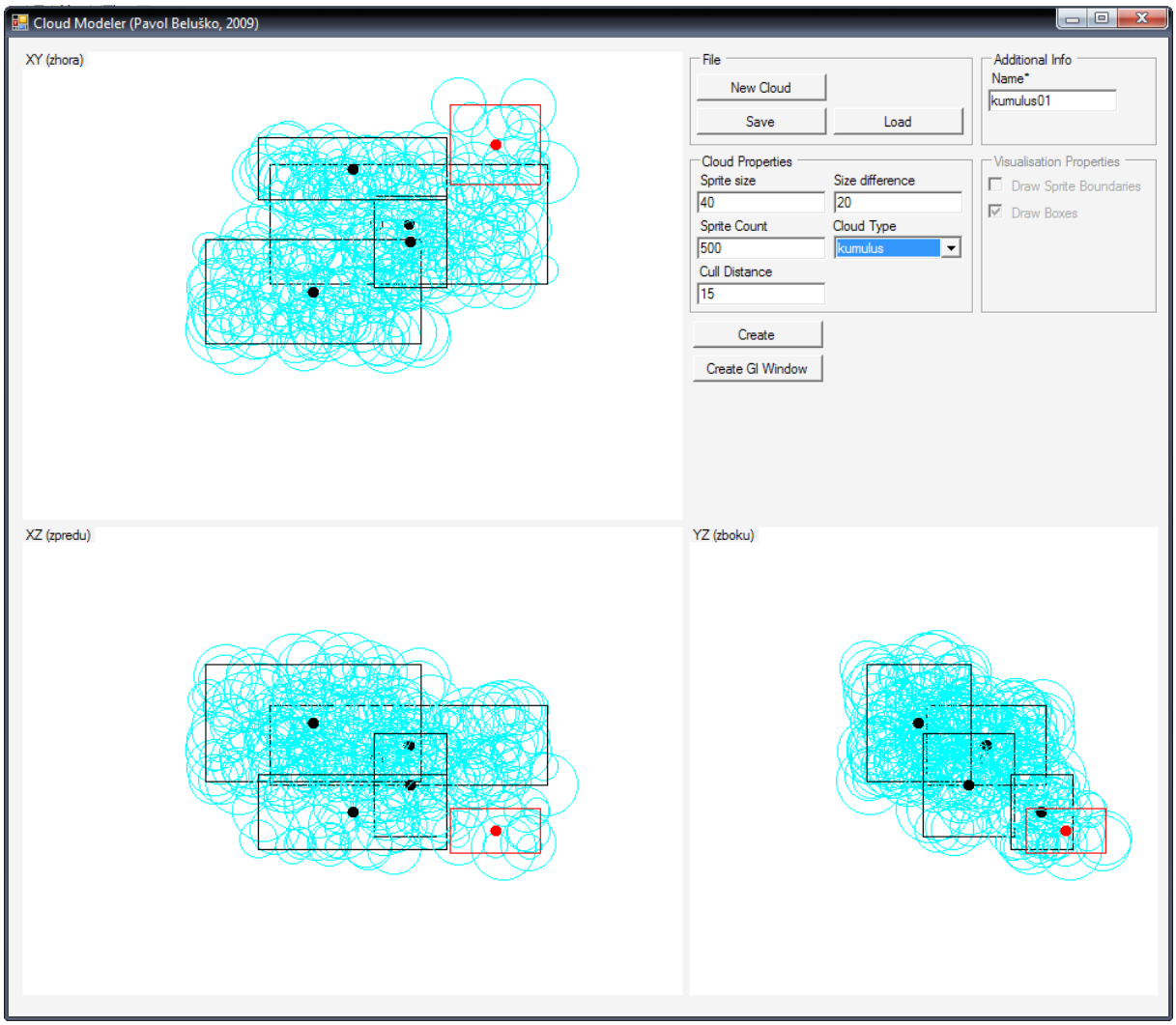
- Vizualizačné nastavenia: zvýraznenie spritov obtiahnutím líniou v OpenGl, vizualizácia boxov v OpenGl (zablokované nastavenia – funkcionalita nebola reimplementovaná)

Špecifikácia tried:

CloudModeler.CloudModelerMainForm

Trieda definuje grafické rozhranie programu. Grafické rozhranie je oddelené od biznis logiky. Využívajú sa teda volania existujúcich biznis tried zo sekcie Cloud Engine. Trieda samotná implementuje techniku doublebufferingu nad panelmi trojpohľadového nákresu na zamedzenie blikania obrazu pri jeho animácii.

Grafické rozhranie (GUI)



Screenshot GUI Cloud Modelera

Ovládacie prvky na paneloch:

Ľavé tlačidlo myši – umiestnenie nového boxu/označenie boxu

Ľavé tlačidlo myši + ťahanie – premiestnenie označeného boxu

Pravé tlačidlo myši + ťahanie – zmena rozmerov označeného boxu

2.3.4. Scene Modeler

Scene Modeler bol vyvinutý ako prostredie na modelovanie formácii oblakov. K dispozícii je náhľad zhora na scénu v mierke 1:5 (mierka je meniteľná).

Funkcionalita:

- Rozmiestňovanie a manipulácia s oblakmi
- Vytváranie kolekcii typov oblakov
- Import/export scén a kolekcií oblakov

Výkon:

- Náhľad na modelovanú scénu v reálnom čase

Externé rozhrania:

- Grafické rozhranie
- Konzolové ovládanie programu nie je podporované

Atribúty

- Aktuálne modelovaná scéna
- Aktuálna kolekcia typov oblakov

Špecifikácia tried:

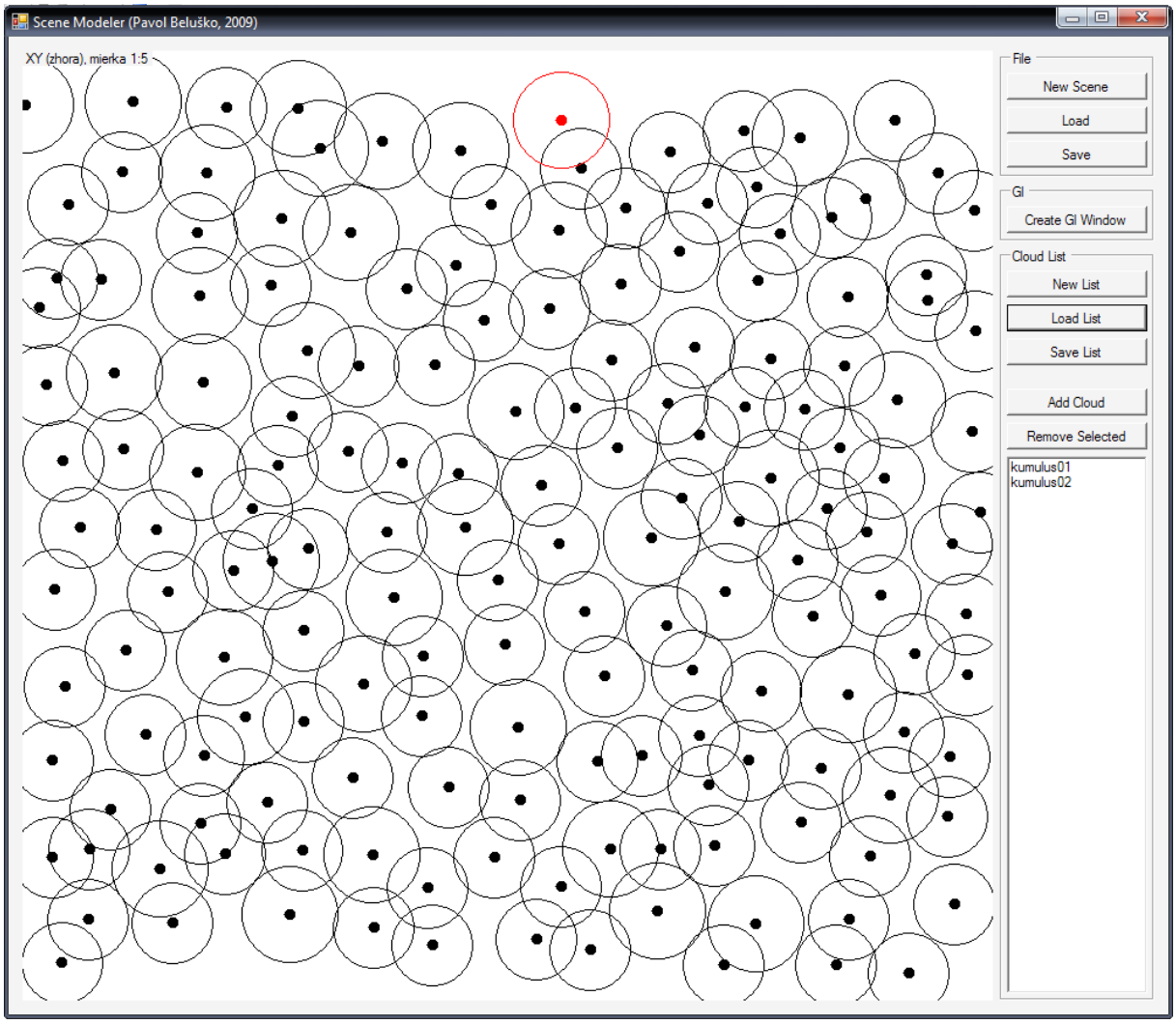
SceneModeler.SceneModelerMainForm

Grafické rozhranie programu. Je oddelené od biznis logiky, využíva volania existujúcich biznis tried.

SceneModeler.CloudList

Trieda CloudList reprezentuje zoznam typov oblakov (trieda Cloud). Tento zoznam je potrebný pre výber oblakov umiestňovaných do scény. Je možné uložiť a natiahnuť ho z Xml súboru.

Grafické rozhranie (GUI)



Screenshot GUI Scene Modelera

Ovládacie prvky na paneloch:

Ľavé tlačidlo myši – umiestnenie nového oblaku/označenie oblaku

Ľavé tlačidlo myši + ťahanie – premiestnenie označeného oblaku

Koliesko myši – zmena výškového umiestnenia oblaku

Ctrl + koliesko myši – rotácia oblaku okolo svojej zvislej osi.

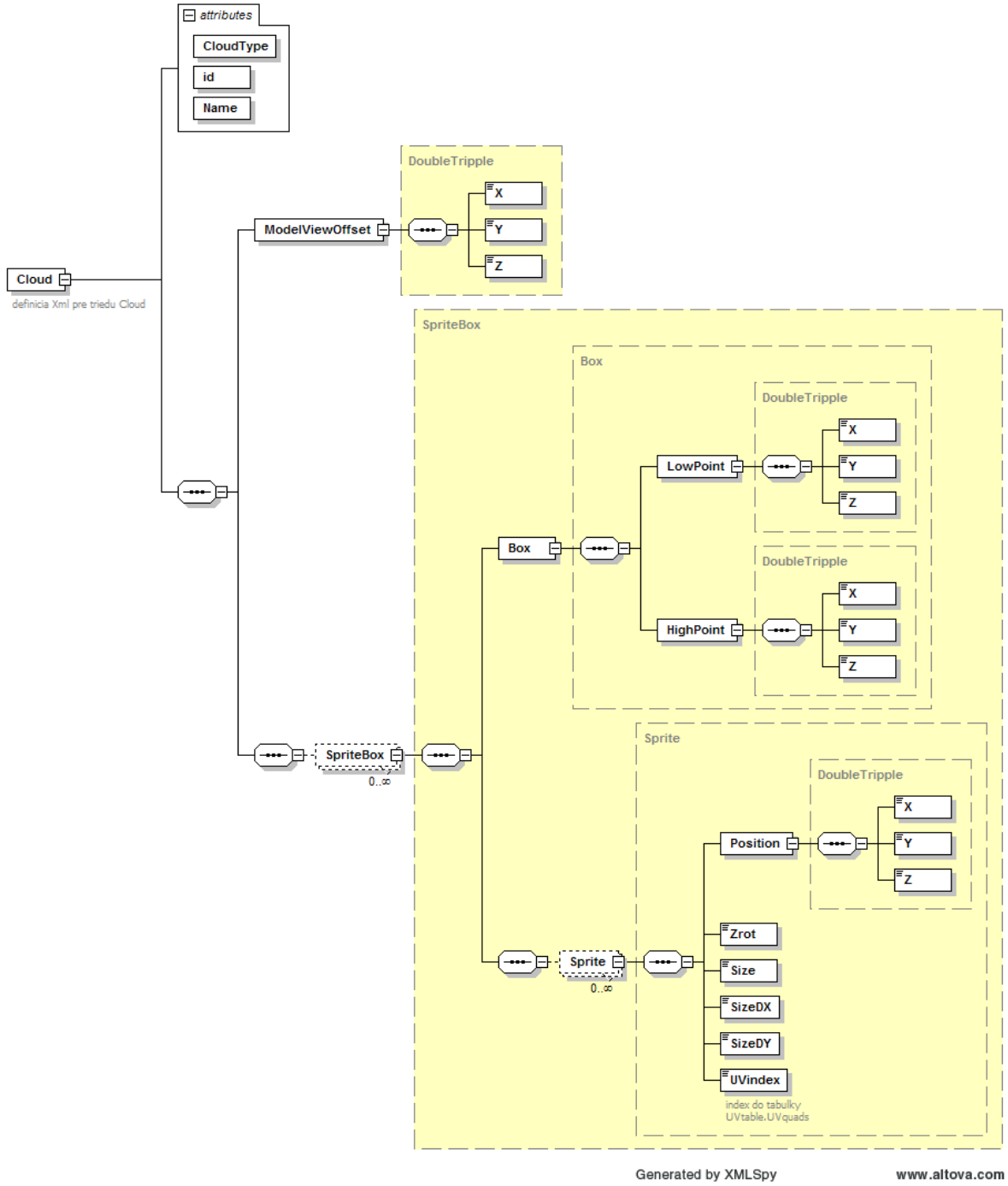
4.4. Špecifikácia transferu dát (Xsd schémy)

Všetky Loadery a CloudList ukladajú dáta do Xml štruktúr. Tento prístup má výhodu v ľahkej manipulácii s dátami a prehľadnosti. Xml je následne možné uložiť do súboru/preniesť po sieti. Systém je tak pripravený na rozšírenie na klient-server model. Pri

importe/exporte do Xml sa vykonáva validácia oproti nasledujúcim Xsd schémam. Zabezpečuje sa tým bezproblémové parsovanie.

Xsd pre Cloud

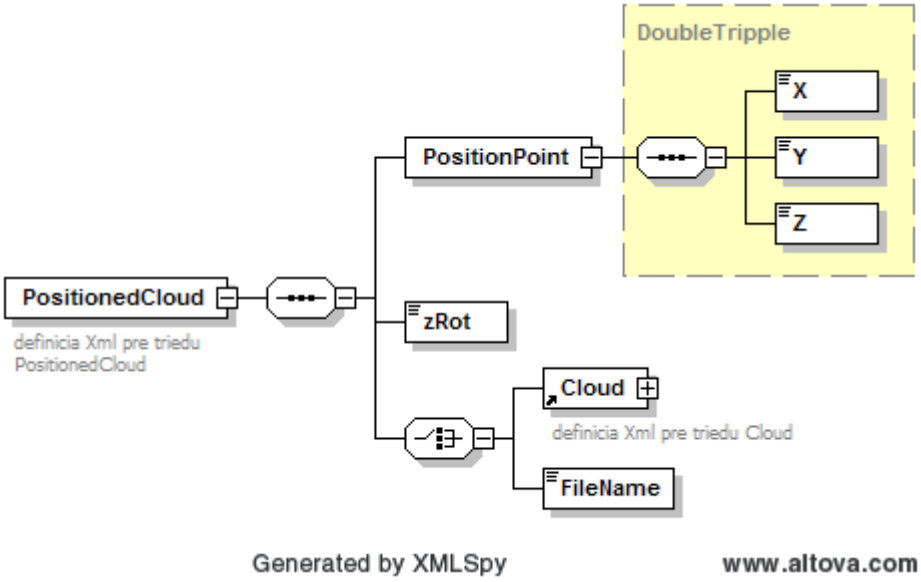
Xsd schéma odráža hierarchiu obsahu triedy Cloud. Cloud obsahuje neznámy počet inštancií SpriteBox.



Cloud.xsd

Xsd pre PositionedCloud

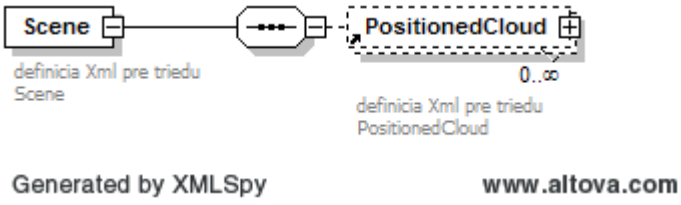
V Xsd schéme pre PositionedCloud je vidieť voľbu medzi plným módom a referenčným módom ukladania triedy PositionedCloud. V plnom móde sa ukladá obsah triedy Cloud, v referenčnom len cesta k súboru, kde je uložený obsah triedy Cloud.



PositionedCloud.xsd

Xsd pre Scene

Scene obsahuje neznámy počet inštancií PositionedCloud.



Scene.xsd

Xsd pre CloudList

CloudList obsahuje neznámy počet typov oblakov.

CloudList.xsd

3. Test výkonu

Referenčný hardware:

CPU: Athlon X2 4850e (2 x 2.5 GHz)

Operačná pamäť: 4 x 1GB DDR2 A-Data Extreme Edition @800Mhz

Grafická karta: Nvidia GeForce 7900GT 256MB

Namerané výsledky:

Renderovaná ukážková scéna obsahuje vyše 46 000 textúrovaných spritov. Na uvedenom hardwari dosiahol systém v priemere 85FPS v rozlíšení 1680x1050x32bit so všetkými efektmi zapnutými (dynamika, color enhancement, skybox). Výkon sa líšil najmä v dôsledku potreby fragmentových operácii. Ak sa oblaky premietali na celú obrazovku, systém dosahoval svoje minimum – približne 70FPS. Beh programu bol po celú dobu plynulý, bez veľkých výkyvov v nameraných FPS. Ukazuje sa tak, že použitie vyššieho interpretovaného jazyk, akým je C# na platforme .NET je možné a na akademické účely pre plynulosť programovania a robustnosť aj vhodné.

Výrez zo screenshotu z aplikácie vo Fullscreen móde

4. Záver

Implementovaný bol editor oblakov podporujúci viacero typov oblakov a editor scén. Podarilo sa ukázať, ako je možné využiť Vertex Shader program a rôzne optimalizačné techniky na vizualizáciu veľkých dynamických scén rôznych typov oblakov. Fakt, že dynamika sa počíta len u blízkych oblakov, je z vizuálneho hľadiska prehliadnuteľný, čo je pozitívom, pretože za predpokladu reálnych výpočtov fyziky ich možno obmedziť na niekoľko oblakov naraz.

Toto otvára ďalší priestor na vývoj klient-server modelu, kde by klient počítal fyziku blízkych oblakov real-time a server by počítal vzdialené oblaky offline.

Takisto ďalším veľkým pozitívom je fakt, že všetky transformácie a dynamika sú počítané na grafickej karte. V dnešnej dobe, keď sa výpočty fyziky presúvajú na grafickú kartu (CUDA, PhysX), je prepojenie tejto vizualizácie s fyzikálnymi výpočtami priamočiare.

5. Zoznam použitého softwaru

Microsoft .NET Framework 3.5 SP1

Microsoft Visual C# Express 2008

Tao Framework

Altova XmlSpy 2009 trial

Sparx Systems Enterprise Architect trial

Fraps

BBflashback trial

6. Použitá literatúra

Všetky internetové zdroje boli v apríli 2009 dostupné a obsahovali popísaný obsah.

[01] Wang Niniane, “Realistic and fast cloud rendering in computer games”, ACM SIGGRAPH, 2003

[02] Harris M., LASTRA A., “Real-time Cloud Rendering”, Computer Graphics Forum, Blackwell Publisher, vôl. 20, 2001

[03] Schaufler G., “Dynamically generated imposters”, Modeling Virtual Worlds - Distributed

Graphics, MVD Workshop, 1995

[04] Wenzel Carsten, “Real-time atmospheric effects in games”, ACM SIGGRAPH, 2006

[05] Akenine-Möller T., Haines E., “Real-time Rendering, 2nd Edition”, A K Peters, 2002

[06] C# versus C++ versus Java performance comparison, <http://reverseblade.blogspot.com/2009/02/c-versus-c-versus-java-performance.html>

[07] Performance comparison C++, C# and Java, <http://www.tommti-systems.de/go.html?http://www.tommti-systems.de/main-Dateien/reviews/languages/benchmarks.html>

[08] XNA, <http://creators.xna.com/en-US/>

[09] Ogre, <http://www.ogre3d.org/>

[10] .NET Framework, <http://www.microsoft.com/downloads/details.aspx?FamilyId=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=en>

[11] Visual C# Express 2008, <http://www.microsoft.com/express/vcsharp/>

[12] Tao Framework, <http://www.taoframework.com/>

[13] Vertex Buffer Object, <http://en.wikipedia.org/wiki/Vertex_Buffer_Object>

[14] Frame Buffer Object, <http://en.wikipedia.org/wiki/Framebuffer_Object>

[15] Shader, <http://en.wikipedia.org/wiki/Shader>