VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÝVOJ APLIKACÍ PRO XBOX 360

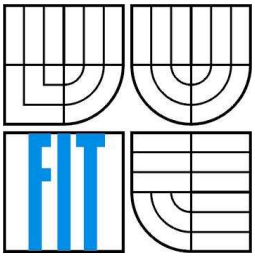
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RUDOLF KAJAN

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÝVOJ APLIKACÍ PRO XBOX 360

APPLICATION DEVELOPMENT FOR XBOX 360

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RUDOLF KAJAN

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ADAM HEROUT, Ph.D.

BRNO 2009

VÝVOJ APLIKACÍ PRO XBOX 360

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Adama Herouta, Ph.D. a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Rudolf Kajan
25. května 2009

Poděkování

Za podměty, trpělivé konzultace a odborné rady děkuji Ing. Adamovi Heroutovi, Ph.D. Také bych chtěl poděkovat Dr. Ing. Daliborovi Kačmářovi z firmy Microsoft za poskytnutí členství v Creator's Club.

Abstrakt

Diplomová práce se zabývá vývojem her na platformě Xbox 360 a vytvořením startovacího balíku pro tuto platformu. Po počátečním představení Xboxu 360 jako moderní a velmi výkonné herní konzoly představuje technologii XNA, umožňující poprvé v historii nejen profesionálním tvůrcům, ale i amatérským nadšencům vytvářet hry pro herní konzoli. Kromě existujících startovacích balíků pro 3D XNA hry je pozornost věnována i přehledu existujících nástrojů na analýzu výkonnosti implementované aplikace a nástrojem na vytváření herního obsahu. Hlavní část práce je věnována návrhu, implementaci a testování startovacího balíku zaměřeného na strategickou hru, přičemž důraz je kladen na efektivnost, srozumitelnost a použitelnost členy XNA komunity.

Klíčová slova

Xbox 360, XNA, Game Studio, .NET, startovací balík, strategická hra, RTS

Abstract

This thesis deals with game development on the Xbox 360 platform and implementation of a starter kit for this platform. After introduction of the Xbox 360 as a modern and very powerful gaming console, the XNA technology that makes development of games for console possible for the first time in history not only for professionals but also for amateurs, is introduced. The next part mentions existing starter kits for 3D XNA games, existing tools widely used to analyze implemented solution with focus on performance and tools commonly used for game content creation. The main part of thesis is dedicated to design, implementation and testing of a real time strategy starter kit prototype, with emphasis on efficiency, understandability and usability by XNA community members in mind.

Keywords

Xbox 360, XNA, Game Studio, .NET, starter kit, real time strategy game, RTS

Citace

Kajan Rudolf: Vývoj aplikací pro XBOX 360, diplomová práce, Brno, FIT VUT v Brně, 2009

Obsah

1 Úvod ... 2
1.1 Členenie práce ... 3
2 Prehľad problematiky ... 4
2.1 Herná konzola Xbox 360 ... 4
2.2 XNA a vývoj hier ... 8
2.3 XNA, .NET Compact Framework a výkonnosť ... 11
2.4 XNA a štartovacie balíky ... 12
3 Prehľad nástrojov použiteľných pri riešení projektu ... 16
3.1 Nástroje na analýzu výkonnosti riešenia ... 16
3.2 Nástroje na tvorbu 3D obsahu do hier ... 19
3.3 Konverzia 3D objektov medzi formátmi ... 23
3.4 Nástroje na tvorbu a úpravu textúr ... 24
4 Cieľ práce ... 25
5 Návrh systému ... 29
5.1 Funkcionálne požiadavky ... 29
5.2 Nefunkcionálne požiadavky ... 30
5.3 Opis systému ... 31
5.4 Hlavné komponenty prototypu ... 31
5.5 Editor prostredia ... 38
6 Použitie Content Pipeline ... 42
6.1 Vytvorenie jednoduchého objektu prostredia ... 42
6.2 Vytvorenie ovládateľného objektu ... 44
7 Testovanie výkonnosti ... 48
7.1 Podmienky testovania ... 48
7.2 Výsledky ... 49
7.3 Zhodnotenie ... 55
8 Záver ... 56
Zoznam príloh ... 61
Príloha 1. Obsah elektronického média ... 62
Príloha 2. Shader použitý pri zobrazovaní terénu ... 63
Príloha 3. Popis objektov prostredia vo formáte XML ... 67

1 Úvod

Nie je pochýb, že počítačové hry sa tešia celosvetovo obrovskej popularite. Avšak niekedy sa zabúda, že predtým, než sa z vytvárania hier stal obrovský obchod s mnohomiliardovým ročným obratom, bolo vytváranie hier hlavne zábavou nadaných nadšencov. Pocit z kvalitnej, odladenej hry ktorá bola zábavou na dlhé mesiace pre mnohých ľudí ďaleko prevyšoval (zväčša minimálne) zisky. Postupne s rozvojom technológií a zvlášť s príchodom grafických akcelerátorov ktoré výrazne zlepšili vizuálnu kvalitu hier sa o vývoj hier začalo zaujímať čoraz väčšie množstvo firiem, ktoré sa začali predháňať v technologickom spracovaní hier. Došlo k obrovskému nárastu komplexnosti vizuálnej stránky hier, no to podstatné – jedinečný zážitok podporený silným príbehom začal zapadať do úzadia. Tvorcovia prichádzajúci s novými nápadmi mali a stále majú problém získať finančné zdroje na svoje projekty, pretože vytvorenie kvalitnej hry je niekoľkoročný proces vyžadujúci veľké množstvo investovaných prostriedkov, pričom riziko že výsledná hra nebude úspešná a rýchlo „zapadne“ je značné. V dôsledku toho sa v posledných rokoch počet herných žánrov znížil – ostali väčšinou len časovo menej náročné žánre priťahujúce občasných hráčov (angl. casual gamers) a vytváranie hier určených pre užšiu skupinu, v minulosti tak typické, v podstate zaniklo. Amatérski tvorcovia majú vývoj hier sťažený aj tým, že hráči očakávajú veľmi vysokú vizuálnu a zvukovú stránku hry, pričom pre amatérskeho tvorcu alebo malý tým je problematické dostať sa k nástrojom a technológiám umožňujúcim vytvorenie takto kvalitného herného obsahu.

Práve amatérskym tvorcom hier a malým tímom sa rozhodol Microsoft pomôcť a uľahčiť im cestu pri vytváraní vlastnej hry vydaním XNA. XNA je voľne dostupný súbor nástrojov – herný framework, integrované vývojárske prostredie a nástroje na prácu s herným obsahom, založených na .NET Frameworku a Compact Frameworku, ktorý zapuzdruje nízkoúrovňové rozdiely medzi cieľovými platformami. S XNA je možné rýchlo a efektívne vytvárať 2D i 3D hry určené pre rôzne populárne platformy a to bez nutnosti zdĺhavého portovania. V súčasnosti XNA podporuje vytváranie hier pre Windows (2D a 3D hry), Xbox 360 (2D a 3D hry) a multimediálny prehrávač Zune (2D hry). Na žiadnej z platforiem nie je nijakým spôsobom obmedzený prístup k hardvéru, takže aj pri vytváraní hry pre Xbox 360 majú vývojári plný prístup k rovnakému výpočtovo silnému hardvéru ako profesionálne tímy, vďaka čomu majú po prvý krát v histórii amatérski vývojári prístup k vytváraniu hier pre takto silnú hernú konzolu.

Frank Savage z tímu stojaceho za XNA nedávno na Professional Developers Conference 2008 oznámil, že: „V súčasnosti sa XNA vyučuje na viac ako tisíc univerzitách v kurzoch zameraných na vývoj hier.” Jedná sa nepochybne o úspech, obzvlášť keď si uvedomíme, že prvá verzia XNA bola sprístupnená relatívne nedávno - 30. augusta 2006. Pre vývojárov začínajúcich s XNA je veľmi dôležitá kvalitná dokumentácia a štartovacie balíky dodávané s XNA venované niektorému hernému

žánru. Práve tieto štartovacie balíky pomáhajú pochopiť základné mechanizmy XNA a postupne budovať hru z určitého základu namiesto zdĺhavého a ťažkopádneho začínania „na zelenej lúke“, resp. z „čistého projektu.“ Problémom však je, že tieto užitočné štartovacie balíky pre 3D hry aj po viac ako dva a pol roku existujú iba tri. A práve vytvorenie ďalšieho, venovaného strategickej hre je cieľom tejto práce.

1.1 Členenie práce

Práca je rozdelená do ôsmich kapitol. Prvá kapitola, ktorú práve čítate, slúži k uvedeniu do širšieho kontextu problematiky a predstavuje jednotlivé kapitoly práce.

Druhá kapitola predstavuje Xbox 360 ako modernú a veľmi výkonnú hernú konzolu. Ďalej predstavuje technológiu XNA umožňujúcu po prvý krát v histórii nielen profesionálnym tvorcom hier, ale aj amatérskym nadšencom vytvárať hry pre (túto) hernú konzolu, popisuje niektoré problémy s výkonnosťou, ktoré pramenia zo špecifík tejto platformy a v závere sa venuje existujúcim štartovacím balíkom pre 3D XNA hry.

Kapitola tri je venovaná prehľadu existujúcich nástrojov analyzujúcich výkonnosť implementovanej aplikácie a nástrojov na vytváranie herného obsahu, pričom kladie dôraz na predstavenie voľne šírených verzií komerčných nástrojov, ktoré sú dôležité pre amatérskych vývojárov.

Štvrtá kapitola obsahuje formuláciu hlavných cieľov práce, venuje sa negatívnym prvkom obsiahnutým v existujúcich štartovacích balíkoch a predstavuje možnosti ako tieto negatíva odstrániť.

Piata kapitola je venovaná návrhu a implementácii štartovacieho balíka zameraného na strategickú hru. Funkcionálne a nefunkcionálne požiadavky kladené na navrhovaný štartovací balík sú sformulované a hlavné komponenty navrhovaného balíka sú predstavené.

Kapitola šesť popisuje vytvorenie a použitie dvoch základných typov objektov – jednoduchého objektu slúžiaceho ako objekt prostredia a dynamického ovládateľného objektu. Jej cieľom je demonštrovať jednoduchosť a priamočiarosť procesu modifikácie a rozširovania štartovacieho balíka, čo je veľkou výhodou implementovaného štartovacieho balíka.

Siedma kapitola popisuje spôsob merania výkonnosti navrhnutej a implementovanej aplikácie. Venuje sa podrobnému popisu konfigurácií na ktorých bola aplikácia testovaná a popisu scény, ako aj samotným testom. Záver kapitoly je venovaný celkovému zhodnoteniu nameraných údajov.

Záverečná ôsma kapitola je zhrnutím doterajších výsledkov a venuje sa pokračovaniu práce.

2 Prehľad problematiky

Táto kapitola predstavuje hernú konzolu Xbox 360 a poskytuje základný prehľad o jej technických parametroch. Zaoberá sa aj technológiou XNA ktorá predstavuje moderný prístup k vývoju hier nielen na amatérskej úrovni, objasňuje vzťah medzi XNA a Xboxom 360, venuje sa hlavným zdrojom straty výkonnosti XNA aplikácií na platforme Xbox 360 oproti aplikáciám založeným na XNA určenými pre platformu PC a potrebe vývoja štartovacích balíkov. Záver kapitoly poskytuje prehľad existujúcich štartovacích balíkov a stručne rozoberá ich prínosy a nedostatky.

2.1 Herná konzola Xbox 360

Xbox 360 je moderná herná konzola siedmej generácie na ktorej vývoji spolupracovali firmy Microsoft, IBM, ATI a SiS. Xbox 360 bol po prvý krát oficiálne odhalený 12. Mája 2005 na E3 (Electronic Entertainment Expo). Jedná sa o priameho nástupcu hernej konzoly Xbox – prvej hernej konzoly od Microsoftu, s ktorou je do značnej miery spätne kompatibilná – pri prvom pokuse o spustenie hry určenej pre pôvodný Xbox sa automaticky stiahne emulátor ktorý v súčasnosti podporuje spustenie viac ako 300 Xbox hier. Xbox 360 bol navrhnutý ako moderné multimediálne centrum poskytujúce zábavu všetkým členom domácnosti.

I po troch rokoch ktoré uplynuli od uvedenia na trh sa Xbox 360 teší značnej popularite. 14. Mája 2008 Microsoft oznámil že v Spojených štátoch bolo predaných 10 000 000 kusov tejto hernej konzoly, čím sa Xbox 360 stal prvou konzolou siedmej generácie ktorej sa podarilo prekonať túto hranicu predajnosti. Celosvetovo bolo predaných viac ako 19 000 000 kusov [1].

Obr. 1: Konzola Xbox 360 vo verzii „Pro“ s bezdrôtovým ovládačom

Na začiatku vývoja Xboxu 360 stál Microsoft pred rozhodnutím, či pre túto novú hernú konzolu použiť jediný procesor s čo najvyššou frekvenciou, alebo radšej na základe existujúcich technológií vytvoriť nový viacjadrový procesor. Napokon sa rozhodol pre druhú možnosť. Procesor vyrobila spoločnosť IBM a jedná sa o trojjadrový SMT (simultaneous multithreading) procesor. Frekvencia každého jadra je 3.2 GHz. Grafickú kartu určenú špeciálne pre túto konzolu navrhla a vyrobila spoločnosť ATI. Jej kódové označenie je Xenos a vyvíjaná bola pod označením C1. V mnohých smeroch sa jedná o predchodcu Radeon R600 série určenej pre PC. Frekvencia grafickej karty je 500 MHz a karta obsahuje 512 MB RAM a 10 MB vstavanej DRAM (eDRAM, embedded DRAM). Základná doska ktorá pôvodne súčasťou Xboxu 360 bola od júla 2007 nahradená novšou základnou doskou s čipsetom s kódovým označením Zephyr. Táto novšia doska priniesla HDMI výstup a nový, účinnejší chladič pre GPU. Neskôr bola vymenená za základnú dosku s čipsetom Falcon. Prítomnosť základnej dosky s Falcon čipsetom je možne určiť podľa redizajnovaného napájania a prítomnosti 175 wattového zdroja. Taktiež rozloženie základnej dosky bolo pozmenené. Tab. 1 poskytuje prehľad základných technických údajov Xboxu 360.

Xbox 360 – Technické údaje	
CPU	▪ IBM PowerPC procesor ▪ 3 symetrické jadrá na frekvencii 3.2 GHz ▪ 2 hardvérové vlákna na jadro ▪ 1 VMX-128 vektorová jednotka na jadro ▪ 128 VMX-128 registrov na hardvérové vlákno ▪ 1 MB L2 vyrovnávacej pamäte
GPU	▪ Frekvencia 500 MHz ▪ 10 MB vstavanej DRAM ▪ 48 paralelných dynamicky nastavovaných shader pipeline schopných pracovať s desatinnou čiarkou ▪ Unifikovaná architektúra shaderov
Pamäť	▪ 512 MB GDDR3 ▪ 700 MHz DDR ▪ Unifikovaná pamäťová architektúra
Vstupy a výstupy	▪ Maximálne 4 súčasne pripojené ovládače ▪ 3x USB 2.0 ▪ 2 sloty na pamäťové karty (kapacita 64 - 512 MB)
Pripojenie	▪ Ethernet port ▪ Wi-Fi: 802.11a, 802.11b a 802.11g

Optická mechanika	▪ 12x dual-layer DVD-ROM
Zvuk	▪ viackanálový priestorový zvukový výstup ▪ 48KHz 16-bit ▪ 320 nezávislých dekomprimačných kanálov ▪ 32bitové spracovanie zvuku ▪ 256 audio kanálov

Tab. 1: Xbox 360 – Technické údaje. Zdroj údajov [2].

V súčasnosti je herná konzola Xbox 360 dostupná v troch verziách líšiacich sa príslušenstvom a cenou. Najzákladnejšia a najlacnejšia verzia nazývaná Arcade obsahuje okrem samotnej hernej konzoly bezdrôtový ovládač v bielom prevedení, pamäťovú kartu s kapacitou 256 MB a AV kábel. Cena tejto verzie je 199.99 amerických dolárov. Arcade verzia od 22. októbra 2007 nahrádza takzvanú Core verziu, ktorá neobsahovala žiadnu pamäťovú kartu ani pevný disk a dodávala sa s káblovým ovládačom [3]. Pro verzia (pôvodne nazývaná Premium) sa od Arcade verzie líši prítomnosťou hybridného a komponentového kábla namiesto kompozitného, ako aj prítomnosťou odnímateľného pevného disku na ktorý je možné ukladať obsah stiahnutý pomocou služby LIVE, pozície z hier a profil hráča. Prítomnosť pevného disku je vyžadovaná aj v prípade hrania hier určených pre pôvodný Xbox. Kapacita pevného disku dodávaného s Pro verziou bola pôvodne 20 GB, no od 1. augusta 2008 je s touto verziou dodávaný pevný disk s kapacitou 60 GB. Cena Pro verzie je 299.99 amerických dolárov. Najnákladnejšou verziou je takzvaná Elite verzia, ktorej cena je 399.99 amerických dolárov. Obsahuje všetko to čo Pro verzia, ale pevný disk má kapacitu 120 GB a obsahuje naviac HDMI kábel a slúchadlá s mikrofónom. Naviac konzola, ovládač, slúchadlá i mikrofón majú čierny povrch, čím sa líšia oproti ostatným verziám, ktoré sú svetlo sivé.

Xbox 360 používa špeciálny operačný systém určený pre túto konzolu ktorý bol navrhnutý od úplného základu, nejedná sa teda o modifikáciu niektorej z verzií operačného systému Windows. Tento operačný systém sa nachádza na systéme súborov ktorý má veľkosť 16 megabytov a má prístup maximálne k 32 megabytom operačnej pamäte [4]. Operačný systém je aktualizovateľný, pričom medzi jednotlivými aktualizáciami je priemerný časový odstup štyri mesiace.

Operačný systém Xboxu 360 disponuje grafickým používateľským rozhraním, ktoré sa nazýva Dashboard (palubná doska). Od svojho uvedenia na trh v roku 2005 až do novembra 2008 pozostával Dashboard zo štyroch záložiek poskytujúcich prístup k jednotlivým službám operačného systému - služba Xbox Live poskytujúca prístup k online obsahu, hranie hier, prehrávanie videa a hudby a nastavenia. Od 19. novembra 2008 je však dostupná aktualizácia operačného systému, ktorá bola nazvaná New Xbox Experience (NXE). NXE prináša mnoho zmien a vylepšení, no pravdepodobnejšie najmarkantnejšou je úplná zmena grafického používateľského rozhrania. Nové používateľské rozhranie je založené na rozhraní Twist UI, ktoré už bolo použité vo Windows Media Center a v multimediálnom prehrávači Zune. Hlavnou motiváciou pre zmenu používateľského

rozhrania bola snaha o uľahčenie a zrýchlenie navigácie. Obr. 2 zachytáva grafické používateľské rozhranie s nainštalovanou NXE aktualizáciou.

Obr. 2: GUI Xboxu 360 s aktualizáciou NXE. Prevzaté z [5].

Herný zážitok takmer z akejkoľvek hry však nie je dokonalý pokiaľ hráčovi nie je umožnené súperiť s ľudským protihráčom, ideálne ľubovoľne geograficky vzdialeným. Myšlienka umožniť pripojenie hernej konzoly na internet a priniesť tak hráčom oveľa silnejší zážitok z hry nie je nová. Po prvý krát ju realizovali tvorcovia hernej konzoly Dreamcast v roku 1999. Táto konzola obsahovala spočiatku zabudovaný modem, ktorý bol neskôr nahradený adaptérom umožňujúcim využiť širokopásmové pripojenie k internetu. Avšak veľmi nízka rýchlosť spojenia v kombinácii s nedostatočným rozšírením širokopásmového internetu spôsobili, že myšlienka online hrania prostredníctvom hernej konzoly upadla na čas do zabudnutia. Konkurenti Dreamcastu herné konzoly Playstation 2 a Nintendo GameCube neobsahovali možnosť online hrania. Túto myšlienku opäť vzkriesila až firma Microsoft keď správne identifikovala dva kľúčové atribúty vedúce k úspechu takejto služby – dostatočne dostupné širokopásmové pripojenie k internetu a prítomnosť dostatočne veľkého pevného disku v konzole slúžiaceho na uchovanie stiahnutého obsahu. Táto služba umožňujúca online hru viacerých hráčov proti sebe, sťahovanie online obsahu a streamovanie zvuku a videa sa nazýva Xbox Live a po prvý krát bola sprístupnená 15.11.2002 majiteľom konzoly Xbox. Postupom času pribúdali nové možnosti ako napríklad hlasová komunikácia v reálnom čase (voice chat), či možnosť pozorovať hru iných hráčov bez možnosti zasahovania do hry. Popularita služby Xbox Live neustále rastie a v novembri 2007 presiahol počet používateľov služby hranicu 8 miliónov. Služba Xbox Live je dostupná aj pre všetkých majiteľov Xboxu 360. Jej súčasťou je okrem iného aj online obchod s hrami s názvom Xbox Live Marketplace, komunikátor Windows Live Messenger a systém na vyhľadávanie súperov s podobnou úrovňou zručnosti v danej hre využívajúci

matematický model neurčitosti odhadujúci dopredu výsledok zápasu nazývaný TrueSkill. Aktualizácia NXE priniesla vylepšenie aj tejto služby, umožnila sledovanie filmov a televíznych relácií prostredníctvom služby NetFlix, hoci momentálne je táto služba dostupná len v Severnej Amerike.

2.2 XNA a vývoj hier

XNA, používajúc rekurzívnu skratku XNA's Not Acronymed, je súhrnné označenie pre niekoľko nástrojov spoločnosti Microsoft súvisiacich s vývojom hier a to nielen na amatérskej úrovni. XNA zahŕňa :

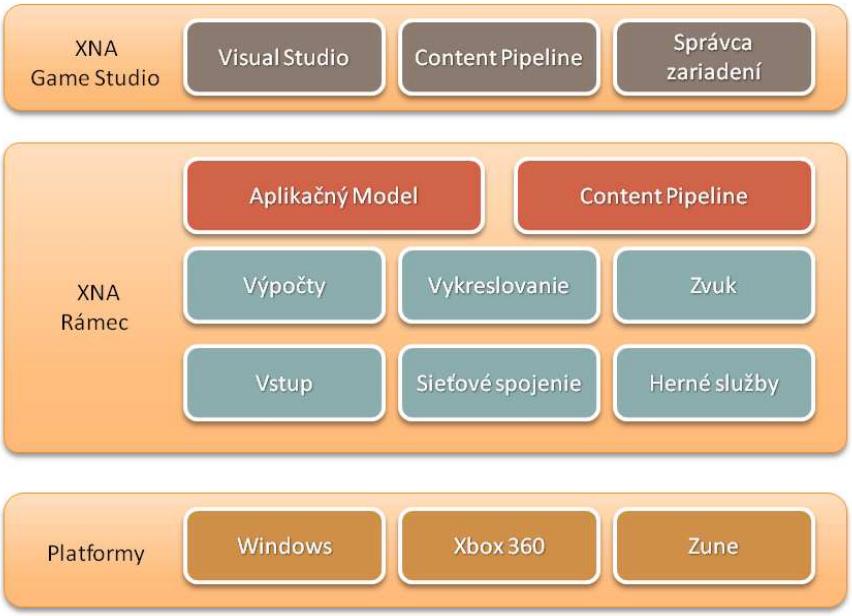
- XNA Framework – rozsiahla množina knižníc v jazyku C# špeciálne navrhnutá pre vývojárov hier ktorej cieľom je zjednodušiť vývoj hier a zapuzdriť nízkoúrovňové rozdiely medzi cieľovými platformami,
- XNA Build – nástroje na optimalizovanie, ladenie, údržbu a odhaľovanie vzťahov medzi herným obsahom, umožňujúc previesť herný obsah do vhodnej formy závislej na použitom hernom stroji,
- XNA Game Studio – modifikácia Visual Studia (v podstate sa jedná o zásuvný modul pre Visual Studio), je integrované vývojárske prostredie určené na vývoj hier založených na XNA Frameworku. Využíva XNA Framework aj XNA Build ktoré sú s ním aj distribuované a preto pojmy XNA a XNA Game Studio sú vzájomne zameniteľné.

Ako Albert Ho, člen tímu stojaceho za XNA uvádza na svojom blogu [6] : „S XNA sa snažíme riešiť problém znižovania nákladov potrebných na vývoj hry a času potrebného na uvedenie hry na trh. Pri riešení tohto problému sa sústreďujeme na dva ciele – súčasný paralelný vývoj hier pre platformu Windows a Xbox 360 a zvýšenie produktivity vývojárov, pretože chceme aby sa vývojári plne sústredili na hrateľnosť a nie na prekonávanie technických rozdielov medzi cieľovými platformami.“ Práve tieto spomínané princípy – zapuzdrenie nízkoúrovňových technických rozdielov cieľových platforiem a možnosť spustenia aplikácie na niekoľkých populárnych platformách bez zdĺhavého a náročného portovania a nutnosti nákupu nákladných vývojárskych balíkov prispeli k značnej popularite XNA – v súčasnosti má XNA komunita viac ako 65 000 členov.

XNA Game Studio Express 1.0 bolo po prvý krát sprístupnené verejnosti 30. augusta 2006, keď Microsoft uvoľnil prvú beta verziu, ktorá bola približne po dvoch mesiacoch nahradená druhou beta verziou a napokon 11. decembra 2006 prvou finálnou verziou. Ďalšie vylepšenia prišli 24. apríla 2007 s verziou nazvanou XNA Game Studio Express 1.0 Refresh, pričom medzi hlavné vylepšenia patrila plná a oficiálna podpora Windows Vista, inkrementálne odosielanie všetkých projektov na Xbox 360, podpora 3D zvuku, 3D textúr a per-pixel osvetlenia. Slovo Express sa v názve objavilo z dôvodu, že Microsoft pôvodne plánoval dve verzie Game Studia – Express verziu založenú na

voľne dostupne verzii Visual Studia Express ktorá mala byť dostupná zdarma a bola určená pre amatérskych vývojárov a profesionálnu komerčnú verziu založenú na Visual Studiu 2005 (VS 2005). Toto delenie však bolo časom zrušené a XNA Game Studio je dostupné zdarma a použiteľné so všetkými verziami Visual Studia. Veľkým míľnikom v histórii XNA bolo vydanie verzie 2.0 13. decembra 2007. Táto verzia priniesla oficiálnu podporu všetkých verzií VS 2005, teda nielen Express verzie, čím bolo vývojárom umožnené pri vývoji hier založených na XNA používať všetky zásuvné moduly určené pre VS 2005, pokročilé ladenie (ladenie v zmiešanom režime, sledovanie vlákien), jednotkové testovanie a verziovanie. Špeciálny integrovaný projekt zastrešuje prístup k všetkému hernému obsahu a vlastným importérom procesorom, zostavenie herného obsahu prebieha pred samotným zostavením kódu čim sa zjednodušilo ladenie, pribudla podpora sieťovej hry a prístup k Xbox Sprievodcovi – grafickému rozhraniu používanom pri interakcii s hráčom. Mnoho zmien nastalo aj v samotnom API, okrem iného bol úplne prerobený systém cieľov renderovania. Momentálne aktuálna verzia XNA Game Studio 3.0 bola zverejnená 30. októbra 2008, pričom jej predchádzala jedna beta verzia. Hlavnými zmenami sú podpora Visual Studia 2008, C# 3.0, LINQ, pribudol nový skúšobný herný režim (angl. trial mode) a podpora ďalšej hernej platformy - 2D hry vytvorené pomocou XNA Game Studio 3.0 je možné spustiť na multimediálnom prehrávači Zune.

Ako ukazuje Obr. 3, XNA Framework pozostáva z dvoch vysokoúrovňových častí - z aplikačného modelu a content pipeline.



Obr. 3: XNA Framework a XNA Game Studio.

Aplikačný model je zodpovedný za vytvorenie nezávislosti od cieľovej platformy - napríklad vytvorenie plochy na ktorú je možné vykresľovať, zasielanie správ medzi oknami, prehrávanie

zvukov a hudby a umožňuje vytvárať znovupoužiteľné komponenty, ktoré je možné zdieľať medzi aplikáciami. Každý XNA projekt využíva triedu `Game`, ktorá obsahuje všetky dôležité herné komponenty, nastavenie okna a manažéra herného obsahu. Jej základné metódy dávajú už svojím názvom tušiť základnú filozofiu toku riadenia v XNA:

- `Initialize ()` – slúži k úvodnej inicializácii objektov
- `LoadGraphicsContent ()` – slúži na načítanie herného obsahu a jeho spracovanie
- `Update (GameTime time)` – táto metóda je volaná pred vykreslením každého snímku na obrazovku. Slúži na získanie vstupu od používateľa, vykonanie výpočtov súvisiacich s hernou logikou, prehrávanie zvukov a pod.
- `Draw (GameTime time)` – slúži na vykreslenie scény
- `UnloadGraphicsContent ()` – pri ukončovaní hry sa stará o odstránenie herného obsahu z pamäte.

Novým konceptom s ktorým XNA prichádza je systém znovupoužiteľných častí kódu, ktoré sa jednoducho vložia do aplikácie. Tieto triedy sa nazývajú herné komponenty a ich základom je trieda `GameComponent`. Tieto komponenty obsahujú metódy `Update` a `Draw`, ktoré sú automaticky volané. Herné komponenty sú všestranne využiteľné – môžu byť použité na vytvorenie menších častí použiteľných v podstate v každej hre, napríklad počítadlo zobrazených snímkov za sekundu, ale i špecifickejších častí, ako je terén.

Content pipeline zodpovedá za zostavenie herného obsahu. Pred samotným použitím v hernom stroji je totiž nutné herný obsah exportovať z aplikácie v ktorej bol vytvorený vo formáte zrozumiteľnom pre herný stroj a následne tento obsah importovať. V prípade problémov s kompatibilitou je nutné vytvoriť pre tento herný obsah konvertor. Content pipeline má snahu tento proces štandardizovať a uľahčiť pomocou dodávaných importérov a procesorov obsahu, pričom samozrejme ponúka možnosť vytvorenia vlastných procesorov a importérov. Dodávané sú importéry pre 3D modely vo formáte DirectX (.x) a FilmBox (.fbx), zvuky vo formáte .wav, .mp3 a .wma, ako aj pre všetky štandardne používané 2D formáty. Vďaka prítomnosti content pipeline je množstvo vlastného kódu potrebného na prístup k hernému obsahu minimálne.

Ako už bolo spomínané, Game Studio je založené na Visual Studiu, z ktorého sú dostupné rozšírenia content pipeline pomocou XNA Build nástrojov, ako aj správca zariadení, ktorý je zodpovedný za pripojenie a komunikáciu s Xboxmi 360 a prehrávačmi Zune. Po identifikácii zariadení je možné skompilovaný kód aplikácie spolu so zostaveným herným obsahom pomocou tohto nastroja preniesť na cieľové zariadenie a následne spustiť a ladiť.

Vývoj aplikácií pre platformy Windows a Zune je bez akýchkoľvek poplatkov, avšak pre vývoj na platforme Xbox 360 je potrebné platené členstvo v „Creator's Club“, ktoré sa dá zakúpiť prostredníctvom služby Xbox Live Marketplace. V súčasnosti existujú dva druhy členstva – štvormesačné, ktorého cena je 49 amerických dolárov a ročné s cenou 99 amerických dolárov. Po

zaplatení poplatku je možné prostredníctvom služby Xbox Live stiahnuť na pevný disk Xboxu 360 .NET Compact Framework a spúšťač XNA aplikácií. Následne je možné pripojiť sa z XNA Game Studia k tomuto Xboxu 360 a po overení platnosti licencie je umožnené odoslať spustiteľnú verziu projektu Xboxu 360.

2.3 XNA, .NET Compact Framework a výkonnosť

Pri vývoji aplikácie pre platformu Xbox 360 je však potrebné uvedomiť si niektoré rozdiely oproti platforme Windows, ktoré majú zásadný vplyv na výkonnosť aplikácie a venovať im náležitú pozornosť. Pretože hoci výsledný neoptimalizovaný kód bude spustiteľný, jeho výkonnosť bude značne obmedzená. Keďže hry sú aplikácie vykonávané v reálnom čase, výsledný dojem veľmi závisí od množstva zobrazených snímkov za sekundu a samotné korektné vykonávanie ešte nie je postačujúce.

XNA Framework na platforme Xbox 360 používa Compact CLR (Common Language Runtime) na vykonávanie kompilovaného IL (Intermediate Language) kódu. Compact CLR je optimalizovaný pre zariadenia ako sú PDA a mobilné telefóny, kde veľkosť je uprednostňovaná pred výkonnosťou. Preto implementácia Compact CLR na Xboxe 360 nevykonáva žiadnu optimalizáciu kódu. Taktiež nevykonáva inlineovanie malých funkcií (napríklad Vector3.CrossProduct, prípadne Vector2.operator+). Dokonca odkladá všetky argumenty na zásobník a nie do registrov, hoci PowerPC disponuje veľkým množstvom registrov.

Naviac PowerPC jadrá v Xboxe 360 sú pomerne nesofistikované z architektonického pohľadu. Žiadna inštrukcia nemôže byť vyradená pokiaľ neboli vyradené všetky inštrukcie ktoré ju predchádzajú. Dôsledok toho je ten, že pokiaľ sa potrebné dáta nenachádzajú vo vyrovnávacej pamäti celý dátovod je pozastavený. Toto je významný rozdiel v porovnaní s modernými CPU s podporou hyper threadingu. Pri nich ak dôjde k pozastaveniu vlákna z dôvodu čakania na pamäť, plánovač spustí iné vlákno.

Dokonca aj odstraňovanie nepoužívaných objektov z pamäte sa na platforme Xbox 360 líši od toho ktoré existuje na PC platforme. Zberač na Xboxe 360 nie je generačný, takže keď dôjde k zberu nepotrebných objektov všetky objekty na halde sú prezreté aby bolo možné určiť či je potrebné ich odstránenie. Toto je obzvlášť spomaľujúci proces ktorý sa negatívne prejaví drastickým znížením počtu zobrazených snímkov za sekundu pokiaľ je objektov veľké množstvo, prípadne ak sú objekty veľké. Pomocou pri problémoch so zberačom sú nástroje, ktoré budú predstavené v kapitole 3.1 nazvanej Nástroje na analýzu výkonnosti riešenia.

Práve tieto faktory znižujú výkonnosť aplikácií a použitím rôznych optimalizácií na strane kódu, v ideálnom prípade skombinovanými s využitím viacerých procesorov, je nutné tieto negatívne vplyvy minimalizovať.

2.4 XNA a štartovacie balíky

Je dobrým zvykom k herným strojom okrem dokumentácie a krátkych kúskov kódu, demonštrujúcich správne použitie technológie s cieľom dosiahnuť istú veľmi špecifickú funkcionalitu, pridávať aj väčšie, komplexnejšie aplikácie. V oblasti herných strojov a herných frameworkov sa tieto komplexné príklady použitia nazývajú štartovacie balíky, anglicky starter kits. Nejedná sa o plné hry, aj keď niektoré štartovacie balíky k nim majú veľmi blízko. Zvyčajne poskytujú základ na ktorom je potrebné vybudovať úplnú hernú logiku a dodať herný obsah, ktorého je v štartovacích balíkoch nevyhnutné minimum.

Poskytovanie týchto štartovacích balíkov je bežné pri komerčných herných strojoch a frameworkoch (hoci niekedy sú dostupné za extra príplatok), pri voľne šíriteľných sa jedná bohužiaľ vo veľkej miere o vzácnosť, nakoľko ich tvorba nie je jednoduchá a ani časovo nenáročná. V tomto prípade ich úlohu často nahrádzajú menšie voľne šíriteľné projekty vytvorené používateľmi daného stroja či frameworku, často však na úkor prehľadnosti a s nedostatočným množstvom komentárov.

Prečo sú ale tieto štartovacie balíky také dôležité a aké je ich výhoda oproti malým príkladom ? V prvom rade demonštrujú veľkú škálu možností danej technológie a slúžia ako veľmi dobrý nástroj na overenie jej možností v praxi. Prítomnosť a kvalita štartovacích balíkov vysiela prvotný signál o stave danej technológie, keďže len veľmi ťažko môže mať „nedospelá“ technológia kvalitnú „živú“ prezentáciu v tejto forme. Takto sa zo štartovacích balíkov stáva jeden z nástrojov na porovnanie možností technológií. Ďalšou nemenej dôležitou funkciou je zoznamovanie nového používateľa so štruktúrou a spôsobom akým daný herný stroj prípadne framework poskytuje svoju funkcionalitu – na rozdiel od veľmi úzko zameraného príkladu ukazuje veci v oveľa širších súvislostiach a pomáha tak používateľovi preniknúť hlbšie do problematiky. Štartovacie balíky sú veľmi vhodné aj k rapídnemu prototypovaniu, keďže do štartovacieho balíka daného žánru je potrebné doplniť vo veľkej miere iba hernú logiku a obmedzené množstvo herného obsahu a tento základ veľmi dobre poslúži na testovanie nových nápadov v praxi ako aj na ich ladenie.

Základné delenie štartovacích balíkov je podľa počtu použitých dimenzií v hre – čiže 2D, 2.5D (izometrické) a 3D a ďalej podľa herných žánrov. Z 2D a 2.5D hier medzi najčastejšie spracovávané žánre patria RPG hry (role playing games, hry na hrdinov) a 2D akčné hry (side scrollers, hry na pohybujúcom sa pozadí), z 3D hier sú to akčné hry hrané z pohľadu prvej osoby (FPS – First Person Shooters) a 3D RPG hry.

Nasledujúce tri podkapitoly predstavujú jednotlivé 3D štartovacie balíky ktoré sú v súčasnosti dostupné pre XNA.

2.4.1 XNA Racing Game

XNA Racing Game [7] je štartovací balík zameraný na tvorbu závodnej hry. Pre Microsoft ho vytvoril Benjamin Nitschke z nemeckej spoločnosti exDream Entertainment [8]. Benjamin Nitschke je autorom knihy [9], v ktorej na príklade vytvorenia šiestich hier ukazuje možnosti XNA a zásady tvorby efektívneho kódu na platforme Xbox 360. Racing Game je jednou z týchto hier a slúži ako demonštrácia sily a možností XNA. Okrem iného prezentuje použitie shaderov (napríklad shadow mapping, detail mapping), generovanie a renderovanie zložitých objektov a vytvorenie vlastného fyzikálneho enginu. Jedná sa o veľmi užitočný štartovací balík zaoberajúci sa mnohými pokročilými témami.

Obr. 4: XNA Racing Game. Prevzaté z [7].

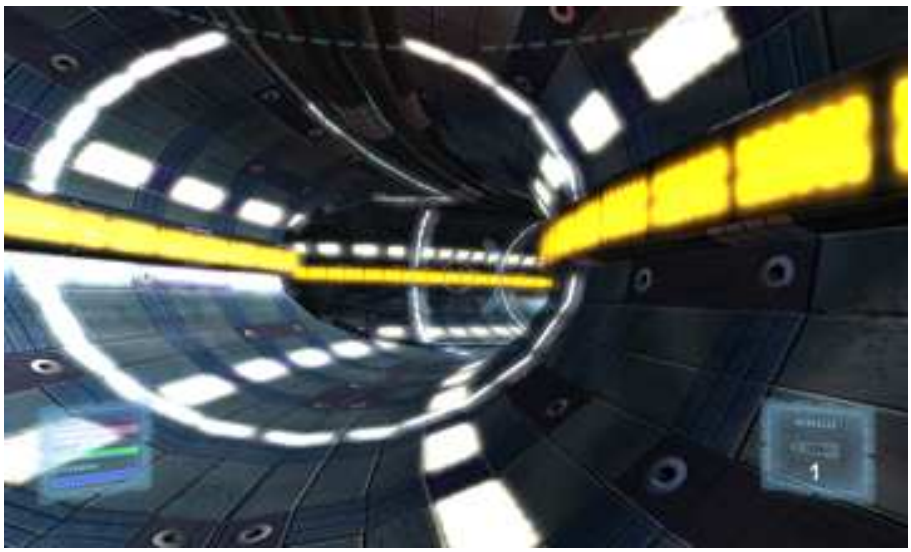
Štartovací balík XNA Racing Game je voľne dostupný ako projekt do Microsoft Visual Studia na [7] pod Ms-PL (Microsoft Permissive License), a to vo verzii pre Windows i Xbox 360. Na spustenie musí grafická karta podporovať vertex a pixel shadre minimálne verzie 2.0.

2.4.2 Ship Game

Ship Game [10] je štartovací balík zameraný na tvorbu trojrozmernej akčnej hry tematicky podobnej hrám Descent [11] a Forsaken [12]. Úlohou hráča je vybrať si jednu z troch ponúkaných lodí

s rôznymi vlastnosťami (rýchlosť, sila štítov, ...) a nej prejsť labyrintom tunelov a zničiť nepriateľské lode.

Štartovací balík je podstatne jednoduchší ako Rancing Game. Zameriava sa na grafickú príťažlivosť prostredia, detekciu kolízií a hru viacerých hráčov. Demonštruje implementáciu normálového mapovania pomocou vlastného procesoru modelov, predstavuje kameru na princípe pružiny s názvom „Chase Camera“ a vlastnú knižnicu na detekciu kolízií s názvom „BoxCollider“. Výhodami tohto štartovacieho balíka sú prehľadnosť, zrozumiteľnosť, dobrá modifikovateľnosť a znovupoužiteľné komponenty, nevýhodou je značná jednoduchosť a štruktúra do ktorej sa pomerne obtiažne vkladá ďalšia herná logika.



Obr. 5: Ship Game. Prevzaté z [10].

Balík XNA Ship Game je dostupný ako projekt do Microsoft Visual Studia iba pre vývojárov s platným členstvom v Creators Club [13] na [10] pod Microsoft Creators Club Premium Content License. Dostupné sú verzie pre Windows i Xbox 360. Na spustenie musí grafická karta podporovať vertex shader minimálne vo verzii 1.1 a pixel shader minimálne verzie 2.0.

2.4.3 Robot Game

Robot Game [14] je Microsoftom označovaný ako minihra, čo znamená že sa jedná o štartovací balík s úplnou hernou logikou a funkcionalitou blížiacou sa plnej hre, no neobsahuje črty typické pre plnohodnotnú hru, napríklad príbehovú líniu, rôznorodé prostredia, viacero režimov hry a podobne. Robot game je trojrozmerná akčná hra, ktorej cieľom je v aréne zničiť všetkých nepriateľov pomocou robota s niekoľkými zbraňami, pričom je potrebné priebežne si doplňať muníciu a vyhýbať sa strelám nepriateľov.

Túto minihru pre Microsoft vyvinula juhokórejská firma Zepetto [15], zaoberajúca sa primárne vývojom hier pre Playstation Portable (PSP). Robot Game demonštruje použitie shaderov, post process efektov, pokročilého systému časticových efektov, trojrozmerného zvuku, manažmentu herných obrazoviek, rozdelenia obrazovky na dve polovice pri hre dvoch hráčov na jednom počítači (split screen) a ukazuje koncept editovateľných herných prvkov – robotov, zbraní a zbierateľných predmetov. Robot Game má veľmi komplexnú vnútornú štruktúru, je postavená na jadre nazvanom „FrameworkCore“. Spolu s minihrou bol uvoľnený manuál popisujúci vzťahy medzi hlavnými objektmi vo FrameworkCore. Hoci Robot Game ukazuje množstvo užitočných techník, prílišná komplexnosť je v tomto prípade trochu na škodu, kód je menej prehľadný (viac ako 40 000 riadkov kódu) oproti Ship Game (takmer 10 000 riadkov) či Racing Game (viac ako 22 000 riadkov) a miestami nedostatočne okomentovaný.

Obr. 6: Robot Game. Prevzaté z [14].

XNA Robot Game je rovnako ako Ship Game dostupný iba pre vývojárov s platným členstvom v Creators Club [13] na [14] pod Microsoft Creators Club Premium Content License. Dostupné sú verzie pre Windows i Xbox 360. Na spustenie musí grafická karta podporovať vertex shader minimálne vo verzii 1.1 a pixel shader minimálne verzie 2.0.

3 Prehľad nástrojov použiteľných pri riešení projektu

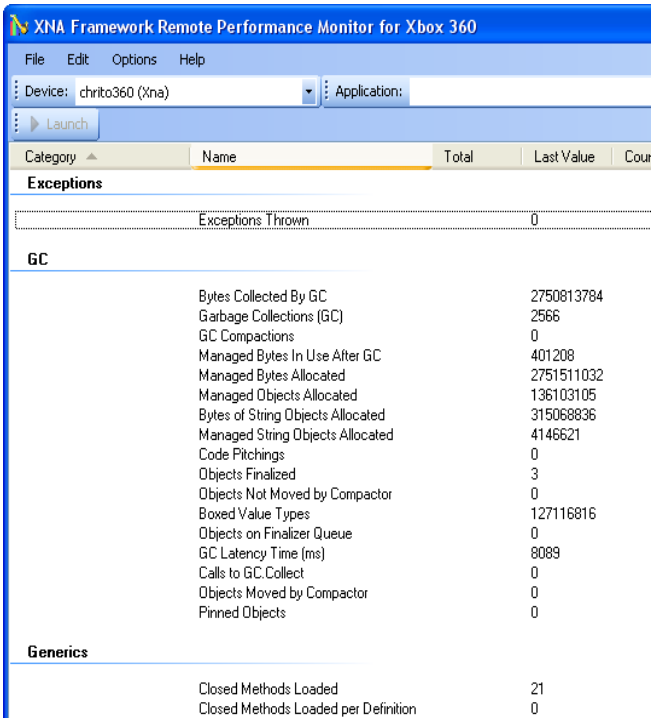
Táto kapitola predstavuje prehľad jednotlivých nástrojov súvisiacich s analýzou výkonnosti riešenia a tvorbou 3D obsahu do hier, kompatibilných s XNA Framework Content Pipeline. Nástroje sú rozdelené do niekoľkých skupín. Prvou popisovanou skupinou sú voľne šírené nástroje použiteľné na analyzovanie výkonnosti riešenia, pomáhajúce identifikovať a lokalizovať zdroje problémov s výkonnosťou aplikácie v reálnom čase. Ďalšou skupinou sú nástroje na tvorbu 3D obsahu do hier. Okrem proprietárnych, finančne značne náročných verzií určených hlavne pre profesionálov z oblasti filmu, televízie a digitálnej zábavy sú predstavené aj odľahčené verzie týchto nástrojov dostupné zdarma, ktoré sú vhodné pre jednotlivcov a malé tímy – cieľovú skupinu na ktorú je XNA zamerané. Veľmi potešujúce je, že spoločnosti si uvedomili potrebu týchto voľne dostupných verzií profesionálnych nástrojov a umožňujú tak širokej komunite amatérskych vývojárov a modifikátorov hier používať nástroje profesionálnych tvorcov hier a grafikov. Následne prehľad pokračuje nástrojmi konvertujúcimi 3D objekty medzi najpopulárnejšími formátmi. Záver kapitoly je venovaný nástrojom na tvorbu a úpravu textúr.

3.1 Nástroje na analýzu výkonnosti riešenia

Nasledujúce podkapitoly 3.1.1 až 3.1.4 bližšie predstavujú jednotlivé nástroje na analýzu výkonnosti riešenia.

3.1.1 XNA Framework Remote Performance Monitor for Xbox 360

Tento nástroj[16] je jediným nástrojom na platforme Xbox 360 slúžiacim na analýzu výkonnosti aplikácie v reálnom čase. Poskytuje používateľské rozhranie zobrazujúce namerané hodnoty, ktoré sú získavané počas vykonávania aplikácie priamo z Common Language Runtime (CLR). Obr. 7 zachytáva výstup XNA Framework Remote Performance Monitor for Xbox 360.



Obr. 7: XNA Framework Remote Performance Monitor for Xbox 360. Prevzaté z [16].

Primárnou úlohou tejto aplikácie je zber údajov týkajúcich sa garbage collectoru – a to najmä frekvencia spúšťania a množstvo času potrebné na zber objektov merané v milisekundách. Keďže práve zber objektov z pamäte je jednou z hlavných príčin zníženia výkonu aplikácií využívajúcich Compact Framework, jedná sa o veľmi užitočný nástroj najmä pri globálnom testovaní aplikácie. Pre hry určené pre Xbox 360 sú obzvlášť zaujímavé hodnoty Garbage Collections (celkový počet zberov) a GC Latency Time (latencia zberača). Čo z týchto údajov môžeme usúdiť ? Ako Shawn Hargreaves, jeden z autorov XNA na svojom blogu [17] vysvetľuje:

Ak je napríklad:

- Počet zberov za poslednú sekundu = 40
- Latencia = 2 ms
- Celkový čas zberača = 40 * 2 = 80 ms

Záver: 80 / 1000 = 8%. Napriek tomu, že hra vytvára v pamäti veľa odpadu, zberač pracuje dostatočne rýchlo a nespôsobuje problémy s výkonnosťou. Problém je treba hľadať na inom mieste.

Ak ale namerané hodnoty ukazujú napríklad :

- Počet zberov za poslednú sekundu = 30
- Latencia = 25 ms
- Celkový čas zberača = 30 * 25 = 750 ms

Záver: 750 / 1000 = 75%. Táto hra je neuveriteľne spomaľovaná práve zberačom a preto je potrebné zmenšiť množstvo nepotrebných objektov v pamäti.

3.1.2 PIX

PIX [18] je predchodcom nástroja XNA Framework Remote Performance Monitor for Xbox 360, a hoci jeho názov pôvodne znamenal „Performance Investigation on Xbox“, je dostupný iba na platforme Windows ako súčasť Microsoft DirectX SDK. Jedná sa o voľne šírený nástroj na ladenie a analýzu aplikácií využívajúcich Direct3D, čo zahŕňa aj XNA aplikácie. PIX dokáže zbierať informácie o volaniach Direct3D API, informácie týkajúce sa časovania a údaje o vrcholoch 3D objektov pred a po transformáciách. Jedno z použití je aj ladenie pixel a vertex shaderov, keďže podporuje vkladanie breakpointov a krokovanie vykonávania do HLSL kódu.

3.1.3 CLRProfiler

CLRProfiler [19] je voľne šírený nástroj na analýzu správania sa aplikácií využívajúcich MSIL – Microsoft Intermediate Language. Primárnym cieľom záujmu je zberač nepotrebných objektov z pamäte a halda s ktorou tento zberač pracuje. Keďže práve réžia zberača je častou príčinou značného zníženia výkonnosti XNA aplikácií, jedná sa o veľmi užitočný nástroj. Pomocou vstavaných pohľadov zberajúcich údaje z aplikácie v reálnom čase dokáže zistiť ktoré metódy alokujú aký typ objektov, ktoré objekty sú (momentálne) aktívne a ktoré referencie zabraňujú ich odstráneniu z pamäte zberačom a množstvo a typy objektov momentálne nachádzajúcich sa na halde. Vstavaný graf volaní dokáže vizualizovať ako sú jednotlivé metódy a objekty volané inými metódami a objektmi a ako často.

3.1.4 NVIDIA Performance HUD

Performance HUD [20] (často skrátene nazývaný PerfHUD) je nástroj od firmy NVIDIA ktorý je zameraním podobný nástroju PIX, no vyžaduje grafickú kartu ktorej výrobcom je NVIDIA. Jedná sa voľne šírený nástroj určený na analýzu aplikácií využívajúcich Direct3D v reálnom čase.

PerfHUD má tri základné režimy v ktorých pracuje:

- Výkonnostný prehľad (Performance Dashboard) umožňuje monitorovanie a ladenie aplikácie ako celku v reálnom čase. Tento režim poskytuje prehľad o počte renderovaných snímkov za sekundu, frekvencii a počte vytvorených 2D textúr, shaderov, bufferov a cieľov renderovania. Taktiež umožňuje v reálnom čase vykonávať experimenty vedúce k lokalizácii hlavných výkonnostných problémov – napríklad vymeniť všetky textúry použité v aktuálnej scéne za malé textúry (veľkosť 2x2 pixely), čo v spojení s nárastom výkonnosti signalizuje problémy s priepustnosťou textúr, či zredukovať všetky objekty v scéne na jediný trojuholník, a sledovať tak výkonnosť aplikácie pri znížení zaťaženia GPU.

- Odladovač snímkov (Frame Debugger) slúži na analýzu jednotlivých snímkov a volaní Direct3D rozhrania ktoré sa v danom snímku uskutočnili. V tomto režime sú zobrazované všetky textúry a ciele renderovania asociované s aktuálnym volaním. Vizualizované sú aj stupne jednotlivých zobrazovaných mipmáp, kde stupeň s najvyšším rozlíšením je zafarbený načerveno, s najnižším namodro a ostatné stupne nadobúdajú rôzne dopredu nadefinované farby. Takto je možné odhaliť prípady kedy sa na scéne nachádza objekt zaberajúci malú plochu, napríklad malý kameň, ktorý ale používa neprimerane veľkú textúru, alebo naopak, identifikovať príčinu neostrého zobrazenia niektorých objektov, ktorou je použitie mipmapy s príliš malým rozlíšením.

- Profilovač snímkov (Frame Profiler) odhaľuje problémy s výkonnosťou pomocou vyhľadávania časovo najnáročnejších volaní v jednotlivých snímkoch. Zvolený snímok je renderovaný niekoľko krát po sebe, pričom sú zbierané informácie z GPU a špeciálneho ovládača používaného PerfHUDom o dobe potrebnej na vykonanie jednotlivých volaní. Tieto informácie sú následne vizualizované pomocou grafov.

3.2 Nástroje na tvorbu 3D obsahu do hier

Nasledujúce podkapitoly 3.2.1 až 3.2.3 bližšie predstavujú jednotlivé nástroje na tvorbu 3D obsahu do hier.

3.2.1 Autodesk Gmax, 3D Studio MAX a Maya

Gmax je aplikácia určená výhradne na modelovanie 3D objektov do hier založená na 3D Studiu MAX. Zatiaľ čo 3D Studio MAX predstavuje veľmi komplexný balík nástrojov určený pre profesionálov z oblasti digitálnych filmových efektov a hier, obsahujúci nástroje na modelovanie 3D objektov, animovanie, renderovanie a post produkciu, Gmax je veľmi úzko zameraný na vytváranie 3D obsahu do hier. Preto hoci s 3D Studiom MAX zdieľa vzhľad, spôsob ovládania a základnú sadu

nástrojov na modelovanie, textúrovanie a animovanie objektov, líši sa absenciou pokročilého renderovania, pokročilých materiálov, shaderov a fyzikálnej simulácie. Gmax je ľahko rozšíriteľný pomocou „herných balíkov“ (angl. game packs) vydávaných firmami vyvíjajúcimi hry, príkladom je BAT – Building Architect Tool firmy Maxis, určený pre hru Sim City 4. Hoci podpora firmy Autodesk bola ukončená 16. Októbra 2005, Gmax je naďalej dostupný zdarma na stránkach Autodesku a predstavuje veľmi silný nástroj na vytváranie 3D obsahu hier, ktorý je plne kompatibilný s XNA Content Pipeline.

Autodesk Maya predstavuje rovnako ako 3D Studio špičkový balík nástrojov zameraný na 3D modelovanie, animovanie, vytváranie vizuálnych efektov a renderovanie [21]. Maya je dostupná popri veľmi nákladných Complete a Unlimited verzií určených pre profesionálov a firmy aj v takzvanej PLE (Personal Learning Edition) verzii. Používatetelia PLE verzie dostupnej zdarma na stránkach Autodesku „majú prístup k hlavným nástrojom dostupným vo verzii Complete, vrátane nástrojov na modelovanie, animovanie, inverznú kinematiku, vytváranie časticových efektov, vytváranie shaderov a rovnako majú prístup k nástrojom na renderovanie. PLE verzia je určená na nekomerčné využívanie [22].“

3.2.2 SoftImage XSI a SoftImage ModTool

XSI je 3D animačný softvér určený pre hry, filmy a televíziu, pričom obsahuje kompletnú sadu nástrojov na 3D modelovanie, animovanie a renderovanie [23]. Svojou kvalitou a možnosťami je veľmi podobný hlavným produktom firmy Autodesk – 3D Studiu MAX a Maya.

Verzia XSI založená na rovnakom 3D engine určená na nekomerčné použitie sa nazýva ModTool a je voľne dostupná na stránkach firmy SoftImage [24]. ModTool bol po prvý krát oficiálne predstavený 17. Augusta 2007 a spomedzi všetkých spomínaných riešení poskytuje najlepšiu a najrýchlejšiu integráciu s XNA Content Pipeline. Je to vďaka priamej spolupráci firiem Microsoft a SoftImage, ktorej výsledkom bolo vytvorenie špeciálnych zásuvných modulov pre ModTool ktoré umožňujú „prepojiť ModTool s XNA Framework Content Pipeline a umožniť tak bezchybný a bezproblémový export [25].“ Výhodou je aj veľké množstvo dokumentácie podrobne a zrozumiteľne zachytávajúcej celý komplexný proces vytvárania herného 3D obsahu – od inštalácie XSI a integráciu s XNA a Visual Studiom po vytvorenie animovaného textúrovaného objektu a jeho zobrazenie v scéne pomocou XNA [26].

3.2.3 Blender

Blender je voľne šírená (open source, pod GNU GPL) aplikácia určená na vytváranie 3D obsahu, pričom jej cieľom je konkurovať „gigantom“ - 3D Studiu MAX, Mayi a XSI, a to funkcionalitou ale

nie cenou. Blender je použiteľný na modelovanie, textúrovanie, renderovanie, podporuje kosti, časticové efekty a animácie. Základňou pre používateľov Blenderu sú komunitné fóra [27]. Blender je uznávaný ako silný nástroj, no jeho hlavnou nevýhodou je ťažkopádne a miestami veľmi nelogicky navrhnuté grafické používateľské rozhranie ktoré odrádza množstvo najmä menej skúsených tvorcov modelov a modifikácií hier. Vďaka tomu si vybudoval povesť ťažko zvládnuteľného nástroja, ktorej sa zbavuje len veľmi pomaly, keďže funkcionalitou je porovnateľný s proprietárnymi nástrojmi.

	Autodesk 3Ds MAX 9 SP 2	**Autodesk Maya Complete 8.5**	**Blender 3D 2.45**	**SoftImage XSI Foundation 6.2**
Platformy	Windows	Windows, OS X, Linux	Windows, OS X, Linux, FreeBSD, Irix, Solaris	Windows, Linux
Odľahčená verzia	Gmax	Maya PLE	-	ModTool
Skúšobná verzia	Existuje	Existuje	-	Existuje
Jazykové verzie	Anglická, francúzska	Anglická	Mnoho	Anglická, japonská
Odhadovaná minimálna doba potrebná na zoznámenie sa s aplikáciou	Menej ako dva mesiace	Menej ako tri mesiace	Menej ako tri mesiace	Menej ako dva mesiace
Podpora pre jednotlivca	Dobrá	Dobrá	Dobrá - prostredníctvom komunitných fór	Vynikajúca, dostupné komunitné fóra
Úroveň dokumentácie	Vynikajúca	Vynikajúca	Dobrá	Veľmi dobrá
Kvalita dostupných video návodov na DVD	Dobrá	Veľmi dobrá	Nízka	Dobrá
Cena	5000 €	2500 €	Zdarma	450 €

Tab. 2: Prehľad vybraných vlastností nástrojov na tvorbu 3D obsahu do hier kompatibilných s XNA Content Pipeline.[1] [2]

	Autodesk 3Ds MAX 9 SP 2	**Autodesk Maya Complete 8.5**	**Blender 3D 2.45**	**SoftImage XSI Foundation 6.2**
Import .X	Prostredníctvom zásuvného modulu	Prostredníctvom zásuvného modulu	Áno	Áno
Export .X	Prostredníctvom zásuvného modulu	Prostredníctvom zásuvného modulu	Áno	Áno
Import .FBX	Áno	Áno	Nie	Áno
Export .FBX	Áno	Áno	Áno	Áno

Tab. 3: Prehľad dostupnosti importérov a exportérov pre hlavné formáty 3D modelov podporované XNA. [1]

[1] Údaje sú pre vzaté z http://wiki.cgsociety.org/index.php/Comparison_of_3d_tools a http://www.tdt3d.be/articles_viewer.php?art_id=99.

[2] Ceny sú platné k aprílu 2007 a nezahŕňajú DPH.

	Autodesk 3Ds MAX 9 SP 2	Autodesk Maya Complete 8.5	Blender 3D 2.45	SoftImage XSI Foundation 6.2
NURBS	Áno	Áno	Áno	Áno
Patch	Áno	Nie	Nie	Nie
SubD	Áno	Áno	Áno	Áno
Polygon	Áno	Áno	Áno	Áno
3gons	Áno	Áno	Áno	Áno
n-gons	Áno	Áno	Nie	Áno

Tab. 4: Porovnanie prítomnosti jednotlivých modelovacích nástrojov[1]

	Autodesk 3Ds MAX 9 SP 2	Autodesk Maya Complete 8.5	Blender 3D 2.45	SoftImage XSI Foundation 6.2
Textúrovanie uzlov	Áno	Áno	Áno	Áno
2D maľovanie textúr	Prostredníctvom zásuvného modulu	Áno	Áno	Áno
3D maľovanie textúr	Prostredníctvom zásuvného modulu	Áno	Áno	Nie
Simultánne nanášanie viacerých materiálov	Áno	Áno	Nie	Nie
Textúrovanie vrcholov	Áno	Áno	Áno	Áno

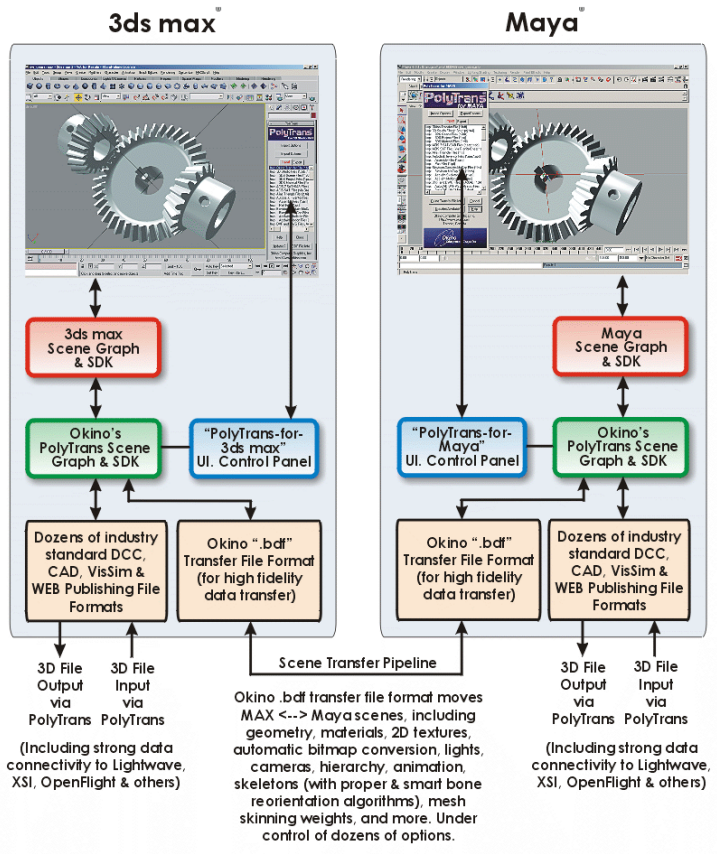
Tab. 5: Porovnanie prítomnosti jednotlivých nástrojov na textúrovanie modelov[1]

3.3 Konverzia 3D objektov medzi formátmi

Nasledujúce podkapitoly 3.2.1 a 3.3.2 sa venujú použitým nástrojom na konverziu 3D objektov medzi formátmi.

3.3.1 Okino Polytrans

Polytrans predstavuje veľmi užitočný nástroj na konverziu 3D modelov medzi jednotlivými formátmi. Je dostupný v troch verziách : ako samostatná aplikácia, ako zásuvný modul pre Autodesk 3D Studio MAX a ako zásuvný modul pre Autodesk Maya. Obojstranná konverzia modelov medzi Mayou a 3D Studiom MAX prebieha pomocou vlastného formátu súborov BDF. Pri konverzii modelov sú podporované všetky bežné používané formáty, rovnako ako formáty používané väčšinou 3D modelovacích aplikácií (napríklad Lightwave, XSI,COLLADA, VRML, SketchUp a iné). Okrem konverzie modelov Polytrans umožňuje aj konverziu kostí a kostrových systémov [28]. Obr. 8 zachytáva spôsoby konverzie 3D formátov podporované aplikáciou Polytrans.



Obr. 8: – Spôsoby konverzie 3D formátov podporované aplikáciou Polytrans. Prevzaté z [29].

3.3.2 Panda DirectX Exporter

Panda DirectX Exporter (PDE) je zásuvný modul pre 3D Studio MAX (podporované verzie sú 8, 9, 2008 a 2009), umožňujúci exportovať modely v DirectX .X formáte. PDE je dostupný zdarma na stránkach autora [30]. PDE predstavuje veľmi stabilný zásuvný modul poskytujúci širokú funkcionalitu – dokáže exportovať objekty, materiály, kostrové systémy, pomocné (dummy) objekty, shadre, podporuje automatickú konverziu textúr do zvoleného formátu a automatickú zmenu rozlíšenia textúry tak, aby výsledné rozmery textúry boli mocninou dvoch (dôležité pre XNA Content Pipeline).

3.4 Nástroje na tvorbu a úpravu textúr

Neodmysliteľnou súčasťou vytvárania 2D ale i 3D obsahu do hier je tvorba a úprava textúr, pretože textúry umožňujú vytvoriť dojem veľmi komplexného objektu i s relatívne malým počtom polygónov. Jedným z najpoužívanejších spôsobov editácie textúr je vyrenderovanie jednej veľkej textúry pre každý 3D model (render to texture) a jej následná editácia v ľubovoľnom grafickom editore. Veľkou výhodou je tohto prístupu je jednoduchosť, priamočiarosť, široká podpora zo strany 3D modelovacích aplikácií a dostupnosť grafických editorov.

Medzi najpoužívanejšie voľne šíriteľné editory patrí Paint.NET [31], ktorý mal byť pôvodne alternatívou k programu Paint dodávaného s operačnými systémami Windows, no rozrástol sa na plnohodnotný grafický editor, alebo dobre známy multiplatformový editor GIMP (GNU Image Manipulation Program) [32].

Komerčné editory sú populárne najmä medzi profesionálmi. Hoci poskytujú veľké množstvo nástrojov a rozšírení funkcionality pomocou zásuvných modulov, ich zvyčajne vysoká cena je dôvodom prečo nie sú veľmi používané amatérskymi vývojármi a grafikmi. Medzi najpopulárnejšie komerčné editory patrí Paint Shop Pro od firmy Corel [33] a Photoshop od firmy Adobe [34]. Photoshop je obzvlášť obľúbený pri tvorbe obsahu do hier, dôkazom čoho je aj množstvo zásuvných modulov vytvorených pre tento program. Zaujímavé voľne šíriteľné moduly poskytuje na svojich stránkach firma NVIDIA vo vývojárskej sekcii [35]. Umožňujú import a export textúr v DDS formáte, vytváranie normálových máp z výškových máp, generovanie mip-máp a korekcie máp prostredia.

4 Cieľ práce

Ako už bolo spomenuté v kapitole 2.4, štartovacie balíky plnia niekoľko veľmi dôležitých úloh, ktorými sú najmä pritiahnutie nových potenciálnych používateľov danej technológie vďaka živej ukážke možností a schopností technológie a poskytnutie komplexného pohľadu, odhaľujúceho základné princípy a mechanizmy na ktorých je technológia vybudovaná.

Hoci každý z XNA štartovacích balíkov je zameraný na iný žáner, vo všeobecnosti môžeme konštatovať, že ich spoločnou črtou je oboznamovanie začínajúcich vývojárov s aplikačným modelom, importovaním, použitím a správnym odstránením herného obsahu, získavaním vstupu zo vstupných zariadení, prehrávaním zvukov, renderovaním a použitím shaderov.

Po bližšom preskúmaní, pracovaní s nimi a modifikácii je možné nájsť v existujúcich štartovacích balíkoch určité nedostatky, medzi ktoré patria:

- **Prílišná komplexnosť** – niektoré štartovacie balíky sú príliš veľké a začiatočníka môžu skôr odradiť. Nie je jednoduché orientovať sa v zdrojovom kóde pozostávajúcom z viac ako 40 000 riadkov, ktorý pokrýva len základnú funkcionalitu. V tomto prípade platí že menej je niekedy viac a komplexná vnútorná štruktúra môže byť na škodu ak nepomáha, ale naopak vedie k nejasnostiam a zmätku.
- **Obtiažna rozšíriteľnosť** – je jedným z často sa opakujúcich problémov štartovacích balíkov, nejedná sa len o problém XNA balíkov. Štartovacie balíky nie sú navrhnuté dostatočne generický a nie je možné podstatným spôsobom vytvoriť robustnejšiu verziu hernej logiky bez predchádzajúcich náročných a rozsiahlych modifikácií zvyšku balíka. Takýto štartovací balík následne stráca vo veľkej miere svoje opodstatnenie, keďže jedným z hlavných cieľov je možnosť jednoduchého rozširovania balíka a postupné pridávanie funkcionality.
- **Prílišné podriadenie sa určitej platforme** – tento nedostatok sa vzťahuje k štartovacím balíkom určeným pre viac ako jednu platformu. Najčastejšie sa jedná o výber nevhodného spôsobu ovládania použitom v štartovacom balíku, kde ovládanie istým vstupným ovládacím zariadením je veľmi vhodné a efektívne na jednej platforme no na druhej platforme sa jedná o veľmi ťažkopádny a nevhodný spôsob ovládania ku ktorému nie je poskytnutá žiadna alternatíva. Ďalším veľmi často sa vyskytujúcim problémom v tejto oblasti sú nedostatky v prezentačnej vrstve štartovacieho balíka, ktorá nerešpektuje špecifiká určitej platformy – príkladom je ignorovanie bezpečnej oblasti (angl. safe area) pri zobrazovaní prvkov grafického používateľského rozhrania, v dôsledku čoho na starších televízoroch pripojených k Xboxu 360, tieto prvky nie sú zobrazené.
- **Použitie negenerického herného obsahu** – pri niektorých štartovacích balíkoch je použitý nevhodný herný obsah. Príkladom sú modely v binárnom tvare s parametrami pre procesor

modelov nastavenými v 3D modelovacej aplikácii bez možnosti ich menenia, shadre referencované priamo z modelov či kostrové systémy vytvorené a exportované netradičným a nedokumentovaným spôsobom. V týchto prípadoch je veľmi obtiažne vytvárať vlastný herný obsah pre tento štartovací balík, čím je jeho celková použiteľnosť značne obmedzená.

☒ **Nedostatočné množstvo komentárov a dokumentácie** – kvalitné, zrozumiteľné komentáre v zdrojovom kóde a dostatočne rozsiahla dokumentácia sú jedny zo základných kameňov kvalitného štartovacieho balíka. Bohužiaľ dosť často bývajú komentáre len veľmi sporadické bez bližšieho vysvetlenia a zdôvodnenia.

Práve prekonanie týchto nedostatkov a vytvorenie užitočného, ľahko použiteľného a rozšíriteľného štartovacieho balíka je cieľom nadväzujúcej diplomovej práce. Ak má byť navrhnutý a implementovaný štartovací balík skutočným prínosom pre XNA komunitu, musí nevyhnutne spĺňať nasledujúce požiadavky:

☑ **Jednoduchá rozšíriteľnosť** – vytvorenie flexibilnej vnútornej štruktúry umožňujúcej značné zmeny hernej logiky a spôsobu prezentovania údajov bez nutnosti modifikácie zvyšných častí štartovacieho balíka.

☑ **Ladenie hrateľnosti bez nutnosti neustáleho kompilovania zdrojového kódu** – využitie formátu XML na zápis nastavení a parametrov súvisiacich s hrateľnosťou, čím dôjde k odstráneniu potreby neustáleho kompilovania kódu pri vykonávaní častých drobných zmien akými sú zmeny parametrov premenných súvisiacich s hrateľnosťou.

☑ **Využitie komponent so znovupoužiteľným kódom** – aplikačný model XNA umožňuje vložiť kód využiteľný vo viacerých projektoch do špeciálnych komponent nazývaných Game Components. Komponenty fungujú na princípe plug-and-play, čo zvyšuje prehľadnosť kódu a odbúrava nutnosť opakovane vytvárať kód s rovnakou funkcionalitou.

☑ **Využitie predností cieľových platforiem** – každá cieľová platforma má svoje výhody ktoré je nutné v čo najširšej miere využiť, pričom je nutné poskytnúť vhodné alternatívy pokiaľ určité prvky jednej platformy spôsobujú problémy pri využívaní štartovacieho balíka na inej platforme.

☑ **Prehľadný systém vytvárania a importovania herného obsahu** – herný obsah musí byť vytváraný jednotným a dobre zdokumentovaným spôsobom nástrojmi ktoré sú všeobecne dostupné za použitia všeobecne dostupných formátov kompatibilných s XNA Content Pipeline.

☑ **Kvalitná dokumentácia** – nielen komentáre v zdrojovom kóde, ale aj celková dokumentácia štartovacieho balíka zachytávajúca najčastejšie scenáre použitia, poskytujúca vysokoúrovňový prehľad o komponentoch štartovacieho balíka a spôsobe akým spolupracujú.

Po sformulovaní týchto všeobecných zásad bolo nutné pristúpiť k rozhodnutiu, či nový štartovací balík pre XNA bude rozširovať, prípadne nahradzovať niektorý z existujúcich balíkov, alebo bude navrhnutý a implementovaný nový štartovací balík venovaný inému hernému žánru. Tomuto rozhodnutiu predchádzala komunikácia s ďalšími členmi XNA komunity, ktorá prebiehala prevažne na fórach Creator's Clubu [36], hlavnom portáli pre vývojárov používajúcich XNA. Na základe získanej spätnej väzby som sa rozhodol navrhnúť a implementovať štartovací balík pre XNA zameraný na tvorbu strategickej hry v reálnom čase, RTS – Real Time Strategy (Game). Obr. 9 ukazuje záber z populárnej RTS hry Command and Conquer 3.

Obr. 9: RTS Command and Conquer 3. Prevzaté z [37].

RTS hry vychádzajú z princípov klasických strategických doskových hier (angl. board games) no ako ich vývoj postupoval stávali sa komplexnejšími a hra na ťahy pri ktorej sa striedali ťahy oponentov bola nahradená dynamickejšou a napínavejšou hrou v reálnom čase, kedy protihráči pri vydávaní povelov svojim jednotkám nie sú obmedzení pridelenými časovými úsekmi. Cieľom hry je zvyčajne zničiť jednotky a/alebo budovy protihráča, prípadne ovládnuť určité územie. K naplneniu tohto cieľa používajú hráči jednotky a budovy ktoré majú pod svojím velením. Jednotky a budovy je možné ovládať priamo a vydávať im rozkazy, ktoré zvyčajne zahŕňajú zmenu pozície, zaútočenie, obranu či niektoré špeciálnejšie príkazy, napríklad opravu poškodenej priateľskej jednotky. Niektoré RTS hry sa sústreďujú iba na strategickú časť – to znamená manévrovanie, jednotiek, útočenie, obranu a dobývanie územia, pričom nie je možné v priebehu hry vytvárať ďalšie jednotky a budovy. Iné umožňujú vytváranie dodatočných jednotiek a budov v priebehu hry za nahromadené zdroje,

ktoré sú typicky kumulované ak hráč obsadí špeciálne body na hernej mape. Tento typ hier je teda obohatený o ďalšiu zložku ktorá predstavuje manažment zdrojov a výrobu nových jednotiek.

Nakoľko RTS predstavuje veľmi populárny a rozšírený žáner a ani viac ako dva a pol roka po vydaní prvej verzie XNA neexistuje žiaden komerčný ani voľne dostupný štartovací balík pre XNA venovaný strategickej hre, rozhodol som sa túto medzeru vyplniť a tento balík navrhnúť, implementovať a sprístupniť ostatným členom XNA komunity. Som si istý že bude predstavovať skutočný prínos a pomôže mnohým začínajúcim herným programátorom pri vytváraní ich vlastných hier.

5 Návrh systému

Táto kapitola sa venuje návrhu prototypu systému – RTS štartovaciemu balíku určenému pre XNA. Tento prototyp je založený na princípoch uvedených v predchádzajúcej kapitole, pričom boli brané na zreteľ špecifiká herného žánru RTS. Tri podkapitoly postupne predstavujú funkcionálne požiadavky kladené na systém, nefunkcionálne požiadavky a hlavné komponenty prítomné v prototype štartovacieho balíka.

5.1 Funkcionálne požiadavky

Pred samotnou implementáciou prototypu štartovacieho balíka boli určené nasledujúce funkcionálne požiadavky zoradené do kategórií podľa komponentu ku ktorému sú logicky priradené:

RTS kamera

FP01 – Vytvorenie kamery ako komponentu poskytujúceho služby

FP02 – Kamera musí zodpovedať štandardnej kamere použitej v RTS hrách
(tzn. musí byť plne ovládateľná myšou, natáčateľná len v určitom uhle, ...)

FP03 – Kamera sa sama musí postarať o to, aby nekolidovala s terénom

FP04 – Kamera musí poskytovať údaje o svojej polohe a natočení ostatným komponentom

Systém ovládania

FP05 – Zachytávanie vstupu z myši, klávesnice aj gamepadu

FP06 – Poskytuje alternatívne ovládanie ak to platforma dovoľuje

FP07 – Možnosť vydávania príkazov jednotkám

Terén

FP08 - Vytvorenie terénu ako komponentu poskytujúceho služby

FP09 – Terén nesmie byť renderovaný ako celok, iba časť videná kamerou

FP10 – Musí podporovať zmiešavanie textúr

FP11 – Musí poskytovať informácie o výške a normále v ľubovoľnom bode

FP12 – Musí podporovať načítanie a ukladanie informácií o váhach textúr vo vrcholoch

FP13 – Ukladanie informácií o objektoch na povrchu terénu

Hierarchia objektov

FP14 – Podpora objektov prostredia aj objektov kontrolovaných hráčmi

FP15 – Implementovanie hľadania optimálnej cesty v teréne

FP16 – Riešenie vzájomnej kolízie jednotiek

FP17 – Riešenie vzájomnej viditeľnosti jednotiek

Systém animovania kostí

FP18 – Podpora animácií budov a všetkých druhov jednotiek

FP19 – Plná kompatibilita minimálne s kostrovým systémom použitom v 3Ds MAX

FP20 – Podpora viacerých kostí

FP21 – Podpora pomocných objektov lokalizujúcich presné miesto na jednotke / budove prípadne v blízkom okolí

Manažér nábojov

FP22 – Podpora priamych nábojov

FP23 – Podpora nábojov letiacich po balistickej krivke

FP24 – Podpora výbušnín

Editor prostredia

FP25 – Možnosť zafarbovania terénu (angl. terrain painting)

FP26 – Možnosť pridávania objektov na terén a ich odoberanie

5.2 Nefunkcionálne požiadavky

Okrem určenia základných funkcionálnych požiadaviek je nevyhnutné venovať sa aj nefunkcionálnym. Medzi tie najpodstatnejšie patria:

NP01 - Štartovací balík musí byť multiplatformový – cieľové platformy sú Windows a Xbox 360. Rozdiely v kóde dané odlišnosťou platforiem musia byť minimálne a žiadna z platforiem nesmie byť preferovaná na úkor tej druhej.

NP02 - Náročnosť na hardvér musí byť vo verzii pre Windows minimálna.

NP03 - Štartovací balík musí byť voľne šíriteľný a dostupný každému členovi XNA komunity.

NP04 - Štartovací balík musí byť dobre škálovateľný a modifikovateľný.

NP05 - Štartovací balík musí byť stabilný a prípadné chybové hlásenia musia byť zrozumiteľné a nevšeobecné.

NP06 - Štartovací balík musí jednoducho spolupracovať s grafickými knižnicami iných autorov.

NP07 - Jednotlivé časti balíka musia byť jednoducho testovateľné.

5.3 Opis systému

Implementovaný štartovací balík je navrhnutý a implementovaný ako dvojúrovňová aplikácia. Pri vytváraní aplikácie bola totiž hlavnou motiváciou snaha oddeliť hernú logiku od zvyšku systému, čím dôjde k zabezpečeniu znovupoužiteľnosti maximálneho možného množstva kódu.

Nižšia, takzvaná systémová úroveň obsahuje triedy zodpovedné za základné operácie nižšej úrovne, ktoré nesúvisia priamo s hernou logikou. Tieto triedy zabezpečujú komunikáciu s XNA, vďaka čomu dokážu komunikovať s hardvérom či už na platforme PC, alebo Xbox 360, slúžia ako pomocné triedy zaobaľujúce často používanú funkcionalitu (pomocné – helper triedy) a predstavujú implementáciu základných algoritmov, akým je napríklad hľadanie cesty.

Vyššia, takzvaná herná úroveň obsahuje všetku hernú logiku ktorá je ušitá presne na mieru cieľovému hernému štýlu, v tomto prípade strategickej hre hranej v reálnom čase. Týmto sa líši od spodnej úrovne, ktorá je nezávislá na zvolenom cieľovom hernom štýle. Táto úroveň obsahuje hlavný komunikačný kanál objektov, ktorý pri hre viacerých hráčov výrazne znižuje množstvo dát ktoré je nutné preniesť sieťou, hierarchiu herných objektov, ako aj triedy spracovávajúce definíciu konkrétnej hry z externých XML súborov. Vďaka tomu je samotná hra konfigurovateľná bez rekompilácie kódu a teda môže byť modifikovaná nielen programátormi, čo predstavuje jednu z hlavných výhod tohto štartovacieho balíka oproti tým ktoré už existujú. Kapitoly 5.4.1 až 5.4.6 bližšie popisujú hlavné komponenty nachádzajúce sa na tejto vrstve, kapitola 5.5 podrobne predstavuje vstavaný herný editor, ktorým je možné vizuálne definovať hru.

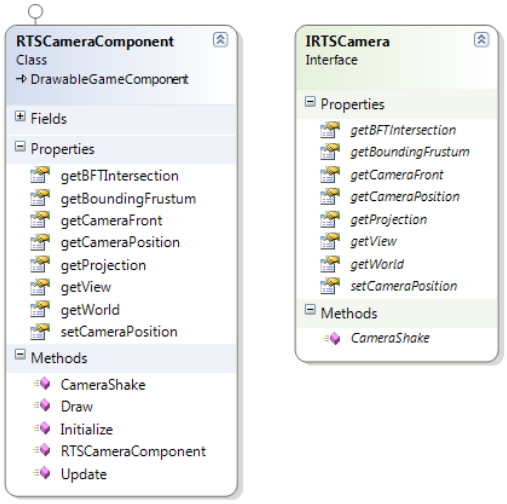
5.4 Hlavné komponenty prototypu

5.4.1 RTS kamera

V 3D aplikáciách je kamera jedným zo základných komponentov. Kamera reprezentuje pohľad používateľa na herný svet. XNA renderuje tento 3D svet na obrazovku tak, ako je videný kamerou. Použitá RTS kamera poskytuje rovnaký pohľad a používa rovnaký spôsob ovládania ako kamery používané štandardne v 3D RTS hrách, čo je veľmi pohodlné pre hráčov tohto typu hier.

Kamera je vytvorená ako herný komponent, takže je použité dedenie od `GameComponent` a obsahuje svoje vlastné verejné rozhranie `IRTSCamera`, cez ktoré je umožnené ostatným komponentom využiť funkcionalitu kamery.

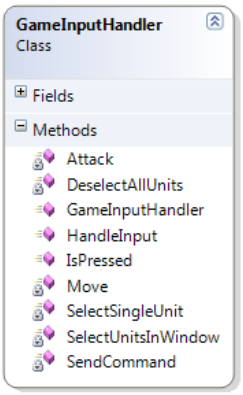
Obrázok 10 ukazuje `RTSCameraComponent` a verejné rozhranie `IRTSCamera` ktoré používa.



Obr. 10: RTSCameraComponent a verejné rozhranie IRTSCamera ktoré používa.

5.4.2 Systém ovládania

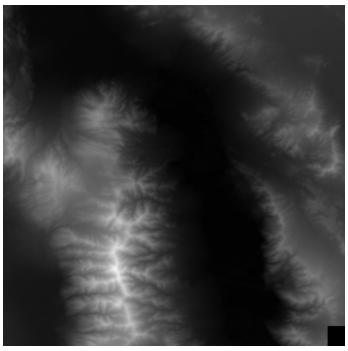
Systém ovládania hry sa nachádza v triede `GameInputHandler`. Trieda obsahuje verejnú metódu `HandleInput`, ktorá uchováva predchádzajúci stav gamepadu, klávesnice a myši. Pri porovnaní súčasného a predchádzajúceho stavu týchto ovládacích zariadení je možné zistiť rozdiely a podľa nich určit aké tlačidlo používateľ stlačil prípadne či a kam presunul myš. Keďže .NET Compact Framework použitý na platforme Xbox 360 neobsahuje knižnice umožňujúce prácu s myšou, pri kompilácii pre túto platformu je potrebné podmienkami zamedziť kompilovaniu niektorých častí kódu. Ako alternatíva k ovládaniu pomocou myši je implementovaná vhodná náhrada využívajúca gamepad. Obrázok 11 ukazuje triedu `GameInputHandler`.



Obr. 11: Trieda GameInputHandler.

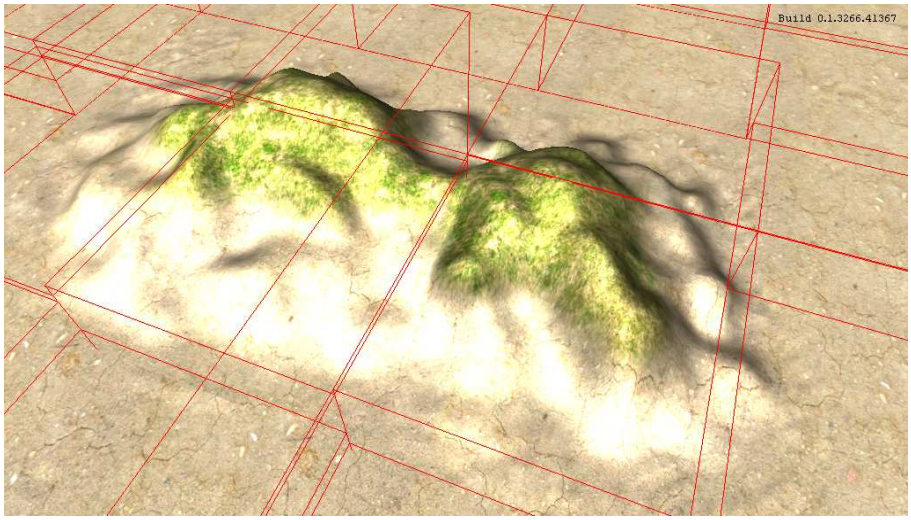
5.4.3 Terén

Terén je jedným z najdôležitejších a najkomplexnejších komponentov navrhnutého štartovacieho balíka. Je založený na výškovej mape zloženej z odtieňov sivej farby. Každý pixel reprezentuje bod s určitou výškou, kde biela farba reprezentuje maximálnu možnú výšku a čierna farba minimálnu. Na obrázku 12 je zachytený príklad použitej výškovej mapy.



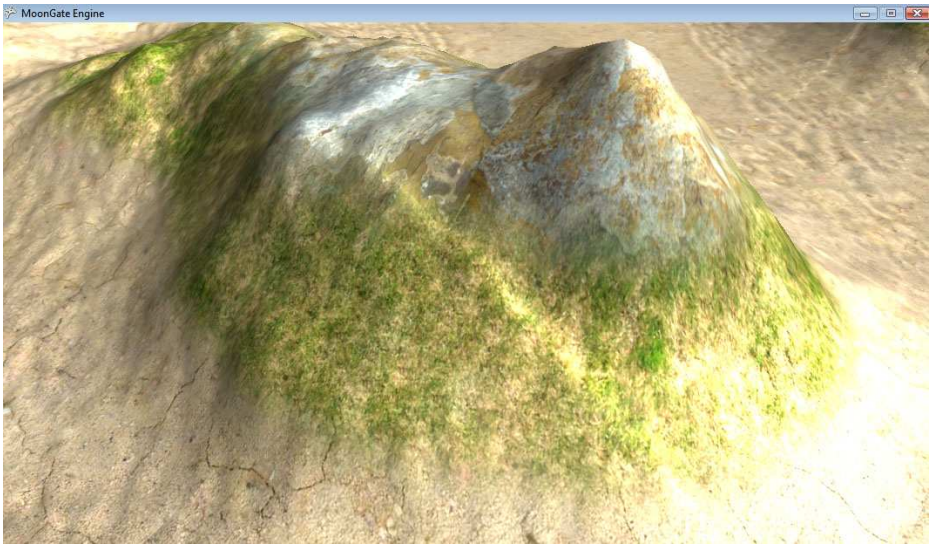
Obr. 12: Príklad použitej výškovej mapy.

Po načítaní výšok z výškovej mapy je pomocou rekurzívneho delenia vytvorená stromová štruktúra QuadTree, kde v listoch stromu sa nachádzajú štvorcové časti terénu a v uzloch osovo zarovnané kvádre (AABB – Axis Aligned Bounding Box). Pred samotným vyrenderovaním terénu je tento strom rekurzívne prechádzaný, pričom v každom uzle je zisťované či dochádza k prieniku plochy videnej kamerou a príslušným AABB. Pokiaľ je táto podmienka splnená aj pre najmenší AABB, dôjde k vyrenderovaniu príslušného štvorca s terénom. Vďaka tomuto princípu je zobrazený len kus terénu ktorý je aktuálne videný kamerou. Na základe experimentov bola stanovená veľkosť štvorca s terénom na veľkosť 32 x 32 vrcholov terénu, čo pri sklone kamery zodpovedajúcom väčšine RTS hier znamená, že sa v priemere renderuje iba 10 štvorcov s terénom zo 64 existujúcich, čo je 15,625% celkového povrchu terénu. Na obrázku 13 je znázornená vizualizácia štvorcov s terénom a AABB

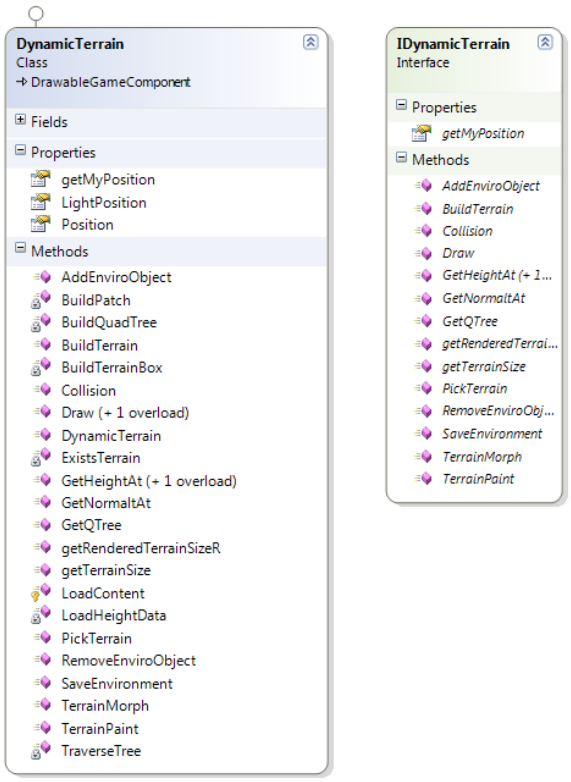


Obr. 13: Vizualizácia štvorcov s terénom a AABB

Terén podporuje bump mapping a texture blending. Textúry sú vo vrcholoch terénu zmiešavané podľa váhy. O tento výpočet sa stará GPU. Terén momentálne podporuje 4 textúry. Obrázok 14 ukazuje zmiešavanie textúr vo vrcholoch terénu. Obrázok 15 znázorňuje komponent `DynamicTerrain` zostavujúci a renderujúci terén a jeho verejné rozhranie `IDynamicTerrain`. Príloha A je venovaná shadru použitom na renderovanie terénu.



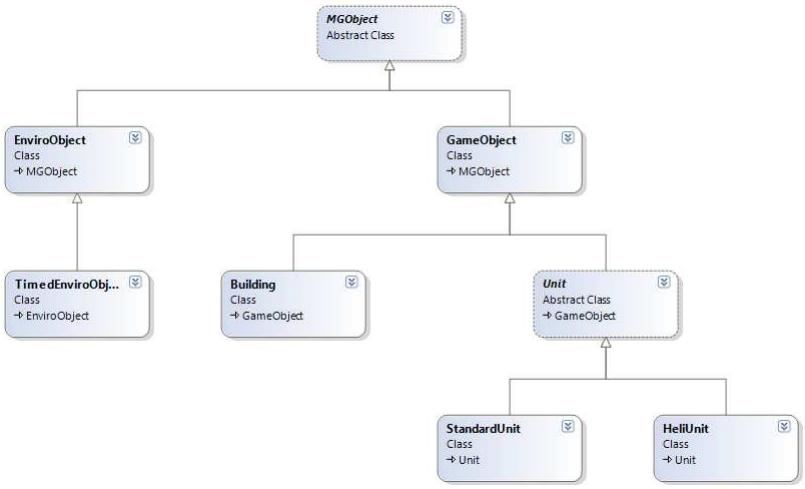
Obr. 14: Zmiešavanie textúr vo vrcholoch terénu



Obr. 15: DynamicTerrain komponent a rozhranie IDynamicTerrain ktoré používa

5.4.4 Hierarchia objektov

Hierarchia objektov predstavuje hierarchiu renderovateľných objektov nachádzajúcich sa v štartovacom balíku. Vrchol hierarchie predstavuje abstraktná trieda `MGObject`. Jedná sa triedu poskytujúcu najvšeobecnejšie atribúty objektov ako napríklad unikátny identifikátor a informácie o polohe a rotácii objektu. Základné delenie renderovateľných objektov spočíva v delení na základe ovládateľnosti hráčmi. `EnviroObjekty` sú objekty prostredia ktoré hráči nemôžu vlastniť, manipulovať s nimi a zničiť ich. Tieto objekty slúžia ako dekorácie. Ide napríklad o stromy, skaly či stĺpy elektrického vedenia. `TimedEnviroObjekty` vychádzajú z `EnviroObjektov` no majú v sebe 3časovač umožňujúci im vykonať nejakú činnosť po uplynutí časového intervalu. Príkladom sú vraky jednotiek ktoré sa objavia po zničení jednotky, ktoré po niekoľkých sekundách zmiznú. Druhým typom sú hráčmi ovládateľné objekty - `GameObjects`, čo sú buď stacionárne objekty – budovy (do tejto kategórie patria však aj ozbrojené veže a míny), alebo pohyblivé jednotky. Mobilné jednotky sú buď pozemné, reprezentované triedou `StandardUnit`, alebo letecké, reprezentované triedou `HeliUnit`. Obrázok 16 znázorňuje spomínanú hierarchiu objektov. Príloha B ukazuje popis objektov prostredia vo formáte XML.

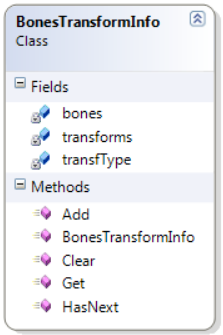


Obr. 16: Hierarchia objektov

5.4.5 Systém animovania kostí

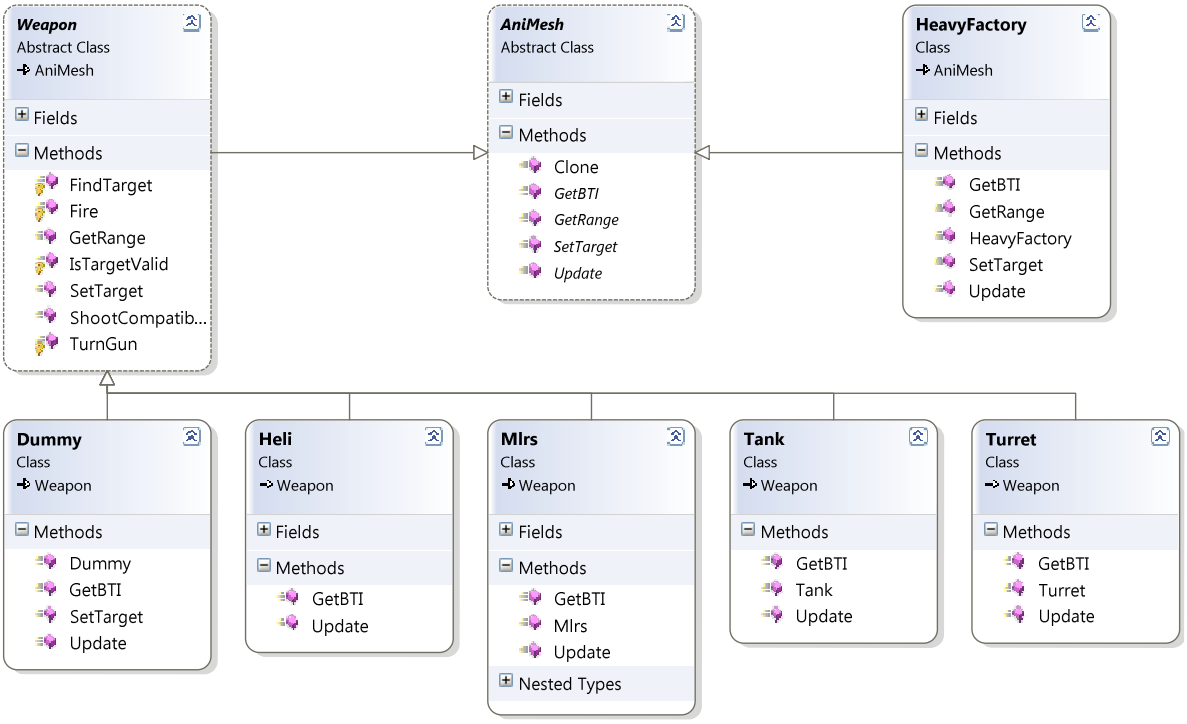
Tak ako každý herný žáner ktorý zažil prechod z 2D zobrazenia do 3D, aj žáner stratégií využíva animovanie kostí na vytváranie efektných animácií objektov namiesto zastaraného spôsobu rýchleho striedania textúr používaného pri 2D hrách. Animovanie objektov v 2D je z programátorského uhla pohľadu relatívne triviálne, keďže sa jedná o vhodné striedanie textúr, prípadne ich častí. Pridanie tretieho rozmeru však okrem zlepšenia vizuálnej kvality prinieslo aj problémy, keďže animovanie kostí v 3D nie je jednoduché ani z pohľadu animátora, ani z pohľadu programátora.

Implementovaný systém kostí je založený na objektoch manipulujúcich s kosťami v reálnom čase, čiže sa nejedná o sadu exportovaných animácií založených na kľúčových snímkoch (angl. key-frame animation). Každý animovaný objekt referencuje inštanciu triedy `AnimController`, ktorá obsahuje referencie na triedy manipulujúce s kosťami modelu určitým spôsobom. Pred samotným vyrenderovaním objektu prislúchajúci `AnimController` zozbiera informácie o kostiach daného objektu zo všetkých tried manipulujúcich jeho kosťami. Tieto informácie sú uložené v triede `BonesTransformInfo`, zobrazenej na obrázku 17.



Obr. 17: Trieda BonesTransformInfo

Na základe týchto informácií o kostiach objektu sú nastavené kosti príslušného 3D modelu, ktorý je následne vyrenderovaný. Obrázok 18 ukazuje ovládače manipulujúce s kosťami. Ovládače sú pomenované podľa objektov pre ktoré sú primárne určené.

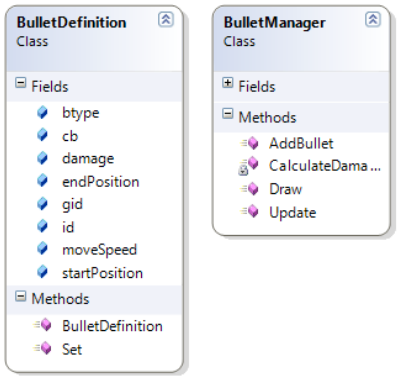


Obr. 18: Ovládače manipulujúce s kosťami.

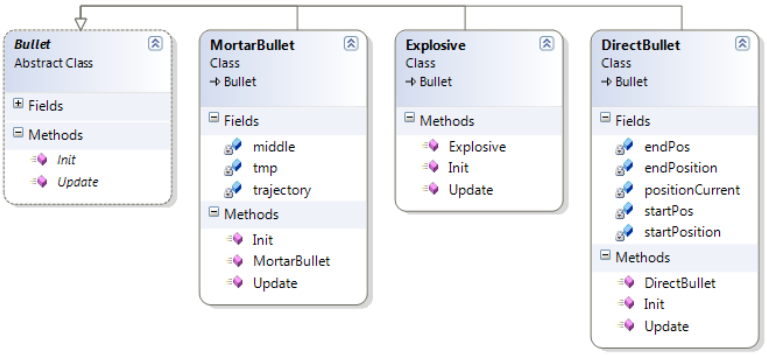
5.4.6 Manažér nábojov

Manažér aktívnych nábojov, `BulletManager` je trieda referencujúca všetky aktívne náboje nachádzajúce sa na mape v určitom momente. Po vystrelení zbraň prislúchajúca objektu poskytne informácie o svojom náboji formou triedy `BulletDefinition` tomuto manažérovi, ktorý na základe tohto popisu vytvorí príslušný objekt a stará sa o jeho vykresľovanie, prepočítavanie trajektórie a zisťovanie kolízií. Tento princíp umožňuje veľmi ľahko vytvoriť rovnako sa správajúce objekty používajúce rovnaký typ projektilov s rôznymi vlastnosťami. Napríklad tank ktorý zničil niekoľko nepriateľov a je stále funkčný môže byť povýšený do stavu „veterán“ a jeho projektily môžu mať väčší dostrel alebo silu. Tento princíp je veľmi často využívaný v RTS hrách. V súčasnosti sú implementované tri základné typy projektilov – `MortarBullet` je náboj vystrelený po balistickej krivke, `Explosive` je okamžite vybuchujúci náboj ktorý sa nepohybuje (napr. mína) a `DirectBullet` je náboj pohybujúci sa po priamke. Obrázok 19 ukazuje triedy

`BulletDefinition` a `BulletManager`, obrázok 20 triedy zodpovedajúce jednotlivým typom nábojov.



Obr. 19: Triedy BulletDefinition a BulletManager

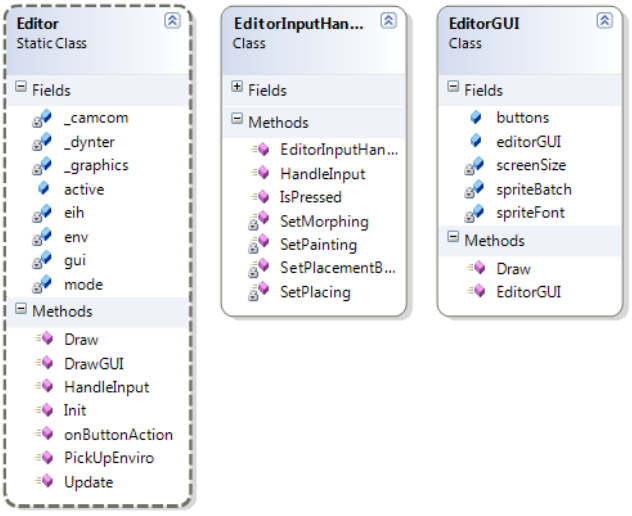


Obr. 20: Jednotlivé typy nábojov

5.5 Editor prostredia

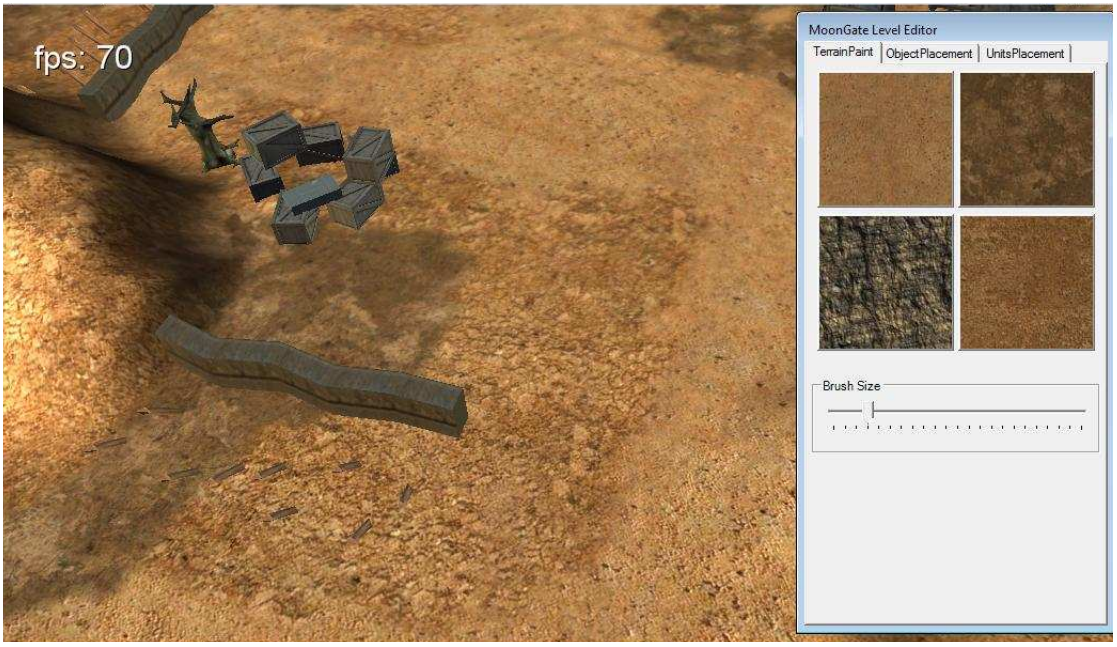
Hoci sa to na prvý pohľad nemusí javiť ako úplná samozrejmosť, kvalitný vizuálny editor prostredia je jedným z najpotrebnejších nástrojov ktoré je nutné implementovať pri vytváraní strategickej hry. Editor výrazne skracuje čas potrebný na vytvorenie prostredia a slúži ako základný nástroj pri vytváraní modifikácií hry nadšencami. Existujú dva základné typy editorov prostredia – editor ako samostatná aplikácia (angl. stand alone editor) a vstavaný editor dostupný priamo z aplikácie. V implementovanom prototype sa nachádza vstavaný editor, pretože hoci je tento typ náročnejší na implementáciu prináša výhodu editovania prostredia priamo počas hry, pričom zmeny sú okamžite viditeľné bez nutnosti reštartovať aplikáciu.

Editor využíva technológiu Windows Forms, ktorá je súčastou .NET Frameworku, vďaka ktorej je možné používať štandardné ovládacie prvky Windows. Keďže Windows Forms pracuje na princípe udalostí, aj editor komunikuje s hlavným oknom aplikácie pomocou udalostí a delegátov. Obrázok 21 ukazuje triedy tvoriace editor prostredia.

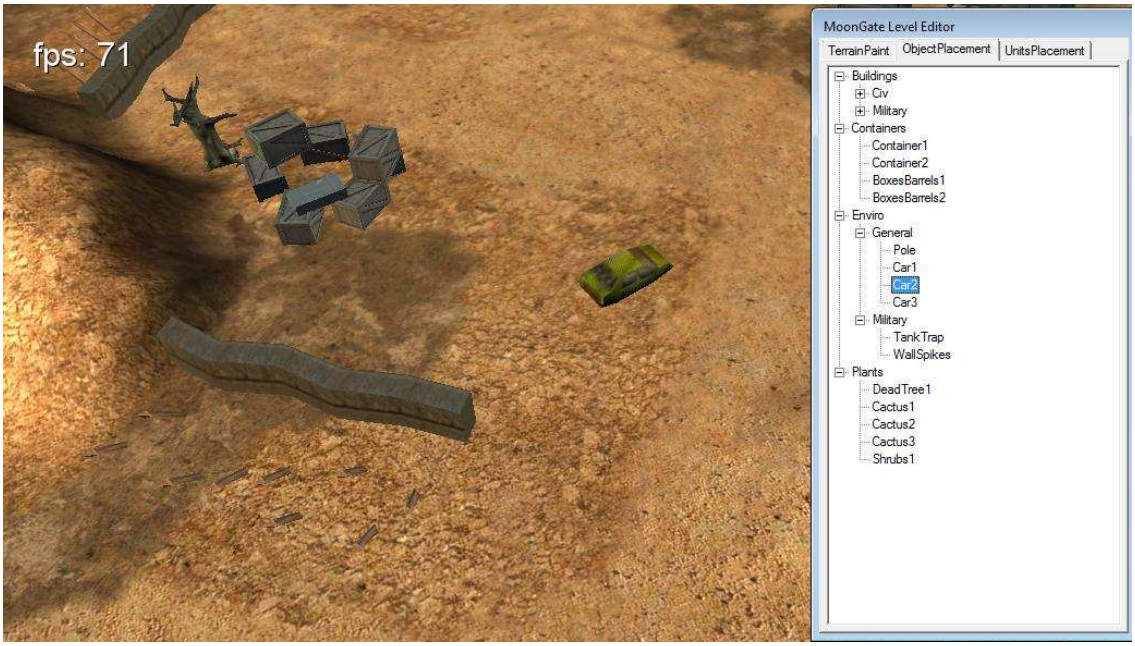


Obr. 21: Triedy tvoriace vstavaný editor - Editor, EditorInputHandler a EditorGUI

Editor pracuje v troch režimoch. Prvým je režim „TerrainPaint“ (Obr. 22), čiže nanášanie textúr na povrch terénu. K zafarbovaniu dochádza pomocou zmeny váh textúr vo vrcholoch terénu. Všetky zmeny zafarbenia sú okamžite viditeľné a automaticky sa ukladajú, takže pri novom spustení aplikácie sa automaticky zobrazí rovnaký stav terénu aký bol pri ukončení aplikácie. Na terén je možné naniesť maximálne štyri textúry, pričom je podporovaný bump mapping. Okrem samotnej textúry je možné vybrať aj polomer štetca ktorým je textúra na terén nanášaná.

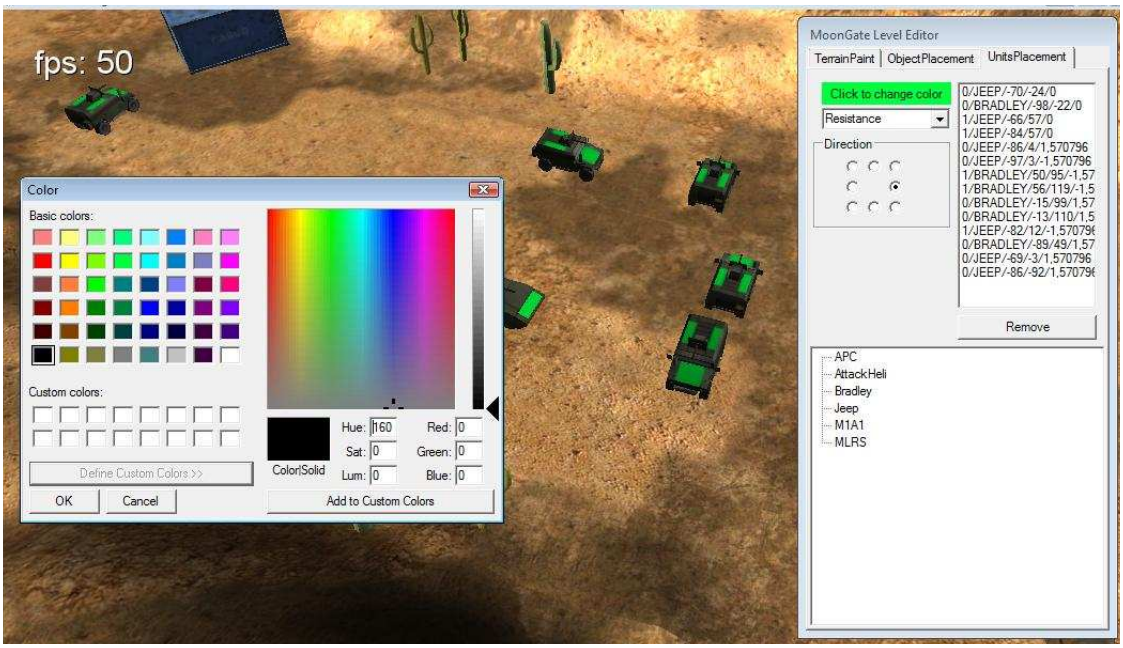


Obr. 22: Editor prostredia – režim nanášania textúry na terén



Obr. 23: Editor prostredia – ukladania objektov na povrch terénu

Druhým režimom je režim „Object Placement“ (Obr. 23), ktorý umožňuje ukladať objekty na povrch terénu. Pred samotným uložením je nutné najprv objekty v engine zaregistrovať. Podrobný popis ako toho dosiahnuť sa nachádza v kapitole 6.1. Objekty prostredia sú založené na triede `EnviroObject`, jedná sa teda o jednoduché statické objekty. Po vybratí objektu je možné objekt rotovať a následne uložiť na povrch terénu, odkiaľ je možné ho opäť uchopiť a premiestniť, alebo odstrániť.



Obr. 24: Editor prostredia – režim vkladania dynamických jednotiek

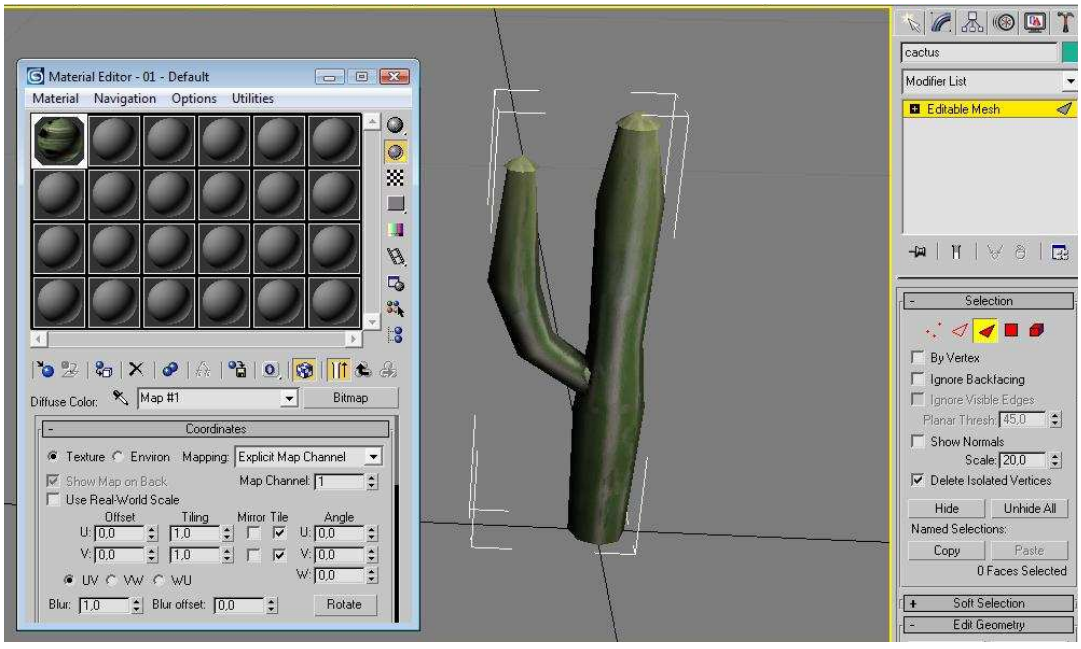
Režim „Unit Placement“ (Obr. 24), teda rozmiestňovanie jednotiek, je tretím režimom ktorý editor ponúka. Slúži na vkladanie dynamických objektov do prostredia. Tieto objekty predstavujú komplexné objekty ovládateľné hráčmi. V tomto režime je možné zvoliť tím do ktorého bude nová jednotka patriť a jej úvodnú rotáciu. Následne je možné jednotku priamo vložiť do prostredia, pričom táto jednotka začne okamžite reagovať na podnety z okolia, takže ak je napríklad umiestnená do blízkosti nepriateľskej jednotky okamžite na ňu začne útočiť. Okrem samotného rozmiestňovania jednotiek je možné meniť tímové farby jednotlivých tímov, pričom zmena je okamžite viditeľná.

6 Použitie Content Pipeline

Vytvorenie kvalitného a atraktívneho herného obsahu a jeho úspešné použitie v akomkoľvek vizualizačnom engine je netriviálnou a komplexnou úlohou, na ktorej úspešnom zvládnutí závisí do značnej miery jeho celková použiteľnosť. Táto kapitola popisuje vytvorenie a použitie dvoch základných typov objektov – jednoduchého objektu slúžiaceho ako objekt prostredia a dynamického ovládateľného objektu. Jej cieľom je demonštrovať jednoduchosť a priamočiarosť procesu modifikácie a rozširovania štartovacieho balíka, čo je veľkou výhodou implementovaného štartovacieho balíka. Herný obsah je možné vytvárať v ľubovoľnom nástroji na vytváranie 3D obsahu, táto kapitola popisuje vytváranie pomocou nástroja 3D Studio Max 8.0 a exportéra Pandasoft DirectX Exporter v. 4.8.63.0.

6.1 Vytvorenie jednoduchého objektu prostredia

Ako bolo spomenuté v kapitole 5.4.4, objekty prostredia sú jednoduché objekty s ktorými nie je možné manipulovať a slúžia primárne ako dekorácie prípadne prekážky na hernej mape. Práve pre ich jednoduchosť je vhodné začať demonštráciu vytvárania herných objektov práve s nimi.



Obr. 25: Vytváranie objektu prostredia v 3D Studiu MAX

Objekt je možné vytvoriť ľubovoľnou modelovacou technikou a nie je nutné aby sa skladal iba z jednej časti (mesh), ako to požadujú niektoré iné enginy resp. štartovacie balíky. Po vymodelovaní je možné na objekt naniesť textúru – štandardne pomocou modifikátora UVW Wrap je potrebné určiť mapovanie a pomocou editora materiálov (Obr. 25 vľavo) priradiť textúru. Na objekt je možné naniesť aj viacero textúr, no kvôli efektívnosti renderovania enginom je výhodnejšie v prípade viacerých textúr použiť funkciu Render to texture, čím dôjde k vytvoreniu jedinej textúry. Následne je potrebné objekt vyexportovať vo formáte .X prípadne .FBX, resp. ľubovoľnom inom, avšak v tom prípade je nutné vytvoriť vlastný procesor pre tento formát.

Po vyexportovaní na vhodné miesto (štandardne do jedného z podadresárov adresára Content\Models) je potrebné upovedomiť engine o tom, že si želáme tento nový model zaradiť do skupiny aktívne používaných objektov, čo sa deje z dôvodu načítania iba nevyhnutného počtu modelov, čím sa skracuje čas štartu aplikácie. Spravíme tak prostredníctvom vytvorenia záznamu do XML súboru „Gamedef“ definujúceho hru nachádzajúceho sa v adresári „Content\Gamedef“. XML záznam sa vkladá do sekcie „Models“ a má tvar:

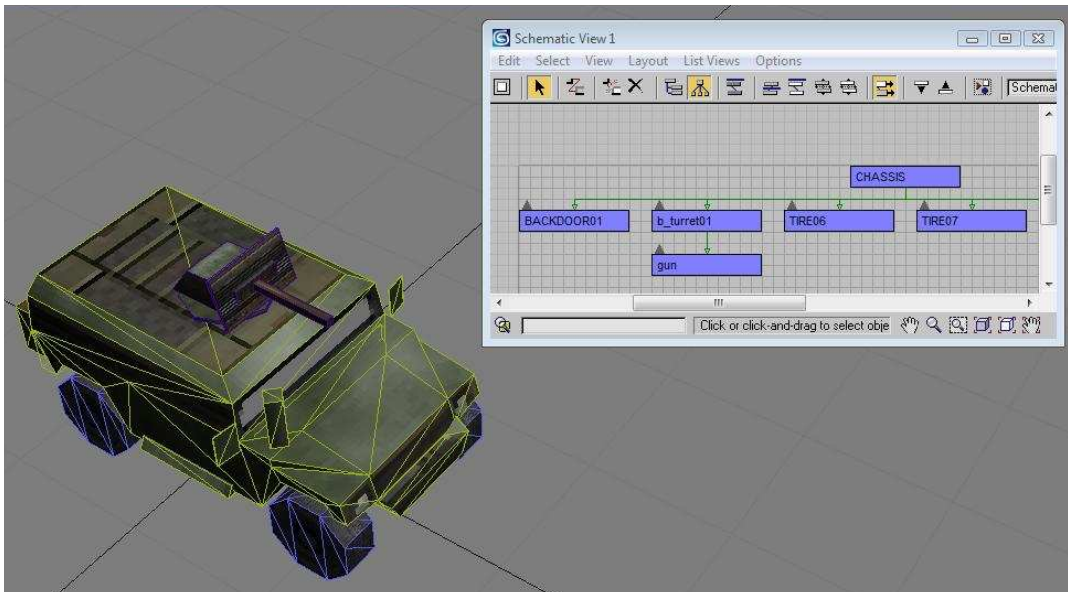
```
<ModelInfo>
    <Path> models/environment/cactus</Path>
    <StarTemplate>
        <X>5</X> <Y>5</Y>
    </StarTemplate>
    <Method>BASIC</Method>
</ModelInfo>
```

Path predstavuje relatívnu cestu k súboru s modelom, StarTemplate veľkosť priestoru okolo modelu cez ktorý A* algoritmus nebude plánovať presun jednotiek (je to efektívnejšie ako počas navigácie zisťovať kolízie s objektmi prostredia) a Method určuje že objekt bude vyrenderovaný základným shaderom. Takto pripravený a enginom rozpoznaný objekt je možné umiestniť na mapu dvojakým spôsobom – buď prostredníctvom herného editora, alebo ručným vložením XML záznamu do súboru definujúceho prostredie na hernej mape „Content/Maps/MENO_MAPY/Map.dat“ v sekcii „Props“, napríklad:

```
<SerialEnviro>
    <Gid>CACTUS</Gid>
    <Position>
        <X>-52</X> <Y>2.82051277</Y> <Z>-49</Z>
    </Position>
    <Rotation>
        <X>0</X> <Y>1</Y> <Z>0</Z>
    </Rotation>
</SerialEnviro>
```

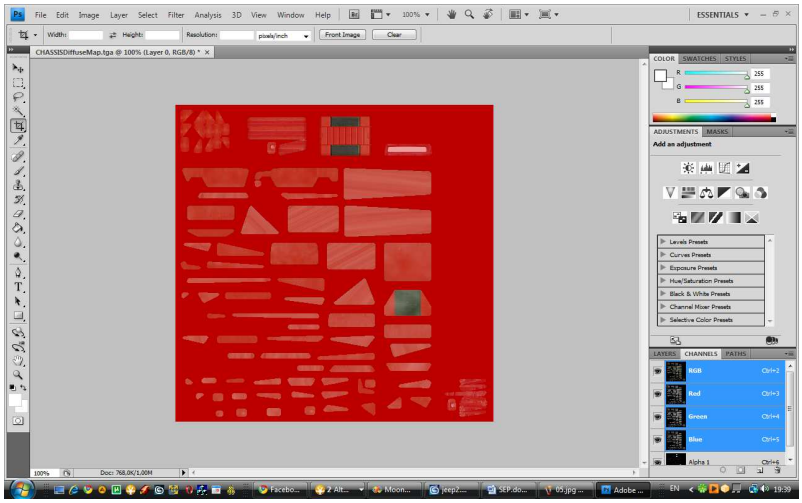
6.2 Vytvorenie ovládateľného objektu

Vytvorenie dynamického, rozkazy prijímajúceho, no do istej miery aj autonómne sa správajúceho objektu je len o niečo zložitejšie ako vytvorenie statického objektu. Prvotná úloha – vymodelovanie a otextúrovanie objektu je zhodná s predchádzajúcim objektom. Rozdielom je, že pokiaľ potrebujeme aby sa niektoré časti objektu otáčali alebo posúvali nezávisle na základni objektu, je nutné ich vymodelovať ako samostatné objekty a spojiť kosťou použitím objektu Bone a operácie Select and Link v 3D Studiu. Takto vznikne hierarchia závislostí. Jedna taká je znázornená na obrázku 26. Základom je karoséria, na ňu sú kosťami napojené kolesá, zadná rezerva a delo na streche. Vďake tomu bude neskôr možné otáčať delom na streche nezávisle od smeru pohybu auta. Pokiaľ ide o objekt ktorý hráč ovláda priamo, je dobré ak nesie tímovú farbu, čím môže byť hráčmi okamžite vizuálne identifikovaný ako priateľský alebo nepriateľský. Aby shader dokázal rozoznať ktoré miesta na textúre sú určené pre tímovú farbu, je potrebné tieto plochy vyznačiť na textúre pomocou alfa kanála, ako to znázorňuje obr. 27 – nečervené plochy budú nahradené na tímovou farbou.



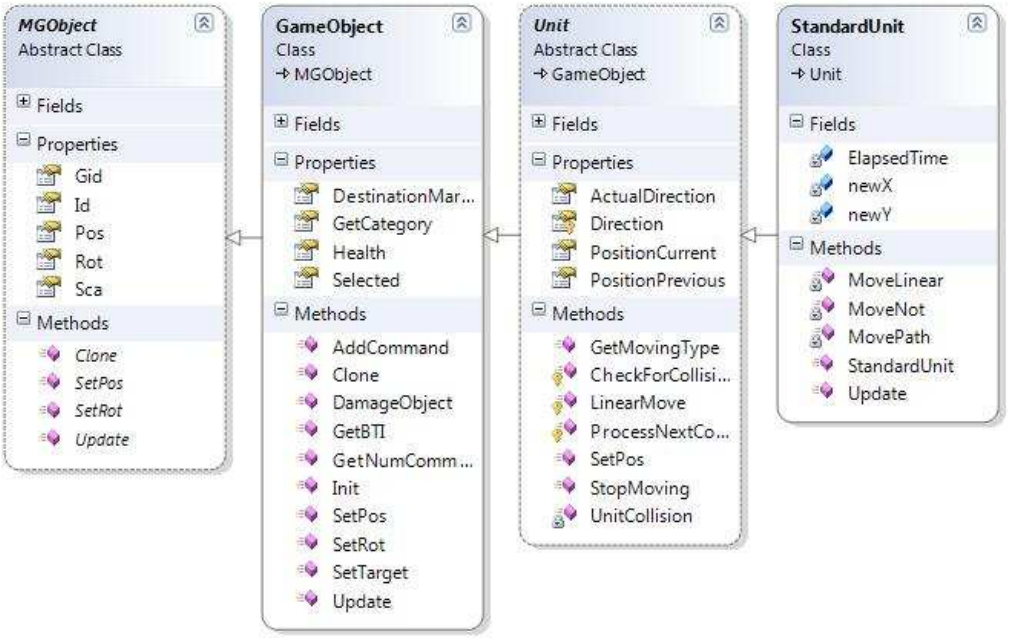
Obr. 26: Spájanie častí objektu pomocou kostí v 3D Studiu MAX

Následné vyexportovanie modelu je rovnaké ako v predchádzajúcom prípade. Rozdielne je však upovedomenie enginu o novom objekte. V tomto prípade nestačí iba zaregistrovať model ako taký, keďže sa jedná o dynamický objekt, ktorý musí reagovať na okolie a akcie vyvolané inými objektmi (napríklad streľba na tento objekt v dôsledku ktorej sa objekt poškodzuje).



Obr. 27: Vymedzenie oblastí pre tímové farby pomocou alfa kanála textúry

Dynamické objekty sú tvorené jednak samotným modelom ktorý je renderovaný, ako aj inštanciou triedy ktorá obsahuje jeho správanie sa (napr. inštancia triedy `StandardUnit`, ktorá dokáže reagovať na príkazy, reagovať na kolízie a pohybovať sa po teréne, viď Obr. 28), ako aj inštanciami tried manipulujúcimi jednotlivými kosťami objektu.



Obr. 28: Triedy obsahujúce logiku jednotiek

V tomto prípade požadujme od tejto jednotky aby sa chovala ako klasická pozemná jednotka, dokázala nájsť cestu terénom, reagovala na kolízie, a dokázala útočiť buď na nepriateľa okamžite ako sa dostane na dostrel, alebo v prípade že cieľ vyberie hráč dokázala ho prenasledovať. Keďže ide

o ľahkú prieskumnú jednotku bude mať slabé pancierovanie, vysokú rýchlosť a bude útočiť strešným delom s vysokou kadenciou.

Rovnako ako v predchádzajúcom príklade v súbore „Gamedef“ v sekcii „Models“ vytvoríme záznam o modeli ktorý pre túto jednotku použijeme:

```
<ModelInfo>
    <Path>models/units/jeep/jeep</Path>
    <Bones>
        <String>b_turret01</String>
    </Bones>
    <StarTemplate>
        <X>2</X> <Y>2</Y>
    </StarTemplate>
    <Method>RCOLOR</Method>
</ModelInfo>
```

Rozdiel je v pridaní zoznamu kostí ktorými budeme chcieť manipulovať (v tomto prípade ide o jednu kosť – b_turret01 otáčajúcu delom) a v zmene renderovacej metódy – použijeme RCOLOR (Replacement Color), ktorá pomocou shaderu pridá na miesta v textúre tímové farby. Ďalej je potrebné do sekcie „Prototypes“ vložiť informácie o tejto novej jednotke, aby sme ju dokázali inštancovať:

```
<ProtoInfo>
    <Key>JEEP</Key>
    <Classname>StandardUnit</Classname>
    <Category>GROUND</Category>
    <Gid>JEEP</Gid>
    <Hp>600</Hp>
    <Sturn>0.03</Sturn>
    <Smove>9</Smove>
    <Los>25</Los>
</ProtoInfo>
```

Jednotlivé položky reprezentujú „herné“ vlastnosti objektu – ide o pozemnú jednotku (Category), ktorej správanie sa je obsiahnuté v triede `StandardUnit`, odolnosť je meraná zásahovými bodmi (HitPoints - Hp), rýchlosť pohybu a otáčania sa je definovaná položkami Sturn a Smove, dohľad zase položkou Los (Line of Sight). Napokon je potrebné určiť ktorý ovládač bude manipulovať delom a aké náboje bude používať. To zaistíme záznamom v sekcii „Animations“:

```
<AnimInfo>
        <Key>JEEP</Key>
        <Controllers>
            <Controller>
                    <ClassName>TANK</ClassName>
                    <At>GROUND</At>
                    <FireDst>80</FireDst>
                    <ReloadT>0.12</ReloadT>
                    <TurnS>0.03</TurnS>
                    <Bullet>
                            <Bt>BULLET</Bt>
                            <Gid>RAPIDBULLET</Gid>
                            <Sound>MachineGun</Sound>
                            <MoveSpeed>130</MoveSpeed>
                            <Damage>5</Damage>
                            <Radius>3</Radius>
                    </Bullet>
                </Controller>
        </Controllers>
</AnimInfo>
```

Takto definovaný objekt je možné inštancovať a teda použiť v hernom prostredí. Vložiť ho do herného prostredia je možné buď pomocou editora, ručne XML záznamom ako v predchádzajúcej ukážke, alebo odkiaľkoľvek z kódu pomocou triedy `CMDGateway` spracovávajúcej príkazy:

```
CMDGateway.AddCommand( new BuildCommand(PlayerID, "JEEP", new Vector2(50,50)) );
```

7 Testovanie výkonnosti

Táto kapitola popisuje spôsob merania výkonnosti navrhnutej a implementovanej aplikácie. V úvode sa venuje podrobnému popisu konfigurácií na ktorých bola aplikácia testovaná a popisu scény. Následne sa venuje samotným testom, ktorými sú meranie počtu zobrazených snímkov za sekundu, priemerného času potrebného na vykonanie daného počtu najčastejšie používaných operácií a zistenie množstva času potrebného na spustenie aplikácie. Záver kapitoly je venovaný údajom o aplikácii ktoré poskytuje nástroj XNA Framework Remote Performance Monitor for Xbox 360 a celkovému zhodnoteniu nameraných údajov.

7.1 Podmienky testovania

Výkon aplikácie bol testovaný na dvoch PC a na konzole Xbox 360. Jednotlivé konfigurácie sa líšili jednak procesormi – Intel Core Duo vs. Intel Core 2 Duo vs. IBM Power PC, ako aj grafickými kartami – od lowendovej mobilnej GeForce Go 7300 cez Radeon HD 3450 po profesionálnu grafickú kartu nachádzajúcu sa v Xboxe 360. Čo sa týka použitých operačných systémov, pri testoch boli použité 32 bitové a 64 bitové Windows Vista Business a operačný systém Xboxu 360. Detaily konfigurácií PC sa nachádzajú v tabuľke 6, prehľad konfigurácie Xboxu 360 sa nachádza v tabuľke 1.

	Konfigurácia 1	**Konfigurácia 2**
CPU	Intel(R) Core(TM) Duo CPU T2450 @ 2.00GHz Cores per Processor : 2 Unit(s) Threads per Core : 1 Unit(s) Type : Mobile, Dual-Core	Intel(R) Core(TM)2 Duo CPU T5550 @ 1.83GHz Cores per Processor : 2 Unit(s) Threads per Core : 1 Unit(s) Type : Mobile, Dual-Core
RAM	3GB DDR2 SO-DIMM	4GB DDR2 SO-DIMM
VGA	NVIDIA GeForce Go 7300 256MB DDR2 PCIe 1.00 x16, SM4.0	ATI RADEON HD 3450 256MB DDR2 PCIe 2.00 x16, SM4.1
HDD	TOSHIBA MK1637GSX 160GB SATA150, 2.5", 5400rpm	TOSHIBA MK2046GSX 200GB SATA150, 2.5", 5400rpm
OS	Microsoft Windows Vista Business 6.00.6001 SP 1 x86	Microsoft Windows Vista Business 6.00.6001 SP 1 x64

Tab. 6: Prehľad konfigurácií počítačov na ktorých bola aplikácia testovaná

Scéna na ktorej boli vykonané testy bola založená na výškovej mape s rozmermi 256x256 pixelov, povrch bol tvorený zmiešavaním štyroch základných textúr a štyroch bump textúr s rozmermi 512x512 pixelov. Mapa obsahovala 45 objektov prostredia, ktoré sa skladali v priemere z 300 polygónov a dvoch hráčov, pričom každý hráč mal k dispozícii desať jednotiek ktoré sa skladali v priemere zo 750 polygónov.

7.2 Výsledky

Nasledujúce štyri podkapitoly sú venované jednotlivým metódam merania výkonnosti aplikácie. Predstavujú komplexný prístup k meraniu výkonnosti – venujú sa nielen meraniu počtu zobrazených snímkov za sekundu, ale i porovnaniu dĺžky trvania najčastejšie vykonávaných operácií a porovnaniu množstva času ktoré uplynie kým sa hra spustí a metódy v ktorých aplikácia pri štarte trávi najviac času.

7.2.1 Zobrazené snímky za sekundu

Počet zobrazených snímkov za sekundu je jedným z najdôležitejších ukazovateľov výkonnosti aplikácie. Na meranie počtu zobrazených snímkov bol použitý vlastný herný komponent. Pred začatím merania bolo potrebné vypnúť synchronizáciu, ktorá zabraňuje aplikácii zobrazovať viac ako 60 snímkov za sekundu:

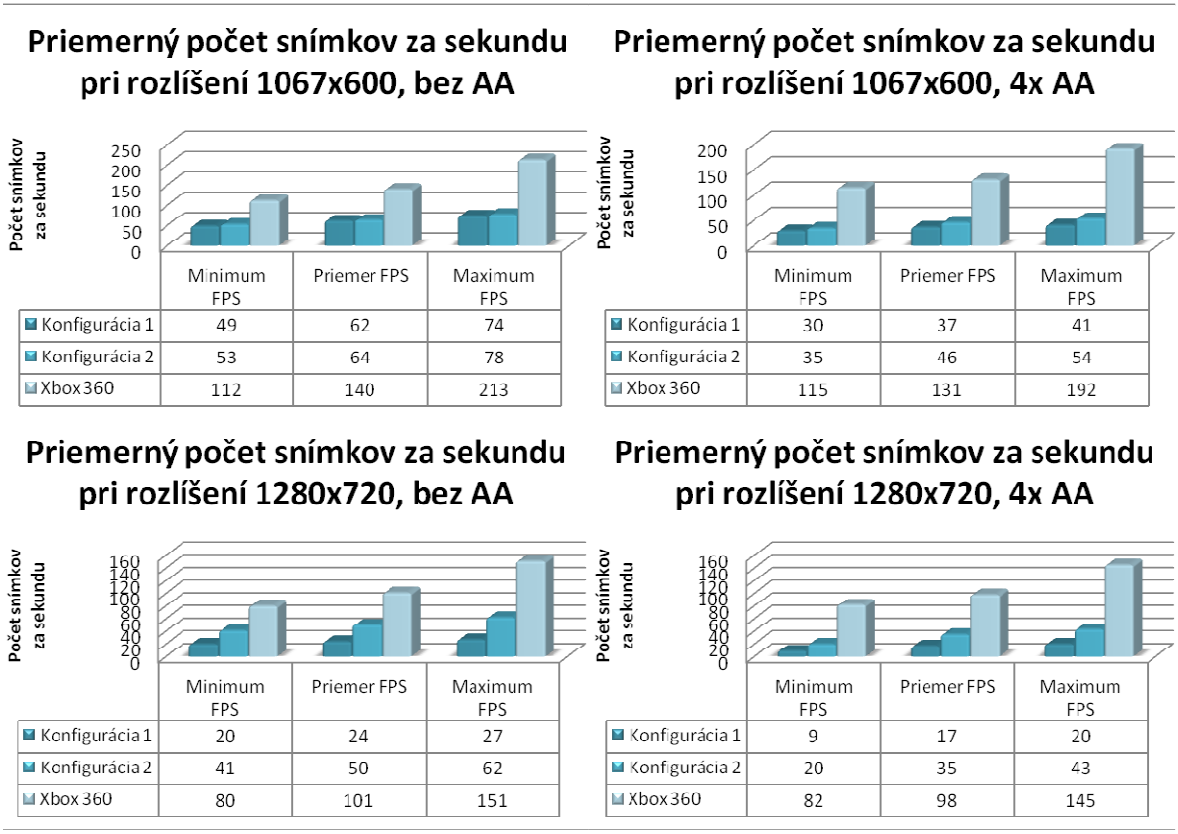
```
graphics.SynchronizeWithVerticalRetrace = false;
IsFixedTimeStep = false;
```

Samotné meranie prebiehalo v Update metóde komponentu:

```
public override void Update(GameTime gameTime)
{
    elapsedTime += gameTime.ElapsedGameTime;

    if (elapsedTime > TimeSpan.FromSeconds(1))
    {
        elapsedTime -= TimeSpan.FromSeconds(1);
        frameRate = frameCounter;
        frameCounter = 0;
    }
}
```

Meranie bolo vykonávané na spomínaných troch konfiguráciách pri rozlíšeniach 1067x600 a 1280x720, a to ako pri zapnutom štvornásobnom vyhladzovaní hrán (AA – antialiasing) tak i pri vypnutom. Výsledky ukazuje obrázok 29.



Obr. 29: Priemerný počet snímkov za sekundu pri danom rozlíšení a nastavení antialiasingu

7.2.2 Vykonávanie najbežnejších operácií

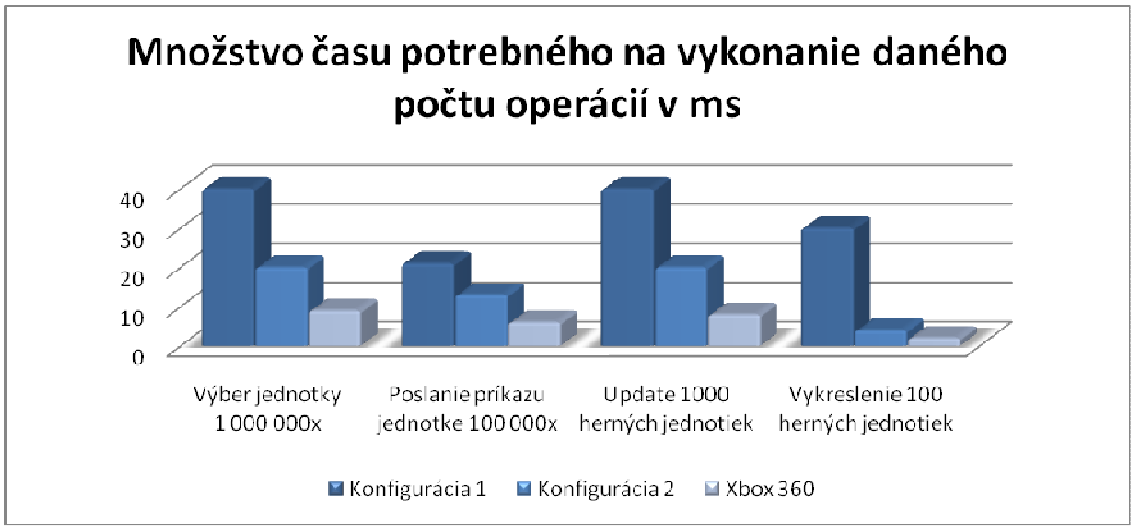
Ďalším testom bolo meranie času potrebného na vykonanie daného počtu operácií. Testované boli často používané operácie:

- **Výber jednotky** – zahŕňa zistenie polohy kurzoru myši, vytvorenie kolízneho lúča a určenie prieniku s kolíznou obálkou niektorej jednotky
- **Poslanie príkazu jednotke** – zahŕňa nájdenie označenej jednotky, určenie správneho príkazu na základe stavu prostredia, zostavenie príkazu a jeho odoslanie
- **Update jednotky** – predstavuje získanie novej polohy jednotky pokiaľ sa pohybuje, prípadné spracovanie príkazov a zmenu polohy všetkých ovládačov kostí danej jednotky
- **Vykreslenie jednotky** – zahŕňa zmenu polohy kostí a vyrenderovanie jednotky

Meranie prebiehalo prostredníctvom štandardného .NET objektu Stopwatch:

```
var stopwatch = System.Diagnostics.Stopwatch.StartNew( );
 // Do something
stopwatch.Stop( );
Console.WriteLine( "Time : {0}", stopwatch.Elapsed );
```

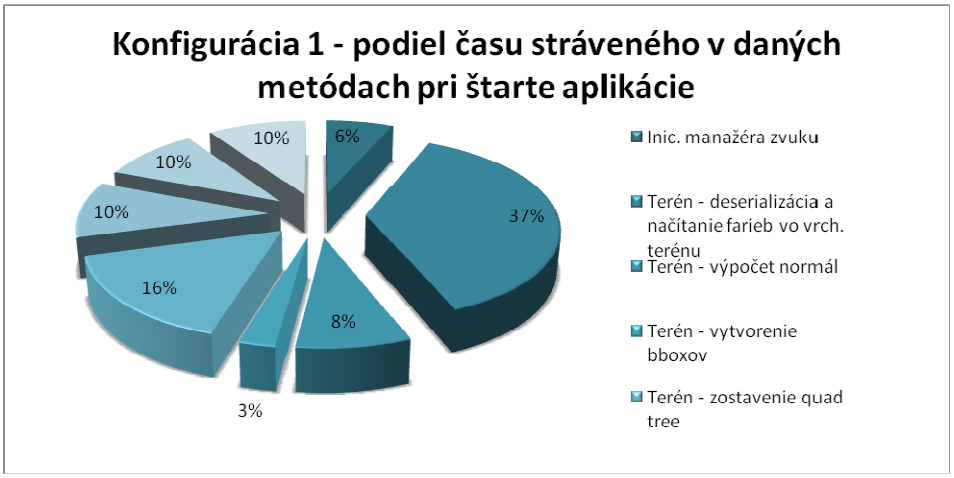
Výsledky ukazuje obrázok 30.



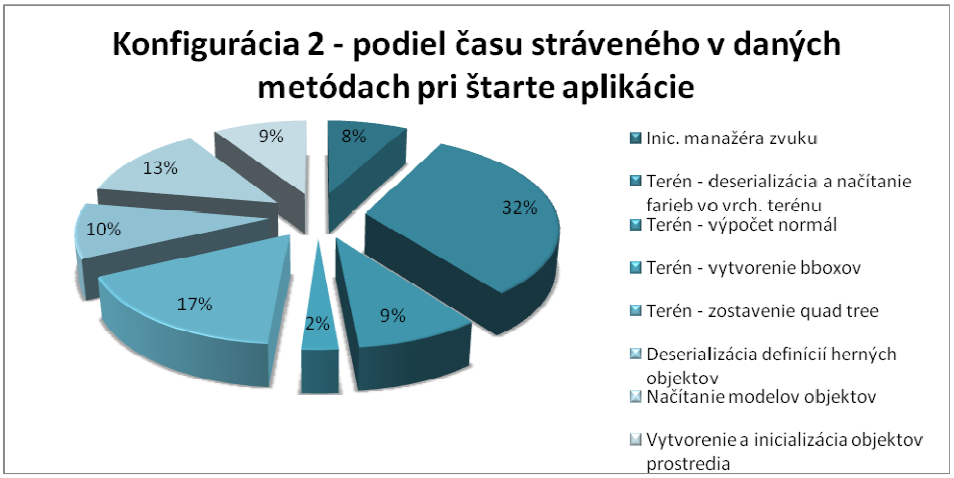
Obr. 30: Množstvo času potrebného na vykonanie daného počtu operácií

7.2.3 Spúšťanie aplikácie

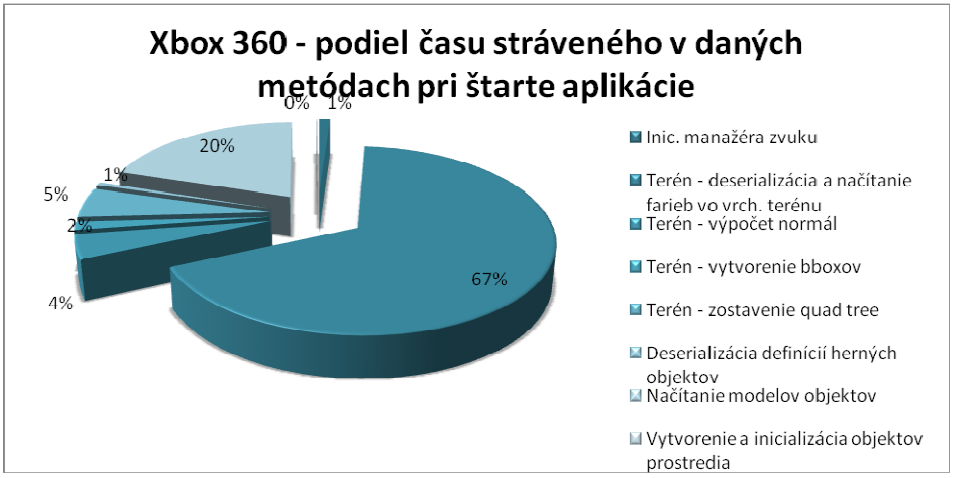
Tretia séria testov bola zameraná na porovnanie výpočtovej sily jednotlivých konfigurácií a rýchlosť načítania dát z pevného disku. Merané bolo množstvo času ktoré uplynie od spustenia aplikácie po začatie hry, tzn. po zobrazenie scény a prijímanie vstupu od hráča. Zisťované bolo celkové množstvo času ktoré uplynie kým sa hra spustí a metódy v ktorých aplikácia pri štarte trávi najviac času. Obrázky 31-33 ukazujú percentuálny podiel jednotlivých metód na celkovom čase spúšťania, obrázok 34 porovnáva jednotlivé časy spúšťania.



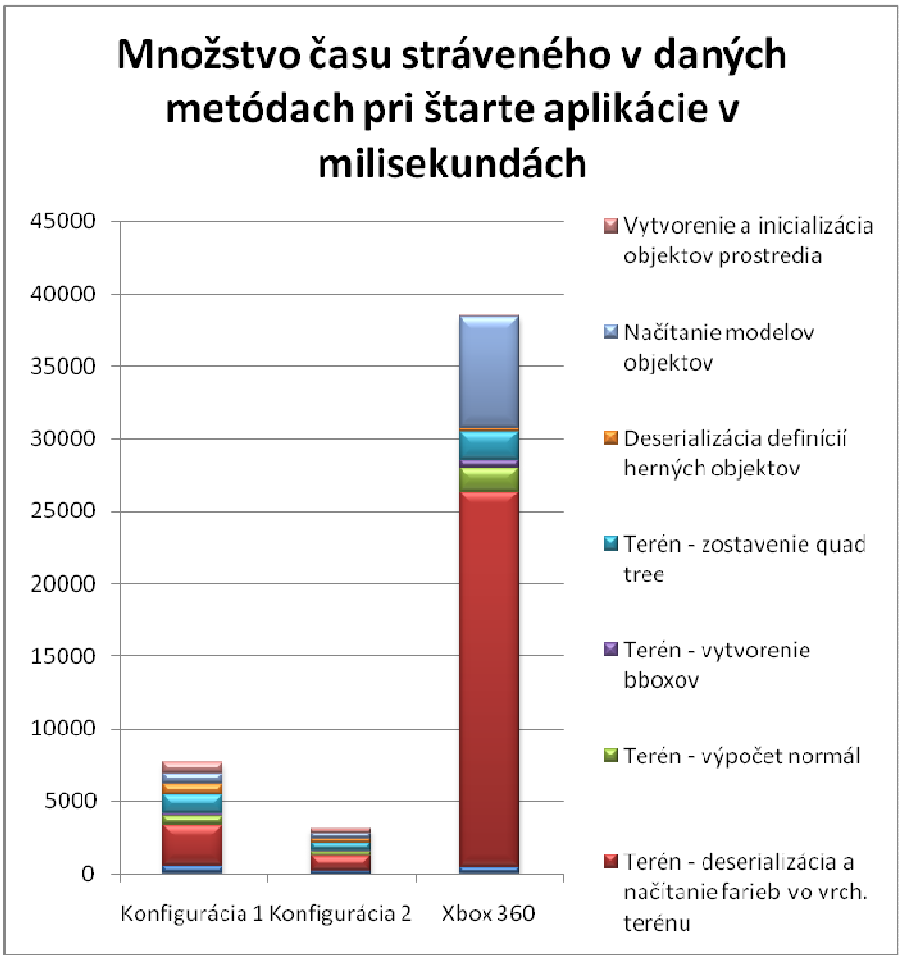
Obr. 31: Konfigurácia 1 – podiel času stráveného v daných metódach pri štarte aplikácie



Obr. 32: Konfigurácia 2 – podiel času stráveného v daných metódach pri štarte aplikácie



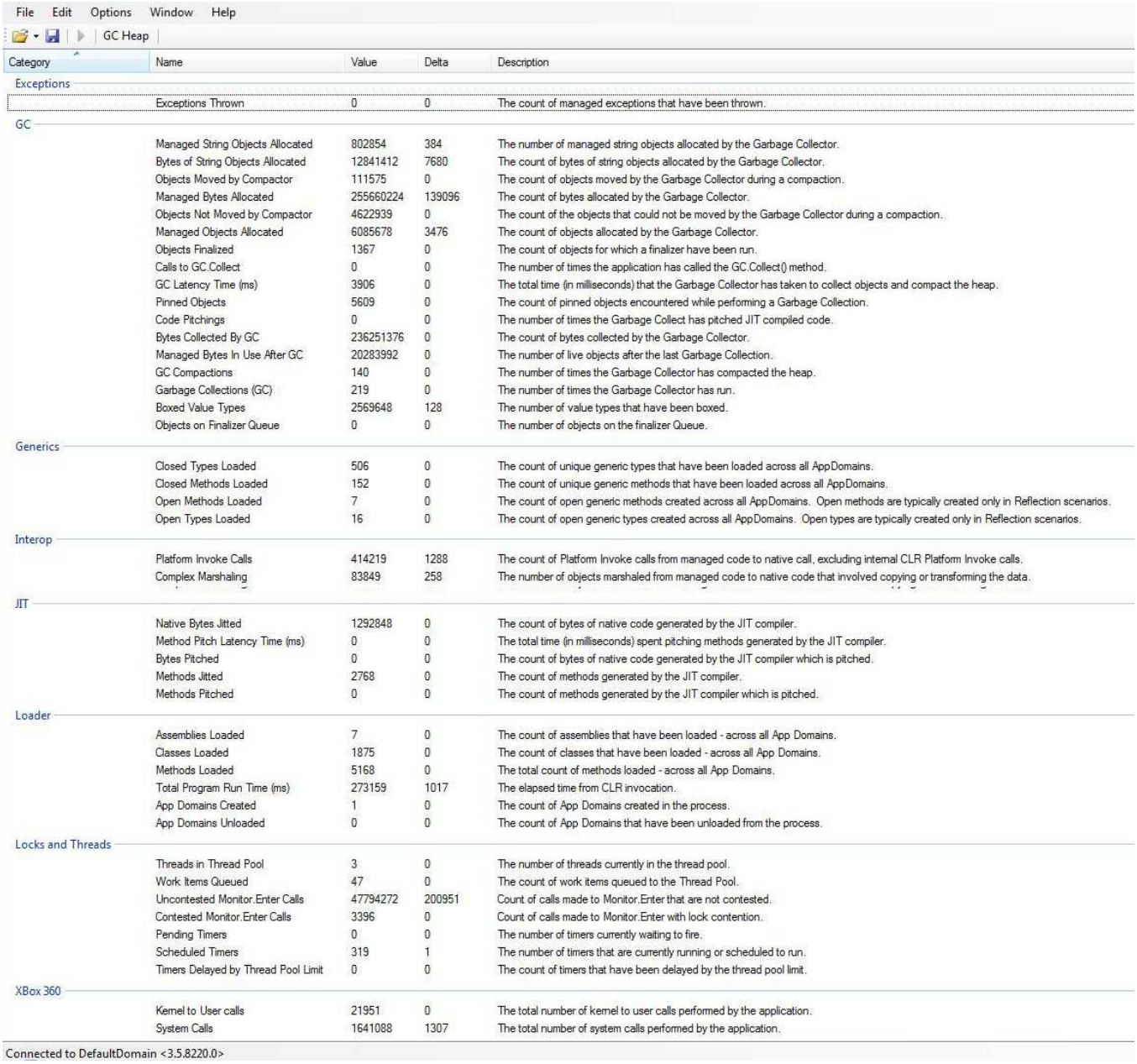
Obr. 33: Xbox 360 – podiel času stráveného v daných metódach pri štarte aplikácie



Obr. 34: Množstvo času stráveného v daných metódach pri štarte aplikácie v milisekundách

7.2.4 XNA Framework Remote Performance Monitor for Xbox 360 - výsledky

Ako už bolo spomenuté v kapitole 3.1.1 XNA Framework Remote Performance Monitor for Xbox 360 je jediným nástrojom na platforme Xbox 360 slúžiacim na analýzu výkonnosti aplikácie v reálnom čase. Poskytuje používateľské rozhranie zobrazujúce namerané hodnoty, ktoré sú získavané počas vykonávania aplikácie priamo z Common Language Runtime (CLR). Obrázok 35 zachytáva výstup z tohto programu pri spustenej aplikácii.



Category	Name	Value	Delta	Description
Exceptions				
	Exceptions Thrown	0	0	The count of managed exceptions that have been thrown.
GC				
	Managed String Objects Allocated	802854	384	The number of managed string objects allocated by the Garbage Collector.
	Bytes of String Objects Allocated	12841412	7680	The count of bytes of string objects allocated by the Garbage Collector.
	Objects Moved by Compactor	111575	0	The count of objects moved by the Garbage Collector during a compaction.
	Managed Bytes Allocated	255660224	139096	The count of bytes allocated by the Garbage Collector.
	Objects Not Moved by Compactor	4622939	0	The count of the objects that could not be moved by the Garbage Collector during a compaction.
	Managed Objects Allocated	6085678	3476	The count of objects allocated by the Garbage Collector.
	Objects Finalized	1367	0	The count of objects for which a finalizer have been run.
	Calls to GC.Collect	0	0	The number of times the application has called the GC.Collect() method.
	GC Latency Time (ms)	3906	0	The total time (in milliseconds) that the Garbage Collector has taken to collect objects and compact the heap.
	Pinned Objects	5609	0	The count of pinned objects encountered while performing a Garbage Collection.
	Code Pitchings	0	0	The number of times the Garbage Collect has pitched JIT compiled code.
	Bytes Collected By GC	236251376	0	The count of bytes collected by the Garbage Collector.
	Managed Bytes In Use After GC	20283992	0	The number of live objects after the last Garbage Collection.
	GC Compactions	140	0	The number of times the Garbage Collector has compacted the heap.
	Garbage Collections (GC)	219	0	The number of times the Garbage Collector has run.
	Boxed Value Types	2569648	128	The number of value types that have been boxed.
	Objects on Finalizer Queue	0	0	The number of objects on the finalizer Queue.
Generics				
	Closed Types Loaded	506	0	The count of unique generic types that have been loaded across all AppDomains.
	Closed Methods Loaded	152	0	The count of unique generic methods that have been loaded across all AppDomains.
	Open Methods Loaded	7	0	The count of open generic methods created across all AppDomains. Open methods are typically created only in Reflection scenarios.
	Open Types Loaded	16	0	The count of open generic types created across all AppDomains. Open types are typically created only in Reflection scenarios.
Interop				
	Platform Invoke Calls	414219	1288	The count of Platform Invoke calls from managed code to native call, excluding internal CLR Platform Invoke calls.
	Complex Marshaling	83849	258	The number of objects marshaled from managed code to native code that involved copying or transforming the data.
JIT				
	Native Bytes Jitted	1292848	0	The count of bytes of native code generated by the JIT compiler.
	Method Pitch Latency Time (ms)	0	0	The total time (in milliseconds) spent pitching methods generated by the JIT compiler.
	Bytes Pitched	0	0	The count of bytes of native code generated by the JIT compiler which is pitched.
	Methods Jitted	2768	0	The count of methods generated by the JIT compiler.
	Methods Pitched	0	0	The count of methods generated by the JIT compiler which is pitched.
Loader				
	Assemblies Loaded	7	0	The count of assemblies that have been loaded - across all App Domains.
	Classes Loaded	1875	0	The count of classes that have been loaded - across all App Domains.
	Methods Loaded	5168	0	The total count of methods loaded - across all App Domains.
	Total Program Run Time (ms)	273159	1017	The elapsed time from CLR invocation.
	App Domains Created	1	0	The count of App Domains created in the process.
	App Domains Unloaded	0	0	The count of App Domains that have been unloaded from the process.
Locks and Threads				
	Threads in Thread Pool	3	0	The number of threads currently in the thread pool.
	Work Items Queued	47	0	The count of work items queued to the Thread Pool.
	Uncontested Monitor.Enter Calls	47794272	200951	Count of calls made to Monitor.Enter that are not contested.
	Contested Monitor.Enter Calls	3396	0	Count of calls made to Monitor.Enter with lock contention.
	Pending Timers	0	0	The number of timers currently waiting to fire.
	Scheduled Timers	319	1	The number of timers that are currently running or scheduled to run.
	Timers Delayed by Thread Pool Limit	0	0	The count of timers that have been delayed by the thread pool limit.
XBox 360				
	Kernel to User calls	21951	0	The total number of kernel to user calls performed by the application.
	System Calls	1641088	1307	The total number of system calls performed by the application.

Obr. 35: Výstup programu XNA Framework Remote Performance Monitor for Xbox 360

7.3 Zhodnotenie

Možno povedať, že výsledky testov dopadli nad očakávania - pri návrhu aplikácie bola stanovená podmienka, že priemerný počet snímkov za sekundu nesmie pri Xbox 360 verzii aplikácie klesať pod hranicu 30 snímkov za sekundu. Túto podmienku sa podarilo splniť. Testy ukázali, že pri oboch PC konfiguráciách sa priemerný počet snímkov pohyboval na úrovni 60 snímkov za sekundu pri vypnutom antialiasingu a okolo 40 snímkov za sekundu pri zapnutom antialiasingu. Pri Xbox 360 verzii sa priemerný počet snímkov za sekundu dokonca pohyboval na dvojnásobnej úrovni. Testy jasne ukázali veľkú výhodu architektúry Xboxu 360, vďaka ktorej použitie antialiasingu neovplyvňuje negatívne výkonnosť aplikácie. Táto vlastnosť predstavuje značný vizuálny a výkonnostný náskok najmä pri porovnávaní s menej výkonnými grafickými kartami, pre ktoré antialiasing predstavuje značnú záťaž.

Hoci pri použití manažovaného kódu na Xboxe 360 býva výkonnosť aplikácií obmedzovaná zvyčajne procesorom a nie GPU, testy preukázali, že ani vykonávanie pomerne komplexných výpočtov nie je pre Xbox 360 problémom. O to prekvapivejšie výsledky prinieslo porovnávanie času ktoré uplynie od spustenia aplikácie po začatie hry, tzn. po zobrazenie scény a prijímanie vstupu od hráča. Zisťované bolo celkové množstvo času ktoré uplynie kým sa hra spustí a metódy v ktorých aplikácia pri štarte trávi najviac času. Aplikácia na Xboxe 360 mala niekoľkokrát horší čas a to najmä vinou metód pristupujúcich k pevnému disku. Ukázalo sa, že Xbox 360 verzia pri načítavaní aplikácie trávila až 67% času pri XML deserializácii veľkého množstva hodnôt (farby vo vrcholoch terénu) a 20% času pri načítavaní modelov. Z toho vyplýva jasné odporúčanie pre XNA aplikácie určené pre Xbox 360 – predpočítavanie komplexných objektov (napr. terén), načítavanie len minimálneho množstva modelov – teda len modelov a textúr ktoré sú použité v aktuálnej mape a XML serializácia len minimálneho množstva objektov.

Oproti tomu výsledky získané z nástroja XNA Framework Remote Performance Monitor for Xbox 360 ukazujú, že použitie rôznych optimalizačných metód v aplikácii (napr. pooling, efektívna hierarchia objektov a pod.) vedú k vynikajúcim výsledkom v oblasti tradične zodpovednej za značné znižovanie výkonu aplikácie, ktorou je uvoľňovanie nepotrebných objektov z pamäte. Aplikácia trávila iba 5,61% času prácou s pamäťou – zberač sa spúšťal v priemere iba raz za sedem sekúnd a pracoval dostatočne rýchlo, takže nedochádzalo ku kolísaniu rýchlosti aplikácie a hra bola plynulá aj pri náročnejších scénach.

8 Záver

Úvod práce predstavil Xbox 360 ako modernú a veľmi výkonnú hernú konzolu a technológiu XNA, ktorá predstavuje moderný prístup k vývoju hier nielen na amatérskej, ale i profesionálnej úrovni. Okrem histórie XNA boli predstavené základné vysokoúrovňové časti XNA Frameworku, objasnené základné princípy vytvárania aplikácií s využitím XNA a identifikované hlavné zdroje straty výkonnosti XNA aplikácií na platforme Xbox 360.

Ďalej je venovaná pozornosť problematike vývoja štartovacích balíkov, pričom nechýba prehľad existujúcich štartovacích balíkov pre 3D XNA hry, s dôrazom na ich dôležitosť pri zoznamovaní sa vývojárov s XNA. Okrem samotných štartovacích balíkov bol čitateľovi poskytnutý podrobný prehľad existujúcich nástrojov analyzujúcich výkonnosť implementovanej aplikácie a nástrojov na vytváranie, úpravu a konvertovanie herného obsahu.

Druhá polovica práce je venovaná návrhu, implementácii a testovaniu prototypu systému, štartovaciemu balíku venovaného strategickej hre prebiehajúcej v reálnom čase (RTS stratégii). Po úvodnej formulácii hlavných cieľov práce, identifikovaní hlavných nedostatkov existujúcich riešení a navrhnutí spôsobov ich prekonania nasleduje popis funkcionálnych a nefunkcionálnych požiadaviek kladených na systém a návrh prototypu štartovacieho balíka. Na základe tohto návrhu bol prototyp štartovacieho balíka implementovaný a následne bola jeho výkonnosť testovaná. Testami sa preukázalo, že vytvorený štartovací balík je vysoko výkonný, počet zobrazovaných snímkov za sekundu pri stredne veľkej náročnej scéne sa pohyboval okolo hodnoty 60 snímkov za sekundu pri testovaní na PC konfiguráciách a na viac ako dvojnásobnej hodnote pri testoch na Xboxe 360. Ako veľmi účinné sa ukázali byť implementované optimalizácie pri vytváraní objektov, vďaka čomu garbage collector pracoval dostatočne rýchlo, takže nedochádzalo ku kolísaniu rýchlosti aplikácie a hra bola plynulá aj pri náročnejších scénach.

Práca spĺňa všetky body zadania a je značným prínosom pre XNA komunitu. Navrhnutý štartovací balík je vhodný a dostatočne zrozumiteľný pre začiatočníkov, no má čo ponúknuť i skúseným herným vývojárom. Vďaka jednoduchej konfigurácii pomocou externých XML súborov je umožnené redefinovať hru i bez predchádzajúcej znalosti programovania, podpora mnohých formátov súborov zase umožňuje vytváranie kvalitného herného obsahu používateľom širokého spektra aplikácií zameraných na túto oblasť. V budúcnosti je možné tento štartovací balík rozšíriť pridaním kvalitnej umelej inteligencie riadiacej skupiny jednotiek ako aj naďalej zlepšovať vizuálnu stránku aplikácie pomocou shaderov.

Literatúra

[1] Xbox 360 First to Reach Ten Million Console Sales in U.S. [online].
Aktualizované 2008-11-28 [cit. 2008-11-28]. Dostupné na URL:
< http://www.xbox.com/en-US/community/news/2008/0514-10million.htm>

[2] The Xbox 360 System Specification [online].
Aktualizované 2008-11-27 [cit. 2008-12-30]. Dostupné na URL:
<http://hardware.teamxbox.com/articles/xbox/1144/The-Xbox-360-System-Specifications/p1>

[3] Microsoft Launches New Xbox 360 Console for Families [online].
Aktualizované 2008-10-20 [cit. 2008-12-28]. Dostupné na URL:
<http://www.microsoft.com/Presspass/press/2007/oct07/10-22ArcadeConsolePR.mspx>

[4] Xbox 360 December 2007 Update [online].
Aktualizované 2007-12-11 [cit. 2008-12-28]. Dostupné na URL:
<http://blogs.msdn.com/xboxteam/archive/2007/11/30/december-2007-system-update.aspx>

[5] Wikipedia – New Xbox 360 Dashboard Image [online].
Aktualizované 2008-10-20 [cit. 2008-12-28]. Dostupné na URL:
<http://upload.wikimedia.org/wikipedia/en/d/df/Newdashboard1912.jpg>

[6] Albert Ho's XNA Redux [online].
Aktualizované 2008-08-20 [cit. 2008-12-28]. Dostupné na URL:
<http://blogs.msdn.com/al_msft/archive/2006/03/20/555065.aspx>

[7] Racing Game Starter Kit [online].
Aktualizované 2007-12-11 [cit. 2008-12-28]. Dostupné na URL:
<http://creators.xna.com/en-US/starterkit/racinggame>

[8] exDream [online].
Aktualizované 2007-11-11 [cit. 2008-12-28]. Dostupné na URL:
<http://www.exdream.com/>

[9] NIETSCHKE, Benjamin. *Professional XNA Game Programming: For Xbox 360 and Windows*. 1st edition. [s.l.] : Wrox, 2007. 504 s.

[10] Ship Game [online].
Aktualizované 2007-07-19 [cit. 2008-12-28]. Dostupné na URL:
<http://creators.xna.com/en-US/starterkit/shipgame>

[11] Descent 3 [online].
Aktualizované 2007-10-21 [cit. 2008-12-28]. Dostupné na URL:
<http://www.descent3.com/index.html>

[12] Forsaken [online].
Aktualizované 2006-06-05 [cit. 2008-12-28]. Dostupné na URL: <http://www.forsakenplanet.tk/>
[13] XNA Creator's Club Membership [online].
Aktualizované 2007-12-11 [cit. 2008-12-28]. Dostupné na URL: <http://creators.xna.com/membership>
[14] Robot Game [online].
Aktualizované 2008-07-11 [cit. 2008-12-28]. Dostupné na URL: <http://creators.xna.com/en-US/minigame/robotgame>
[15] Zepetto [online].
Aktualizované 2008-13-08 [cit. 2008-12-28]. Dostupné na URL: <http://www.zepetto.com>
[16] Managed Code Preformance on Xbox 360 for XNA [online].
Aktualizované 2007-12-21 [cit. 2008-12-28]. Dostupné na URL: <http://blogs.msdn.com/netcfteam/archive/2006/12/22/managed-code-performance-on-xbox-360-for-xna-part-2-gc-and-tools.aspx>
[17] Shawn Hargreave's blog [online].
Aktualizované 2007-12-16 [cit. 2008-12-28]. Dostupné na URL: <http://blogs.msdn.com/shawnhar/archive/2007/06/29/how-to-tell-if-your-xbox-garbage-collection-is-too-slow.aspx>
[18] PIX [online].
Aktualizované 2008-10-20 [cit. 2008-12-28]. Dostupné na URL: <http://msdn.microsoft.com/en-us/library/bb173085.aspx>
[19] CLR Profiler [online].
Aktualizované 2008-11-11 [cit. 2008-12-28]. Dostupné na URL: <http://msdn.microsoft.com/en-us/library/ms979205.aspx>
[20] Nvidia Performance HUD [online].
Aktualizované 2008-12-07 [cit. 2008-12-28]. Dostupné na URL: <http://developer.nvidia.com/object/nvperfhud_home.html>
[21] Autodesk Maya [online].
Aktualizované 2008-12-10 [cit. 2008-12-28]. Dostupné na URL: <http://usa.autodesk.com/adsk/servlet/index?id=7635018&siteID=123112>
[22] Autodesk Maya PLE [online].
Aktualizované 2008-12-04 [cit. 2008-12-28]. Dostupné na URL: <http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=7679012#section1>

[23] SoftImage XSI [online].
Aktualizované 2008-11-11 [cit. 2008-12-28]. Dostupné na URL:
<http://www.softimage.com/products/xsi/>
[24] SoftImage ModTool [online].
Aktualizované 2008-08-22 [cit. 2008-12-28]. Dostupné na URL:
<http://www.softimage.com/products/modtool/getmod.aspx>
[25] SoftImage Unleashes XSI 6 ModTool [online].
Aktualizované 2008-12-27 [cit. 2008-12-28]. Dostupné na URL:
<http://www.cgw.com/ME2/dirmod.asp?sid=&nm=&type=news&mod=News&mid=9A02E3B96F2A415ABC72CB5F516B4C10&tier=3&nid=9E3D5862A46B4474A6FF3831561F518D>
[26] XNA Integration in XSI [online].
Aktualizované 2008-12-09 [cit. 2008-12-28]. Dostupné na URL:
<http://www.softimage.com/products/modtool/pdf/XNA_GSE_integration_in_XSI.pdf>
[27] Blender [online].
Aktualizované 2008-12-03 [cit. 2008-12-28]. Dostupné na URL:
<http://www.blender.org/community/forums/>
[28] Okino Polytrans [online].
Aktualizované 2008-09-13 [cit. 2008-12-28]. Dostupné na URL:
<http://www.okino.com/conv/conv.htm>
[29] Moving Data Between Maya and 3D Studio Max [online].
Aktualizované 2008-10-18 [cit. 2008-12-28]. Dostupné na URL:
<http://www.okino.com/conv/pt4maya8.htm>
[30] Panda X Exporter [online].
Aktualizované 2007-10-11 [cit. 2008-12-28]. Dostupné na URL:
<http://www.andytather.co.uk/Panda/directxmax_downloads.aspx>
[31] Paint.NET [online].
Aktualizované 2008-12-11 [cit. 2008-12-28]. Dostupné na URL:
<http://www.getpaint.net>
[32] GNU Image Manipulation Program [online].
Aktualizované 2008-12-16 [cit. 2008-12-28]. Dostupné na URL:
<http://gimp.org/>
[33] Corel Paint Shop Pro [online].
Aktualizované 2008-10-11 [cit. 2008-12-28]. Dostupné na URL:
<http://www.corel.com/servlet/Satellite/ca/en/Product/1184951547051#versionTabview=tab0&tabview=tab0>

[34] Adobe Photoshop [online].
Aktualizované 2008-09-14 [cit. 2008-12-28]. Dostupné na URL:
<http://www.adobe.com/products/photoshop/compare/>
[35] Nvidia Photoshop Plugins [online].
Aktualizované 2008-12-20 [cit. 2008-12-28]. Dostupné na URL:
<http://developer.nvidia.com/object/photoshop_dds_plugins.html>
[36] XNA Creator's Club [online].
Aktualizované 2008-11-01 [cit. 2008-12-28]. Dostupné na URL:
<http://creators.xna.com/en-US/>
[37] IGN [online].
Aktualizované 2008-12-14 [cit. 2008-12-28]. Dostupné na URL:
<http://www.ign.com/>

Zoznam príloh

Príloha 1. Obsah elektronického média

Príloha 2. Shader použitý pri zobrazovaní terénu

Príloha 3. Popisom objektov prostredia vo formáte XML

Príloha 1. Obsah elektronického média

Súčasťou práce je aj elektronické médium na ktorom sa nachádza nasledujúci obsah:

/ Dokumenty	
DP-Kajan.doc	Tento dokument vo formáte MS Word
DP-Kajan.pdf	Tento dokument vo formáte PDF
/ MoonGate-Binary	Spustiteľná verzia projektu. Pred spustením je potrebné nainštalovať XNA Game Studio 3.0, ktoré sa nachádza v adresári MoonGate-Prereq
/ MoonGate-Source	Adresár so zdrojovými kódmi projektu vrátane projektu do Visual Studia 2008
/ MoonGate-Prereq	Inštalátor XNA Game Studia 3.0
/ Poster	Poster

Príloha 2. Shader použitý pri zobrazovaní terénu

```
float4x4 wvp : WorldViewProjection;
float4x4 world : World;
float3 LightPosition : LightDirection;
float3 EyePosition : CAMERAPOSITION;

// Texture size modifier 1 = no change.
float tileSizeMod = 1;

// Terrain Textures.
texture  LayerMap0;
texture  LayerMap1;
texture  LayerMap2;
texture  LayerMap3;

// Terrain Normals for above texture.
texture BumpMap0;
texture BumpMap1;
texture BumpMap2;
texture BumpMap3;

// Normal samplers
sampler BumpMap0Sampler = sampler_state
{
	Texture = <BumpMap0>;
	MinFilter = Linear;
	MagFilter = Linear;
	MipFilter = Linear;
	AddressU = mirror;
	AddressV = mirror;
};

sampler BumpMap1Sampler = sampler_state
{
	Texture = <BumpMap1>;
	MinFilter = Linear;
	MagFilter = Linear;
	MipFilter = Linear;
	AddressU = mirror;
	AddressV = mirror;
};

sampler BumpMap2Sampler = sampler_state
{
	Texture = <BumpMap2>;
	MinFilter = Linear;
	MagFilter = Linear;
	MipFilter = Linear;
	AddressU = mirror;
	AddressV = mirror;
};
```

```
sampler BumpMap3Sampler = sampler_state
{
	Texture = <BumpMap3>;
	MinFilter = Linear;
	MagFilter = Linear;
	MipFilter = Linear;
	AddressU = mirror;
	AddressV = mirror;
};

// Texture Samplers
sampler LayerMap0Sampler = sampler_state
{
   Texture   = <LayerMap0>;
   MinFilter = LINEAR;
   MagFilter = LINEAR;
   MipFilter = LINEAR;
   AddressU  = WRAP;
   AddressV  = WRAP;
};

sampler LayerMap1Sampler = sampler_state
{
	Texture   = <LayerMap1>;
	MinFilter = LINEAR;
	MagFilter = LINEAR;
	MipFilter = LINEAR;
	AddressU  = WRAP;
   AddressV  = WRAP;
};

sampler LayerMap2Sampler = sampler_state
{
   Texture   = <LayerMap2>;
   MinFilter = LINEAR;
	MagFilter = LINEAR;
	MipFilter = LINEAR;
	AddressU  = WRAP;
   AddressV  = WRAP;
};
sampler LayerMap3Sampler = sampler_state
{
   Texture   = <LayerMap3>;
   MinFilter = LINEAR;
	MagFilter = LINEAR;
	MipFilter = LINEAR;
	AddressU  = WRAP;
   AddressV  = WRAP;
};

// Vertex Shader input structure.
struct VS_INPUT
{
	float4 posL	: POSITION0;
   float3 normalL	: NORMAL0;
   float4 tiledTexC	: TEXCOORD0;
   float4 TextureWeights : TEXCOORD1;
   float3 Tangent : TANGENT;
};
```

```
// Vertex Shader output structure/Pixel Shaer input structure
struct VS_OUTPUT
{
	float4 posH	: POSITION0;
   float  shade	: TEXCOORD0;
   float4 tiledTexC	: TEXCOORD1;
   float4 TextureWeights : TEXCOORD2;
	float4 Light : TEXCOORD3;
	float3 lView : TEXCOORD4;
};

// Vetex Shader
VS_OUTPUT BumpVS(VS_INPUT input)
{
	// Clean the output structure.
	VS_OUTPUT Out = (VS_OUTPUT)0;

	// Calculate tangent space.
	float3x3 worldToTangentSpace;
	worldToTangentSpace[0] = mul(input.Tangent,world);
	worldToTangentSpace[1] = mul(cross(input.Tangent,input.normalL),world);
	worldToTangentSpace[2] = mul(input.normalL,world);

	// Get world pos for texture and normal.
	float4 PosWorld = mul(input.posL,world);

	// Get light position.
	Out.Light.xzy = LightPosition;
	Out.Light.w = 1;

	// Set position for pixel shader
   Out.posH  = mul(input.posL, wvp);

   // Set lighting.
   Out.shade = saturate(saturate(dot(input.normalL, normalize(LightPosition))));

   // Set view direction for normals.
	Out.lView = mul(worldToTangentSpace,EyePosition - Out.posH);

   // Set tile TexCoord.
   Out.tiledTexC = input.tiledTexC * tileSizeMod;

	// Set Texture weight.
   Out.TextureWeights = input.TextureWeights;

	return Out;
}
// Output to screen.
struct PixelToFrame
{
   float4 Color : COLOR0;
};

// Pixel shader.
PixelToFrame BumpPS(VS_OUTPUT input) : COLOR
{
	// Clean output structure.
	PixelToFrame Out = (PixelToFrame)0;
```

```
        // Get pixel color.
        float4 Col = tex2D(LayerMap0Sampler, input.tiledTexC)*input.TextureWeights.x;
    Col += tex2D(LayerMap1Sampler, input.tiledTexC)*input.TextureWeights.y;
    Col += tex2D(LayerMap2Sampler, input.tiledTexC)*input.TextureWeights.z;
    Col += tex2D(LayerMap3Sampler, input.tiledTexC)*input.TextureWeights.w;

    // Set light directon amd view direction.
    float4 LightDir = normalize(input.Light);
        float3 ViewDir = normalize(input.lView);

        // Get prominent normal.
        //float2 nearTextureCoords = input.tiledTexC;
        float3 Normal;
        Normal = tex2D(BumpMap0Sampler,input.tiledTexC)*input.TextureWeights.x;
        Normal += tex2D(BumpMap1Sampler,input.tiledTexC)*input.TextureWeights.y;
        Normal += tex2D(BumpMap2Sampler,input.tiledTexC)*input.TextureWeights.z;
        Normal += tex2D(BumpMap3Sampler,input.tiledTexC)*input.TextureWeights.w;
        Normal = 2 * Normal - 1.0;

        // Set diffuse, reflection and specular effect for Normal.
        float Diffuse = saturate(dot(Normal,LightDir));

        float Reflect = normalize(2 * Diffuse * Normal - LightDir);
        float Specular = min(pow(saturate(dot(Reflect,ViewDir)),3),Col.w);


        float4 final;

        // Do color calculation depending if bump feature is on or off.
        final = 0.2 * Col * Diffuse * (input.shade * 12);
        //final = (0.2 * Col * (Diffuse + Specular)) * (input.shade * 12);

        Out.Color = final;

        return Out;
}

technique Terrain_MultiTex
{
    pass P0
    {
                    vertexShader = compile vs_2_0 BumpVS();
       pixelShader  = compile ps_2_0 BumpPS();
    }
}
```

Príloha 3. Popis objektov prostredia vo formáte XML

```
<?xml version="1.0"?>
<ArrayOfSerialEnviro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SerialEnviro>
    <gid>BUILDING1</gid>
    <position>
      <X>-30</X>
      <Y>1.97530866</Y>
      <Z>-64</Z>
    </position>
    <rotation>
      <X>0</X>
      <Y>1</Y>
      <Z>0</Z>
    </rotation>
  </SerialEnviro>
  <SerialEnviro>
    <gid>BUILDING1</gid>
    <position>
      <X>-19</X>
      <Y>1.97530866</Y>
      <Z>-86</Z>
    </position>
    <rotation>
      <X>0</X>
      <Y>1</Y>
      <Z>0</Z>
    </rotation>
  </SerialEnviro>
  <SerialEnviro>
    <gid>CONTAINER1</gid>
    <position>
      <X>40</X>
      <Y>1.97530866</Y>
      <Z>55</Z>
    </position>
    <rotation>
      <X>0</X>
      <Y>1</Y>
      <Z>0</Z>
    </rotation>
  </SerialEnviro>
</ArrayOfSerialEnviro>
```