VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

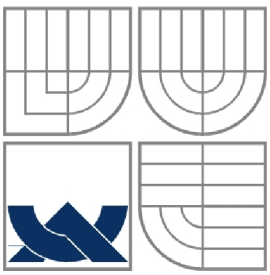
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

JEDNODUCHÝ LETECKÝ SIMULÁTOR NA WINDOWS PHONE 7

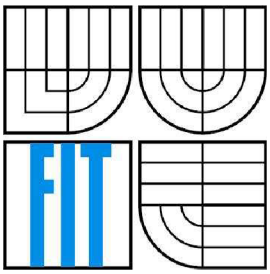
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Marián Hacaj
AUTHOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

JEDNODUCHÝ LETECKÝ SIMULÁTOR NA WINDOWS PHONE 7

SIMPLE FLIGHT SIMULATOR FOR WINDOWS PHONE 7

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Marián Hacaj

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. Adam Herout, Ph.D.

BRNO 2011

Abstrakt

Práce popisuje programování 3D aplikací, zejména her, na platformě Windows Phone 7 a lehce jej porovnává s přístupem platformy Silverlight. V práci je podrobně popsán framework XNA a dále problematika programování leteckých simulátorů. V druhé části práce je kompletně popsaná implementace jednoduchého leteckého simulátoru pro platformu Windows Phone 7, která je založená na frameworku XNA. Celá hra pozůstává z implementace terénu, oblohy, letadla, krajiny ale také fyziky a principu hry.

Abstract

Thesis describes programming of 3D applications, mainly games, on Windows Phone 7 platform and lightly compares this approach with Silverlight platform. It also describes XNA framework in detail and programming plane simulators problems. In the second part of this thesis, reader can find complete description of implementation of a simple airplane simulation for Windows Phone 7 platform, which is based on the XNA framework. The game consists of implementation of terrain, sky, plane, scene and also of physics and the principle of the game.

Klíčová slova

Windows Phone 7, XNA, Silverlight, DirectX, C#, 3D Grafika, Letecký simulátor, Level of detail, LOD, Skybox, Terén, Mobil, Microsoft, Hra, Marketplace.

Keywords

Windows Phone 7, XNA, Silverlight, DirectX, C#, 3D Graphics, Plane simulator, Level of detail, LOD, Skybox, Terrain, Mobile, Microsoft, Game, Marketplace.

.

Citace

Hacaj Marián: Jednoduchý letecký simulátor pre Windows Phone 7, semestrální projekt, Brno, FIT VUT v Brně, 2011

Jednoduchý letecký simulátor pre Windows Phone 7

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. Ing. Adama Herouta, Ph.D. Další informace mi poskytli Andrej Novosad a Martin Polák.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Marián Hacaj
25.5.2011

Poděkování

Chcel by som sa poďakovať svojmu vedúcemu doc. Ing. Adamovi Heroutovi, Ph.D. za venovaný čas pri písaní tejto práce, ale aj za cenné rady, ktoré mi poskytol. Ďalej by som chcel poďakovať Martinovi Polákovi za poskytnutie modelu lietadla do aplikácie a Andrejovi Novosadovi za mnohé nápady a rady. V neposlednej rade chcem poďakovať aj svojej rodine, priateľke, kamarátom, ktorí pri mne po celú dobu stáli a všetkým, ktorí aplikáciu testovali a uľahčovali mi tým prácu.

Obsah

Zoznam obrázkov

1 Úvod

V poslednej dobe sa stále viac rozširuje možnosť programovania hier pre mobilné zariadenia. Hlavnými prenosnými platformami na hry sa stávajú mobilné telefóny. Je to dané tým, že na rozdiel od prenosných konzol ako PSP a pod., má mobilný telefón (*smartphone*) už takmer každý. Terajší vývoj ukazuje na to, že mobilné telefóny by sa hlavne vďaka veľkému rozšíreniu mohli v budúcnosti stať hlavnou platformou pre vývoj aplikácií či hier. K veľkému úspechu mobilného priemyslu prispieva aj konkurenčný boj hlavných výrobcov operačných systémov pre tieto zariadenia (Apple – iOS, Google-Android, Samsung-Bada, Nokia-Symbian a i.) Táto práca poukazuje na možnosť programovania 3D aplikácií, hlavne hier (leteckých simulátorov), na novej platforme od firmy Microsoft – Windows Phone 7.

Práca je rozdelená na 6 hlavných kapitol. Prvou z nich je úvod. V druhej kapitole je podrobne popísaná platforma Windows Phone 7, jej hardwarová konfigurácia, softwarové vybavenie, vývoj a distribúcia aplikácií. Celá práca je zameraná hlavne na framework XNA, ktorý sa používa, mimo iné, na programovanie hier pre Windows, Xbox a Zune. Na Windows Phone 7 je možné vyvíjať aj pomocou frameworku Silverlight. Na konci druhej kapitoly je ľahké porovnanie XNA s týmto frameworkom.

Tretia kapitola je zameraná detailne na framework XNA. Obsahuje históriu, rozdiely medzi stolovou (Xbox a PC) verziou a verziou pre Windows Phone 7 a nakoniec podporu sociálnej hracej siete Xbox Live.

Štvrtá kapitola sa zaoberá problematikou ohľadom budúcnosti platformy Windows Phone 7. Zameriava sa na partnerstvo so spoločnosťou Nokia a taktiež detailne na ohlásené aktualizácie, ktoré by užívateľom mali byť prístupné koncom roka 2011.

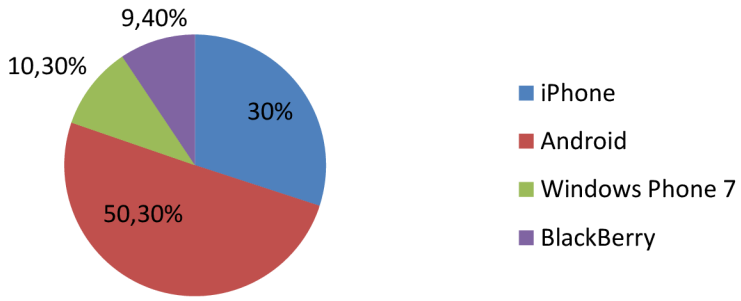
Piata kapitola práce sa zaoberá problematikou programovania leteckých simulátorov. Čitateľ sa v nej dočíta o veciach typických pre programovanie leteckých simulátorov - o algoritmoch zobrazovania terénu (o implementácii Level of Detail), ďalej o zobrazovaní oblohy a nakoniec obsahuje aj ľahký úvod do fyziky lietadla.

Šiesta kapitola je určená veľmi detailnému popisu implementácie jednoduchého leteckého simulátoru pre platformu Windows Phone 7 vo frameworku XNA. Čitateľ sa v nej dozvie niektoré klasické konštrukcie potrebné pre framework XNA a samozrejme náznak, ako by sa mali na platformu Windows Phone 7 vyvíjať 3D aplikácie (hry). Vývoj lietadla je zameraný hlavne na framework XNA, preto sa z tejto kapitoly čitateľ nedozvie napr. o umelej inteligencii. Úrovne lietadla sú náučného charakteru s dôrazom na rozvíjanie pamäte a multitasking myslenia.

2 Windows Phone 7

Windows Phone 7 [1][3][4][11] je nový operačný systém pre mobilné telefóny z dielne Microsoftu. Microsoft ho predstavil začiatkom októbra (11.10.2010) ako úplne nový produkt nenadväzujúci na predchádzajúce verzie Windows Mobile. Firma si pripúšťa, že mierne zaspala v porovnaní s konkurenciou – tak ako aktuálny CEO Microsoft-u, Steve Ballmer povedal: „*Windows Mobile 'screwed up' so far*“. Preto sa rozhodli ísť inou cestou a vytvorili na trhu originálny operačný systém. Vývoj Windows Mobile bol s nástupom Windows Phone 7 ukončený [3].

Čo sa týka stavu telefónu na trhu, v dobe písania tohto textu tento systém ešte len nastupoval, ale očakáva sa, že bude konkurencie schopný operačný systém, hlavne vďaka veľmi intuitívnemu grafickému rozhraniu, ktoré je (podľa renomovaných serverov – *Gizmodo*, *Engaged*...) dokonca lepšie ako rozhranie na *Apple iPhone*. V porovnaní s konkurenciou – *iPhone, Android, Symbian* alebo *Bada* – kráča Windows Phone najviac v stopách *iPhonu* a je cielený na bežného užívateľa. Pre profesionálnu prácu nie je úplne vhodný, hlavne keď bol človek zvyknutý na Windows Mobile – chýba tu napríklad synchronizácia s Outlookom alebo Microsoft Exchange.



Obrázok 1 - Predaj na Vianoce 2010

2.1 Hardwarová konfigurácia

Microsoft sa rozhodol (správnym smerom) obmedziť hardwarovú stránku telefónu a nastolil celkom prísne požiadavky. Je to kvôli doterajšiemu komplikovanému vývoju, kde Windows Mobile, ako nejednotná platforma, spôsobovala rôzne problémy. Procesor musí mať takt minimálne 1GHz architektúry ARM (Snapdragon od Qualcomm). Požiadavka na pamäť RAM je minimálne 256 MB a FLASH minimálne 8 GB. Display musí disponovať rozlíšením 800 x 480 bodov, musí byť kapacitný a podporovať multitouch minimálne štyroch dotykov naraz. Zo senzorov sú povinné akcelerometer, digitálny kompas, A-GPS, ambientný svetelný senzor a proximity senzor pre automatické vypnutie obrazovky pri hovore. Ďalšou povinnou položkou je minimálne 5Mpx fotoaparát s bleskom, pričom každý telefón musí disponovať hardwarovým tlačidlom na fotenie.

Okrem tohto tlačidla sú potrebné ďalšie tri – štart, späť a hľadať. Pre grafiku a zvuk je povinný hardwarovo akcelerovaný čip kompatibilný s DirectX 9 [4].

2.2 Softwarové vybavenie

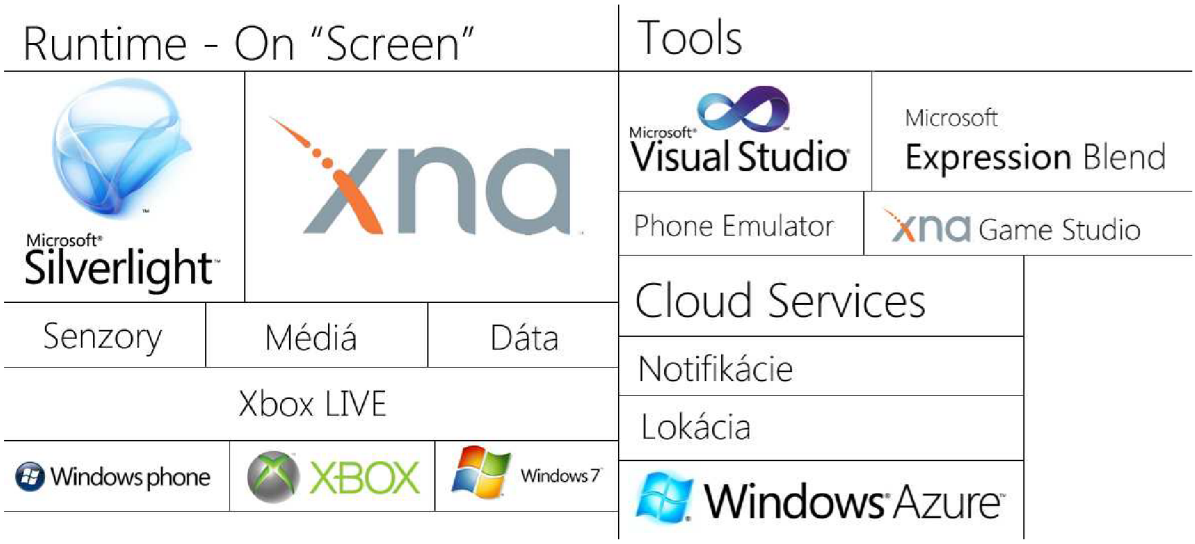
Jediné, čo zostalo sčasti rovnaké ako pri Windows Mobile, je jadro operačného systému. Windows Phone 7 je postavený na Windows CE 6.0 s viacerými výraznými vylepšeniami oproti staršej verzii Windows CE 5.2, na ktorej je ešte postavený Windows Mobile 6.5. Vo Windows Phone 7 prišlo zároveň k úplnému prepracovaniu užívateľského rozhrania, ktoré sa nepodobá na žiadny zo súčasných populárnych mobilných OS [1]. Windows Phone 7 využíva dizajnérsky jazyk Metro[2], ktorý bol použitý napríklad v multimediálnom prehrávači Zune (ktorý je sčasti súčasťou Windows Phone 7) alebo v programe Windows Media Center. Tento jazyk si potrpí hlavne na čistotu užívateľského rozhrania a na perfektnú typografiu. Typograficky je preto Windows Phone 7 najkrajší spomedzi konkurencie. Hlavná obrazovka telefónu obsahuje tzv. „live tiles“ – dlaždice, ktoré sa menia podľa push-notifikácií. Microsoft chce týmto ľudí oslobodiť od telefónov tak, aby užívateľ len pozrel na úvodnú obrazovku a hneď vedel, čo sa deje. Celý systém je rozdelený do tzv. hubov, kde každý hub spojuje obsah, ktorý spolu súvisí – napríklad huby *People*, *Pictures*, *Office*, *Games* a pod. Tieto huby sú poprepájané so sociálnymi sieťami Windows Live a Facebook. Ako bolo zmienené vyššie, telefón obsahuje hardwarové tlačidlá späť, štart a hľadať. Tlačidlo späť slúži na návrat medzi aplikáciami, ktoré sa ukladajú na zásobník – *backstack*. Ak začína dochádzať pamäť, systém automaticky začne mazať najstaršie položky na zásobníku. Ďalšie tlačidlo štart slúži na návrat na hlavnú obrazovku, kde sa nachádzajú dlaždice. Tlačidlo hľadať má v každom hube funkciu hľadania len v tomto hube. Na hlavnej obrazovke toto tlačidlo spustí prehliadač s vyhľadávačom *Bing*.

Systém nepodporuje multitasking aplikácií. To znamená, že nie je možné mať zapnuté dve bežiace aplikácie naraz. Je to z dôvodu väčšej výdrže batérie a z dôvodu väčšej rýchlosti momentálne bežiacej aplikácie. Pri nečakanej udalosti (telefonát, sms, budík…) sa aplikácia deaktivuje (tiež sa to nazýva *tombstonin*g) a následne sa aktivuje. Tieto udalosti má na starosti programátor.

2.3 Vývoj aplikácií na Windows Phone 7

Vývoj na tento systém sprevádzajú tri technológie a to Silverlight [3][11], XNA [3][8][19][20][21] a .NET Compact Framework 4. Silverlight je na klasické aplikácie v štýle Metro. Práca s ním je podobná ako so Silverlight na PC. Windows Phone 7 využíva verziu Silverlight 3.0 s niektorými prvkami zo Silverlight 4.0. XNA je framework na vývoj grafických aplikácií (hlavne hier) pomocou jazyka C#. API disponuje špecifickými rozhraniami na prácu s telefónom ako napríklad – akcelerometer, GPS, telefonovanie, posielanie správ apod. Chýba tu ale zatiaľ rozhranie pre prácu s kompasom a taktiež tu neexistuje rozhranie pre prácu s rozšírenou realitou (augmented reality).

Architektúra platformy je rozdelená do troch hlavných komponentov: *Runtime-On "Screen"*, *Tools* a *Cloud Services* [7].



Obrázok 2 - Architektúra Windows Phone 7 [7]

2.3.1 Runtime - On "Screen"

Aplikácie napísane v Silverlighte pre PC alebo v XNA pre Xbox bez problémov pobežia aj na systéme Windows Phone 7. Treba urobiť len minimálne zmeny (rozlíšenie obrazovky a pod.).

<table>
<tr><td colspan="9">Frameworky Windows Phone 7</td></tr>
<tr><td colspan="2">Senzors</td><td colspan="2">FMRadio</td><td colspan="2">Camera</td><td>Device Integration</td><td>Launchers & Choosers</td><td>Bing Maps</td></tr>
<tr><td colspan="3">PhoneApplicationFrame</td><td colspan="3">PhoneApplicationPage</td><td>Push Notifications</td><td colspan="2">WebBrowserControl</td><td>Pause/ Resume</td></tr>
<tr><td colspan="5">Silverlight Presentation and Media</td><td colspan="4">XNA Frameworks</td></tr>
<tr><td colspan="2">Controls</td><td colspan="2">Drawing</td><td colspan="2">IsolatedStorage</td><td>Input</td><td>Media</td><td>Content</td></tr>
<tr><td colspan="9">Application Object</td></tr>
<tr><td colspan="9">Common Base Class Library</td></tr>
<tr><td>Runtime</td><td>Resources</td><td>Globalization</td><td>Reflection</td><td>Location</td><td>Text</td><td>IO</td><td>Net</td><td>Diagnostics</td></tr>
<tr><td>Security</td><td>Threading</td><td>Collections</td><td colspan="2">ComponentModel</td><td>Configuration</td><td colspan="2">ServiceModel</td><td>Linq</td></tr>
</table>

Silverlight je ideálny framework na tvorbu aplikácií v štýle internetových stránok. Pre grafické rozhranie používa jazyk XAML (xml) a pre logiku C#. Používa sa takmer identicky ako jeho verzia na Windows. Obsahuje typické "internetové" prvky ako napr. zoznamy, checkboxy a pod. Pre programovanie 3D aplikácií, hlavne hier, je tu *framework XNA*. Je to multiplatformový *framework*, takže jednu aplikáciu je možné pustiť na počítači, Xbox-e, Zune HD alebo na Windows Phone 7.

Dôležitým rozhraním pre prácu s telefónom je rozhranie *Senzors*. Pomocou neho je možné mať prístup k *multitouch* vstupu, akcelerometru, mikrofónu alebo k GPS. Spolu s nástupom Windows Phone došlo aj k prerobeniu rozhrania na prácu s médiami. Toto rozhranie je prístupné rovnako z XNA, tak zo Silverlightu. Ďalším typickým rozhraním pre Windows Phone 7 je rozhranie *Data*. Aplikácie si môžu ukladať dáta do tzv. izolovaného úložiska (isolated storage). Tento priestor má k dispozícii len odpovedajúca aplikácia. Pre všetky platformy je prístupné API na prácu so sociálnou hracou sieťou Xbox Live. Pomocou neho je možné vytvárať celosvetové tabuľky so skóre a pod.

2.3.2 Nástroje (Tools)

Pre vývoj je potrebné *Visual Studio* v minimálnej verzii 2010 a nástroje *Windows Phone Developer Tools*. Ak nie je nainštalované žiadne *Visual Studio*, nainštaluje sa verzia *Express*, ktorá je zdarma. Ak už je v systéme *Visual Studio*, nainštaluje sa doň plugin.

Ďalšie nástroje, ktoré je možné využiť na vývoj aplikácií pre Windows Phone 7, sú *Expression Blend*, ktorý slúži ako dizajnérsky program pre tvorbu grafických komponentov pre platformu Silverlight. Ďalším dôležitým nástrojom je *Windows Phone Emulator*, ktorý nahrádza skutočný telefón. Nie je to ale úplná náhrada, nepodporuje totiž emuláciu GPS modulu a akcelerometra. Je automaticky integrovaný do Visual Studia a do programu Expression Blend. Posledným nástrojom je *XNA Game Studio*. Je to integrované prostredie pre tvorbu hier na Windows Phone, Windows, Xbox alebo Zune.

2.3.3 Cloud Services

Tzv. *cloud services* ponúkajú programátorovi pohodlnú prácu tak, že skupiny podobných služieb zoskupujú do jednej. Dvoma hlavnými rozhraniami typickými pre Windows Phone 7 sú *Notifications* a *Locations*. Notifikácie slúžia na sprostredkovanie informácií zo servera tak, že server dá vedieť o tom, že sa niečo zmenilo. Typické využitie to má pre komunikačné programy ako napr. ICQ a pod. *Locations* API slúži na prístup k aktuálnej pozícii telefónu pomocou senzoru A-GPS, triangulácie z BTS alebo z informácií o Wi-Fi sieti. Programátor sám ale nenastavuje jednu z týchto možností. Programátor si len vyžiada lokáciu a systém sám zaistí jej zdroj. Zaujímavým rozhraním je rozhranie *Azure*, ktoré dovoľuje vývojárom nechať bežať svoje služby na strojoch v dátových centrách Microsoftu.

2.4 Marketplace a jeho politika

Bežní užívatelia majú k aplikáciám prístup len cez Marketplace. Je to podobné ako konkurenčný App Store od Apple. Vývojári si môžu reálne zariadenie odomknúť a aplikácie si naň dávať prostredníctvom programu Application Deployment.

Registrácia na Marketplace vyžaduje *Windows Live ID*. Registrácia na *Windows Live* je zdarma, ale registrácia na vývoj je platená – 99$ na rok. Študenti sa môžu zaregistrovať cez *DreamSpark* a následne môžu vyvíjať na Windows Phone 7 zdarma. Študent má ale malé obmedzenie v podobe možnosti mať len jedno zaregistrované reálne zariadenie, pričom vývojár môže mať tri. Okrem typu registrácie „Študent“ je možné sa zaregistrovať ako nezávislý vývojár (*independent*) alebo ako firma (*business*).

Aby nevznikali zbytočné aplikácie, pre každého vývojára je tu obmedzenie v podobe vydaných neplatených aplikácií za rok – maximum je 5. Za každú ďalšiu je potrebné zaplatiť 20$ a viac.

Pre dokončenie registrácie je nutné overiť užívateľa. Tento proces sa koná prostredníctvom služby GeoTrust – SSL certifikát.

Pri platených aplikáciách je rozdelený zárobok tak, že vývojár dostane 70% z ceny aplikácie a ostatok spoločnosť Microsoft. Platba prebieha priamo na účet každý mesiac, pričom minimálny zárobok musí presiahnuť 200$. Pre Českú resp. Slovenskú republiku registrácia zatiaľ nie je prístupná, mala by sa sprístupniť do konca roku 2011.

Marketplace podporuje trial mód pri kupovaní aplikácie, takže užívateľ si ju môže najskôr skúsiť pred tým, ako ju kúpi. Vývojár toto musí naprogramovať jednoduchým spôsobom – volaním jednej metódy:

```
if (CurrentLicense.IsTrial().Equals(true))...
```

Po vytvorení aplikácie vzniká súbor s príponou XAP. Je to obyčajný ZIP, takže je možné prezerať si jeho obsah. V jeho vnútri sa nachádzajú dva manifest súbory: *AppManifest* – určuje aké DLL súbory sa majú použiť a *WMAppManifest* – v ňom sa nachádza zoznam prostriedkov, ktoré môže aplikácia použiť (networking, location, senzors, microphone...). Ďalej sa v tomto súbore nachádza informácia, ako sa má aplikácia nahrať. Hry sa nahrávajú do hubu Games – toto treba zabezpečiť vlastnosťou `Genre=Apps.Games`, aplikácie majú túto vlastnosť `Genre=Apps.Normal`. Tieto manifest súbory je možné priamo editovať vo Visual Studiu:

```
                    ...
    <Capability Name="ID_CAP_LOCATION" />
     <Capability Name="ID_CAP_SENSORS" />
    <Capability Name="ID_CAP_MICROPHONE" />
    <Capability Name="ID_CAP_NETWORKING" />
                    ...
```

Maximálna veľkosť XAP súboru pri tzv. *Over the Air* (OTA – cez mobilnú sieť) režime je 20 Mb. Celková veľkosť súboru nemôže presiahnuť 400 Mb. Zaregistrovaný vývojár si môže XAP súbor importovať do telefónu sám pomocou nástroja *XAP Deployment Tool*.

Marketplace ponúka možnosť zverejnenia beta verzie aplikácie. Aplikácia ale nie je vidieť, je nutné rozoslať informáciu záujemcom. Po krátkom čase sa aplikácia z Marketplace-u vytratí.

Proces zverejnenia aplikácie prebieha prostredníctvom nástenky (dashboard) vývojára, kde stačí odoslať XAP súbor a nasledovať niekoľko certifikačných krokov [3].

2.5 Životný cyklus aplikácie (XNA)

Windows Phone 7 nepodporuje multitasking aplikácií. Namiesto toho tu ale existuje mechanizmus zvaný *tombstoning* – aktivácia a deaktivácia aplikácie. Je to akýsi kompromis medzi naozajstným multitaskingom a rýchlosťou telefónu a zároveň výdržou batérie. Na popredí teda dokáže bežať len jedna aplikácia (mimo originálnych aplikácií ako prehrávač hudby a pod.) a ostatné sú deaktivované. Pre programátorov sú tu nachystané rôzne udalosti, ktoré sa volajú pri spustení, aktivácii, deaktivácii a vypnutí aplikácie.

Existujú dva hlavné spôsoby pri ktorých je XNA aplikácia prerušená:

1. Aplikácia je deaktivovaná (*tombstoning*). Signalizuje to dočasné prerušenie aplikácie vyvolané výnimkou (telefonát, sms, budík...). Taktiež nastane keď užívateľ stlačí jedno z tlačidiel *Home, Search* alebo *Power* a ak sa zamkne obrazovka.
2. Aplikácia je ukončená. Signalizuje to úplné ukončenie aplikácie. Nastane keď užívateľ úplne vypne danú aplikáciu alebo je aplikácia príliš dlho deaktivovaná (systém ju automaticky "uprace") alebo sa telefón z nejakého dôvodu vypne resp. reštartuje (slabá batéria a pod.) alebo v *backstack*-u (zásobníku deaktivovaných aplikácií) nie je miesto a tým pádom ju systém automaticky uvoľní.

XNA ponúka rôzne metódy, ako zistiť, či sa aplikácia aktivuje, deaktivuje a pod (viď Obrázok 3 – Životný cyklus aplikácie [6]):

- `OnDeacvitation` – volá sa pri deaktivácii, pri vypnutí a ak sa ukáže dialógové okno (napr. budík).
- `OnActivation` – volá sa pri aktivácii, novej inštancii aplikácie a ak dialógové okno zmizne.
- `Exiting` – volá sa pri úplnom vypnutí.
- `Initialize` – volá sa pri novej inštancii aplikácie a ak je aplikácia reaktivovaná.

Ďalším spôsobom, ako zistiť reaktiváciu aplikácie je použiť vlastnosť:

```
PhoneApplicationService.Current.StartupMode
```

Táto vlastnosť sa nastaví ešte pred tým, ako je zavolaný konštruktor aplikácie. Môže nadobudnúť hodnoty `Activate` – aplikácia je prebudená a `Launch` – spúšťa sa nová inštancia aplikácie [5][6].

2.5.1 Dočasné a perzistentné dáta

Pre dosiahnutie „falošného“ multitaskingu je možnosť využívať dočasné a perzistentné dáta.

Dočasné dáta sú všetky dáta, ktoré sú nevyhnutne potrebné na prebudenie aplikácie, ktorá je v deaktivovanom stave. Je to napríklad situácia, kedy je v hre zobrazené menu, aplikácia sa deaktivuje, tak pri následnej aktivácii sa musí toto menu znova zobraziť. Ak sa užívateľ hral hru, po aktivácii sa stav hry musí zobraziť presne tam, kde skončil. Tieto dáta sa ukladajú pomocou objektu `PhoneApplicationService.Current`. Proces ukladania týchto dát nesmie presiahnuť dobu desiatich sekúnd, inak systém aplikáciu ukončí úplne. Funguje to na princípe kľúč – hodnota:

```
PhoneApplicationService.Current.State["pPos"]=p.Pos;
p.Pos=(Vector2)(PhoneApplicationService.Current.State["pPos"]);
```

Perzistentné dáta sú dáta potrebné k uloženiu vývoja hráča (uloženie pozície v hre a pod.). Nie je ale potrebné obnoviť stav hry presne tam, kde skončil. Ak sa užívateľ vráti do hry, je vhodné vytvoriť napríklad položku v menu, ktorá bude odpovedať pokračovaniu v predtým rozohranej hre. Tieto dáta sa ukladajú pomocou objektu `IsolatedStorage`. Sem je možné ukladať súbory a pracovať s nimi ako v bežnom súborovom systéme alebo je možné využiť objekt `IsolatedStorageSettings`, kde to funguje podobne ako pri dočasných dátach mechanizmom kľúč – hodnota. Podobne ako pri dočasných dátach aj tu je nutné tieto dáta uložiť do desiatich sekúnd, inak systém ukončí aplikáciu úplne.[6]:

```
IsolatedStorageSettings.ApplicationSettings.Add("pPos", p.Pos);
p.Pos=(Vector2)IsolatedStorageSettings.ApplicationSettings["pPos"];
```

2.5.2 Porovnanie s aplikáciou v rozhraní Silverlight

Silverlight API ponúka veľmi podobné metódy pre aktiváciu, deaktiváciu, spustenie a vypnutie aplikácie: *Deactivated*, *Activated*, *Closing*, a *Launching*. Udalosti, pri ktorých sa volajú sú ale mierne odlišné [5]:

XNA:

	Initialize	OnActivation	OnDeactivation	Exiting
Aplikácia sa spúšťa	X	X		
Aplikácia je deaktivovaná			X	
Aplikácia je reaktivovaná	X	X		
Je zobrazené dialógové okno			X	
Dialógové okno zmizne		X		
Aplikácia sa vypne			X	X

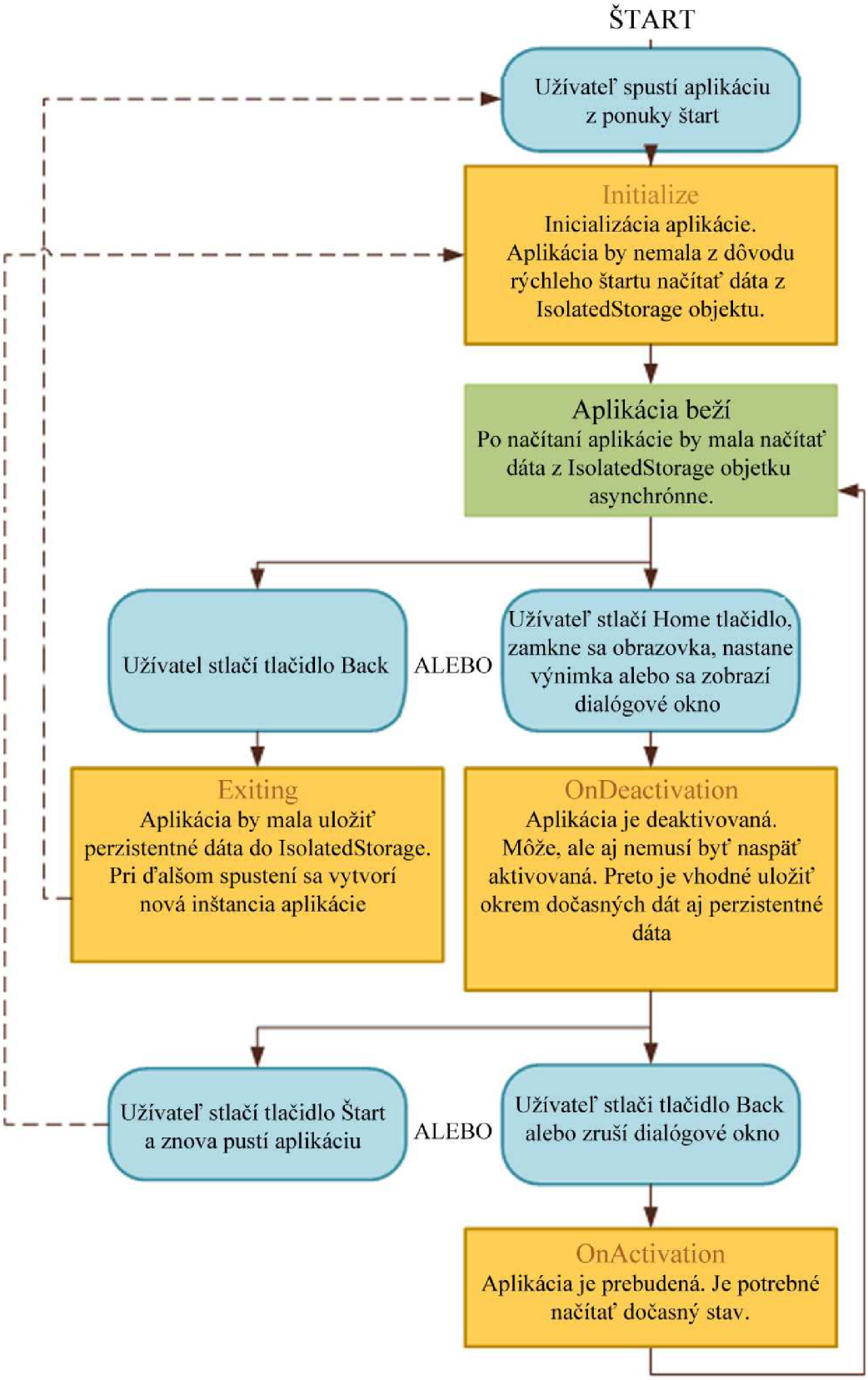
Silverlight:

	Launching	Activated	Deactivated	Closing
Aplikácia sa spúšťa	X			
Aplikácia je deaktivovaná			X	
Aplikácia je reaktivovaná		X		
Je zobrazené dialógové okno				
Dialógové okno zmizne				
Aplikácia sa vypne				X

Framework Silverlight je možné rovnako využívať aj na hry. Je to vhodné hlavne v prípade ak je napr. potrebné využitie videa v hre alebo prístup ku klávesnici (rôzne slovné hry). Taktiež sa viac hodí ak je k hre potrebné zobrazovanie internetových stránok alebo interaktívne hranie pomocou máp (Bing maps).

Vhodným príkladom využitia Silverlight frameworku vo svete hrách je vytvoriť aplikáciu, ktorá zobrazuje štatistiky z plnej hry založenej na XNA. Dovoľuje vývojárovi využiť vstavané prvky a tým si ušetriť prácu.

Pre Silvelight existuje aj 3D engine – Balder [21], ktorý dovoľuje vytvárať 3D aplikácie pomocou tohto frameworku. Nie je to ale prioritne robené pre hry, ale skôr pre prezentáciu napr. výrobkov alebo stavbu virtuálnych domov v obchode s nehnuteľnosťami a pod.



Obrázok 3 – Životný cyklus aplikácie [6]

3 XNA Framework

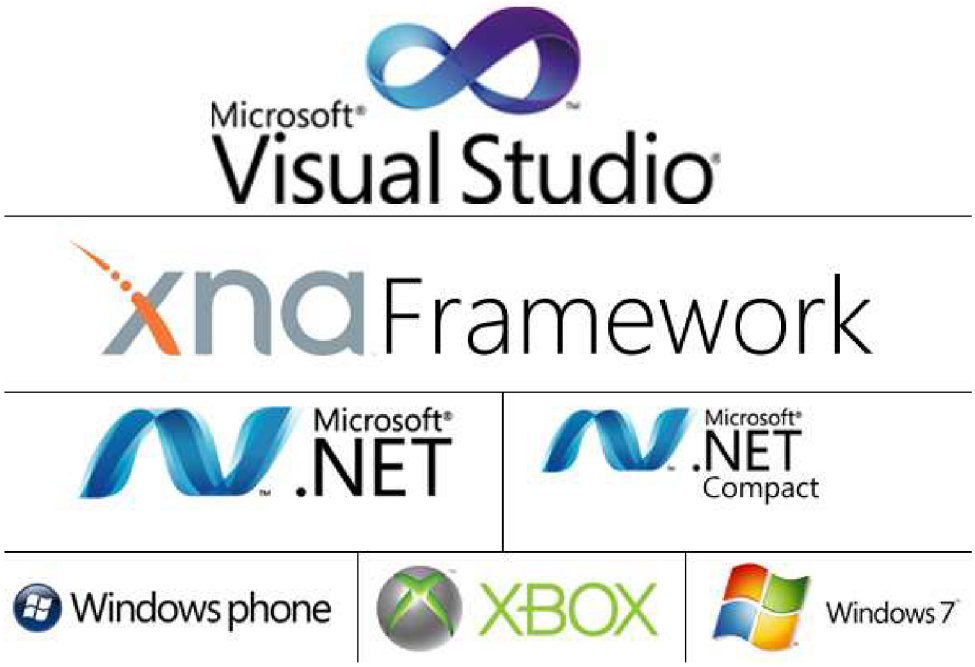
Microsoft XNA je integrované vývojové prostredie navrhnuté k ľahšiemu vývoju hier pre Microsoft Windows, Xbox 360 a pre Windows Phone. XNA Game Studio rozširuje Microsoft Visual Studio s podporou XNA framework-u a jeho funkcií. Inak povedané, XNA Game Studio dáva programátorovi možnosť pridať do svojej hry ľahko grafiku a zvuky.

XNA je navrhnuté na základe platformy .NET Framework. Programátor môže pri programovaní používať ako XNA, tak .NET Framework. XNA pre Xbox a Windows Phone využíva .NET Compact Framework. Týmto je zaručená kompatibilita medzi platformami a hry tak môžu bežať na rôznych platformách s minimálnou alebo žiadnou úpravou.

XNA Framework obsahuje nízkoúrovňové technológie, ktoré sú typické pre programovanie hier. Podporuje obe 2D aj 3D hry a pri platformách Xbox 360 a Windows Phone dovoľuje prístup k vibráciám [19][20].

3.1 XNA Build

XNA Build je množina tzv. *asset pipeline* nástrojov, ktorých cieľom je definovanie, spravovanie, ladenie a optimalizácia *asset pipeline*. *Asset pipeline* popisuje proces, ktorým sa obsah, ako textúry a 3D modely, modifikujú do podoby vhodnej pre hru. XNA Build dokáže identifikovať závislosti *pipeline* a taktiež dokáže analyzovať zbytočný obsah a tým znížiť veľkosť hry.



Obrázok 4 - XNA Game Studio [8]

3.2 História XNA

Prvá množina nástrojov pod označením XNA bola vydaná 24. marca 2004. Prvý XNA Build bol sprístupnený 14. marca 2006 pod označením XNA Game Studio Express. Microsoft sa snažil zapuzdriť funkcie knižnice DirectX to jednotného systému pre rôzne platformy.

3.2.1 XNA Game Studio Express a XNA Game Studio 2.0

XNA Game Studio Express bol navrhnutý pre študentov, nadšencov a nezávislých programátorov. Bol dostupný zdarma. Poskytoval základné funkcie – tzv *starter kits* – pre vývoj špecifických hier, ako napr. plošinové hry, real-time stratégie a hry typu first-person. Vývojári mohli na na platformu Windows vyvíjať zdarma, ale keď si chceli hru spustiť na platforme Xbox 360, museli zaplatiť poplatok 99$ pre prístup k *Microsoft XNA Creator's Club*. V tejto verzii nebolo možné posielať si medzi sebou binárne súbory na Xbox 360. To sa zmenilo s verziou XNA Game Studio Express 1.0 Refresh, kde bolo možné zdieľať binárne súbory s užívateľmi portálu *Microsoft XNA Creator's Club*. Táto verzia bola vydaná 24. apríla 2007 [8].

S nástupom XNA Game Studio 2.0 bola hlavnou zmenou podpora všetkých verzií Visual Studia, vrátane vtedy najnovšej verzie - Visual Studio 2005. Taktiež pribudlo API na podporu „sociálneho hrania" – Xbox LIVE. Bolo taktiež zdarma a bol vydaný 13. decembra 2007 [8].

3.2.2 XNA Game Studio 3.0

Pri verzii 3.0 pribudla podpora Visual Studia 2008. Ďalej pribudla podpora *LINQ* a C# 3.0, ktorá obsahovala niekoľko vylepšení: *Query expressions*, *extension methods*, *lambda expressions*, *expression trees*, *partial methods*... S príchodom multimediálneho prehrávača Zune bolo možné vyvíjať hry aj na túto platformu. Bolo možné vytvárať *cross-platform* aplikácie, aplikácie, ktoré mohli bežať na rôznych platformách (projekty na rôzne platformy mohli zdielať jeden obsah). Pribudli tu aj vylepšenia ohľadom API Xbox Live, kde vznikli pozvánky na multiplayer hry. Od tejto verzie pribudla podpora zvukových formátov WAV, WMA a MP3, ktoré sú integrované do *Content Pipeline*. Vznikla taktiež Media API (hlavne kvôli Zune). K pohodlnosti programátorov prispela podpora trial módu v aplikáciách [8].

3.2.3 XNA Game Studio 3.1

Verzia 3.1 bola vydaná 11. júna 2009. API v sebe zahŕňa podporu pre prehrávanie videa a upravené API pre audio. Videá sa dajú premietať aj na 3D modely. Ďalej pribudol Xbox LIVE Party systém s podporou pre tzv. Avatarov. Avatari sú animované 3D postavy, ktoré reprezentujú hráča. Sú určené pre každého. S Avatarmi je možnosť využiť aj tzv. Avatar Marketplace, kde je možné dokúpiť pre postavu predmety, oblečenie a pod. Programátor s Avatarom pracuje cez obsiahnuté API. Avatara je

možné využívať teda v hre ako postavu. Postava Avatara obsahuje 31 vstavaných animácií (*idle, celebrate, clap, wave, cry, yawn…*). Vlastné animácie je možné pridať napríklad pomocou programu *Softimage Mod Tool Pro*, ktorý je zdarma pre členov *Microsoft XNA Creator's Club*. Taktiež je možný export Avatara do formátu FBX a následná práca ako s akýmkoľvek iným modelom. Spolu s Avatarmi súvisí aj ich politika, kde sú špecifikované pravidlá ako napr. nemožnosť vulgarizmov a pod [8].

3.2.4 XNA Game Studio 4.0

XNA verzia 4.0 nadobudla svoju finálnu podobu 16. septembra 2010. Hlavným prínosom je podpora platformy Windows Phone 7. S tým súvisí aj pridané API pre multi-touch a mikrofón. Ďalej pribudla podpora konfigurovateľných efektov (shaders) a samozrejme integrácia do Visual Studia 2010. Novinkou v tejto verzii sú aj tzv. *State Objects*, ktoré spravujú stavy, ktoré sa týkajú vykresľovania, napr. *BlendState, DepthStencilState*... S prístupom platformy Windows Phone 7 je tu snaha o väčšiu kompatibilitu medzi platformami a zaviedla sa tu podpora pre automatické (systémom riadené) zistenie rozlíšenia obrazovky a jednotný *key-input* [8].

3.2.5 XNA Framework Content Pipeline

XNA Framework Content Pipeline slúži na pridávanie obsahu (modely, textúry, zvuk…) do hry. Najlepšie sa dá popísať ako "rozšíriteľné spracovanie obsahu spravované vo Visual Studiu". Tento framework sa snaží vývojárovi uľahčiť prácu s importovaním objektov a necháva mu priestor sústrediť sa len na vývoj aplikácie.

Vďaka tomu, že celý proces riadi Visual Studio, je možné pridávať objekty neskôr a zároveň celý obsah zdieľať medzi viacerými projektmi bez nutnosti duplikácie.

Content Pipeline sa skladá z dvoch hlavných častí. Po pridaní objektu do obsahu je potrebné zvoliť tzv. *Importer*. Ten načíta dáta a normalizuje ich. To znamená, že vývojára nemusí napríklad zaujímať, ako je objekt natočený a pod. Inak povedané, nezáleží na tom, z akého zdroja dáta prišli, *Importer* si ich vezme a prerobí do vhodnej podoby.

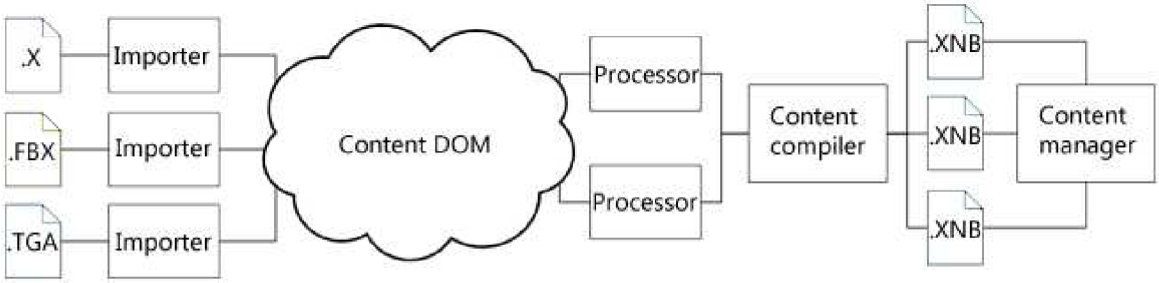
3D formáty	2D formáty	Materiály	Zvukové formáty	Iné
.FBX .X	.BMP .DDS .DIB .HDR .JPG .PFM .PNG .TGA	.FX (shader)	.XAP (XACT)	.SPRITEFONT .XML

Po tom, ako *Importer* dokončí svoju prácu, dáta sa uložia vo formáte DOM. DOM reprezentuje zbierku tried alebo schému ako napríklad XML súbor. Sú uložené ako prísne typované

dáta, čo napríklad znamená, že séria vrcholov alebo textúra vyzerajú rovnako bez ohľadu na to, z akého formátu súboru prišli. Tento súbor sa môže byť pre ladenie uložiť na pevný disk, ale je potrebné vedieť, že je stratový, pretože *Importer* si ukladá len tie dáta, ktoré potrebuje pre ďalšiu činnosť.

Ďalšiu činnosť prevezme *Processor*. Ten je zodpovedný za načítanie dát z DOM-u a za vytvorenie objektu na základe týchto dát. Pre *Processor* je veľmi dôležité to, že *Importer* dáta zo súboru uložil rovnako nezávisle na formáte súboru. Preto *Processor* pre vytvorenie modelu vyzerá vždy rovnako. Štandardné procesory sú [10]:

- *EffectProcessor* – prekladá obsah prenechaný *FX* importerom. Ide o shader.
- *ModelProcessor* – parametrizovaný procesor, ktorého výstupom je 3D model.
- *PassThroughProcessor* – používa sa pre už predspracované súbory (napr. pripravený DDS súbor alebo špeciálny XML súbor).
- *FontDescriptionProcessor* – konvertuje .spritefont súbor na font.
- *FontTextureProcessor* – konvertuje textúru na font.
- *TextureProcessor* – parametrizovaný procesor, ktorého výstupom je objekt textúry.
- *XACTProcessor* – generuje zvukový obsah.



Obrázok 5 - Content Pipeline [9][10]

3.3 XNA Framework pre Windows Phone 7

XNA Framework sa čiastočne líši od XNA pre Xbox alebo Windows. Je to dané platformou a jej novými možnosťami a zároveň jej menším výkonom.

3.3.1 Rozlíšenie a orientácia obrazovky

Telefón podporuje pri XNA aplikáciách 3 orientácie – *LandscapeLeft, Portrait* a *LandscapeRight*. Predvolene je nastavená orientácia *LandscapeLeft*. Samozrejme je možné v hre používať všetky tri orientácie a detekovať ich zmenu:

```
Graphics.SupportedOrientations = DisplayOrientation.Portrait |
                                 DisplayOrientation.LandscapeLeft |
```

```
                                    DisplayOrientation.LandscapeRight;
This.Window.OrientationChanged += new EventHandler <EventArgs>
                                        (orient_changed);
```

Rozlíšenie obrazovky je možné nastaviť od 240x240 až do 800x480 (480x800). Pokiaľ pomer strán rozlíšenia nesedí s pomerom strán telefónu, sú medzery doplnené čiernymi bodmi. Zároveň systém zabezpečí, že dotykový vstup bude len v rozmedzí daného rozlíšenia.

```
graphics.PreferedBackBufferWidth = 480;
graphics.PreferedBackBufferWidth = 800;
```

O rozlíšenie a o zmenu orientácie sa stará hardwarový komponent *Scaler*. Keďže je hardwarový, nespomaľuje priebeh hry.

Tak ako na platforme Windows, aj na Windows Phone 7 je možné využiť možnosť celej obrazovky – *full screen*. V ponímaní telefónu ide len o odstránenie ukazovateľa stavu baterky, signálu a pod.

Užívateľ si môže na svojom telefóne nastaviť vypnutie obrazovky (a jej zamknutie) po určitej dobe nečinnosti. Systém túto nečinnosť kontroluje pomocou dotyku a hardwarových tlačidiel. Hry, ktoré využívajú len akcelerometer je treba ošetriť. Je možné zakázať vypínanie obrazovky na pevno, ale treba to používať s citom. Ak to nie je potrebné (napr. v menu), treba možnosť vypnutia obrazovky zase zapnúť [3].

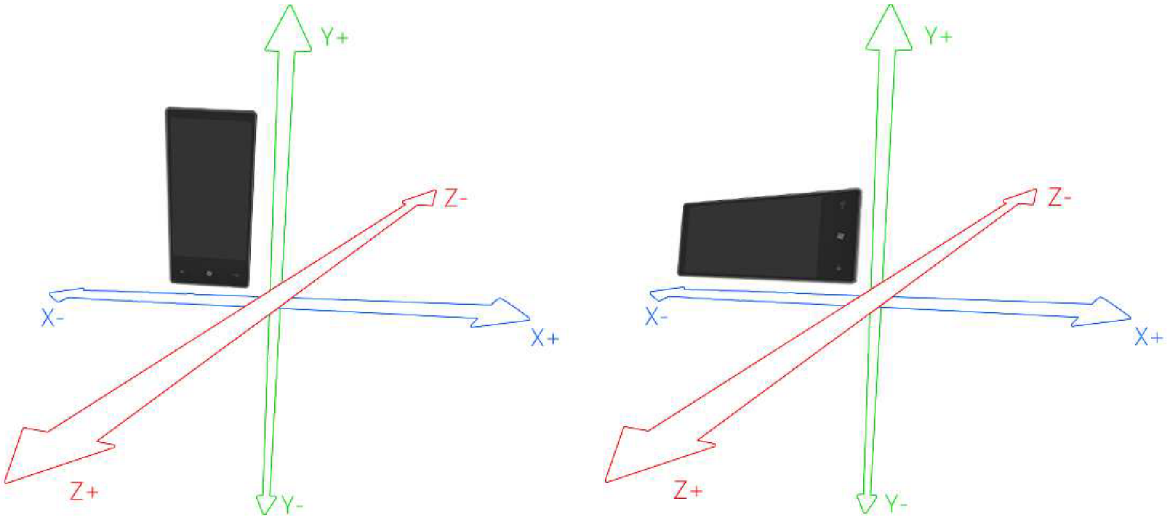
3.3.2 Akcelerometer

Akcelerometer na Windows Phone 7 meria zrýchlenie vo všetkých troch osách – X, Y, Z. Je samozrejme možné merať len v niektorých a ostatné vynechať. Hodnoty, ktoré akcelerometer vracia pre jednotlivé osi, sú v rozmedzí -1 až +1. Keď je hodnota danej osi 0, znamená to, že je telefón položený priamo na os.

Na rozdiel od iných vstupných zariadení (klávesnica, myš...), kde je potrebné manuálne čítať stav, akcelerometer generuje udalosti na základe zmeny. Je teda potrebné nastaviť metódu, ktorá sa volá automaticky so zmenou akcelerometru. Toto je rozdiel oproti akcelerometru na Zune HD, kde ten funguje tak ako klávesnica. Hlavným dôvodom fungovania akcelerometru týmto spôsobom je rovnaký princíp akcelerometru na silverlight-e pre Windows Phone 7 [3].

```
Accelerometer acc = new Accelerometer();
Acc.ReadingChanged+=new EventHandler<AccelerometerReadimgEventArgs>
                            (acc_ReadingChanged);
acc.Start();
```

Telefón môže byť otočený v tzv. *portrait* móde (vertikálne) alebo v tzv. *landscape* móde (horizontálne). V oboch prípadoch sa hodnoty akcelerometru v smere všetkých osí menia rovnako. Napríklad keď je telefón v polohe *portrait* a nakloní sa vľavo, hodnota akcelerometra v osi X sa vychýli do mínusovej hodnoty. Rovnako to funguje aj pri *landscape* polohe – viď oObrázok 6 - Smer zmeny jednotlivých osí.



Obrázok 6 - Smer zmeny jednotlivých osí

3.3.3 Zvukové efekty

Načítanie zvukov funguje obdobne ako pri klasickom XNA – načítava sa cez *Content Pipeline*. Rozdielom oproti klasickému XNA je, že tu nie je podpora pre XACT a je možné mať pustených menej zvukov naraz.

Existujú tu dva rôzne objekty pre prácu so zvukom. Prvým je *SoundEffect*, ktorý je prvotným štádiom pri načítaní zvuku. Ten je možné načítať vo formáte MP3, WAV a WMA. Tento objekt je možné len prehrať. Na zložitejšie operácie slúži objekt *SoundEffectInstance*, ktorý sa vytvorí práve z objektu *SoundEffect*. S týmto objektom je už možné robiť operácie ako zmena tónu, slučka, pauza, hlasitosť a pod [3].

3.3.4 Dotykový vstup (Touch input)

Dotykový vstup tu funguje rovnako ako pri Zune HD. Funguje to podobne ako klávesnica, číže je potrebné manuálne pýtať si aktuálny stav. XNA rozlišuje 3 typy vstupu: *Pressed, Released, Dragged*. Je možné si požiadať o kompletný zoznam aktuálne aktívnych bodov. Je potrebné vedieť, že pri každom zavolaní sa vygeneruje aktuálny stav, preto je nutné si po volaní metódy uložiť stav dotykového vstupu a ten potom zdieľať medzi objektmi, ktoré ho potrebujú [3].

```
TouchCollection touchState = TouchPanel.GetState();
foreach(TouchLocation touch in touchState){
      Point tPoint = new Point(touch.Position.X, touch.Position.Y);
}
```

3.3.5 Textový vstup v XNA

XNA pre Windows alebo Xbox využíva USB klávesnicu pre textový vstup. Na platforme Windows Phone je to vyriešené cez objekt *Guide*. *Guide* je množina zdrojov, ktorá poskytuje interakciu užívateľa s XNA. *Guide* sa používa na zobrazenie klávesnice. Keď je zobrazená klávesnica pomocou *Guide*, aplikácia nie je vidieť, ale stále beží. Preto je nutné pred zobrazením klávesnice bežiacu hru pozastaviť.

Pre zobrazenie textu na obrazovke sa používa objekt *SpriteFont*. *SpriteFont*-y sú spravované ako obsah (content) a načítavajú sa teda cez *Content Pipeline* [3].

3.3.6 XNA a Windows Phone OS

XNA aplikácia môže spúšťať iné programy ako napr. poslanie SMS, uloženie telefónneho čísla a pod. Tieto úlohy sú rozdelené medzi tzv. *Launchers* a *Choosers*. *Launcher* len púšťa program, *Chooser* môže vrátiť výsledok. Typickou ukážkou pre *Chooser* je fotoaparát [3]:

```
cameraTask = new CameraCaptureTask();
cameraTask.Completed +=new EventHandler<PhotoResult>(cam_Completed);
cameraTask.Show();
```

3.3.7 Windows Phone 7 a Shadere

V aktuálnej verzii nie je možné použiť vlastné shadere (*effects*, ako to nazýva Microsoft). Je to ale zrejme dočasné riešenie. Namiesto toho je tu niekoľko vstavaných *effect*-ov [11]:

BasicEffect	Poskytuje možnosti jednoduchého textúrovania kombinovaného s osvetlením.
DualTextureEffect	Poskytuje *blending* medzi dvoma textúrami. Je s ním možné tvoriť svetelné mapy z preddefinovaných máp.
EnvironmentMappedEffect	Poskytuje možnosti tzv. *Environmental mapping*. Ide o dosiahnutie veľmi lesklého objektu za veľmi nízku cenu.
AlphaTestEffect	Poskytuje *Aplha Test*. Vykresľujú sa body, ktoré majú kanál alpha v istom rozsahu.
SkinnedEffect	Umožňuje hardwarový *Skinning*.

3.4 Xbox Live

Xbox Live je služba pre online multiplayer hranie a prezeranie digitálnych médií na platformách Xbox, Windows a Windows Phone 7. Je to momentálne jediná služba tohto druhu, za ktorú je možné platiť. Prvýkrát bola spustená v roku 2002 pre vtedajší Xbox. Neskôr bola vydaná aktualizácia s nástupom Xbox 360 v roku 2005. Ďalšie rozšírenie prišlo v roku 2007 na platforme Windows, ktoré sa nazýva *Games for Windows – Live*. V roku 2010, kedy Microsoft vydal Windows Phone 7, bola služba rozšírená aj o podporu tejto platformy.

Niektoré aktuálne možnosti Xbox Live:

- *Gamercard* – zobrazuje informácie o hráčovi.
- *Avatar* – virtuálna postavička reprezentujúca hráča.
- *Friends* – kamaráti hráča, s ktorými je možné hrať.
- *Windows Live Messenger* integrácia – možnosť písania s kamarátmi cez túto službu.
- *Xbox Live Marketplace* – ponúka stiahnuteľný obsah pre hry, hudbu a filmy.
- *Party System* – hromadné hranie alebo pozeranie filmov.
- *Netflix* – video služba, ktorá prehráva televízne relácie a filmy.
- *Zune* – instantné prehrávanie video obsahu v HD kvalite.

Profil užívateľa sa skladá z niekoľkých častí. *Gamertag* je univerzálne meno hráča – prezývka. *Gamerscore* – systém dosiahnutých úspechov hráča. Tieto úspechy zahŕňajú jednotlivé body v hrách alebo aj na turnajoch a pod. *Gamercard* je informačný panel, ktorý zobrazuje hráčove informácie vrátane *Avatara*.

Ako je spomenuté vyššie, Xbox Live je jediná platená aplikácia tohto druhu. Je možné mať program, ktorý je zdarma. S ním však nie je možné hrať hry online. Na to je potrebné vylepšenie na tzv. *Gold* účet, ktorý má aj niekoľko ďalších vylepšení. Najdrahším programom je *Gold Family*, kde je možné nastaviť prístup k jednotlivým hrám pre rôznych členov rodiny a pod.

3.4.1 Xbox Live a Windows Phone 7

API v XNA 4.0 pre Windows Phone 7 ponúka podobné možnosti ako pre Xbox 360. Na začiatku je potrebná inicializácia hráča a následný login. Z ďalších možností je tu samozrejme prístup k *Gamertag*-u, k obrázkom hráča, jeho úspechom a k jeho skóre. Je tu možnosť pozývať hráčov na multiplayer hranie, ale nie je tu možnosť služby *Party System*, tak ako na Xbox 360. Ďalším obmedzením oproti Xbox 360 je nemožnosť využitia *Avatar*-a v hre. Na tom ale Microsoft pracuje a s najbližšou aktualizáciou by to malo fungovať. Ďalšou z možností služby Xbox Live je podpora trial verzií. To znamená, že hráč si môže vyskúšať obmedzenú hru (časovo aj funkcionálne) a až potom kúpiť [12].

4 Budúcnosť Windows Phone

Budúcnosť sa dá vo všeobecnosti predpovedať len veľmi ťažko. V prípade platformy Windows Phone 7 je už ale známy čiastočný budúci vývoj.

4.1 Microsoft a Nokia

Jednou z najväčších udalostí tejto platformy bolo bezpochyby uzatvorenie partnerstva so spoločnosťou Nokia 11.2. 2011. Týmto sa chcela firma Nokia zachrániť od postupného úpadku, ktorý jej zapríčiňoval nie veľmi populárny systém Symbian. Spoločne sa budú snažiť poraziť firmu Google a ich systém Andorid.

Čo sa týka obchodov s aplikáciami, Nokia Ovi Store bude integrovaný do Microsoft Marketplace-u. Nokia bude využívať služby Microsoftu ako napr. vyhľadávač Bing a Microsoft do svojich služieb zaintegruje veľmi populárne mapy Ovi Maps.

Nokia navyše dostane práva na modifikáciu operačného systému. To znamená, že sa na trhu objavia aj lacnejšie telefóny s horším hardwarovým vybavením. Je to jedna zo súčastí plánu, s ktorým chcú bojovať proti Andoridu, ktorý má veľké množstvo práve tých lacnejších mobilných zariadení.

4.2 Aktualizácie

V apríli 2011 Microsoft vydal prvú aktualizáciu na platformu Windows Phone. Ide o tzv. *NoDo* aktualizáciu (*NoDo – No Donuts*, odpoveď na aktualizáciu Androidu 2.0 s kódovým označením *Donut*), ktorá priniesla do systému vlastnosti *copy-paste*, vylepšené hľadanie v Marketplace, zrýchlenie systému na základe úpravy správy pamäte, vylepšenie WiFi, Outlook-u, správ, kamery a zvuku. Nasadenie aktualizácie ale neprebiehalo úplne podľa predstáv. Prvé komplikácie sa ukázali hlavne na zariadeniach firmy Samsung, kde z nich aktualizácia spravila na pár dní drahé "ťažidlo".

Na jeseň roku 2011 by mala vyjsť ďalšia aktualizácia, ktorá je už ale pomerne väčšia ako tá predchádzajúca. Nesie kódové označenie *Mango* a podľa oficiálnych informácií Microsoft-u ide o systém verzie 7.5. Jej hlavnými črtami z pohľadu užívateľa sú: Integrácia aplikácie Internet Explorer 9, ktorá by mala byť takmer totožná s počítačovou verziou. To znamená podporu HTML 5, totožný JavaScript engine a pod. HTML 5 bude plne hardwarovo akcelerované, čo znamená plynulé prehrávanie videa, animácií atď. Ďalšou významnou zmenou oproti predchádzajúcej verzii je pridanie tzv. *fast app switching* multitasking systému. Užívateľ už nebude musieť prechádzať aplikáciami pomocou neustáleho stláčania tlačidla späť. Podržaním tlačidla späť sa zobrazí horizontálny zoznam aplikácií, ktorý pozostáva z obrázkov, ktoré odzrkadľujú stav aplikácie pred tým, ako bola aplikácia vypnutá (uspaná). Vylepšený má byť znova aj Marketplace, ktorý pridá nastavenia a lepšie

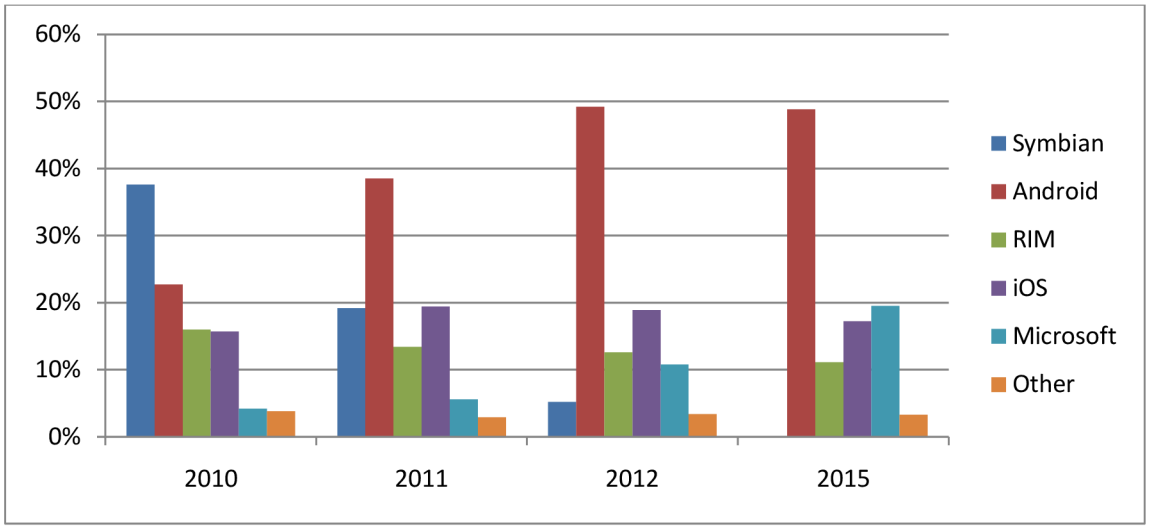
hodnotenia aplikácií. Taktiež bude vylepšená práca s pamäťou, čo výrazne zvýši rýchlosť načítavania aplikácií, v niektorých príkladoch aj viac než dvojnásobne. K integrácii Facebook-u pribudne sociálna sieť Twitter. Pre vývojárov je noviniek viacero. Jednou z nich je spomínaný multitasking, o ktorý sa budú musieť starať hlavne programátori. Tento multitasking bude podporovať aj čiastočné bežanie aplikácií na pozadí, napr. zber GPS dát, prehrávanie zvuku a sťahovanie dát z internetu. Taktiež pribudne prístup k senzorom telefónu, kde okrem povinného kompasu má pribudnúť aj voliteľný gyroskop. Zaujímavou možnosťou je kooperácia frameworkov XNA a Silverlight. To znamená, že vývojár bude môcť v XNA aplikácii (hre) využívať prvky zo Silverlight-u a tým vytvoriť napríklad menu. Novinky sa týkajú aj tzv. live-tiles, ktoré budú obsahovať niekoľko zaujímavých možností pre vývojára:

- Lokálne notifikácie (*Local Notifications*) –nejde o *push* notifikácie, ale o čisto lokálne. Sú ideálne pre alarmy, pripomienky atď.
- *Deep Toast* – Ak nejaká aplikácia zobrazí dôležitú notifikáciu, tak po otvorení aplikácie sa táto naviguje priamo na informácie o zmienenej notifikácii. Napríklad ak aplikácia o počasí zobrazí notifikáciu o záplave, tak po otvorení aplikácie sa zobrazia okamžite podrobné informácie o zaplavenej oblasti.
- Multi dlaždice (*Multi-tiles*) – Jedna aplikácia môže mať viac dlaždíc. Napríklad aplikácia so správami o politike, počasí, športe môže mať dlaždicu pre každú tému zvlášť.
- Zvýši sa počet živých dlaždíc z 15 na 30 a cyklus kontrolovania nových notifikácií bude 15 minút namiesto doterajších 60.

Čo sa týka *live-tiles*, pribudne možnosť vytvorenia tzv. agenta. Je to vlastne živá dlaždica s tým rozdielom, že tento agent je vytvorený priamo v aplikácii a týka sa prevažne jednej služby. Napríklad pri aplikácii na rezervovanie leteniek môže byť agentom práve rezervovaná letenka, ktorá užívateľa oboznamuje o odchode lietadla, pomocou GPS kontroluje jeho pozíciu a zisťuje, či daný let môže stihnúť a pod. Ďalšou zmenou je využitie lokálnej SQL CE databázy a taktiež možnosť pristúpiť ku kontaktom a kalendáru telefónu. Vývojár dostane prístup ku kamere, čím je daná možnosť vývoja tzv. aplikácií rozšírenej reality (augmented reality). V emulátore pribudne možnosť vytvorenia fotky obrazovky (*screen shot*) a podpora emulácie GPS a akcelerometra. Zmeny nastanú aj mimo systému ako takého. Najdôležitejšou zmenou pre Českú republiku je podpora ďalších krajín v marketplace-i. Tých má byť spolu 19 a je medzi nimi aj Česká republika. To znamená, že od mája roku 2011 bude môcť český vývojár publikovať svoje aplikácie na Marketplace. Slovensko zatiaľ, bohužiaľ, chýba [23].

Budúcnosť platformy Windows Phone je otvorená a z počiatočných predajov to nevyzeralo veľmi dobre. Partnerstvo so spoločnosťou Nokia a spomínaná aktualizácia *Mango* zrejme prilákajú mnoho ďalších záujemcov. Atraktivita platformy spočíva hlavne v jednoduchosti ovládania a pre vývojára v bezkonkurenčne najlepšom vývojovom prostredí. Zodpovedá tomu doterajší nárast

aplikácií na Marketplace – 15000 za prvých 6 mesiacov, tzn. približne 83 nových aplikácií denne. Spoločnosť Gartner zverejnila predikciu na rok 2015 v oblasti mobilných zariadení, kde stanovila 20% podiel Microsoft-u na trhu [24]:



Obrázok 7 - Predikcia podľa spoločnosti Gartner [24]

Dôležitou a veľmi zaujímavou novinkou má byť prepojenie platformy Windows Phone s konzolou Xbox 360 a hlavne s perifériou Kinect. SDK pre Kinect by malo vyjsť na jar roku 2011.

5 Letecký simulátor v hrách

Letecké simulátory v hrách majú spoločných niekoľko hlavných rysov. Sú to predovšetkým terén a jeho prostredie, obloha, fyzika lietadla a poprípade voda. Každú z týchto súčastí je možné implementovať niekoľkými spôsobmi. V nasledujúcej kapitole sú popísané niektoré najčastejšie metódy.

5.1 Terén a jeho implementácia

Terén pre letecký simulátor je špecifický svojou rozsiahlosťou. S tým samozrejme súvisí aj rýchlosť vykresľovania s využitím optimalizácií. Rozsiahlosť terénu sa dá zabezpečiť rôznymi spôsobmi. Najčastejšie je postupné načítanie obrovského terénu, ktorý je rozdelený na časti. Ďalšou možnosťou je opakovanie terénu po určitej dobe.

S terénom a jeho optimalizáciou sa spája pojem *Level of Detail* (LOD). Je to metóda pri ktorej objekty, ktoré sú vo väčšej vzdialenosti od kamery, nie je potrebné vykresľovať v plnej kvalite. Existuje niekoľko algoritmov pre dosiahnutie techniky *Level of Detail* pre terén:

- *Continuous LOD* – znamená, že LOD je počítaný každým vykreslením snímku. Je založený na zložitosti terénu, vzdialenosti kamery, pozorovacom uhle atď.
- *ROAM (Realtime Optimally-Adapting Mesches)* – dynamicky zo snímku na snímok mení konfiguráciu primitív (trojuholníkov) podľa pozície kamery a členitosti terénu v každom mieste. Trojuholníky pokrývajúce celú štvorcovú krajinu sú hierarchicky delené podľa potreby vždy na polovicu. Vzniká tak strom trojuholníkov pokrývajúci celú plochu výškovej mapy. S rastúcou hĺbkou stromu raste úroveň detailu [13] [18].
- *Chunked LOD* – štvorcová výšková mapa je hierarchicky delená na identické kvadranty do tzv. kvadrantového stromu – Quadtree. Každý uzol stromu obsahuje optimalizovanú reprezentáciu odpovedajúcej časti výškovej mapy vo vhodnej, konštantnej úrovni detailu. Pri zobrazovaní dochádza k priechodu kvadrantovým stromom [13].
- *SOAR (Stateless One-pass Adaptive Refinement)* – elegantný a veľmi rýchly algoritmus. Algoritmus zjemňovania terénu generuje pre každý snímok nový model aproximujúci zdrojový terén. Pre dosiahnutie spojitého povrchu je nutné pred zobrazením zjemňovanie dokončiť a nie je ho teda možné prerušiť ako pri metóde *ROAM* [16] [17].

Súčasťou terénu je samozrejme aj jeho textúra. Tu je znova na výber niekoľko možností. Prvou z nich je *multi-texturing* pomocou shaderov. Táto technika sa používa najčastejšie, ale vzhľadom na to, že

Windows Phone 7 zatiaľ vlastné shadere nepodporuje, je nutné využiť iný spôsob. Jedným z nich je použiť vstavaný shader – *DualTextureEffect*, kde je možné využiť dve textúry. To je pre terén ale dosť málo a tak sa ponúka najjednoduchšie riešenie – vytvoriť celú textúru na celý terén. Toto riešenie je síce jednoduché, ale veľká textúra zaberá veľký priestor, čo sa hlavne na mobilnú aplikáciu nehodí. Je to ale zatiaľ asi jediná možnosť.

5.2 Obloha pomocou skyboxu

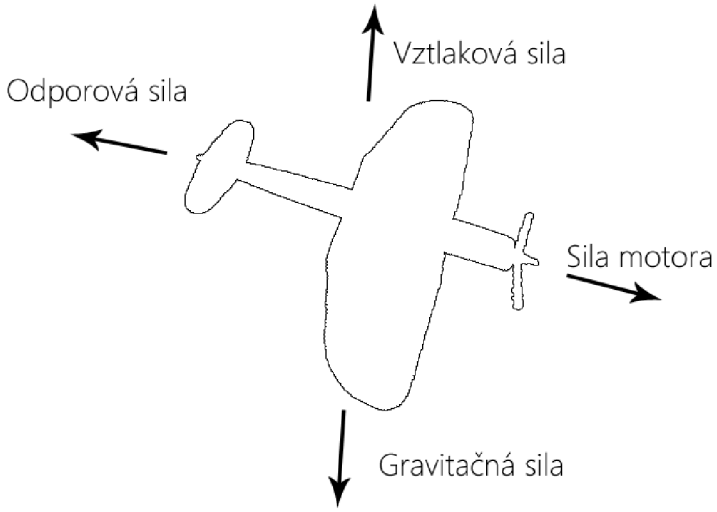
Obloha sa dá vytvoriť dvomi spôsobmi. Jedným je dobre známy *skybox*, kedy sa na prázdny kváder znútra nanesie textúra technikou zvanou *cube mapping*. Toto dáva efekt vzdialenej prírody, domov a pod. a užívateľovi dáva pocit obklopenia vzdialenou prírodou. Tento kváder sa musí pohybovať spolu s hráčom, inak by sa efekt veľkého priestoru stratil a mohlo by dôjsť prechodu za oblohu.

Druhou metódou je veľmi podobná metóda ako *skybox*, s rozdielom, že namiesto kvádra sa použije guľa, resp. valec.

5.3 Fyzika

Fyzika je veľmi podstatná časť pri tvorení leteckých simulátorov. Na prvom mieste je treba rozhodnúť, či ide o plnohodnotnú simuláciu, alebo o aproximáciu.

V jednoduchosti na lietadlo pôsobia 4 sily:



Obrázok 8 - Sily pôsobiace na lietadlo

Sila motora závisí od výrobnej sily motora. Gravitačná sila závisí od váhy lietadla a gravitačnej konštanty. Vztlaková a odporová sila spolu súvisia a počítajú sa na základe naklonenia krídla voči smeru letu. Pomocou týchto síl, fyzikálnych rovníc a tabuľkových hodnôt je možné vypočítať takmer presné vlastnosti lietadla. Pre aplikáciu typu "jednoduchý letecký simulátor" je to ale príliš zložité.

Výsledná sila sa dá aproximovať niekoľkými jednoduchšími spôsobmi. K tomu veľkou mierou môže dopomôcť využitie akcelerometra, ktorý vyrieši nakláňanie lietadla vo všetkých smeroch. Ďalej je nutné doplniť nejakú gravitačnú silu, ktorá bude tlačiť lietadlo k zemi a samozrejme silu motora, ktorá ho tlačí vpred.

5.3.1 Dostupné implementácie leteckých simulátorov

Namiesto programovania kompletného fyzikálneho jadra simulátora je možnosť prevziať časť zo známych *open-source* simulátorov:

- *FlightGear* – multi-platformový projekt. Zdrojový kód je plne dostupný pod licenciou *GNU General Public License* [14].
- *Palomino Flight Simulator* – multi-platformový, plne open-source [15].

6 Letecký simulátor – Brain Plane

Hlavným cieľom tejto práce je prezentovať možnosti novej mobilnej platformy Windows Phone 7 v oblasti hier. Hra leteckého simulátoru je preto vhodná z viacerých dôvodov. V hre sa ukáže rozsiahle prostredie, je možné využiť možnosti telefónu ako napr. akcelerometer a pod. Ďalšou výhodou je, že nie je nutne potrebné vytvárať umelú inteligenciu, čo výrazne zníži zložitosť problému.

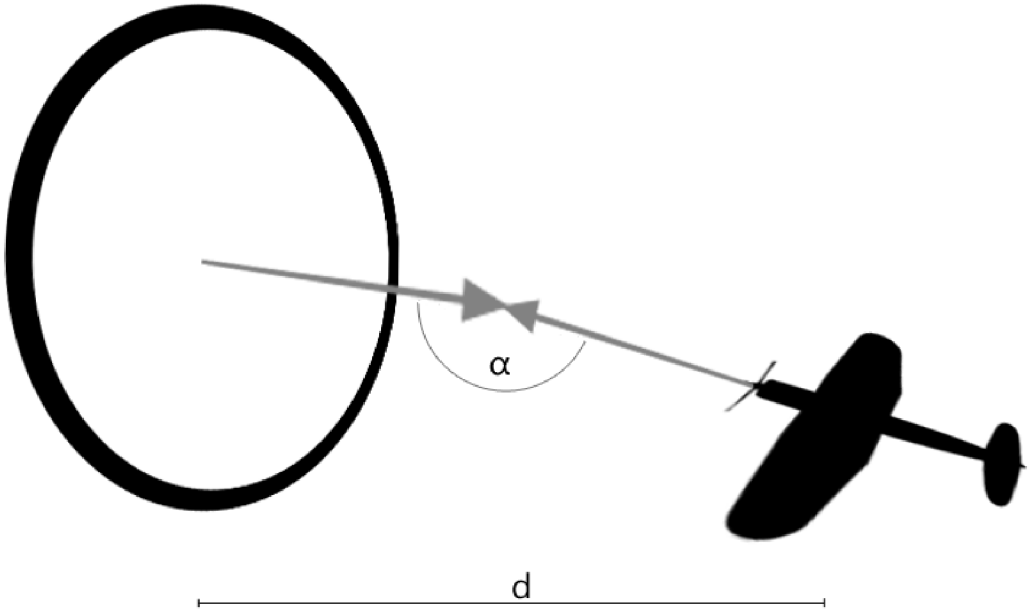
Celé riešenie je rozdelené na viacero projektov v programe Visual Studio 2010. Hlavným je samozrejme samotná hra, ďalej obsah k hre (modely, zvuky, textúry a pod.). V poradí tretím projektom je vlastná *pipeline* pre vytvorenie vlastných importujúcich algoritmov, napr. pre terén. Poslednou časťou je projekt s nastavením časticových efektov.

Táto práca bola takisto prihlásená do medzinárodnej súťaže Imagine Cup 2011 spoločnosti Microsoft v kategórii Windows Phone 7. Táto súťaž sa silne odzrkadľuje v princípe hry kvôli téme súťaže: "Predstavte si svet, kde technológie pomáhajú riešiť najpálčivejšie problémy". Preto je hra nenásilná a dôraz je skôr kladený na rozvíjanie niektorých intelektuálnych vlastností človeka. Súťaž pozostáva z troch kôl, kde v prvom bolo potrebné napísať krátku správu s popisom aplikácie. Túto správu je možné nájsť v prílohách tejto práce. Pri postúpení do druhého kola bolo nutné vytvoriť krátke video s prezentáciou aplikácie. Posledným, tretím kolom, je svetové finále, kde postupuje najlepších 5 tímov z celého sveta.

6.1 Princíp hry a úrovne

Hra sa skladá z dvoch úrovní:
V prvej si má hráč za úlohu zapamätať poradie piatich farieb, ktoré sa mu zobrazí pred začatím hry na dobu 7 sekúnd. Toto poradie sa náhodne mení každým spustením úrovne. Následne musí v hre preletieť 5 farebných kruhov v tom istom poradí. Táto úroveň je navrhnutá pre trénovanie tzv. krátkodobej pamäte, tzn. pamäte, ktorej obsah si človek zapamätá len na určitú chvíľku, pokiaľ ho potrebuje.

Prelet kruhom je signalizovaný na základe vzdialenosti lietadla od kruhu spolu s uhlom ktorý lietadlo zviera s osou kruhu. To znamená, že ak sa lietadlo nachádza v tesnej blízkosti kruhu a skalárny súčin smeru lietadla a osi kruhu sa blíži k jednej, lietadlo preletelo kruhom.



Obrázok 9 - Prelet lietadla kruhom

Výsledok úrovne spočíva v čo najkratšom čase, za ktorý hráč preletí kruhy v danom poradí. Ak sa náhodou netrafí do správnej farby kruhu, lietadlo vybuchne a lietadlo je premiestnené naspäť na počiatočnú pozíciu, čím hráč stratí množstvo času a zároveň môže stratiť prehľad o poradí farieb.

Druhou úrovňou v hre si hráč precvičuje paralelné spracovanie informácií. Hráč má za úlohu ovládať viacero lietadiel, medzi ktorými si môže prepínať a každé z nich musí dopraviť do kruhu, ktorý odpovedá farbe lietadla. Hra začína s dvoma lietadlami. Ďalšie lietadlo pribudne buď po 20 sekundách, alebo ak hráč trafí všetky aktívne lietadlá do kruhu. Ďalšie lietadlo pribudne po 19 sekundách, ďalšie po 18 sekundách atď. Maximálny počet lietadiel naraz je obmedzený na 7 kusov. Pri dosiahnutí hranice 3 sekundy na ďalšie lietadlo sa táto hranica ďalej neznižuje. Výsledok tejto úrovne spočíva v počte preletených kruhov, tzn. čím viac kruhov užívateľ preletí, tým lepšie. Ak jedno z lietadiel vybuchne, resp. ak sa dve lietadlá zrazia, hráč končí a zapíše sa mu aktuálne skóre.

Pri prepínaní lietadiel to funguje tak, že v tom stave, v akom ho hráč nechal, sa lietadlo pohybuje ďalej. Platí to pre rýchlosť, smer a rotáciu. Každé z týchto lietadiel si uchováva informácie o uložení dlaždíc terénu, aby bolo možné prepozicovať terén pri zmene ovládaného lietadla.

6.2 Objekty v hre

Samotná hra pozostáva s niekoľkých grafických, či negrafických objektov ako napr. samotné lietadlo, krajina, obloha, stromy, voda a pod.

6.2.1 Lietadlo

Hlavným a jediným pohybujúcim sa objektom v hre je model lietadla. Je to externý model vyrobený v programe 3D Studio MAX. Skladá sa z dvoch častí a to z tela lietadla a vrtule. Celý model je vo formáte FBX z dôvodu ľahkého importu pomocou základnej funkcie *content pipeline* v XNA. V modely je taktiež uložená jeho textúra, čím sa nanesenie textúry prenecháva zmienenej *content pipeline*.

Celý objekt je popísaný triedou `Plane.cs`. Okrem očakávaných vlastností ako sú napr. rýchlosť lietadla, smer, maximálna rýchlosť, váha atď. má lietadlo aj vlastnosti späté s frameworkom XNA. Sú to hlavne metódy `Update` a `Draw`, ktoré by mal mať každý grafický objekt. Ako názov napovedá, v nich sa každým vykreslením snímku aktualizuje objekt (rýchlosť, smer a pod.), resp. sa tu objekt vykresľuje.

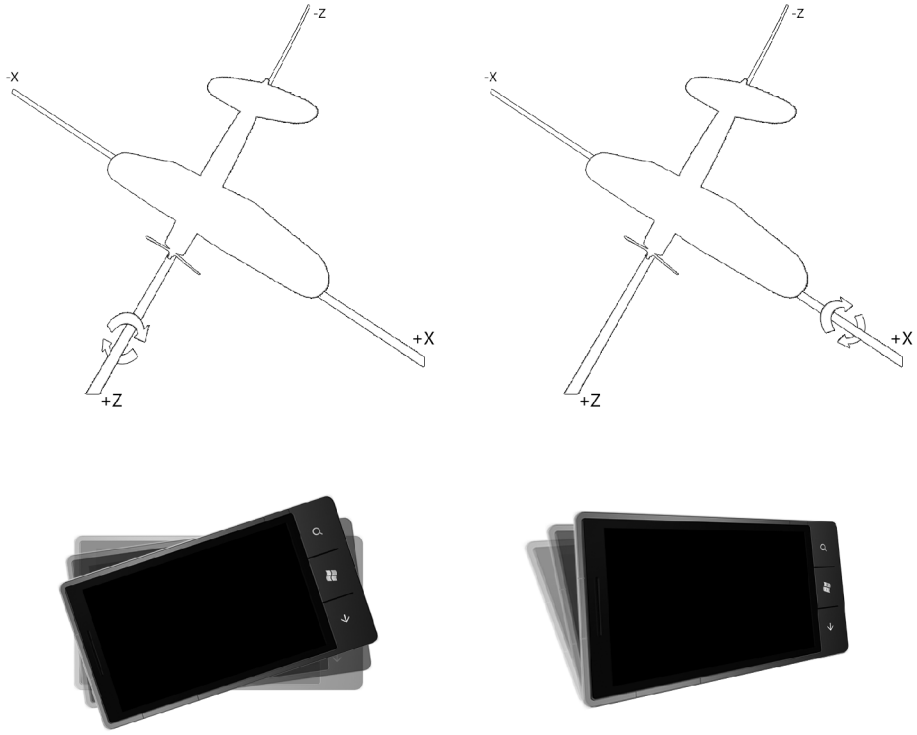
Pri vykresľovaní sa model najskôr rozdelí na objekty triedy `ModelMesh`. Sú to zhluky polygónov, ktoré spolu logicky súvisia. V prípade lietadla je to napríklad vrtuľa, telo lietadla, kolesá atď. Týmto sa dá zaručiť otáčanie vrtule nezávisle od zvyšku lietadla. Tieto objekty triedy `ModelMesh` sa ďalej rozdeľujú na objekty typu `BasicEffect`, čo už sú jednotlivé polygóny, ktorým je potrebné nastaviť transformácie a následne vykresliť celý `ModelMesh`. Takto to funguje pri každom vykresľovaní modelu v XNA, takže pri ďalších objektoch už nebudem spomínať detailný proces vykresľovania. Proces kreslenia vyzerá v jednoduchosti nasledovne:

```
Matrix[] transforms = new Matrix[plane.Bones.Count];
plane.CopyAbsoluteBoneTransformsTo(transforms); //aktuálna transformácia
foreach (ModelMesh mesh in plane.Meshes)
{
      foreach (BasicEffect effect in mesh.Effects)
      {
            //nastavenie matíc pre model
            effect.View = view;
            effect.Projection = projection;
            effect.World = transforms[mesh.ParentBone.Index] * world;
            //nastavenie svetla a farby pre model
            effect.EnableDefaultLighting();
            effect.SpecularColor = new Vector3(0.6f, 0.4f, 0.2f);
            effect.SpecularPower = 8;
      }
      mesh.Draw();
}
```

Túto funkciu je možné použiť spravidla na akýkoľvek model, preto ju už pri ďalších 3D objektoch v hre nebudem uvádzať.

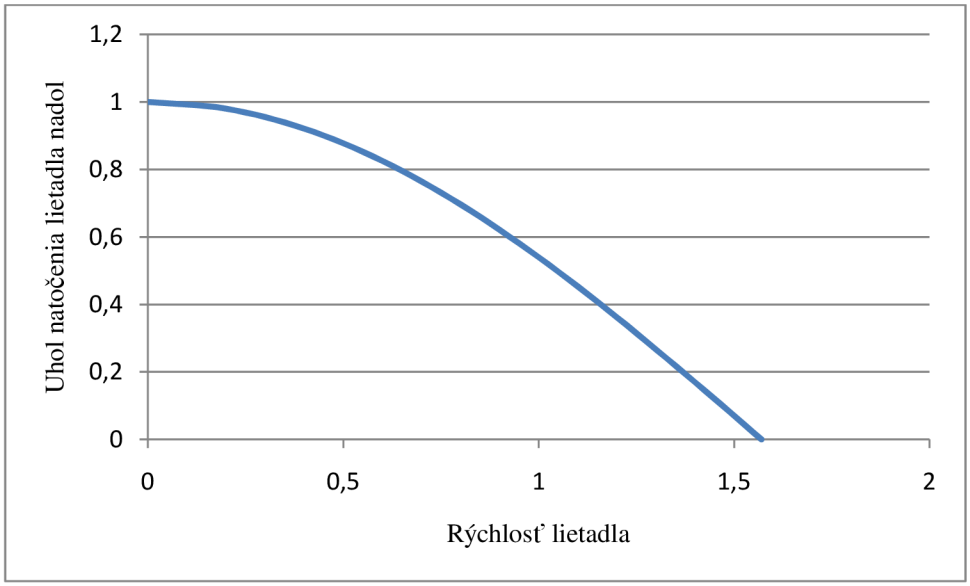
Pri ovládaní lietadla je vhodné využiť potenciál telefónu, najmä akcelerometer, ktorý sa na ovládanie lietadla hodí veľmi dobre. V XNA je API pre akcelerometer robené na spôsob udalostí, tzn. že akonáhle sa zmení hodnota akcelerometra, vyvolá sa interná udalosť, v ktorej je možné odchytiť

hodnoty akcelerometra. V projekte sa o to stará statická trieda `Accelerometer.cs`. Model lietadla využíva dve z troch hodnôt akcelerometra – X a Z. Pomocou naklonenia telefónu vľavo, resp. vpravo sa lietadlo točí okolo svojej osi Z. Nakláňaním telefónu vpred a vzad sa točí okolo svojej lokálnej osi X.



Obrázok 10 - Ovládanie lietadla pomocou akcelerometra

V spomínanej metóde `Update` sa aktualizuje celá transformácia lietadla. Lietadlo ide automaticky vpred bez potreby interakcie hráča. Rýchlosť lietadla závisí aj od toho, či lietadlo klesá, resp. stúpa. K rýchlosti lietadla je prirátaný normalizovaný rozdiel aktuálnej a predchádzajúcej výšky lietadla. Pre brzdenie je potrebné dotknúť sa niektorej časti na pravej strane obrazovky. Ak lietadlo nemá dostatočnú rýchlosť, začne sa klopiť dole a padať. Pre túto operáciu je použitá goniometrická funkcia kosínus, ktorá má ako argument rýchlosť lietadla a ktorá udáva uhol naklápania lietadla pri spomaľovaní. Z priebehu funkcie kosínus je zrejmé, že čím bude rýchlosť lietadla menšia, uhol naklopenia lietadla bude väčší.



Obrázok 11 - Uhol naklápania lietadla nadol

Výsledná rotácia lietadla sa počíta z troch jednoduchších rotácií – natočenie okolo osi X, natočenie okolo osi Z a natočenie spôsobené spomaľovaním lietadla. Matica rotácie sa spočíta pomocou internej funkcie `CreateFromAxisAngle`, ktorá má ako parametre os, okolo ktorej sa má otáčať a uhol natočenia:

```
Matrix.CreateFromAxisAngle(GravityVector, gravityAngle) *
Matrix.CreateFromAxisAngle(Direction, Rotation.Z) *
Matrix.CreateFromAxisAngle(Right, Rotation.Y);
```

Kde rotácie okolo X a Z sú počítané pomocou hodnôt akcelerometra následovne:

```
Rotation -= new Vector3((((int)(currentAccelerometerState.X)) +
defaultAccState) * 0.07f, 0, 0);
Rotation -= new Vector3(0, 0, angularVelocity *
currentAccelerometerState.Y * 3);
```

K hodnote X vektora rotácie je nutné prirátať konštantu `defaultAccState`, ktorá je rovná 0.5 z dôvodu vyváženého stavu lietadla v jemnom naklonení. Inak by rovnovážny stav bol pri vodorovnom položení telefónu, čo je pre užívateľa nevhodné.

Lietadlo samotné navyše obsahuje ďalšie objekty ako sú kruh, cez ktorý ma lietadlo preletieť, šípku ktorá ukazuje na lietadlo, keď je aktívne iné lietadlo, zvuky lietadla a taktiež s ním súvisí kamera.

6.2.2 Kamera

Kameru v projekte zastupuje trieda `ChaseCamera.cs` [28]. Ide o kameru, ktorá sa drží za svojím objektom, preto má rôzne vlastnosti ako pružnosť (udáva ako sa môže kamera od objektu vzdialiť), pozíciu sledovaného objektu, jeho smer a pod. Na základe týchto údajov sa počíta projekčná a pohľadová matica celej hry:

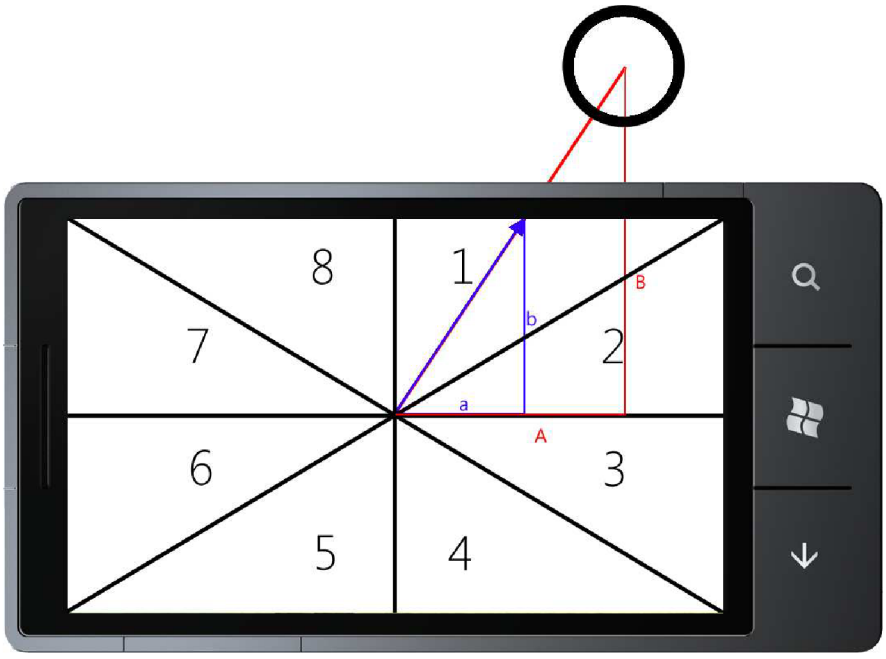
```
Matrix transform = Matrix.Identity;
transform.Forward = ChaseDirection;
transform.Up = Up;
transform.Right = Vector3.Cross(Up, ChaseDirection);
desiredPosition = ChasePosition +
Vector3.TransformNormal(DesiredPositionOffset, transform);

lookAt = ChasePosition + Vector3.TransformNormal(LookAtOffset, transform);
view = Matrix.CreateLookAt(this.Position, this.LookAt, this.Up);
projection = Matrix.CreatePerspectiveFieldOfView(FieldOfView, AspectRatio,
NearPlaneDistance, FarPlaneDistance);
```

Kde vektor `Up` je totožný s vektorom `Up` lietadla. Vektor `desiredPosition` je pozícia kamery, ktorá je od lietadla posunutá o hodnotu `DesiredPositionOffset`. Vektor `lookAt` udáva smer, ktorým sa kamera pozerá upravený o vektor `LookAtOffset`. Pomocou týchto vektorov je možné vyrátať projekčnú a pohľadovú maticu.

6.2.3 Kruh, cez ktorý má lietadlo preletieť

V hre sa v rôznych úrovniach nachádzajú kruhy, cez ktoré má lietadlo preletieť. Tieto kruhy sú tvorené pomocou tried `GeometricPrimitive.cs` a `TorusPrimitive.cs`. Trieda `GeometricPrimitive.cs` slúži na zoskupenie geometrických primitív. Obsahuje zásobník vrcholov, metódy na vykreslenie a pod. Trieda `TorusPrimitive.cs` dedí túto triedu. Navyše obsahuje inicializáciu vrcholov kruhu do zásobníka vrcholov.

Objekt kruh, takisto ako lietadlo, obsahuje objekt šípky, ktorá naň ukazuje. Ide o 2D šípku, ktorá sa pohybuje po okraji obrazovky telefónu a ukazuje, ktorým smerom je kruh. Pre určenie pozície a rotácie šípky je nutné prepočítať súradnice kruhu v scéne na súradnice obrazovky.



Obrázok 12 - Výpočet pozície šípky ku kruhu

Obrazovku je možné rozdeliť na 8 rovnakých častí, v ktorých sa pozícia šípky počíta zvlášť. Interná metóda triedy `Viewport Project` premietne 3D súradnice objektu v scéne na súradnice obrazovky telefónu. Pomocou nich je potrebné vypočítať pozíciu a rotáciu šípky na kraji obrazovky. Premietnuté súradnice sú vo formáte 3D vektoru z dôvodu, že súradnica Z určuje podľa znamienka, či sa objekt nachádza pred, resp. za kamerou. Na základe týchto údajov je možné najskôr vypočítať rotáciu šípky:

Vektor smerujúci hore:

$$v_1 = (0,1,0) \tag{6.1}$$

Vektor smerujúci vpravo (pomôže pri počítaní uhlu):

$$v_2 = (1,0,0) \tag{6.2}$$

Vektor výsledky metódy `Project`, teda ten, ktorý ukazuje na kruh na obrazovke (poprípade mimo obrazovky):

$$v_3 = (x, y, 0) \tag{6.3}$$

Spočítame uhol so znamienkom:

$$a = v_1 . v_2 \tag{6.4}$$

$$b = v_1 . v_3 \tag{6.5}$$

$$uhol = acos(a) * sign(b) \tag{6.6}$$

Pozícia šípky na okraji obrazovky sa dá vypočítať z podobnosti trojuholníkov znázornených na obrázku (viď Obrázok 12 - Výpočet pozície šípky ku kruhu). Najskôr je ale potrebné zistiť, v ktorej časti obrazovky sa má šípka nachádzať. To je možné pomocou súradníc a následne pomocou smernice od stredu obrazovky. Pomocou podobnosti trojuholníkov je možné vypočítať súradnice šípky na okraji obrazovky (príklad pre prvú časť obrazovky):

$$\frac{a}{b} = \frac{A}{B} \tag{6.7}$$

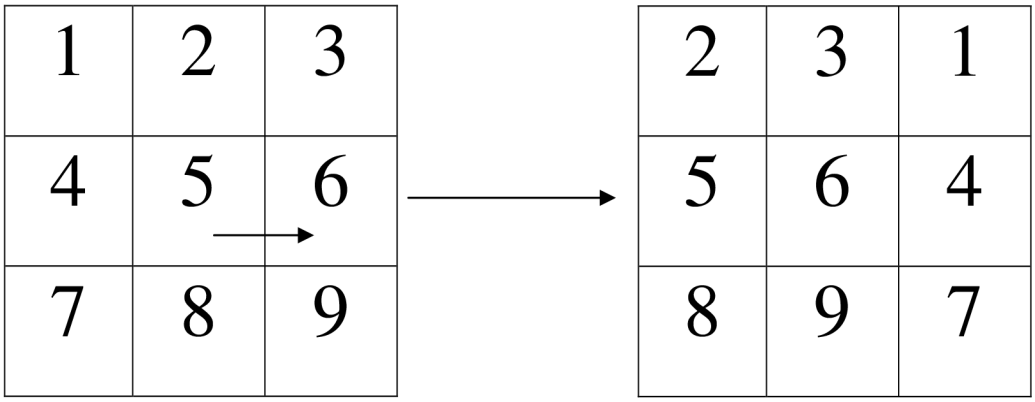
$$b = \frac{disp\_width}{2} \tag{6.8}$$

$$a = b * \frac{A}{B} \tag{6.9}$$

Pretože výpočet prebieha so stredom v bode [0,0], je na konci výpočtu potrebné pričítať vektor `halfDisp`, ktorý obsahuje polovičné rozlíšenie obrazovky. Podobne sa to počíta aj pre ďalších 7 častí obrazovky.

6.2.4 Terén

Hra typu letecký simulátor vyžaduje rozsiahlu scénu. Preto som zvolil možnosť teoreticky neobmedzenej scény s nevýhodou opakovania sa. Terén v hre sa skladá z 9 častí – dlaždíc, ktoré sú usporiadané do štvorca o rozmeroch 3 x 3. Lietadlo sa vždy nachádza nad prostrednou dlaždicou. Pri preletení na ďalšiu sa 3 zadné dlaždice preklopia tak, aby lietadlo bolo znova nad prostrednou dlaždicou.



Ak očíslujeme jednotlivé terény od 1 do 9 a lietadlo sa nachádza nad strednou, čiže piatou dlaždicou a prechádza na v poradí šiestu dlaždicu, dlaždice 1, 4 a 7 sa preklopia tak, aby šiesta dlaždica, čiže tá, nad ktorou je aktuálne lietadlo, bola znova v strede. Podobne to funguje aj pre iné smery. Tým sa dá zabezpečiť "nekonečnosť" terénu, i keď za cenu opakovania, čo ale pri veľkosti dlaždíc nie je až tak podstatné.

Tento systém nemusí byť výhradne obmedzený iba na rozloženie dlaždíc 3x3. Preto je vhodnejší matematický popis algoritmu. Terén pozostáva z niekoľkých dlaždíc usporiadaných do štvorca. Toto usporiadanie reprezentuje maticu ***M***:

$$\begin{bmatrix} x_{00} & \cdots & x_{i0} \\ \vdots & \ddots & \vdots \\ x_{0j} & \cdots & x_{ij} \end{bmatrix} \qquad 6.10$$

Kde i > 1, j > 1 ∧ i, j sú nepárne. Lietadlo sa nachádza vždy v strede matice, tzn. nad dlaždicou:

$$x_{\frac{i}{2}\frac{j}{2}} \qquad 6.11$$

Pri preletení lietadla na niektorú susednú dlaždicu sa matica ***M*** zmení na maticu ***M':***

$$\begin{bmatrix} x_{10} & \cdots & x_{i0} & x_{00} \\ \vdots & \ddots & \vdots & \vdots \\ x_{1j} & \cdots & x_{ij} & x_{0j} \end{bmatrix} \qquad 6.12$$

Tento stav nastane pri prechode lietadla zľava vpravo pri pohľade na maticu zvrchu. Podobne to platí aj pre ostatné tri prechody:

Sprava vľavo:

$$\begin{bmatrix} x_{i0} & x_{00} & \cdots & x_{(i-1)0} \\ \vdots & \vdots & \ddots & \vdots \\ x_{ij} & x_{0j} & \cdots & x_{(i-1)j} \end{bmatrix} \quad 6.13$$

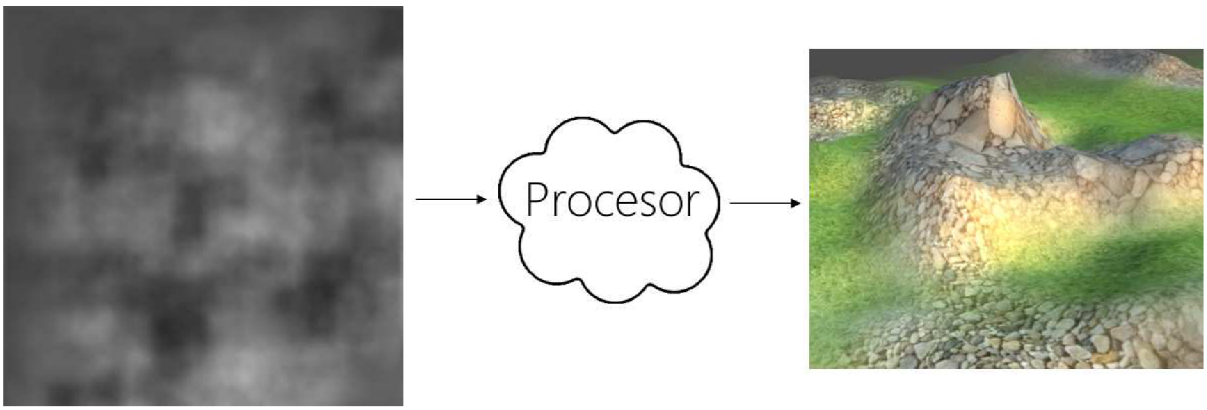
Zhora dole:

$$\begin{bmatrix} x_{01} & \cdots & x_{i1} \\ \vdots & \ddots & \vdots \\ x_{0j} & \cdots & x_{ij} \\ x_{00} & \cdots & x_{i0} \end{bmatrix} \quad 6.14$$

Zdola hore:

$$\begin{bmatrix} x_{0j} & \cdots & x_{ij} \\ x_{00} & \cdots & x_{i0} \\ \vdots & \ddots & \vdots \\ x_{0(j-1)} & \cdots & x_{i(j-1)} \end{bmatrix} \quad 6.15$$

Dlaždice terénu sú tvorené pomocou tzv. výškovej mapy – čiernobiely obrázok popisujúci reliéf terénu na základe farieb – čím belšia farba, tým vyšší kopec. Tento obrázok je spracovávaný explicitným procesorom [26].



Obrázok 13 - Tvorba terénu z výškovej mapy

Procesor má ako vstup 2D textúru – výškovú mapu. Jeho výstupom je objekt triedy `TerrainContent`, ktorý je následne serializovaný na objekt triedy `Terrain` v samotnej hre. Postup prevodu výškovej mapy na terén je nasledovný: Obrázok sa reprezentuje ako pole desatinných čísiel. Následne sa pomocou parametrov `terrainScale` a `terrainBumpiness` naplní pole vektorov, ktoré obsahujú súradnice a výšku na danej súradnici. Spolu s tým sa ukladajú iba samotné výšky do poľa, ktoré je súčasťou triedy `TerrainContent`. Objekt triedy `MeshBuilder` vytvorí nové body pre vykresľovanie. Ďalej je potrebné z týchto bodov vytvoriť trojuholníky pomocou metódy `AddVertex`. Popri vytváraní bodov sa týmto bodom nastavuje aj materiál, ktorý obsahuje pred pripravenú textúru pre každú dlaždicu terénu zvlášť. Keďže platforma Windows Phone 7 zatiaľ podporuje len duálne multitexturovanie, bolo potrebné vytvoriť pomerne veľkú textúru, ktorá presne sadne na každú dlaždicu. Ako bolo spomenuté, výsledkom procesoru je objekt triedy

`TerrainContent`, ktorý obsahuje objekt triedy Model, ktorý predstavuje samotný model terénu, ďalej pole výšok terénu, aby bolo možné zistiť, či sa objekt na daných súradniciach nachádza pod terénom. Ďalšími vlastnosťami terénu sú jeho šírka (rovnaká ako výška), mierka a kopcovitosť. Tento objekt sa pomocou tzv. `ContentSerializer`-u prevedie na objekt triedy `Terrain` priamo v hre.

Výpočet preklápania terénu sa deje priamo v objekte lietadla. Je to z dôvodu možnosti ovládania viacerých lietadiel, čiže každé musí mať svoje vlastné výpočty. Lietadlo má informácie o aktuálnej pozícii a taktiež o aktuálnej pozícii v rámci všetkých dlaždíc, ktorá sa vypočíta jednoduchou operáciou modulo z aktuálnej pozície v scéne. Z nej sa dá ľahko určiť nad ktorou pozíciou sa lietadlo nachádza. Potrebné je ale ošetriť možnosť pohybu lietadla do záporných hodnôt, kedy sa výpočet zrkadlí.

Pre posun dlaždíc existujú 4 rôzne spôsoby vzhľadom na orientáciu terénu:

- Lietadlo ide zľava vpravo,
- lietadlo ide sprava vľavo,
- lietadlo ide zhora dole,
- a lietadlo ide zdola hore.

Pre každú možnosť existuje mierne odlišný výpočet. Napríklad pre smer zdola hore:

```
if (old_x < x) // signalizuje prechod zdola hore
{
      a = (a + 2) % 3; //index do poľa pozícií dlaždíc
      for (int i = 0; i < 3; i++)
      {
            Vector3 actualPosition = TerrainPositions[i, a];
            //posunieme dlaždicu
            TerrainPositions[i, a] = new Vector3(actualPosition.X +=
terrain_width * 3, actualPosition.Y, actualPosition.Z);
      }
}
```

6.2.5 Stromy

Pre vyššiu reálnosť prostredia sú do hry vložené takisto jednoduché stromy. Sú to modely vyrobené, podobne ako lietadlo, v programe 3D Studio MAX vo formáte fbx, pre ktorý má framework XNA predvolený importér. Windows Phone 7, ako je zmienené v úvode, zatiaľ nepodporuje možnosť programovania shaderov, preto nie je možné využiť tzv. instancing, čiže techniku vykresľovania toho istého modelu niekoľkokrát za účelom zrýchlenia vykresľovania. Preto je v hre stromov o niečo menej, ako by v skutočnosti malo byť.

Ako bolo spomenuté, stromy sú jednoduché modely, ktoré sú v oblasti koruny stromu priehľadné. Pri vykresľovaní priehľadných objektov striktne závisí na poradí vykresľovania, preto je

potrebné stromy vykresľovať od vzdialenejších objektov ku kamere k bližším. Tým pádom je nutné stromy každou aktualizáciou scény zoraďovať.

Podobne ako dlaždice terénu, má lietadlo informácie aj o pozíciách stromov v scéne. To znamená, že ak sa preklápa dlaždica terénu, preklopia sa s ňou aj všetky stromy na tomto teréne.

Objekt stromu vzniká z triedy `Tree.cs`, ktorá má takisto metódy `Draw` a `Update`. V metóde `Update` sa aktualizuje vzdialenosť stromu od kamery a v metóde `Draw` sa daný strom vykresľuje.

6.2.6 Voda

Ďalším prvkom scény je vodná hladina. Tá sa bežne tvorí pomocou shaderov, ale keďže, ako už bolo viackrát povedané, Windows Phone 7 zatiaľ možnosť shaderov neponúka, bolo nutné prejsť k inému riešeniu. Voda je preto jednoducho spravená jednou veľkou plochou, ktorá sa nachádza presne pod lietadlom, čiže lietadlo je nad jej stredom, a spolu s lietadlom sa pohybuje. Aby ale nebolo možné spozorovať tento pohyb, menia sa textúrovacie súradnice plochy, čím je vytvorený efekt nekonečnej vodnej hladiny. Textúra vodnej plochy sa pohybuje rovnakou rýchlosťou akou letí lietadlo, ale v opačnom smere. Textúra má mimo tento pohyb aj svoj vlastný pohyb, ktorý tvorí efekt tečúcej vody:

```
//zmena textúrovacích súradníc na základe času (tok vody) a pohybu
//lietadla(kamery)
Vector2 topLeft = new Vector2(0.0f + time - cameraOffset.X, 0.0f +
cameraOffset.Z);
Vector2 topRight = new Vector2(1.0f + time - cameraOffset.X, 0.0f +
cameraOffset.Z);
Vector2 bottomLeft = new Vector2(0.0f + time - cameraOffset.X, 1.0f +
cameraOffset.Z);
Vector2 bottomRight = new Vector2(1.0f + time - cameraOffset.X, 1.0f +
cameraOffset.Z);
```

6.2.7 Obloha

Podobne ako terén, aj obloha využíva explicitný procesor pre načítanie [26]. Jeho vstupom je 2D textúra, ktorá sa má mapovať ako obloha a výstupom objekt triedy `SkyContent`, ktorý je následne serializovaný na objekt triedy `Sky` priamo v hre. Činnosť procesoru spočíva vo vytvorení uzavretého valca, ktorý je zvnútra textúrovaný vstupným obrázkom. Tvorba vrcholov je robená pomocou objektu triedy `MeshBuilder`.

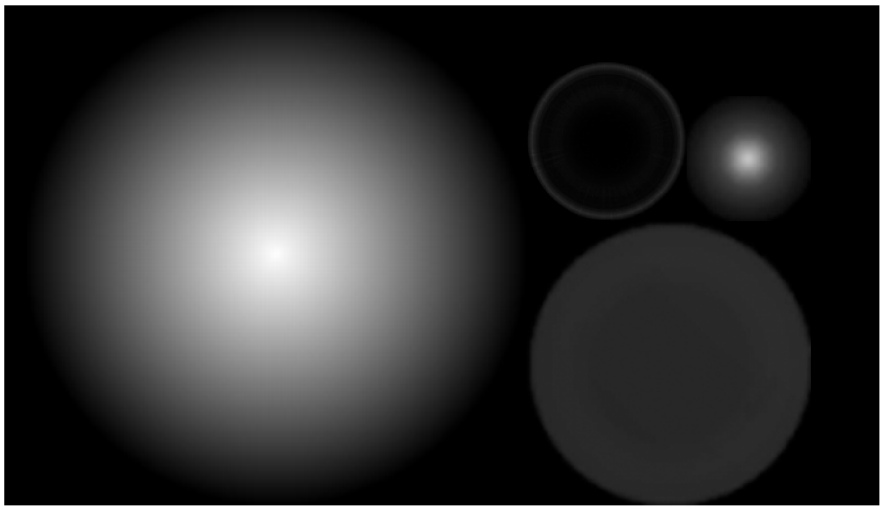
```
for (int i = 0; i < cylinderSegments; i++)
{
      float angle = MathHelper.TwoPi * i / cylinderSegments;
      float x = (float)Math.Cos(angle) * cylinderSize;
      float z = (float)Math.Sin(angle) * cylinderSize;
      topVertices.Add(builder.CreatePosition(x, cylinderSize, z));
      bottomVertices.Add(builder.CreatePosition(x, -cylinderSize, z));
}
```

Výsledkom v hre je objekt triedy `Sky` obsahujúci samotný model valca spolu s textúrou. Pri oblohe je nutné aby sa stále zobrazovala za všetkými objektmi v oblasti najvzdialenejšej vykresľovanej roviny (far clip plane). Tento efekt sa dá dosiahnuť upravením projekčnej matice pri vykresľovaní oblohy:

```
projection.M13 = projection.M14;
projection.M23 = projection.M24;
projection.M33 = projection.M34;
projection.M43 = projection.M44;
```

6.2.8 Odlesk objektívu (Lens flare)

Súčasťou grafických prvkov v hre je aj odlesk objektívu (lens flare). Je to známy efekt systému šošoviek fotoaparátov. V projekte ho predstavuje trieda `LensFlare.cs` [29]. Táto trieda obsahuje internú pomocnú triedu `Flare`, ktorá reprezentuje jeden prstenec celého odlesku. Celý odlesk sa skladá z desiatich takýchto prstencov a z jedného žiarivého kruhu.



Obrázok 14 - Žiarivý kruh a prstence odlesku objektívu

Pri tvorbe odlesku je potrebné nastaviť pozíciu slnka a následne pozíciu všetkých prstencov, ktorá sa udáva ako desatinné číslo. Pozícia prstenca sa počíta nasledovne: Najskôr je potrebné zistiť, či sa slnko nachádza na obrazovke telefónu. To je možné pomocou metódy Project objektu triedy Viewport, ktorá vracia vektor so súradnicami na obrazovke spolu s informáciou, či sa objekt nachádza pred kamerou alebo nie. Ak sa slnko nachádza pred kamerou, vypočítajú sa jeho súradnice na obrazovke. Ďalej sa vypočíta vektor od slnka do stredu obrazovky, po ktorom sa pohybujú prstence odlesku. Pozícia prstenca na tomto vektore je určená práve zmienenou inicializačnou hodnotou v tvare desatinného čísla. Žiarivý kruh sa vykresľuje podobne ako prstence, ale pozíciu má pevne danú na pozícii slnka.

```
//nový prstenec s pozíciou 0.5 a mierkou 0.7
flare = new Flare(0.5f, 0.7f, new Color( 50,  25,  50), "flare1"),
//stred obrazovky
Vector2 screenCenter = new Vector2(viewport.Width, viewport.Height) / 2;
//vektor, po ktorom sa pohybujú prstence
Vector2 flareVector = screenCenter - lightPosition;
//pozícia prstenca na základe jeho inicializačnej pozície
Vector2 flarePosition = lightPosition + flareVector * flare.Position;
```

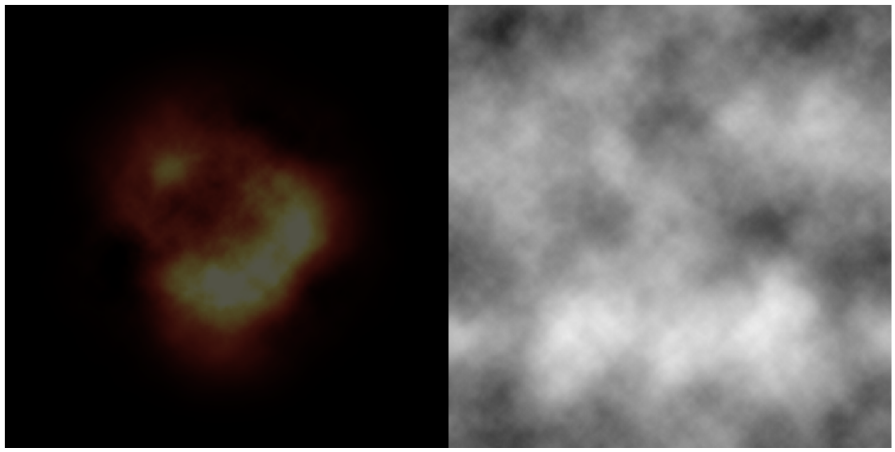
6.3 Cesta častíc (Particles pipeline)

Systém častíc (Particle system) je technika pre vykresľovanie špeciálnych efektov, typicky organických, tekutých alebo prírodných. V hrách sa často používajú pre vykreslenie dymu, ohňa, iskier a pre špliechanie vody.

Časticový systém pozostáva spravidla z viacerých malých častíc. Každá táto častica má svoje vlastné fyzikálne vlastnosti, typicky pozíciu, rýchlosť, akceleráciu a pod. Komplexnejšie systémy môžu obsahovať oveľa viac vlastností. Častice sú inicializované počiatočnými vlastnosťami, ktoré sú dané celým systémom, ale v momente ako sa systém rozbehne, každá častica sa správa nezávisle. Častice sú typicky vykresľované ako 2D priehľadné textúry. Keď ich je dostatočne veľa a vykresľujú sa rôzne cez seba, vyzerá to ako chaotický a prirodzený systém.

Bežne sa častice, hlavne v 3D prostredí, tvoria pomocou shaderov. Tie Windows Phone 7 žiaľ nepodporuje, takže je nutné prísť na iné riešenie. Častice, ktoré by sa pohybovali v 3D prostredí by boli bez možnosti využitia shaderov príliš náročné. Preto je jediným vhodným riešením, i keď za cenu kvality, zvoliť 2D riešenie v podobe tzv. spritov. V hre je len jediný časticový efekt v podobe výbuchu lietadla. Ten sa hráčovi zobrazí len na chvíľku a to v momente, keď sa hra prestane hýbať. To znamená, že to že sú častice 2D bude ťažšie postrehnuteľné [27].

Výbuch lietadla sa skladá z dvoch častí: z dymu a z ohňa. Preto je potrebné vytvoriť časticový systém, ktorý sa skladá z dvoch podsystémov.



Obrázok 15 - Textúra ohňa a dymu

Platforma Windows Phone zatiaľ nepodporuje žiadne rozhranie pre tvorbu časticových systémov. Preto je nutné vytvoriť nový v podstate od nuly. V projekte som zvolil spôsob načítania vlastností častíc z XML súboru. Je to z dôvodu lepšej rozšíriteľnosti do budúcna a ďalej je možné jednoduchou zmenou XML súboru vytvoriť úplne iný časticový systém. Tento XML súbor obsahuje informácie o počiatočných nastaveniach systému. Tie sa spravidla udávajú v nejakom intervale, napr. počiatočná rýchlosť častice musí byť v intervale od 40 do 500. Tým sa do systému vloží akási náhodnosť, ktorá prispieva k reálnosti systému. Okrem rýchlosti sa v XML súbore nastavujú ďalšie vlastnosti: počet častíc v systéme, textúra častice, rýchlosť rotácie, koncová rýchlosť častice, životnosť častice a veľkosť častice. Výsledné XML pre výbuch vyzerá takto:

```
<?xml version="1.0" encoding="utf-8"?>
<XnaContent>
  <Asset Type="ParticlesSettings.ParticleSystemSettings">
    <MinNumParticles>10</MinNumParticles>
    <MaxNumParticles>12</MaxNumParticles>
    <TextureFilename>explosion</TextureFilename>
    <MinInitialSpeed>40</MinInitialSpeed>
    <MaxInitialSpeed>500</MaxInitialSpeed>
    <AccelerationMode>EndVelocity</AccelerationMode>
    <EndVelocity>0</EndVelocity>
    <MinRotationSpeed>-45</MinRotationSpeed>
    <MaxRotationSpeed>45</MaxRotationSpeed>
    <MinLifetime>.5</MinLifetime>
    <MaxLifetime>1.0</MaxLifetime>
    <MinSize>.3</MinSize>
    <MaxSize>1.0</MaxSize>
    <SourceBlend>SourceAlpha</SourceBlend>
    <DestinationBlend>One</DestinationBlend>
  </Asset>
</XnaContent>
```

`AccelerationMode EndVelocity` znamená, že akcelerácia častice sa počíta na základe veku častice a jej počiatočnej a koncovej rýchlosti.

Pre načítanie XML je potrebné vytvoriť zodpovedajúcu triedu, ktorá zahŕňa všetky vlastnosti daného XML. Ide o jednoduchú triedu, kde sú vymenované všetky časti XML súboru, ktoré sú serializované. Táto trieda sa v projekte volá `ParticleSystemSettings.cs` a nachádza sa v podprojekte `ParticleSettings`. Serializácia jednej vlastnosti môže byť voliteľná, pretože sa nemusí hodiť pre každý časticový systém:

```
[ContentSerializer(Optional = true)]
public float MinDirectionAngle = 0;
[ContentSerializer(Optional = true)]
public float MaxDirectionAngle = 360;
```

Tieto nadstavenia z XML súboru sa načítajú obdobne ako všetky iné prvky v hre – cez Content Pipeline frameworku XNA.

Samotný časticový systém sa skladá z niekoľkých tried – `Particle.cs`, `ParticleSystem.cs` a `ParticleHelpers.cs`. Ako názov napovedá, prvá z nich predstavuje objekt jednej častice, druhá objekt celého časticového systému. Tretia obsahuje pomocné metódy pre zobrazovanie.

Najhlavnejšou triedou pre zobrazovanie časticového systému je `ParticleSystem.cs`. Pomocou nej sa tvorí objekt systému definovaný nastaveniami v XML súbore. Taktiež obsahuje `Update` a `Draw` metódy pre aktualizáciu, resp. vykresľovanie celého systému.

Pri vytvorení objektu triedy `ParticleSystem` sa vytvoria jednotlivé častice. Tie sa hneď vložia do radu, ktorý má optimalizačný účel. Tento rad uchováva častice, ktoré sa nepoužívajú. To znamená, že pri vyvolaní napr. výbuchu sa použijú častice z tohto radu. Po dokončení systému výbuchu sa jednotlivé častice znovu vrátia do zmieneného radu. Tým je zaručené, že sa nebudú pri každom výbuchu vytvárať nové a nové častice. Trieda ďalej obsahuje dôležitú metódu `AddParticles`, ktorá má ako parametre vektor, ktorý určuje miesto na obrazovke a rýchlosť častice. Táto metóda vyvoláva štart časticového systému. Ak sú voľné častice v rade voľných častíc, tieto sa nainicializujú pomocou nastavení z XML súboru. Ak nie sú, tak sa vytvoria nové. Inicializácia častíc prebieha v metóde `InitializeParticle`. Pomocou intervalov hodnôt a náhodnosti sa každá častica vytvorí ako jedinečná.

```
velocity *= settings.EmitterVelocitySensitivity;
Vector2 direction = PickRandomDirection();
float speed = ParticleHelpers.RandomBetween(settings.MinInitialSpeed,
settings.MaxInitialSpeed);
velocity += direction * speed;
float lifetime = ParticleHelpers.RandomBetween(settings.MinLifetime,
settings.MaxLifetime);
float scale = ParticleHelpers.RandomBetween(settings.MinSize,
settings.MaxSize);
float rotationSpeed =
ParticleHelpers.RandomBetween(settings.MinRotationSpeed,
settings.MaxRotationSpeed);
rotationSpeed = MathHelper.ToRadians(rotationSpeed);
Vector2 acceleration = Vector2.Zero;
acceleration = (velocity * (settings.EndVelocity - 1)) / lifetime;
```

Ako bolo spomenuté vyššie, v hre je len jeden časticový efekt a to výbuch lietadla, ktorý sa skladá z dvoch podsystémov. Vďaka návrhu časticového systému je nutné len vytvoriť dva objekty, ktoré načítajú nastavenia dymu, resp. ohňa z XML súboru a pri kolízii lietadla s terénom zavolať metódu `AddParticles`. Kolízia s lietadla s terénom je na základe výškových údajov samotného terénu, ktoré obsahujú konkrétnu výšku na daných súradniciach.

6.4 Zvuky v hre

Hra samozrejme obsahuje aj zvuky. Spolu ich je v aplikácii 9:

- Motor lietadla,
- výbuch lietadla,
- oznámenie o novom lietadle v hre,
- prelet kruhom (6):
  - *amazing,*
  - *good work,*
  - *very well,*
  - *splendid,*
  - *nice,*
  - *excelent.*

Všetky zvuky sú vo formáte *wav* a sú načítavané pomocou predvoleného importéra a procesora frameworku XNA. XNA ponúka dve triedy pre prácu so zvukmi – `SoundEffect` a `SoundEffectInstance`. Pre jednoduché prehrávanie zvukov, bez potrebných efektov, je vhodné použiť len triedy `SoundEffect`. Tá má metódu `Play`, ktorá iba prehrá to, čo naozaj obsahuje *wav* súbor bez úprav. Všetky zvuky preletu kruhom, oznámenie o novom lietadle a výbuch sú spúšťané touto metódou. Motor lietadla potrebuje využitie slučky, tzn. že musí hrať neustále dokola. Ďalej je nutné zvyšovať, resp. znižovať výšku tónu keď lietadlo spomaľuje, resp. zrýchľuje. Preto je nutné využiť spomínanú triedu `SoundEffectInstance`, ktorej objekt sa tvorí z objektu triedy `SoundEffect`. Tento objekt už obsahuje vlastnosti ako použitie slučky, zmenu výšky tónu, poprípade zmenu hlasitosti. Motor lietadla vyžaduje meniť všetky tieto vlastnosti. Výška tónu sa mení s rýchlosťou lietadla, hlasitosť sa mení, ak letí okolo práve ovládaného lietadla ďalšie lietadlo, tzn. že hlasitosť jeho motora sa mení v závislosti vzdialenosti od kamery.

```
float volume = 50 / Vector3.Distance(camPosition, this.Position);
```

Všetky tieto vlastnosti nadobúdajú hodnoty od -1 po 1. Z toho vyplýva, že ak je lietadlo ku kamere bližšie ako 50, motor hrá najhlasnejšie, inak postupne stráca na intenzite.

6.5 Menu hry a obrazovky

V každej hre sa prepína niekoľko typov obrazoviek – od niekoľkých menu obrazoviek až po hraciu obrazovku. V aplikácii sa o jednoduché prepínanie medzi obrazovkami stará niekoľko pomocných tried. Hlavnou z nich je trieda `ScreenManager.cs` [25]. Táto trieda manažuje všetky ostatné obrazovky, pridáva ďalšie, odoberá stávajúce a pod. Trieda dedí `DrawableGameComponent` triedu z frameworku XNA. Pri spúšťaní hry sa vytvorí práve objekt triedy `ScreenManager` a vložia sa doň

objekty tried, ktoré tvoria menu. Trieda `ScreenManager` teda obsahuje zoznam aktuálnych obrazoviek, ktoré sa majú zobrazovať. Taktiež obsahuje pomocné objekty ako napríklad objekt triedy `SpriteBatch` pre vykresľovanie spritov, objekt triedy `SpriteFont` pre vykresľovanie fontov, `InputState` pre získanie vstupu od užívateľa atď. V klasických metódach `Update` a `Draw` sa jednotlivé obrazovky, ktoré sa nachádzajú v spomínanom zozname, po jednom aktualizujú a vykreslia. Pre vloženie novej obrazovky do zoznamu slúži metóda `AddScreen`, na druhej strane pre odobratie metóda `RemoveScreen`.

Všetky obrazovky dedia triedu `GameScreen.cs`. Táto trieda má len vymenované vlastnosti, poprípade objekty ako napríklad, či má byť obrazovka na vrchu nad ostatnými (`IsPopUp`), či sa má pri navigovaní na obrazovku táto animovať a s akými parametrami (`Transition`), taktiež obsahuje objekt triedy `ScreenManager` atď.

Obrazovku v menu predstavuje trieda `MenuScreen.cs`. Tá samozrejme dedí spomínanú triedu `GameScreen`, ale navyše obsahuje niekoľko vlastností, resp. objektov. Hlavnou časťou je zoznam tzv. `MenuEntry` objektov, tzn. zoznam položiek v menu. Po pridaní položiek do tohto zoznamu sa tieto položky vykresľujú a aktualizujú v preťažených metódach `Draw` a `Update`. Položky sú rozmiestnené v strede obrazovky pod sebou. Medzeru medzi položkami je možné nastaviť pomocou vlastnosti `menuEntryPadding`. Pre identifikáciu stlačenia niektorej položky z menu slúži virtuálna metóda `getMenuEntryHitBounds`, ktorá vracia obdĺžnik ohraničujúci položku na základe jej veľkosti a spomínaného paddingu:

```
protected virtual Rectangle GetMenuEntryHitBounds(MenuEntry entry){
      return new Rectangle(0, (int)entry.Position.Y - menuEntryPadding,
                  ScreenManager.GraphicsDevice.Viewport.Width,
                  entry.GetHeight(this) + (menuEntryPadding * 2));
}
```

Položku v menu predstavuje zmienená trieda `MenuEntry.cs`. Tá má viacero vlastností ako napr. pozíciu, text a pod. Taktiež má klasické metódy `Update` a `Draw`. V metóde `Update` sa položka aktualizuje hlavne ak je prechod obrazovky animovaný. V metóde `Draw` sa vykresľuje podľa nadstavenej pozície.

6.5.1 Obrazovky

Hra pozostáva z viacerých obrazoviek. Prvou obrazovkou, ktorá sa zobrazí hráčovi pri spustení je *MainMenuScreen*. Táto obrazovka dedí spomínanú triedu `MenuScreen` a zobrazuje menu položky ako nová hra, skóre a koniec hry. Po výbere položky *nová hra* sa zobrazí obrazovka *LevelSelectScreen*, ktorá, ako názov napovedá, ponúka hráčovi výber úrovne. Hra obsahuje spomínané dve úrovne, takže tieto predstavujú dve položky v tejto obrazovke. Pri výbere položky *skóre* v hlavnom menu sa zobrazí obrazovka *HighScoreSelectScreen*, kde si hráč môže vybrať

z dvoch položiek, ktoré odpovedajú výkonnostným tabuľkám v jednotlivých úrovniach. Týmto tabuľkám odpovedajú obrazovky *HighScoreScreen* a *PlaneHighScoreScreen*. Pri výbere úrovne závisí ďalšie zobrazenie od výberu hráča. Pre oba výbery platí zobrazenie obrazovky *LoadingAndInstructionScreen*, ktorá len indikuje načítavanie hry a zobrazuje inštrukcie k hraniu. Ak si hráč vyberie úroveň na trénovanie krátkodobej pamäte, tak ešte pred začatím hry sa na 7 sekúnd zobrazí obrazovka *TorusesInstructionScreen*, ktorá zobrazuje 6 náhodne zoradených farieb, ktoré si hráč musí zapamätať. Následne sa zobrazí obrazovka *GamePlayScreen*, ktorá sa zobrazí hneď po načítaní úrovne s viacerými lietadlami. Predtým je nutné nadstaviť aká úroveň má byť spracovávaná. Tú predstavuje vlastnosť lev v `GamePlayScreen` a priraďuje sa k nej hodnota z vyčísleného zoznamu (*enum*) `Level`. V tomto momente už začína samotná hra a jej vykresľovanie a aktualizácia. Ak hráč stlačí počas hrania tlačidlo späť na telefóne, zobrazí sa obrazovka *PauseScreen*. Pri jej zobrazení sa uloží objekt aktuálnej obrazovky *GamePlayScreen* zo *ScreenManager*-a, odstráni sa a pri návrate sa doň vráti. Obrazovka *PauseScreen* obsahuje menu položky *Return*, *Restart* a *Quit Game*. Po výbere položky *Return* sa zo ScreenManager-a odoberie obrazovka pauzy a vráti sa tam uložená hracia obrazovka. Podobný proces sa vykoná aj pri výbere položky reštartovania, ale navyše sa zavolá metóda `Restart`, ktorá nadstaví všetko (umiestnenie, natočenie lietadla, inicializáciu kruhov atď.) do pôvodného stavu. Položka *Quit Game* vráti hráča do úvodného menu.

7 Záver

Windows Phone 7 je úplne nová platforma, ktorá si už ale pomaly nachádza svoje miesto na trhu. Microsoft sa rozhodol ísť správnym smerom hlavne v smere cieľovej skupiny. Nie je ale jasné, prečo bol ukončený vývoj platformy Windows Mobile, pretože Windows Phone 7 sa vôbec nejaví ako manažérsky schopný nástupca Windows Mobile.

S vývojom na Windows Phone sa spájajú hlavne dva pojmy a to XNA a Silverlight. Prvý z nich slúži na programovanie hlavne hier, resp. 2D/3D aplikácií založených na hardwarovej akcelerácii pomocou DirectX 9.0. Je to teda framework nad DirectX, kde vývojár píše aplikácie pomocou jazyka C#. Zapuzdruje v sebe základné operácie, ktoré sa často využívajú v hrách ako napr. maticové operácie, časticové efekty, práca s kamerou a pod.

Silverlight prioritne slúži na programovanie aplikácií podobných webovým. Práca je veľmi podobná stolovej verzii frameworku. Jazyk je tu opäť C# a vzhľad je definovaný pomocou XAML súborov. Mimo iné je možné využiť Silverlight aj na programovanie 3D aplikácií napr. pomocou enginu Balder.

Celkovo je vývoj na túto platformu riešený veľmi dobre. Kombinácia Visual Studio + Windows Phone Emulator + C# je pre programátora veľmi pohodlná. K pohodlnosti prispieva aj obmedzenie hardwaru na tejto platforme (1Ghz, 256 Mb RAM, 800x480px...), takže sa vývojár nemusí prispôsobovať viacerým zariadeniam. Čo ale zamrzí, je absencia niekoľkých základných technológií ako hlavne nemožnosť pracovať s digitálnym kompasom, neexistencia API pre rozšírenú realitu, nemožnosť pracovať s Xbox Live Avatarmi a neexistencia internej databázy ako napr. SQLite. Tieto nedostatky by mala vyriešiť nadchádzajúca aktualizácia s kódovým označením *Mango* koncom roku 2011.

V práci je taktiež popísaná detailná implementácia jednoduchého leteckého simulátoru vo frameworku XNA. Táto hra pozostáva z dvoch úrovní. Prvá rozvíja u hráča tzv. sémantickú pamäť tým, že si musí zapamätať poradie 6 farieb a následne lietadlom preletieť farebné kruhy v tomto poradí. Druhou úrovňou si hráč zlepšuje multitasking schopnosti. Hráč tu musí ovládať viacero lietadiel a každé z nich dostať do rovnako farebného kruhu. Lietadlo sa ovláda pomocou akcelerometra telefónu a zároveň má implementovanú veľmi jednoduchú fyziku hlavne ohľadom gravitácie, zrýchľovania a spomaľovania lietadla. Terén hry je prakticky nekonečný, čo je zásluhou algoritmu "dlaždicovania", ktorý rozloží terén na 9 dlaždíc, ktoré sú usporiadané do štvorca a lietadlo sa nachádza presne nad strednou dlaždicou. Pri prechode na inú dlaždicu sa ostatné dlaždice usporiadajú tak, aby vznikol znova štvorec s lietadlom nad strednou dlaždicou. Nevýhodou je opakovanie terénu, ktoré je ale vzhľadom na veľkosť dlaždíc zanedbateľné. Okrem terénu sú súčasťou prostredia aj stromy a voda. Stromy sú tvorené jednoduchými externými modelmi. Tým, že sú priehľadné, je nutné ich každým vykreslením snímku triediť vzhľadom k pozícii kamery. Keďže

Windows Phone 7 zatiaľ nepodporuje shader-e, voda je tvorená jednoduchou priehľadnou plochou, na ktorej je textúra vody, ktorá sa pohybuje opačným smerom ako lietadlo. Celá vodná plocha sa pohybuje spolu s lietadlom. Aplikácia obsahuje veľmi dobre prepracovanú cestu častíc (*Particles Pipeline*), ktorá nadstavenie výbuchu lietadla načítava z XML súboru. Výbuch je pre urýchlenie vykresľovania 2D, čo ale vôbec nie je pri tak krátkom zobrazení badateľné. O správu obrazoviek sa stará tzv. manažér obrazoviek (*Screen Manager*), do ktorého je možné obrazovky jednoducho vkladať a naopak zasa odoberať. Hra obsahuje niekoľko zvukov – zvuk motora, výbuch, preletenie kruhom (1 náhodne zvolený zo 6 zvukov) a signalizácia nového lietadla pri úrovni s viacerými lietadlami. Výsledné skóre v úrovniach závisí od času (úroveň s kruhmi), resp. od počtu preletených lietadiel kruhom (úroveň s lietadlami). Tieto skóre sa zapisujú do internej pamäte telefónu, tzv. *Isolated Storage*.

V budúcnosti by bolo vhodné do hry doplniť podporu shaderov (ak bude dostupná), ktorá sa veľmi hodí pri vykresľovaní veľkého množstva stromov, vody, výbuchu, poprípade implementácii *Level of Detail* terénu. Taktiež v hre chýba podpora multitasking vlastností telefónu. So spomínanou aktualizáciou *Mango* by bolo vhodné prepojiť hru s menu vo frameworku SilverLight.

Literatúra

[1] DSL.sk: MS sa pokúša opäť konkurovať v mobiloch, uviedol Windows Phone 7 [online]. 2010. Dostupné na URL: <http://dsl.sk/article.php?article=9845&title=>

[2] Windows Phone Design System: Codenamed 'Metro' [online]. 2010. Dostupné na URL: <http://go.microsoft.com/fwlink/?LinkID=189338>

[3] Wigley, A., Miles, S., R.: Windows Phone 7 Virtual JumpStart Series [online]. 2010. Dostupné na URL: < https://channel9.msdn.com/blogs/egibson/720-22-windows-phone-7-virtual-jumpstart-sessions>

[4] Hardware Specifications for Windows Phone [online]. 2010. Dostupné na URL: <http://msdn.microsoft.com/en-us/library/ff637514%28v=VS.92%29.aspx>

[5] Hennessey, E.: You've Been Tombstoned! [online]. 2010. Dostupné na URL: <http://create.msdn.com/en-US/education/catalog/article/tombstoning_wp7_games>

[6] Execution Model Overview for Windows Phone [online]. 2010. Dostupné na URL:<http://msdn.microsoft.com/en-us/library/ff817008%28v=VS.92%29.aspx>

[7] Application Platform Overview for Windows Phone [online]. 2010. Dostupné na URL: <http://msdn.microsoft.com/en-us/library/ff402531%28VS.92%29.aspx>

[8] Kajan, R.: SlimDX, XNA [online]. 2010. Dostupné na URL: <https://www.fit.vutbr.cz/study/courses/GZN/private/lect/GZN-XNA.pdf>

[9] Klucher, M.: The XNA Framework Content Pipeline [online]. 2006. Dostupné na URL: <http://blogs.msdn.com/b/xna/archive/2006/08/29/730168.aspx>

[10] Adding Content to a Game [online]. 2010. Dostupné na URL: <http://msdn.microsoft.com/en-us/library/bb203887%28v=XNAGameStudio.40%29.aspx>

[11] Best Practises for Windows Phone 7 Games [online]. 2010. Dostupné na URL: <http://create.msdn.com/en-US/education/catalog/article/best_practices_wp7_games>

[12] Adding Xbox LIVE Features to Your Game [online]. 2010. Dostupné na URL: <http://msdn.microsoft.com/en-us/library/dd254747%28XNAGameStudio.40%29.aspx>

[13] Herout, A.: Počítačová grafika – Studijní opora. [2008].

[14] Introduction to FlightGear [online]. 2010. Dostupné na URL: <http://www.flightgear.org/introduction.html>

[15] Palomino Flight Simulator [online]. 2010. Dostupné na URL: <http://www.palomino3d.org/>

[16] Lindstrom P., Pascucci V.: Terrain Simplification Simplified: A General Framework for View-Dependent Out-of-Core Visualization. 2002. IEEE Transactions on Visualization and Computer Graphics, 8(3):239-254.

[17] Lindstrom P., Pascucci V.: Visualization of Large Terrains Made Easy. 2001. Proceedings of IEEE Visualization 2001, pp. 63-370, 574.

[18] Duchaineau M., Wolinsky M., Sigeti E. D., Miller C. M., Aldrich Ch., Mineev-Weinstein B. M.: ROAMing Terrain: Real-time Optimally Adapting Meshes. 1997. UCRL-JC-127870.

[19] Reed A.: Learning XNA 3.0. 2009. O'Reilly Media, Inc. ISBN: 978-0-596-52195-0.

[20] Nitschke B.: Professional XNA Programming: Building Games for Xbox 360 and Windows with XNA Game Studio 2.0. 2008. ISBN: 9780470261286.

[21] Balder – 3D engine for Silvelight, Windows Phone 7, Xna and OpenGL [online]. 2010. Dostupné na URL: <http://balder.codeplex.com/>

[22] Miles, R.: Introduction to Programming Through Game Development Using Microsof XNA Game Studio. 2010. Microsoft Press. Library of Congress Control Number: 2009932322.

[23] Everything you need to know from MIX '11 [online]. 2011. Dostupné na URL: <http://www.wpcentral.com/everything-you-need-know-mix-11>

[24] Gartner Says Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012 [online]. Dostupné na URL: <http://www.gartner.com/it/page.jsp?id=1622614>

[25] Marble Maze Lab [online]. Dostupné na URL: <http://create.msdn.com/en-US/education/catalog/lab/marble_maze>

[26] Generated Geometry Lab [online]. Dostupné na URL <http://create.msdn.com/en-US/education/catalog/sample/generated_geometry>

[27] Particles Pipeline Lab [online]. Dostupné na URL: <http://create.msdn.com/en-US/education/catalog/sample/particles_pipeline>

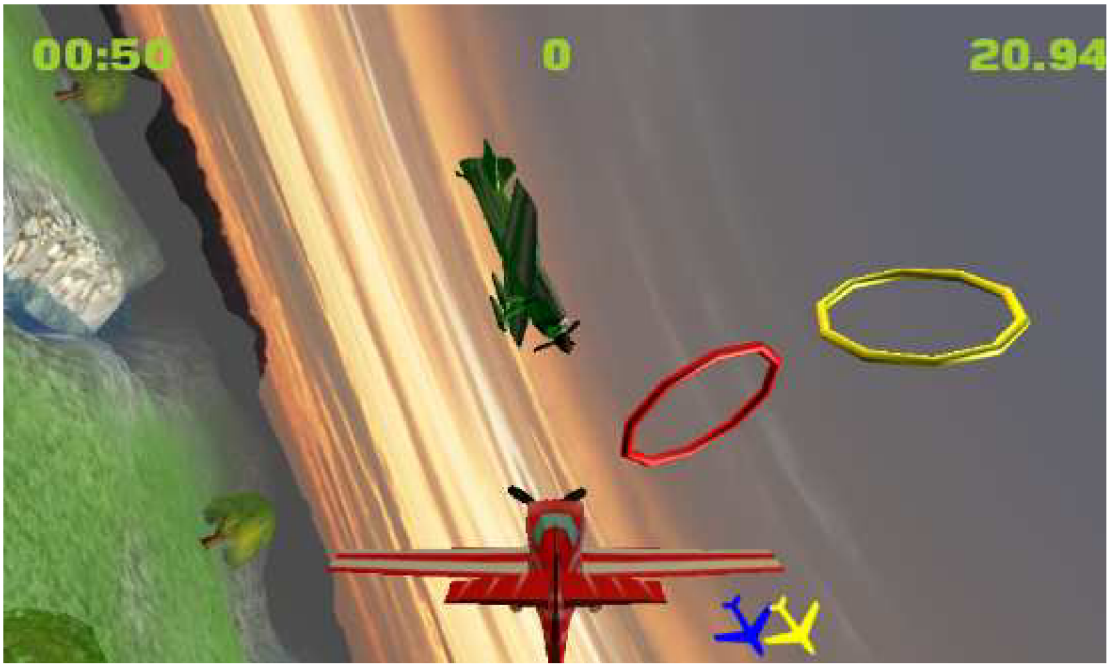
[28] Chase Camera Lab [online]. Dostupné na URL: <http://create.msdn.com/en-US/education/catalog/sample/chasecamera>

[29] Lens Flare Lab [online]. Dostupné naURL: <http://create.msdn.com/en-us/education/catalog/sample/lens_flare>

Zoznam príloh

1 Príloha – Obrázky z hry


00:29
0
27.48


00:50
0
20.94


00:07
0
Tap screen to continue

Brain Plane

Marián Hacaj (marian.hacaj@gmail.com), Imagine Cup 2011

Introduction

More demands are laid on people recently. Therefore it is required to find a **better way to educate** people. One of the best methods is to add educational features to some kind of a game, perhaps to a mobile phone **game**. Recent mobile phones, especially **Windows Phone 7** phones, are very appropriate for multimedia and gaming.

Brain Plane is a simple airplane arcade game which tries to practice and develops brain activity in several ways. It is possible to make literally hundreds of different levels. For example these **three levels** as a start:

This level is designed for training the **short-term memory**. The goal of the game is to fly through differently colored rings in the correct order, which is randomly generated on the start of each level. The player has a few seconds to memorize it before the level starts.

The goal of the second level is to improve **multitasking skills**, which are also essential for everyday use. The player has to fly several planes at once and he/she has to navigate each of them to its corresponding ring, which has the same color as the plane. The player can switch to other plane by tapping the left side of the display. When the player switches to next plane the direction, rotation and speed of the previous plane does not change.

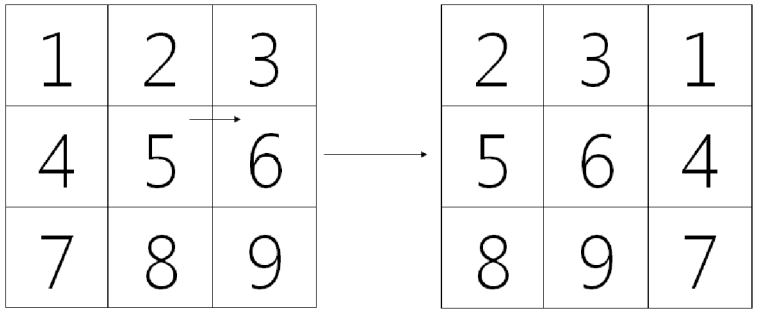
The third level is for training **semantic memory**. This memory separates things in our world that make sense from the ones that do not. For example, we know that a building is bigger than a human or that grass is usually green. We know it because we memorized it. The goal of this level is to aim and select objects in the game world, which are semantically wrong.

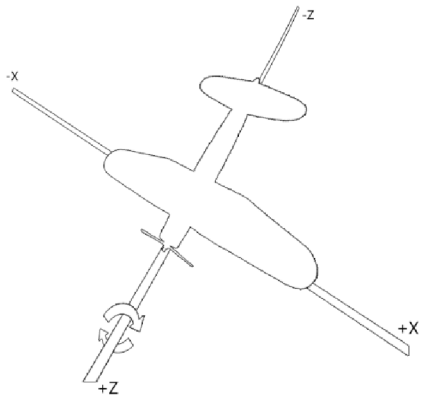
The game also focuses on its graphic side and the player's **visual experience**. Terrain and trees are very detailed and clear water is making much more realistic feeling. Particle explosion and lens flare completes graphic elements of the game.

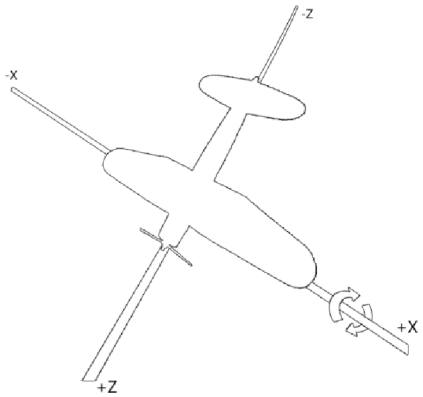
Little closer look

As mentioned in the introduction, the Brain Plane is a **simple 3D airplane arcade game** with some brain developing features. It is created with **XNA Game Studio** and it is using information from tutorials at education section in www.create.msdn.com . Game is by default running under full screen in landscape left mode.

The terrain for the game is generated from height maps. Plane simulation games require **large terrain**. In this game the terrain is made up from "tiles", which are joined together and form one big square. The plane is always over the middle one. When a plane reaches an end of the middle tile, three tiles on back side are transformed to new positions, so the plane stays in the middle. For example, when a plane moves from tile number 5 to tile number 6, tiles 1, 4, 7 are moved to new positions and the plane is in center again.







The plane rotation is performed by phone's **accelerometer**. Rotation along the roll axis of the plane is controlled by the X value of the accelerometer. So, when the phone is tilted to the right or to the left (in landscape mode), the plane starts to change its roll. Other rotation of the plane is along the pitch axis. This rotation is controlled by the Z value of the accelerometer. So, in order for the plane to turn left it has to rotate along its Z axis first and then along its X axis to actually make the turn.