EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/I/2014/2726555322

TVORBA INTERAKTÍVNEJ MULTIMEDIÁLNEJ APLIKÁCIE V PLATFORME MICROSOFT SILVERLIGHT

Diplomová práca

2014 **Bc.Tomáš Kobza**

EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

TVORBA INTERAKTÍVNEJ MULTIMEDIÁLNEJ APLIKÁCIE V PLATFORME MICROSOFT SILVERLIGHT

Diplomová práca

Študijný program: Manažérske rozhodovanie a informačné technológie

Študijný odbor: 6258 Kvantitatívne metódy v ekonómii

Školiace pracovisko: Katedra aplikovanej informatiky

Vedúci záverečnej práce: Ing. Magdaléna Cárachová, PhD.

2014 **Bc.Tomáš Kobza**

Čestné vyhlásenie

Čestne vyhlasujem, že záverečnú prácu som vypracoval samostatne a že som uviedol všetku použitú literatúru.

Dátum: 28.4.2014 **.....**

Poďakovanie

Na tomto mieste by som chcel poďakovať vedúcej svojej diplomovej práce Ing. Magdaléne Cárachovej, PhD. za pomoc, ochotnú spoluprácu, podnetné pripomienky a cenné rady, ktoré mi pri vypracovávaní diplomovej práce vždy ochotne poskytla.

Abstrakt

Kobza, Tomáš: *Vybrané Tvorba interaktívnej multimediálnej aplikácie v platforme Microsoft Silverlight.* – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. – Vedúci záverečnej práce: práce Ing. Magdaléna Cárachová, PhD.Bratislava: FHI EU, 2014, 71 s.

Cieľom záverečnej práce je navrhnúť interaktívnu mutimediálnu aplikáciu v platforme Microsoft Silverlight. Práca je rozdelená do štyroch kapitol. Práca obsahuje 3 tabuľky, 11 obrázkov, a štyri prílohy. Prvá kapitola je venovaná teoretickému opisu platformy Microsoft Silverlight. V ďalšej časti sa charakterizuje cieľ práce a metódy skúmania. Posledná časť opisuje tvorbu interaktívnej aplikácie.

Kľúčové slová: Silverlight, WPF, XAML, Multimediálna aplikácia

Abstract

Kobza, Tomáš: Selected Creating an interactive multimedia application in Microsoft Silverlight platform. University of Economics in Bratislava. Faculty of Economic Informatics; Department of Applied Informatics. - Thesis Supervisor: Ing. Magdaléna Cárachová, PhD. -Bratislava: FHI EU , 2014 , 71 p.

The aim of the thesis is to design an interactive multimedia application in Microsoft Silverlight platform. Thesis is divided into four chapters. It contains 3 tables, 11 images, and 4 additions. The first chapter contains theoretical descritpion of Microsoft Silverlight platform. The next section describes the objective of the work and methods of examination. The last part describes the development of interactive application.

Keywords: Silvelright, WPF, XAML, multimedia application

Obsah

Úvod

Mnoho súčasných aplikácií získava za svojho behu dáta zo servera prostredníctvom počítačovej siete. U niektorých aplikácií môžeme hovoriť až o závislosti na týchto dátach. Pokiaľ sú dáta uložené na serveri, o prezentáciu informácií sa stará klient. Takéto aplikácie sa zvyknú označovať ako distribuované.

Technológia .NET framework má v ponuke hneď niekoľko nástrojov na zjednodušenie tvorby distribuovaných aplikácií. Windows Presentation Foundation (WPF) je vhodný na tvorbu aplikácií určených na inštaláciu na pracovnej stanici. WPF predstavuje nový prístup v tvorbe grafického systému .NET Frameworku. Vďaka WPF je jednoduchšie tvoriť aplikácie bohaté na multimediálny obsah, prostredníctvom ktorého je možné odpútať sa od klasického „formulárového“ vzhľadu užívateľského rozhrania. Využíva sa predovšetkým grafická akcelerácia zodpovedného hardvérového komponentu, vďaka čomu sa odľahčuje výpočtová kapacita procesora. WPF podporuje animácie, geometrické tvary, vektorovú grafiku, farebné prechody, a dátové väzby.[1]

Na vytvorenie klienta, dostupného odkiaľkoľvek prostredníctvom webového prehliadača, je vhodným nástrojom technológia Silverlight, ktorej vývoj bazíruje na technológii WPF.

Windows Communication Foundation uľahčuje vývoj distribuovaných aplikácií, pre ktoré sa tiež používa označenie SOA. Táto skratka znamená Service-Oriented Architecture, teda architektúra orientovaná na služby, ktoré sú využívané klientmi. WCF zaisťuje sieťovú komunikáciu, čo zabezpečuje, že volanie služieb sa takmer nelíši od volania metód v rámci jednej aplikácie. Klientom webovej služby môže byť napríklad WPF alebo Silverlight aplikácia.

Táto technológia má tiež vhodné využitie pre tvorbu tzv. „business aplikácií“, kde napomáha funkcii dátových väzieb. Dátové väzby umožňujú dynamicky pripojiť získané dáta do ovládacích prvkov užívateľského rozhrania. Zdrojové dáta pre dátové väzby Silverlight klient získava napríklad volaním webových služieb, ktoré sú poskytované rôznymi servermi umiestnenými na internete.

1. Teoretické východiská platformy Silverlight

1.1.Čo je Silverlight?

Silverlight je webová platforma fungujúca za podpory .NET framework, ktorá je nezávislá na operačnom systéme a na webovom prehliadači. Je to zároveň aj vývojárska platforma, slúžiaca na tvorbu RIA aplikácií, multimediálnych aplikácii pre web, desktop a mobilné zariadenia.

Silverlight má k dispozícii virtuálne behové prostredie Common Language Runtime (CLR) obsahujúce hlavné triedy, garbage collector a just-in-time kompilátor. Aplikácie môžu byť vytvárané pomocou tradičných .NET vývojárskych prostriedkov ako Visual Studio, ktorý slúži hlavne na editáciu kódu a ladenie, alebo Expression Blend slúžiaci hlavne na editáciu používateľského rozhrania. Kód aplikačnej logiky môže byť písaný v ľubovoľnom .Net programovacom jazyku ako napríklad C# alebo Visual Basic.

Na tvorbu užívateľského rozhrania sa používa značkovací jazyk XAML, ktorý je separovaný od jazyka C#, ktorý je zodpovedný za tvorbu aplikačnej logiky. Táto skutočnosť umožňuje súbežnú prácu vývojárov a dizajnérov používateľského rozhrania. Aplikácie sú distribuované v XAP balíkoch, ktoré sa skladajú z XAML súborov a aplikačného kódu.

Za podpory vektorovej grafiky, prehrávania audia a videa, prístupu k dátam, textu či animácií sa stáva Silvelright vhodnou platformou na vytváranie rozličných aplikácií. Technológia Silvelright je implementovaná vo veľkom množstve web stránok, či už hovoríme o prezentácii obrázkov, prehrávaní videa či hudby alebo o prezentovaní štatistických údajov. Silverlight sa tiež používa na tvorbu hier a business aplikácií, ktoré môžu byť tiež vyvíjané ako Out-Of-Browser aplikácie, a tým umožniť používateľovi ich spustenie priamo na jeho počítači.

1.2.Verzie platformy Silverlight

V dobe písania tejto diplomovej práce bola dostupná technologická platforma Silverlight vo verzii 5. Kompatibilita s predošlými verziami však bola zachovaná. Na počiatku vývoja sa táto technológia označovala ako Windows Presentation Foundation/Everywhere. Názov napovedá, že Microsoft chcel preniesť do Silverlightu čo

možno najviac rysov nového prezentačného rozhrania WPF (Windows Presentation Foundation), ktoré je súčasťou .NET Frameworku od verzie 3.0.

Prvá verzia platformy bola predstavená v roku 2007. Webové prehliadače, ktoré ju podporovali boli: Internet Explorer, Mozilla Firefox, SeaMonkey, Safari, Opera. Prvá verzia bola prístupná na platformy Windows (Windows Vista, Windows 7, Windows Server 2008, Windows XP) a Mac (Mac OS). Architektúra platformy Silverlight 1.0 bola rozdelená na dve základné časti. Prezentačná vrstva obsahovala elementy a funkcie potrebné na vytváranie užívateľského rozhrania a interakciu s užívateľom. V používateľskom rozhraní bolo implementované renderovanie vektorovej a bitmapovej grafiky, textový výstup, animácie a prezentácie mediálneho obsahu vo formátoch WMV, VC-1, MP3 a WMA. Interakcia s užívateľom zahrňovala obsluhu udalostí generovaných užívateľom pomocou myši a klávesnice. Druhou vrstvou je inštalačný a aktualizačný modul pre internetový prehliadač. Na programovú manipuláciu s prezentačnou vrstvou sa používal Javascript, ktorý pracoval s objektovým elementom pracovnej plochy nazývanom „Canvas“. Ten znázorňoval viditeľnú prezentačnú vrstvu Silverlight aplikácie. Prepojenie s dátovým zdrojom bolo realizované prostredníctvom ASP.NET AJAX, ktoré podporuje volanie webových služieb z Javascriptu s výmenou údajov vo formáte JSON (Javascript Object Notation) a tiež volanie jednoduchých webových služieb založených výhradne na XML, tzv. POX služieb (Plain Old XML).

Verzia Silverlight 2 priniesla niektoré nové ovládacie prvky, štýly a šablóny i dátové väzby. Obsahovala bohatú knižničnú štruktúru, v ktorej boli implementované preddeklarované triedy. Táto verzia umožňovala ukladať dáta na strane klienta do izolovaného lokálneho úložiska s veľkosťou 10 MB. Od verzie 2.0 je tak súčasťou zásuvného modulu do prehliadača aj vlastný .NET Framework, ktorý je zminimalizovaný a optimalizovaný tak, že sa všetky jeho funkcie, ktoré sú potrebné na beh aplikácií v rámci internetového prehliadača podarilo uložiť do súboru veľkosti 4 MB. Týmto vznikla nová architektúra fungovania internetových aplikácií, ktorá kladie len minimálnu záťaž na server a presúva hlavné časti fungovania na klienta. Knižničná štruktúra, možnosť použitia štandardných programovacích jazykov .NET objektov, vytváranie graficky a mediáne bohatého užívateľského rozhrania bez nutnosti enormnej záťaže na server, dáva vývojárom do rúk silnú technológiu na konštrukciu webov novej generácie, ktorý sa začal nazývať web 2.0.

Jednou z najvýznamnejších noviniek platformy Silverlight 3 je možnosť spustenia aplikácie na desktope klienta mimo webový prehliadač. Táto vymoženosť nesie názov „Out

Of Browser“, inštaluje sa do lokálneho operačného systému, kde je exekuovaná v izolovanom priestore, v tzv. Sandboxe. Z pohľadu koncového užívateľa sa inštalácia javí ako desktopová. Jej rýchly prístup je reprezentovaný zástupcom a funguje bez nutnosti inštalácie akéhokoľvek ďalšieho softvéru, a to aj pri dočasnej strate internetového pripojenia.[5] V oblasti prezentácie grafiky, fotografií a obrázkov prišlo zásadné zlepšenie v podobe interaktívneho nástroja Deep Zoom, pomocou ktorého je možné vykresľovať obrázky vo vysokom rozlíšení. Implementovaná bola podpora ďalších kodekov, a prehrávanie videa v HD kvalite, v celoobrazovkovom režime, spolu s funkciou na ochranu digitálnych práv DRM. K dispozícii je tiež živé streamovanie videa v HD rozlíšení v kombinácii s technológiou Smooth Streaming. Táto technológia umožňuje prehrávanie videa, počas ktorého sa bez akéhokoľvek prerušenia proces prehrávania adaptuje na kvalitu linky medzi serverom a klientom a grafickými kapacitami počítača. Pribudla možnosť využívať 3D transformácie, ktoré uľahčujú tvorbu efektívnejších používateľských rozhraní. Tieto umožňujú umiestniť rovinu, v ktorej sú ovládacie prvky, ľubovoľne do priestoru. Všetky tieto funkcie bolo možné implementovať pridaním podpory hardvérovej akcelerácie grafického rozhrania, ktorá prenesie záťaž z CPU na GPU.

Vo verzii 4 boli pridané nové prvky, ktoré posúvajú Silverlight do roviny platformy vhodnej pre graficky bohaté „business aplikácie“. Do tejto skupiny je možné zaradiť podporu tlače, alebo nový ovládací prvok RichTextBox, ktorý komplexne zastrešuje prácu s formátovaným textom. Element DataGrid získal vo verzii 4 nové možnosti. Umožňuje nastavenie automatickej šírky stĺpcov, ale predovšetkým kopírovať údaje z tabuľky do Clipboardu a následne do tabuľkového procesora. Taktiež pribudla funkcia „drag and drop“ pre pohodlné vkladanie dát z klientského počítača. Zavedenie funkcie .NET Common Runtime (CLR) umožnilo spúšťať skompilované kódy ako pre webové, tak aj pre desktopové aplikácie. Po hardvérovej stránke je táto verzia menej náročná na výpočty a je dvakrát rýchlejšia ako predošlé. Aplikácie v tejto verzii umožňujú priamo spúšťať desktopové aplikácie, ako sú MS Word alebo MS Excel a prostredníctvom nich je možné priamo pracovať s dátami. Sprístupnilo sa tiež používanie webovej kamery a mikrofónu.[4]

Ďalšia verzia Silverlight nenechala na seba dlho čakať. Vo verzii 5 pribudli ďalšie užitočné komponenty. Pivot Viewer, dokáže spravovať a zobrazovať veľké množstvo dát. Jedno z jeho najlepších využití spočíva v uľahčení hľadania informácií pre koncových užívateľoch, ktorí môžu z veľkej kolekcie dát jednoduchšie vyhľadávať informácie, ktoré potrebujú. V Silverlight aplikáciach je odteraz možné vytvárať súbory a ukladať ich na

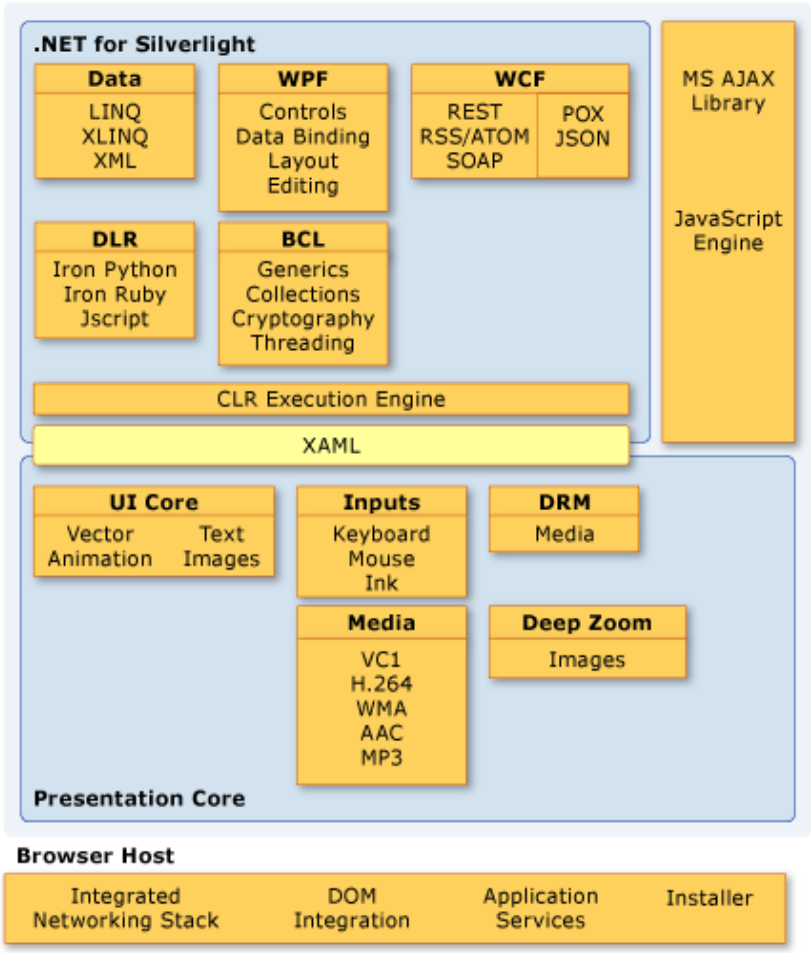
lokálny disk. O túto funkciu sa postarali triedy „openFileDialog“ a „SaveFileDialog“, z ktorých prvá realizuje definovanie cesty k súboru a druhá definuje jeho meno a príponu. K dispozícii je tiež nástroj s názvom „CharacterSpacing“, ktorý má implementovaný mechanizmus na zvyšovanie a znižovanie medzier medzi jednotlivými písmenami textu. Jedným z najdôležitejších prínosov je pokročilá hardvérová akcelerácia. Aplikácie podporujú natívne 64 bitové prehliadače, čo znamená posun v spravovaní veľkého množstva dát. Veľký pokrok bol taktiež zaznamenaný v štarte aplikácii, v tejto verzii zaberie inicializovanie aplikácie omnoho menej času. Podstatné zlepšenia sú viditeľné tiež v oblasti 3D akcelerácie, renderovania textu a kvality animácií. [3]

1.3.Architektúra platformy Silvelright

Platforma Silverlight sa skladá z dvoch častí, ktoré dopĺňa množina komponentov pre inštaláciu a aktualizáciu. Prvá časť Core presentation framework zahŕňa komponenty a služby orientované na užívateľské rozhranie a interakciu s používateľom. Táto časť pokrýva užívateľský vstup, prehrávanie médií, viazanie dát (data binding), správu digitálnych práv (Digital Rights Management - DRM) a prezentačné funkcie pre vektorovú grafiku, text, animácie a obrázky. Obsahuje taktiež jazyk XAML.[17]

Druhá časť .NET framework for Silverlight je podmnožina .NET framework. Obsahuje komponenty a knižnice zahŕňajúce integráciu dát, prácu so sieťou, integráciu dát ovládacie prvky Windows knižnice základných tried, Garbage Collection a Common Language Runtime (CLR). Aplikácia však môže vyžadovať ďalšie knižnice, ktoré nie sú v .Net Framework for Silverlight obsiahnuté. Tieto zahŕňa nástroj Silverlight SDK (Software Development Kit). Knižnice sú pribalené v aplikácií a stiahnuté do prehliadača. Sú to napríklad nové ovládacie prvky užívateľského rozhrania, XLINQ, Syndication (RSS/Atom) serializácia XML alebo Dynamic Language Runtime (DLR).

Množina komponentov pre inštaláciu a aktualizáciu zjednodušujú proces inštalácie pre užívateľov, spúšťajúcich aplikáciu prvýkrát a následne umožňujú jej jednoduchú aktualizáciu. Nasledujúca schéma ukazuje časti architektúry Silverlight, spolu so súvisiacimi komponentami a službami.[17]



Obrázok 1 Architektúra Silverlight [9]

Časť Core Presentation Components sa skladá z prvkov ktoré sú popísané v nasledujúcej tabuľke:

Tabuľka č.1

Input	Prijíma vstupy z hardvérových komponentov ako napríklad klávesnica a myš, zariadenia na kreslenie alebo iné vstupné zariadenia
UI rendering	Má za úlohu vykresľovať vektorovú a bitmapovú grafiku, animácie a text.
Media	Podporuje prehrávanie a správu rôznych typov audio a video súborov, ako napríklad .WMP alebo MP3 súbory.
Deep Zoom	Podporuje približovanie obrázkov vo vysokom rozlíšení
Controls	Podporuje ovládacie prvky, ktoré sú prispôsobiteľné pomocou štýlov a šablón
Layout	Umožňuje polohovanie prvkov aplikačného rozhrania

Data binding	Umožňuje viazanie dátových objektov s prvkami aplikačného rozhrania
DRM	Zahŕňa správu digitálnych práv multimediálneho obsahu (DRM).
XAML	Poskytuje analyzátor pre jazyk UML.

[17]

Vývojári majú k dispozícii nástroj na prácu s aplikačným rozhraním – XAML, ktorý je hlavným prostredníkom medzi .NET Frameworkom a prezentačnou vrstvou. Porovnateľnú funkcionalitu je možné dosiahnuť aj spravovaným kódom na pozadí, ktorý však býva menej prehľadný a z dôvodu nutnosti použitia veľkého množstva tohto kódu sa stáva neefektívny. [17]

Tabuľka č.2 popisuje čiastočný zoznam funkcií .NET Framework for Silverlight

Tabuľka č.2

Funkcia	Popis
Data	Podporuje LINQ a LINQ to XML ktoré uľahčujú integráciu a prácu s dátami z rôznych zdrojov. Podporuje tiež prácu s XML a serializačné triedy pre spracovanie dát.
Base class library	Kolekcia .NET Framework knižníc, ktoré poskytujú dôležité programovacie funkcie, ako napríklad prácu s reťazcami, regulárne výrazy, vstupy a výstupy, kolekcie a globalizáciu.
WCF (Windows Communication Foundation)	Poskytuje funkcie pre zjednodušenie prístupu ku vzdialeným službám a dátam.
CLR (Common Language Runtime)	Zabezpečuje správu pamäte, kontrolu typovej bezpečnosti a zachytávanie výnimiek.
WPF (Windows Presentation Foundation) controls	Poskytuje bohatú kolekciu ovládacích prvkov obsahujúcich Button, CheckBox HyperlinkButton, ListBox, RadioButton a ScrollViewer.
DLR (Dynamic Language Runtime)	Zabezpečuje dynamickú kompiláciu a exekúciu skriptovacích jazykov, ako napríklad JavaScript a IronPython, pre tvorbu aplikácií založených na Silverlight.

[17]

.NET Framework for Silverlight je podmnožina celého .NET Framework. Zabezpečuje základy pre robustný, objektovo - orientovaný vývoj aplikácií pre typy aplikácií, pre ktoré táto podpora v minulosti nebola k dispozícii. Silverlight ponúka niekoľko ďalších funkcií, ktoré pomáhajú programátorom vytvárať bohaté a interaktívne aplikácie, niektoré z nich sú uvedené v tabuľke č.3.

Tabuľka č.3

Funkcia	Popis
Isolated storage	Zabezpečuje bezpečný prístup Silverlight klienta k súborom lokálneho počítača. Umožňuje lokálne ukladanie dát, ktoré sú určené konkrétnemu užívateľovi a ich ukladanie do medzipamäte.
Asynchronous programming	Programové úlohy vykonáva vlákno na pozadí, zatiaľ čo aplikácia samotná interaguje s používateľom.
File management	Sprostredkuje dialóg pre bezpečné a jednoduché nahrávanie súborov.
HTML - managed code interraction	Poskytuje vývojárom možnosť priamo manipulovať s HTML prvkami aplikačného rozhrania webovej stránky.
Serialization	Zabezpečuje podporu pre serializáciu typov CLR do Json a XML.
Packaging	Sprostredkuje Application triedu a nástroje pre vytváranie XAP balíkov, ktoré obsahujú aplikáciu a vstupný bod, pomocou ktorého sa má spustiť Silverlight plugin.
XML libraries	Triedy XMLReader a XMLWriter zjednodušujú prácu s XML dátami z webových služieb. Vymoženosť Xlinq zabezpečuje vývojárom vytvárať XML dotazy priamo bez potreby použitia .NET programovacích jazykov.

[17]

1.4.Silverlight verzus Windows Presentation Foundation (WPF)

Spoločnosť Microsoft poskytuje paralelne dve moderné technológie pre tvorbu prezentačného rozhrania – Windows Presentation Foundation a Silverlight, ktorá bola v etape vývoja označovaná ako Windows Presentation Foundation Everywhere (WPF/E). V tomto označení sa skrýva aj vysvetlenie rozdielov medzi technológiami WPF a Silverlight. Do verzie Silverlight 2 by sa dal rozdiel vysvetliť zjednodušene tak, že technológia Silverlight je určená pre webové aplikácie a WPF pre klasické aplikácie spúšťané na lokálnych počítačoch. Od verzie Silverlight 3 sa však aplikácie dajú nainštalovať a spúšťať už aj lokálne, takže je potrebné hľadať rozdiely trochu hlbšie, ani nie až tak v architektúre, ale v oblasti nasadenia. Slovo „Everywhere“ v pôvodnom kódovom označení vysvetľuje totiž multiplatformovosť technológie Silverlight. Zatiaľ čo WPF aplikácie bežia len pod operačnými systémami Windows s nainštalovanou technologickou platformou .NET Framework, Silverlight aplikácie bežia aj na iných klientskych platformách, napríklad Mac.[4]

1.5.Programovacie jazyky platformy Silverlight

1.5.1. Jazyk XAML

Informácie o polohe jednotlivých ovládacích prvkoch aplikačného rozhrania a spôsob, akými sa prvky zobrazujú, sú uložené v textovom súbore. Jazyk, v ktorom sú informácie zapísané, patrí do skupiny značkovacích jazykov a nazýva sa XAML. Vychádza z jazyka XML a tak prísne dodržuje syntaktické a sémantické pravidlá XML dokumentu. Výhodou XAML je predovšetkým využívanie textového formátu na komunikáciu , čo predstavuje výhodu oproti iným webovým technológiám (napr. Active X, Java Applet-y a Flash), ktoré na komunikáciu s prehliadačom používajú binárny formát. Ten môže spôsobovať problémy s firewallmi užívateľov. Zdrojové súbory majú príponu .xaml.

XAML aplikácie ponúkajú širšie možnosti používateľského prostredia, vyššiu bezpečnosť a jednoduchší vývoj. Prostredníctvom ľahko zrozumiteľného a editovateľného formátu je možné definovať aplikačné rozhrania oddelene od aplikačnej logiky. Silverlight aplikácie obsahujú dva podstatné súbory a to jeden deklaratívny s príponou XAML a druhý s programovým kódom aplikačnej logiky s príponou .cs, Vizuálna reprezentácia jazyka XAML je tvorená vektorovú grafikou. Jednotlivé XAML tagy majú rovnaké názvy, ako im

zodpovedajúce triedy a atribúty. Vnorené elementy môžu byť inštanciami ďalších tried, prípadne vlastností.

Nasledujúci príklad porovnáva vytvorenie rovnakého elementu (tlačidlo) v jazyku C# a XAML. Použitie jazyka XAML výrazne zjednodušuje zdrojový kód, dôjde k jeho podstatnému skráteniu oproti C# zápisu.

C#
```
Button myButton = new Button();
myButton.Background = Color.Blue;
myButton.Content = "Nápis na tlačidle";
this.Children.Add(myButton);
```

XAML
```
<Button Background="Blue">Nápis na tlačidle</Button>
```

Ako už bolo spomenuté, XAML vychádza zo značkovacieho jazyka XML, ktorého dokument vždy pozostáva z elementov. Každý element má svoj začiatočný a koncový tag, kde oba tagy obsahujú meno elementu, koncový tag má pred názvom lomítko. Názov elementu nesmie obsahovať medzery a niektoré špeciálne znaky, musí začínať písmenom. Je case sensitive – rozlišuje veľké a malé písmená.

Elementy sa môžu do seba vnárať, to znamená, že element môže obsiahnuť ďalší element. XML dokument má hierarchickú stromovú štruktúru. Každý dokument musí mať jeden koreňový (root) element, úroveň vnorenia jednotlivých elementov nie je obmedzená, ale jednotlivé elementy sa nesmú krížiť. [1]

Nasledujúci príklad obsahuje blok kódu v jazyku XAML, ktorý zobrazuje jednoduchý formulár obsahujúci textové pole a tlačidlo slúžiace na odoslanie formulára. Koreňovým elementom je v tomto prípade element UserControl.

XAML
```
<UserControl x:Class="SilverlightApplication5.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">
        <TextBox x:Name="tbFilter" Width="215" Height="25" />
        <Button Content="Filtrovať" Width="80" Height="25" Click="Button_Click" />
    </Grid>
</UserControl>
```

Elementy TextBox a Button sú vnorené v elemente Grid a ten je vnorený v koreňovom elemente UserControl. V jazyku XAML sa komentáre zapisujú pomocou párových značiek <!--a -->

1.5.2 .NET programovacie jazyky

C#

C# je objektovo orientovaný programovací jazyk vyvinutý firmou Microsoft. Používa sa na vývoj aplikácii postavených na technológii .NET Framework. Bol navrhovaný s úmyslom vyvážiť silu jazyka C++ a tú spojiť s možnosťou rýchleho programovania "rapid application development", ktoré ponúkali iné jazyky, ako napríklad Visual Basic a Delphi. C# je jazyk zameraný na tvorbu aplikácií pre hosťované, ako aj embedded systémy s ohľadom na veľkú škálovateľnosť - od veľmi veľkých systémov používajúcich sofistikované operačné systémy až po veľmi malé zariadenia pre špecializované úlohy.
Medzi hlavné výhody C# patrí automatické prideľovanie pamäte – ”Automatic Memory Management“, práca s výnimkami, správca verzií a “Garbage Collector“ – prostriedok prideľovania a uvoľňovania pamäte.

Visual Basic

Visual Basic je objektovo-orientovaný jazyk, riadený udalosťami a predstavuje vývojárske prostredie od firmy Microsoft. Svojimi schopnosťami dokáže konkurovať mnohým komerčným i nekomerčným jazykom. Je založený na komponentovom programovaní. Tvorcovia tohto jazyka vychádzali z jeho staršieho predchodcu, z jazyka BASIC, ktorého prvá verzia bola uvedená v roku 1964. Visual Basic sa často označuje aj ako Rapid Application Development (RAD) systém, pretože vývojárom umožňuje rýchly vývoj aplikačných prototypov.

1.6 Typy súborov

Aplikačný projekt Silverlight obsahuje súbory so zdrojovými kódmi a konfiguračné súbory.

***.csproj** - XLM súbor obsahujúci informácie o projekte, o súboroch z ktorých sa skladá, verzii platformy .NET Framework, pripojených objektoch a o nastaveniach niektorých parametrov.

AppManifest.xaml - súbor, ktorý zabezpečuje generovanie aplikačného balíka. Obsahuje súbor elementov, ktoré zahrňujú nasledujúce atribúty: runtime verzia – atribút potrebný na identifikáciu požadovanej verzie Silverlightu, Entry Point Type a EntryPointAssembly - atribúty potrebné na identifikovanie vstupného bodu aplikácie.

AssemblyInfo.cs alebo AssemblyInfo.vb - tento súbor obsahuje meno, popis a verzie metadát, ktoré sú súčasťou .xap súboru

xap súbor - predstavuje Silverlight balík. Je generovaný pri vytváraní Silverlight aplikácie. Takýto aplikačný balík je skomprimovaný zip súbor s príponou .xap a obsahuje všetky súbory potrebné na štart aplikácie.

Súbory MainPage - trieda MainPage obsahuje všetky elementy potrebné na vytvorenie užívateľského rozhrania v Silverlight aplikácii. Táto trieda dedí od triedy UserControl. Definuje vizuálne prvky, pozadie, obrázky a pod.

App.xaml - nastavenie dôležitých informácií pre zostavenie objektu.

App.xaml.cs - obsahuje inicializačný kód, ktorý je spustený po zavedení Silverlight aplikácie a kód pre obsluhu kritických chýb.

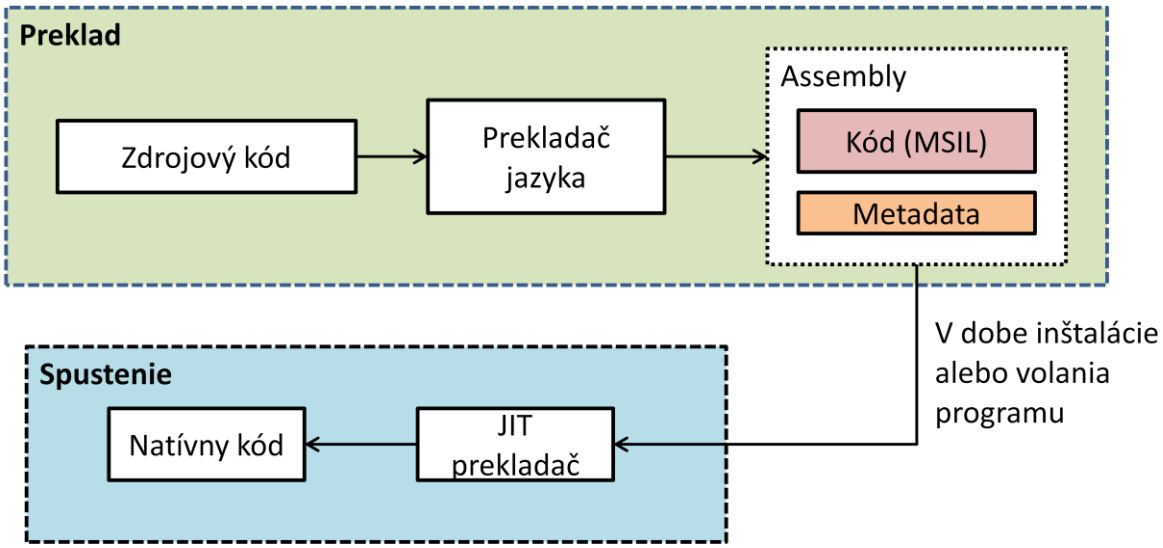
TestPage.html - jedná sa o HTML súbor, ktorý je generovaný na uvádzanie aplikácie.

1.7 Exekučné prostredie

Väčšina súčasných aplikácii býva vyvíjaná a kompilovaná priamo pre konkrétnu platformu. To znamená, že zdrojový kód býva kompiláciou transformovaný do strojového kódu počítača. Výsledkom tohto procesu je aplikácia, ktorá má vysokú spoľahlivosť a veľmi dobrú rýchlosť behu. Avšak tento proces má aj nevýhody, ako napríklad neprenositeľnosť

medzi rôznymi platformami, alebo verziami operačných systémov. V .NET frameworku je tento problém riešený prostredníctvom použitia riadeného prostredia.

Princíp riadených prostredí pridáva k procesu prevodu zdrojového kódu do kódu strojového jednu vrstvu, ktorá je reprezentovaná medzikódom, do ktorého sú skompilované zdrojové kódy. Tento medzikód je exekučným prostredím na cieľovej platforme (Windows, Linux) transformovaný do strojového kódu (viď obrázok 1).[11]



Obrázok 2: Princíp riadených prostredí (vlastná tvorba)

Táto transformácia je na cieľovej platforme realizovaná pri inštalácii alebo až pri volaní určitej časti programu. Pri exekúcii aplikácie potom nedochádza k prekladu celej aplikácie naraz, ale využíva sa tzv. JIT kompilátor (Just In Time). JIT kompilácia znamená, že je transformáciu len potrebná časť medzikódu do strojového kódu. Pri opätovnom použití tejto časti sa spúšťa jej skompilovaná forma, čo sa priaznivo prejaví na rýchlosti aplikácie. Medzikód, do ktorého sú skompilované zdrojové kódy v prostredí .NET, sa nazýva Microsoft Intermediate Language, označovaný skratkou MSIL. Pojednáva o jazyku relatívnych adries a je spúšťaný dôležitou časťou .NET Frameworku pomenovanou CLR (Common Language Runtime).[11]

1.8 Spúšťanie Silverlight aplikácií

V súčasnej dobe je možné spúšťať Silverlight aplikácie v prostredí Windows vo všetkých bežných webových prehliadačoch po inštalácii Silverlight Pluginu. Na platforme Linux je pre spúšťanie Silverlight aplikácií vyvíjaný projekt Mono (www.mono-project.com).

1.8.1 Začlenenie Silverlight aplikácie do HTML stránky

Ako už bolo vyššie uvedené, jedným zo spôsobov zobrazenia Silverlight aplikácie používateľovi je jej začlenenie do klasickej HTML stránky. Existujú dva hlavné spôsoby ako toto začlenenie zrealizovať.

Použitie HTML tagu object

V HTML slúži pre vkladanie obsahu, ktorý je interpretovaný zásuvnými modulmi prehliadača (pluginy), špeciálny tag object. Podľa uvedených atribútov tohto tagu webový prehliadač spozná, aký zásuvný modul sa má spustiť a pomocou neho zobrazí potrebný obsah.

XAML
```
<div id="myAppHost" style="height:100%;">
    <object  id="silverlightApp"
                data="data:application/x-silverlight-2,"
                type="application/x-silverlight-2"
                width="100%" height="100%">
        <param name="source" value="ClientBin/SilverlightApp.xap" />
    </object>
</div>
```

Hodnota x-silverlight-2 atribútu *data* a type neznamená, že sa jedná o verziu Silverlight 2, ale ide o použitý MIME typ spúšťajúci daný zásuvný modul. Element *param* vnorený do elementu *object* určuje, kde je umiestnená samotná Silvelright aplikácia.

Použitie Silvelright.js

Pokročilejšiu možnosť pri vkladaní Silverlight aplikácie do webovej stránky ponúka použitie Javascriptu. Konkrétne, použitie súboru Silvelright.js, ktorý je súčasťou Silverlight SDK a tiež súčasť projektu vo Visual Studiu pre tvorbu Silverlight aplikácií. Tento súbor treba referencovať v HTML stránke pomocou tagu *script*. Základný inicializačný skript, ktorý vloží aplikáciu do stránky môže mať napríklad nasledovný formát:

XAML

```
<div id="mySilverlightHost" style="height:100%;">
    <script type="text/javascript">
        Silverlight.createObjectEx({
                source: "ClientBin/MySilverlightApp.xap",
                parentElement: document.getElementById("myAppHost"),
                id: "mySilverlightControl",
                properties: {
                width: "100%",
                height: "100%",
                version: "4.0"
                },
                events: {}
            });
    </script>
</div>
```

Týmto kódom dôjde taktiež k vytvoreniu elementu *object*, preto v internetovom prehliadači povedú oba spôsoby k rovnakému výsledku.

1.9 Vývojové nástroje

Pre tvorbu aplikácii využívajúcich technológiu Silverlight majú vývojári k dispozícii dva základné nástroje. Prvým je komplexné vývojové prostredie Microsoft Visual Studio 2013, prípadne jeho verzie 2010 a 2008. Druhým je návrhové prostredie Microsoft Expression Blend, ktoré je v súčasnosti dostupné vo verzii 4. U oboch týchto nástrojov možno spozorovať dve vrstvy, v ktorých sa Silverlight aplikácie navrhujú a vyvíjajú. Sú to vývoj aplikačnej logiky a vývoj užívateľského rozhrania. Vývojové prostredie má v ponuke buď návrhové zobrazenie, kde sú jednotlivé elementy a ich štruktúry usporiadané presne tak, ako ich vidí užívateľ v aplikácii, alebo editor XAML kódu, pomocou ktorého je sú tieto elementy a štruktúry definované. [1]

1.9.1 Microsoft Visual Studio

Pre vývoj aplikácií vo vývojovom prostredí Visual Studio 2013 je nutná inštalácia rozširujúceho balíka nástrojov Silverlight Tools for Visual Studio, a tiež balík Microsoft Silverlight 5 SDK, ktorý zahŕňa knižnice a potrebné nástroje na vývoj. V predošlej verzii Visual Studia 2010 je možné po nainštalovaní doplnku vytvárať Silverlight aplikácie iba vo verzii 4.0.

Pri vytváraní nového projektu ponúka Visual Studio 2012 hneď niekoľko šablón pre vývoj Silverlight aplikácie

- Silverlight Application
- Silverlight Class Library
- Silverlight Navigation Application

1.9.2 Microsoft Expression Blend

Návrhové prostredie Microsoft Expression Blend je flexibilné a produktívne vývojové prostredie, využiteľné hlavne na grafické a dizajnové účely. Napomáha pri tvorbe moderných a vizuálne prepracovaných aplikácií s interaktívnou podporou 3D zobrazovania a prehrávania multimédií. Umožňuje vytvorenie a úpravy prezentačnej vrstvy graficky bohatých aplikácií, či už webových, využívajúcich technológiu Silverlight, alebo klasických desktopových aplikácií, založených na technológií Windows Presentation Foundation (WPF). [1]

Aplikácie vyvinuté v jednom z týchto vývojových prostredí je možné otvárať a editovať v druhom vývojovom prostredí a naopak. Dizajnér sa tak môže venovať práci na aplikačnom rozhraní v prostredí Expression Blend a programátor aplikačnej logiky môže pokračovať vo vývoji v prostredí Visual Studia. Možnosť takejto kooperácie výrazne zrýchľuje proces tvorby aplikácie a zvyšuje efektivitu práce.

Napriek tomu, že oba nástroje disponujú možnosťou vytvárania aplikačného rozhrania a aplikačnej logiky, býva vo veľa prípadoch efektívnejšie na každú úlohu použiť samostatný nástroj. Grafický návrh, animácie a priestorová grafika sa efektívnejšie vytvárajú v prostredí Expression Blend, a naopak kód aplikačnej logiky sa lepšie edituje vo Visual Studiu.

1.10 Out-of-browser

Aplikácie na platforme Silvelright od verzie 3.0 už nie sú viazané na hosťovanie v internetovom prehliadači. Platforma Silverlight umožňuje inštaláciu aplikácie priamo do lokálneho operačného systému. Takéto aplikácie sa nazývajú Out Of Browser. Nasadenie aplikácie – jej inštalácia na lokálny počítač prebehne jediným klepnutím na položku menu. Pretože Silverlight aplikácie bežia v izolovanom priestore tzv. Sandboxe, nie sú pre inštaláciu potrebné administrátorské oprávnenia. Takáto izolácia zabraňuje spusteniu prípadného škodlivého kódu stiahnutého z internetu. Z pohľadu koncového používateľa sa Out-Of-Browser aplikácia správa presne ako desktopová. Môže mať zástupcu v ponuke Štart,

prípadne na ploche a beží bez nutnosti inštalácie akéhokoľvek podporného software. A to aj pri dočasnom odpojení od internetu.

Pri každom štarte takejto aplikácie sa v prípade funkčného internetového pripojenia skontroluje verzia a v prípade potreby sa vykoná aktualizácia. Údaje je možné ukladať na server, ak nie je k dispozícii pripojenie, dajú sa dočasne uložiť v lokálnom úložisku typu Isolated Storage [1]

1.11 Rozmiestnenie a polohovanie prvkov aplikačného rozhrania

Ako už bolo spomenuté vyššie, Silvelright vychádza z koncepcie WPF. Vývojári majú pre tvorbu aplikačných rozhraní takmer identické prvky. Pre potreby rozmiestnenia a polohovania prvkov prezentačnej vrstvy ponúka jazyk XAML viacero elementov elementov. Polohovaniu a rozmiestneniu prvkov na ploche aplikácie musia venovať dizajnéri aplikácií veľkú pozornosť, aby bolo v prípade potreby možné meniť ich veľkosť, alebo vzájomnú polohu, napríklad v prípade zmeny veľkosti okna. Na takýto účel majú vývojári k dispozícii tzv. kontajnerové prvky, ktoré zapuzdrujú objekty v nich umiestnené, vďaka čomu určujú niektoré ich vlastnosti, a to najmä veľkosť, vzájomnú polohu a umiestnenie.

Pre polohovanie prvkov sú určené objekty:

Grid – mriežka, ktorá je vytvorená z riadkov a stĺpcov.

Canvas – súradnicový systém (plátno), ktorý zjednodušuje absolútne a relatívne polohovanie ostatných prvkov.

Stack panel – umožňuje vodorovné alebo zvislé ukladanie prvkov.

Wrap panel – je podobný ako Stack panel, pokiaľ by sa prvky nezmestili do jedného stĺpca alebo riadku, vykreslia sa v ďalšom.

Dock Panel – kontajnerový prvok, zarovnaný k niektorému okraju nadradeného prvku.

Ďalšie prvky ako Button, TextBox, TextBlock, ComboBox, ListBox sú i v tomto jazyku samozrejmosťou.

Balík Silvelright 5 Tools pre Visual Studio 2013 poskytuje ývvojárom aj ďalšie zaujímavé prvky. Pre prezentáciu dát v tabuľke slúži element DataGrid alebo

DomainDataSource. Na prácu s kalendárom sú určené prvky Calendar, DatePicker. K dispozícii je tiež niekoľko prvkov pre vykresľovanie grafov. [15]

Obrázok 3 Príklad polohovania prvkov

XAML

```
<Grid HorizontalAlignment="Left" Height="300" VerticalAlignment="Top" Width="400">
    <Grid.RowDefinitions>
        <RowDefinition Height="43*"/>
        <RowDefinition Height="109*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="43*"/>
        <ColumnDefinition Width="157*"/>
    </Grid.ColumnDefinitions>
    <StackPanel Grid.Column="1" Orientation="Horizontal" HorizontalAlignment="Left"
Height="65" Margin="0,20,0,0" VerticalAlignment="Top" Width="304">
        <Button Content="Tlačidlo 1" Width="75"/>
        <Button Content="Tlačidlo 2" Width="75"/>
        <Button Content="Tlačidlo 3" Width="75"/>
        <Button Content="Tlačidlo 4" Width="71"/>
    </StackPanel>
    <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Height="88"
Margin="20,0,0,0" Grid.Row="1" VerticalAlignment="Top" Width="66">
        <Button Content="Tlačidlo 5"/>
        <Button Content="Tlačidlo 6"/>
        <Button Content="Tlačidlo 7"/>
        <Button Content="Tlačidlo 8"/>
    </StackPanel>
    <Canvas Grid.Column="1" HorizontalAlignment="Left" Height="215" Grid.Row="1"
VerticalAlignment="Top" Width="314">
        <TextBox Height="205" TextWrapping="Wrap" Text="TextBox" Width="304"/>
    </Canvas>
</Grid>
```

Dizajnová plocha je rozdelená pomocou prvku *Grid* na štyri časti. V prvom riadku a v druhom stĺpci je umiestnený prvok *StackPanel*, ktorý je orientovaný horizontálne. V druhom riadku a prvom stĺpci je umiestnený tiež prvok *StackPanel*, ale je orientovaný vertikálne.

V oboch týchto prvkoch sú umiestnené štyri tlačidlá. V druhom stĺpci a druhom riadku je umiestnený element *TextBox*.

Ukážka demonštruje jednoduchosť polohovania jednotlivých elementov, ktorá je ešte intuitívnejšia pri práci vo vývojovom prostredí Visual Studio, alebo Blend 4.

1.12 Menné priestory

Na začiatku predošlého príkladu môžeme vidieť niekoľko riadkov s adresami URL. Jedná sa o definíciu tzv. Menných priestorov (namespaces). Vo všeobecnosti menné priestory slúžia k zabráneniu kolízie medzi menami identifikátorov. V dokumentoch postavených na báze XML sa menný priestor deklaruje pomocou atribútu *xmlsn*. Názov môže byť ľubovoľný, v praxi sa ale bežne používa ľubovoľné URL. elrightu dajú využívať až po doinštalovaní patričného rozšírenia. Hlavným rozšírením je Silvelright

Pri vývoji sú niektoré nové ovládacie prvky Silverlight Toolkit. Nové ovládacie prvky sú dostupné až po pridaní menného priestoru v nasledujúcom formáte.

```
XAML
xmlns:data="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.
Data"
```

Za sekvenciou *clr-namespace* nasleduje názov menného priestoru nachádzajúci sa v knižnici, ktorá je zapísaná za označením *assembly*. Použitie prvku z tohto menného priestoru potom vyzerá nasledovne.

```
XAML
<data:DataGrid Name="dgrdPortfolios"/>
```

1.13 Rozšírený zápis väzby XAML

V XAML sa zapisujú väzby prostredníctvom špecifického syntaxu rozšíreného zápisu, alebo pomocou .NET kódu. Oba zápisy sú efektívne a využívajú sa v rôznych prípadoch.

Súprava atribútov jazyka XAML, ktorá sa využíva pre zápis dátových väzieb bude predstavená v nasledujúcich prípadoch.

XAML

```
<ovladaciPrvok vlastnost="{Binding}" …/>
<ovladaciPrvok vlastnost="{Binding nazovVlastnostiZviazanehoElementu}" …/>
<ovladaciPrvok vlastnost="{Binding Path=nazovVlastnostiZviazanehoElementu }" …/>
<ovladaciPrvok vlastnost="{Binding dalsieVlastnosti}" …/>
<ovladaciPrvok vlastnost="{Binding nazovVlastnostiZviazanehoElementu,
                          dalsiaVlastnosti }" …/>
```

Názov „ovladaciPrvok“ sa nahradí požadovaným ovládacím prvkom (TextBox, TextBlock apod.) a „vlastnost“ názvom vlastnosti ovládacieho prvku, ktorá bude cieľom väzby. Atribút *Binding* určuje, že hodnota vlastnosti bude pochádzať z operácie dátovej väzby. Väzba získa svoj zdroj z vlastnosti *DataContext* prvku. Ak prvok nemá vlastnosť DataContext, zisťuje sa, či v hierarchii XAML nemá vlastnosť *DataContext* niektorý nadriadený kontajnerový prvok XAML. Pre vytvorenie väzby prvku k objektu, ktorý má vlastnosť *DataContext*, je možnosť zapísať iba atribút *Binding* (prvý riadok príkladu). Využitie je napríklad pri vytváraní väzby prvku *ListBox* ku objektu zoznamu.

Ďalší spôsob prepojenia vlastnosti zdroja s vlastnosťou ovládacieho prvku je možné vidieť v druhom a treťom riadku príkladu. Kľúčové slovo Path môže, ale nemusí byť použité. Za „dalsieVlastnosti“ sa dajú dosadiť ďalšie voliteľné kľúčové slová [16]:

Converter - prevod hodnôt pre operáciu dátovej väzby v prípade, že hodnota je v dátovom zdroji v inom formáte, než v ktorom má byť zobrazená užívateľovi (dátum, mena atď.).

ConverterCulture - parameter pre konverziu.

ConverterParameter – ďalší parameter pre konverziu

Mode – režim dátovej väzby.

NotifyOnValidationError – nastavenie správania v prípade chyby užívateľského vstupu.

Path – cesta k vlastnosti zdrojového objektu.

Source – odkaz na zdrojový objekt; ak je táto vlastnosť nastavená, potlačí sa vlastnosť DataContext ako dátový zdroj pre túto väzbu.

ValidatesOnExceptions – nastavenie, či budú hlásené výnimky v prípade chyby v užívateľskom vstupe.

1.14 Animácie

Jednou z najdôležitejších a najužitočnejších vymožeností nástroja Silverlight je tvorba animácií. Animácia zvyšuje atraktivitu aplikácie. Pohybujúci sa obsah na seba upúta pozornosť a mnohé postupy, princípy procesov alebo činnosti, sa dajú lepšie vysvetliť na názorných animovaných ukážkach.

Animácie je spôsob vytvárania zdanlivo sa pohybujúcich objektov a scén. Tvorí sa ako postupnosť sekvenčných krokov, ktoré sú statické, len sa od seba v detailoch líšia, každý z nich znázorňuje ďalšiu mikrofázu pohybu. Pri rýchlom sekvenčnom zobrazovaní týchto krokov vzniká dojem pohybu.[1]

Z hľadiska spôsobu vytvárania je možné grafické animácie rozdeliť na dve skupiny:

2D – animácie predstavujú klasický, veľmi často používaný spôsob, ktorého základom je vytváranie jednotlivých snímok animácie, neskôr spojených do súvislého, zdanlivo plynulého, obrazu. Príkladom môže byť aj jednoduché posúvanie obrazcov alebo blokov textu po obrazovke.

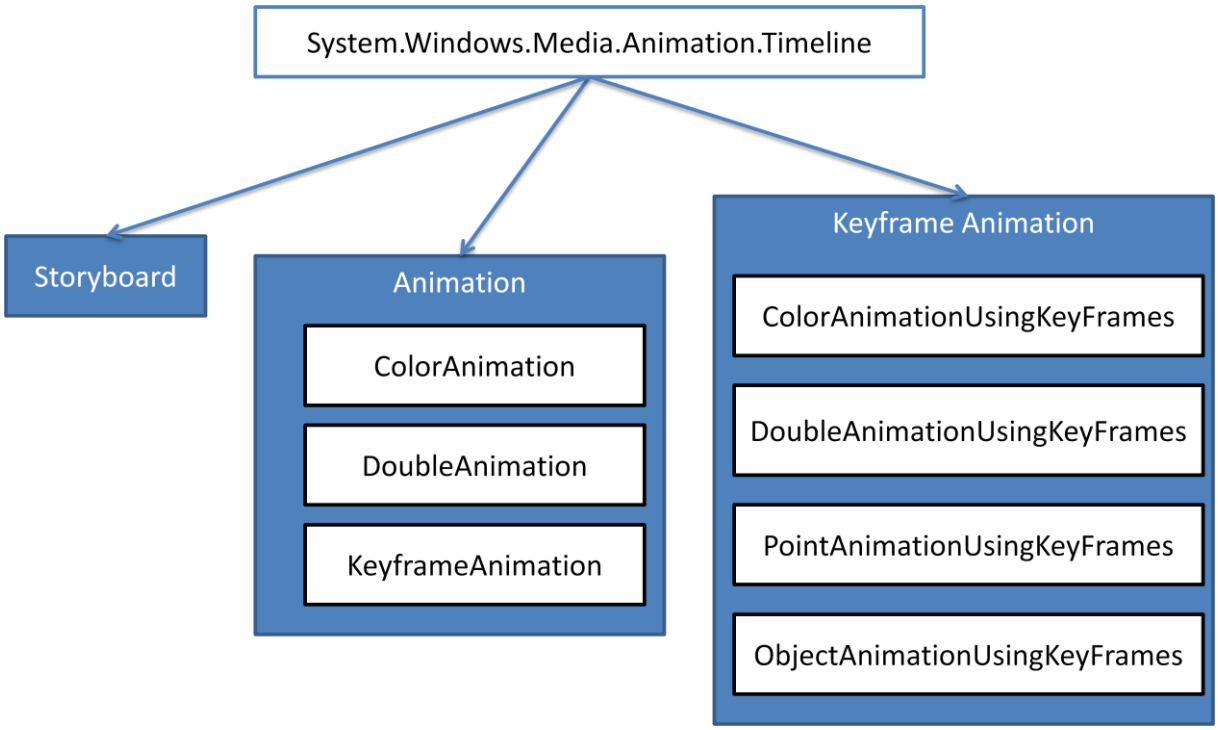
3D - animácie vznikajú na základe matematického modelu trojrozmerného sveta, zloženého z rôznych objektov, ktoré majú určité vlastnosti.

Pre jednotlivé objekty sú v platforme Silverlight k dispozícii rôzne typy parametrov, ktoré je možné pre nastavanie počas každého kroku animácie na časovej osi meniť. Je možné meniť absolútnu alebo relatívnu polohu objektu voči vzťažnej sústave, prípadne navzájom rozmery objektov, alebo podobne. Za animáciu sa môže považovať aj zmena rôznych špecifických vlastností objektu. Napríklad plynulá zmena farebnosti alebo priehľadnosti.[1]

Objekty pre animácie

Základom animácie je dej na časovej osi. Tento princíp akceptuje aj hierarchie objektov použitých pre implementáciu animácie na platforme Silvelright. Základom hierarchie objektov je trieda System.Windows.Media.Animation.Timeline. Je možné identifikovať dva typy animácie

- Jednoduchá animácia označovaná v anglickej terminológii tiež ako from/to/by.
- Keyframe animácia umožňujúca paralelne meniť viacero hodnôt parametrov.



Obrázok 4 System.Windows.Media.Animation.Timeline [1]

Objekt Storyboard môže zapuzdrovať, teda popisovať a ovládať jednu alebo viac animácií.

XAML
```
<Storyboard x:Name="Storyboard"/>
```

Je možné ho vytvoriť vo vizuálnom návrhovom prostredí alebo prostredníctvom kódu. Objekt Storyboard môže byť aj prázdny, v takom prípade sa dá využiť ako časovač.

Silvelright ponúka niekoľko typov animácii

- PointAnimation
- DoubleAnimation
- ColorAnimation

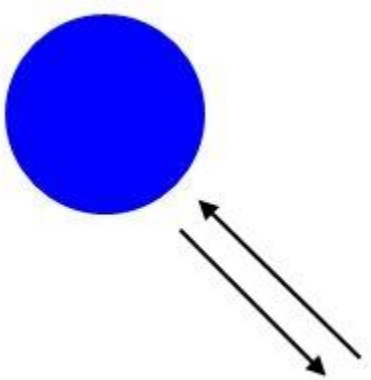
V prvom príklade bude použitá transformácia PointAnimation. Pri vytvorenom objekte animovania je potrebné vybrať taký parameter animovaného objektu, ktorého zmena sa prejaví vizuálne. V tomto prípade je to kruh.

XAML kód pre vykreslenie kruhu

XAML
```
<EllipseGeometry x:Name="Kruh" Center="70,70" RadiusX="50" RadiusY="50" />
```

Z geometrického hľadiska je poloha kruhu v rovine jasne definovaná polohou jeho stredu. Animácia typu PointAnimation umožní meniť súradnice bodu. Pre kruh sa bude meniť poloha stredu.

Pri animácii je potrebné definovať počiatočnú a cieľovú hodnotu parametrov ktoré sa menia a čas za ktorý táto zmena prebehne. Objekt sa bude pohybovať z počiatočnej pozície 100,100 do cieľovej pozície 400,400. Pomocou parametru *RepeatBehavior* je možné určiť opakovanie animácie. Táto možnosť môže mať tri typy hodnôt: počet iterácií, časovú hodnotu alebo špeciálnu hodnotu *Forever*, vďaka ktorej sa bude animácia opakovať donekonečna.



XAML

```
<Canvas x:Name="LayoutRoot" Background="White">
  <Canvas.Resources>
    <Storyboard x:Name="sbPohyb">
      <PointAnimation Storyboard.TargetName="Kruh"
        Storyboard.TargetProperty="Center"
        From="70,70" To="200,200" Duration="0:0:5"
        AutoReverse="True" RepeatBehavior="Forever" />
    </Storyboard>
  </Canvas.Resources>
  <Path Fill="Blue">
    <Path.Data>
      <EllipseGeometry x:Name="Kruh" Center="70,70"
        RadiusX="50" RadiusY="50" />
    </Path.Data>
  </Path>
</Canvas>
```

Pre korektné spustenie animácie za behu aplikácie je nutné pridať potrebný kód pre spustenie.

C#

```
namespace SilverlightAnimacia1
{
	public partial class MainPage : UserControl
	{
		public MainPage()
		{
			InitializeComponent();
			 sbPohyb.Begin();
		}
	}
}
```

Po spustení tejto aplikácie nastane okamžité uvedenie kruhu do pohybu.

1.14.1 Animácia zmenou parametra pomocou Double Animation

Objekt *DoubleAnimation* umožňuje objekty animovať prostredníctvom zmeny hodnoty parametra typu double, teda desatinného čísla s dvojnásobnou presnosťou. Jeho implementácia bude demonštrovaná na obdĺžniku (*Rectangle*). Modifikovaná hodnota double bude v tomto prípade predstavovať hodnotu priehľadnosti (*Opacity*)

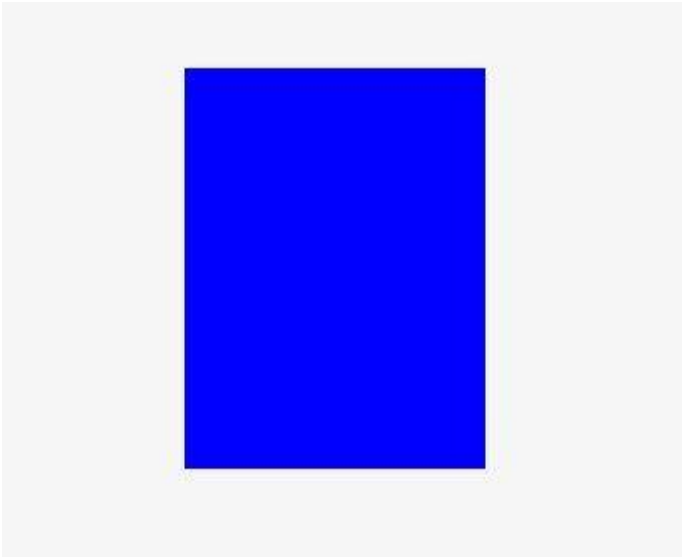
XAML

```
<Grid x:Name="LayoutRoot" Background="White">
	<Grid.Resources>
		<Storyboard x:Name="Priehladnost">
		    <DoubleAnimation
		     Storyboard.TargetName="Obdlznik"
		     Storyboard.TargetProperty="Opacity"
		     From="1.0" To="0.0" Duration="0:0:3"
		     AutoReverse="True" RepeatBehavior="Forever" />
		</Storyboard>
	    </Grid.Resources>
      <Rectangle x:Name="Obdlznik" Width="150" Height="100" Fill="Blue" />
</Grid>
```

C#
```
namespace SilverlightAnimacia2
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();
            Priehladnost.Begin();

        }
    }
}
```

1.14.2 Animácie farebnej zmeny pomocou Color Animation

Pri vytváraní animácií sa často vyskytne požiadavka na zmenu farby elementu. Pre tento účel je možné využiť Double Animation, keďže hodnoty parametrov charakterizujúcich farby sú celé čísla. Z tohto dôvodu je pre zmenu farby k dispozícii špecializovaný objekt pre animáciu Color Animation.



XAML
```
<Grid>
    <Grid.Resources>
        <Storyboard x:Name="Animacia">
            <ColorAnimation Storyboard.TargetName="Obdlznik"
                        Storyboard.TargetProperty="Color"
                        From="Blue" To="Yellow" Duration="0:0:5" />
        </Storyboard>
    </Grid.Resources>
    <Rectangle x:Name="Obdlznik" Width="150" Height="200">
        <Rectangle.Fill>
            <SolidColorBrush Color="Blue" />
        </Rectangle.Fill>
    </Rectangle>
</Grid>
```

C#

```
namespace SilverlightAnimacia3
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();
            Animacia.Begin();
        }
    }
}
```

1.15 Bezpečnosť platformy Silverlight

Na základe vyššie spomínaných faktov je zrejmé, že základný koncept platformy je vykonávanie plnohodnotného objektového kódu na strane klienta a klientskeho prehliadača. Z tohto dôvodu je otázka zabezpečenia Silverlight aplikácií veľmi dôležitá.

.NET Framework pracuje s bezpečnosťou assembly (balíčkov), obsahujúcich jednotlivé knižnice na základe tzv. Code Access Security (CAS) modelu. Podľa tohto modelu CLR (Common Language Runtime) pred tým, ako nahrá assembly, mu na základe jeho informácií pridelí skupinu, code group. Code group zároveň určuje, aké akcie sa môžu vykonávať. Kedykoľvek má kód vykonať akciu vyžadujúcu autorizáciu, CLR overí tieto povolenia. Pokiaľ nie sú povolenia platné, preruší vykonávanie. Povolenia k jednotlivým skupinám kódov môžu byť prideľované aj administrátorom, pretože ten je zároveň zodpovedný za stanovenie bezpečnostnej politiky. Pokiaľ ide o .NET Framework, môže každá metóda, ktorá prešla CAS, vykonať akúkoľvek činnosť, na ktorú má privilégiá. Implantácia CLR, ktorá sa nachádza v jadre Silverlight 2.0 (nazýva sa aj coreCLR), obracia tieto princípy naruby. Akýkoľvek zdrojový kód, ktorý prejde cez coreCLR, získava čiastočné práva a nemá oprávnenie volať metódy vyžadujúce vyššie privilégiá.[12]

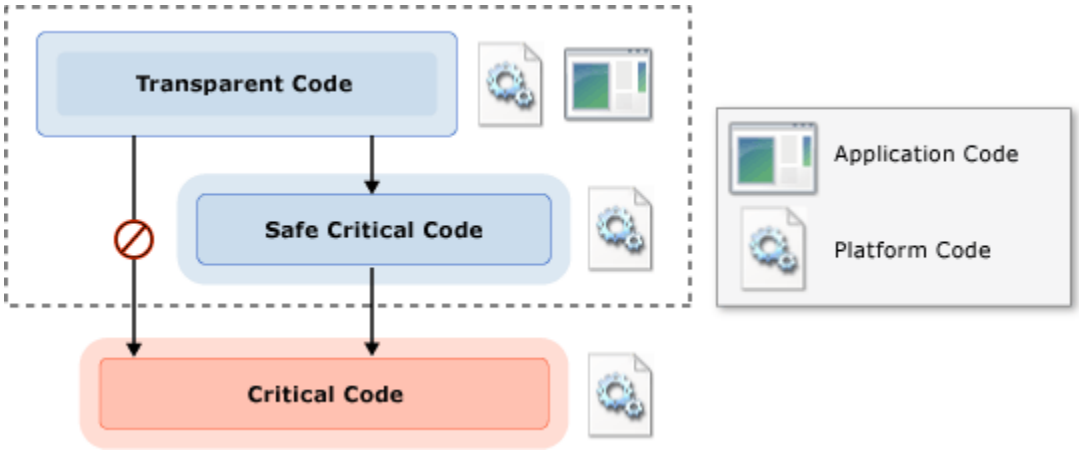
Bezpečnosť zdrojového kódu je v rámci jadra aplikácií Silverlight založená na nasledujúcich atribútoch:

SecurityTransparent – kód s čiastočnou dôveryhodnosťou, platný od verzie .NET Frameworku 2.0

SecuritySafeCritical – každý kód, ktorý je označený týmto atribútom, sa považuje za

dôveryhodný a môže byť volaný akýmkoľvek transparentným kódom. **SecurityCritical** – kód označený týmto atribútom má plnú dôveru jadra.

Každý atribút definuje inú úroveň zabezpečenia vykonávania kódu. Koncept bezpečnostne transparentného kódu znamená, že takýto kód nemôže vykonávať volania do dôveryhodného kódu alebo vykonávať akúkoľvek aktivitu, ktorá vyžaduje vyššiu úroveň povolení. Na rozdiel od .NET Frameworku Silverlight 2.0 coreCLR za takýto kód považuje všetky zdrojové súbory, ktoré nie sú označené iným atribútom, teda aj súbory bez označenia bezpečnostnej úrovne. Znamená to, že akýkoľvek kód obsiahnutý v aplikácii Silverlight 2.0 nemôže vykonávať akékoľvek kritické operácie alebo pristupovať na operačný systém. Ako možno vidieť v rámci jednotlivých druhov bezpečnostných atribútov, safe-critical atribút v .NET Frameworku neexistuje. Bol pridaný len pre bezpečnostný model aplikácií Silverlight. V skutočnosti identifikuje kód, ktorý je medzi aplikačným kódom (to, čo používatelia stiahnu z internetu) a platformovým kódom (miesto definovania kritických metód).



Obrázok 5 Atribúty bezpečnosti zdrojového kódu [1]

1.15.1 Aplikačný kód a platformový kód

Kód Silverlight sa dá rozdeliť na dve skupiny – aplikačný a platformový. Aplikačný je kód, ktorý vytvárajú vývojári a používatelia sťahujú do svojich počítačov. Platformový kód je ten, ktorý je potrebný na beh aplikácie Silverlight. Aplikačný kód môže byť len transparentný, akýkoľvek pokus o volanie metódy označenej ako critical má za následok okamžité vyvolanie výnimky bezpečnostného prístupu. Platformový kód môže byť akéhokoľvek typu. CoreCLR považuje za platformový kód len taký, ktorý je podpísaný špecifickým verejným kľúčom

spoločnosti Microsoft a je stiahnutý z bezpečného miesta na serveroch spoločnosti Microsoft. Tento kód má povolenie volať metódy, ktoré sú definované ako critical alebo safe-critical.

Pokiaľ aplikačný kód, ktorý sa spracúva, je v zmysle predchádzajúcej definície transparentný, možno považovať prostredie behu aplikácií Silverlight za zabezpečené. Stačí, ak vývojári produkujú transparentný kód a vytvárajú volania označené ako critical len prostredníctvom smart proxy, ktorý reprezentujú metódy bezpečnostného typu safe-critical.[12]

1.15.2 Šifrovanie údajov

Z hľadiska bezpečnosti je dôležité zabezpečiť údaje, napríklad proti zneužitiu, za pomoci šifrovania. Šifrovanie je proces zmeny otvoreného obsahu, vo väčšine prípadov otvoreného textu, do nezrozumiteľnej podoby zašifrovaného textu alebo postupnosti binárnych údajov. Dešifrovanie je opačný proces šifrovania, je to zmena zašifrovaného textu do otvoreného textu. Šifra je dvojica algoritmov, presnejšie predpisov na spracovanie, ktorými sa vykonáva šifrovanie a dešifrovanie. Operácia šifry je riadená algoritmom a kľúčom. Kľúč je tajný parameter, ktorý by mal byť známy iba príjemcovi a odosielateľovi pre špecifickú zámenu obsahu správy. Zašifrované údaje je možné uložiť do úložiska Isolated Storage

Pre prácu so šifrovacími algoritmami je možné použiť referenciu

```
using System.Security.Cryptography;
```

1.15.3 Autentizácia

Pri téme autentizácie je potrebné si uvedomiť, že v Silverlight projektoch je Silverlight aplikácia hosťovaná ako zásuvný modul v ASP.NET aplikácii. ASP.NET aplikácie pri autentizácii spolupracujú so službou IIS (Internet Information Services). Táto autentizácia používa cookies na overenie identity užívateľa a umožňuje aplikácii vykonať vlastné overenie prihlasovacích údajov.[11]

Autentizačné služby ASP.NET sú podriadené autentizačným službám poskytnutých IIS. V rámci URL autorizácie sa anonymný užívateľ porovnáva s konfiguračným údajom ASP.NET aplikácie. Ak je prístup povolený pre dané URL, požiadavke je autorizovaná.

Pre aktiváciu autentizačnej služby ASP.NET je potrebné nastaviť element *autentification* v konfiguračnom súbore aplikácie. Výhodnejšie je pre tento účel použiť nástroj ASP.NET.

WebSite Administration Tool

Pre Silverlight sa nastavuje autentizácia typu Forms. Autentizačné služby ASP.NET riadia cookies a smerujú neautentizovaných užívateľov na prihlasovaciu stránku.

Autentizácia sa spravidla odohráva zadaním mena a hesla v prihlasovacom dialógu alebo pomocou autentizačného hardware, napríklad prístupových kľúčov alebo biometrických zariadení. Pri pokuse užívateľa o prístup na chránenú stránku je tento najprv presmerovaný na autentizačný dialóg. Presmerovanie sa vykoná na základe údajov z konfiguračného súboru web.config zo sekcie *authentication.* [11]

```
<configuration>
   <system.web>
      <authentication mode=”Forms”>
         <forms loginUrl=“login.aspx“/>
      </authentication>
   </system.web>
</configuration>
```

V tejto sekcii sú uložené aj autorizačné údaje. Každý podadresár môže mať svoj vlastný súbor web.config s osobitným nastavením autorizačných parametrov. Napríklad, ak by bolo nutné zabrániť prístupu anonymných užívateľov, bolo by možné to dosiahnuť nastavením parametra *deny users* na hodnotu „?“. Kde otáznikový symbol zastupuje anonymných užívateľov.

```
<configuration>
   <system.web>
      <authorization>
         <deny users="?"/>
      </authorization>
   </system.web>
</configuration>
```

Kód aplikácie Silverlight môže obsahovať závislosti (dependencies) od iných, vlastných knižníc a balíčkov. No existujú aj obmedzenia. Projekt Silverlight môže odkazovať len na knižnice, ktoré boli vytvorené a kompilované ako Silverlight Class Libraries. Existujúce knižnice tak musia byť prekompilované, prípadne čiastočne prispôsobené tak, aby mohli byť prekompilované ako knižnice typu Silverlight. Bezpečnostný model coreCLR umožňuje programátorovi derivovať nové triedy len z tried, ktoré sú označené aplikačnou platformou ako security transparent. Za takú sa považuje každá, ktorá neobsahuje bezpečnostný atribút typu SafeCritical alebo Critical. Rozsah tohto použitia je oveľa širší, najmä z pohľadu bezpečnostných definícií metód obsiahnutých v jednotlivých knižniciach, resp. z hľadiska možnosti preťažovania jednotlivých metód.

1.16 MS Silverlight vs. Adobe Flash

Flash a Silverlight predstavujú platformy na tvorbu webu multimediálneho obsahu, ktoré umožňujú prehliadačom s potrebným nainštalovaným pluginom, zobrazovať weby s výraznou interaktivitou, pokročilou animáciou a bohatým multimediálnym obsahom. Flash platforma, bola pôvodne vyvinutá spoločnosťou Macromedia, v súčasnosti ju vlastní spoločnosť Adobe, ktorá kúpila Macromedia.

Popularita

Flash má predovšetkým na internete širokú podporu. Podľa niektorých štatistík je Flash nainštalovaný až na 96% osobných počítačoch, ktoré sú pripojené na internet. Keďže Silverlight je platforma, ktorá prišla na trh podstatne neskôr, má v porovnaní s Flashom omnoho menšie zastúpenie na trhu. Pomocou Flash je možné vytvárať nielen animácie webov. Takmer všetky webové stránky, ktoré streamujú video alebo zvuk, používajú Flash na zdieľanie svojho obsahu. Silverlight taktiež nezaostáva v tejto oblasti, streamovanie videa je lacnejšie, a efektívnejšie ako v platforme Flash.[18]

Hlavný rozdiel medzi platformami Silverlight a Flash spočíva v ich prístupe k tvorbe aplikácií. Flash ide cestou blízkou vývoju webových aplikácií, založených na JavaScripte, HTML alebo CSS. Silverlight pristupuje k tvorbe aplikácií inou cestou. Pre vývoj klasických desktopových aplikácií boli prevzané a prispôsobené technológie tak, aby sa dali implementovať pri tvorbe web stránok s bohatým multimediálnym obsahom.

Animácie

Animácie v platforme Flash sú závislé od počtu zobrazovaných snímok, z ktorých sa animácia skladá. Naopak, animácie v platforme Siverlight vychádzajú z WPF (Windows Presentation Foundation) a ich vlastnosti závisia viac na čase prehrávania animácie ako na počte zobrazovaných snímok. Silverlight týmto umožňuje stanoviť štartovacie a konečné podmienky animácie bez nutnosti kalkulácie pozícií jednotlivých snímok.

Vývojové prostredia

Microsoft ponúka široké množstvo vývojových prostredí, v ktorých je možné Silverlight aplikácie vyvíjať. Microsoft Visual Studio 2012, spolu so staršími verziami a Microsoft Expression Studio predstavujú dva hlavné nástroje, ktoré môžu vývojári využiť na tvorbu Silverlight animácií a aplikácií. Obe prostredia využívajú XAML a .NET programovacie jazyky.

Adobe taktiež disponuje vlastnými developerskými nástrojmi pri vytváraní Flash aplikácií, najpopulárnejšie sú Adobe Flash a FlashDevelop. Vývoj aplikácií sa realizuje prostredníctvom skriptovacieho jazyka Action Script, ktorý zahŕňa objektovo orientované funkcie a podporuje jazyky ako napríklad PHP, ASP alebo HTML.

Podpora operačných systémov

Obidve platformy sú schopné fungovať na operačných systémoch Windows 8/7/Vista/XP/2000, Windows Server 2003/2008/2012, Mac OS (Power PC), Mac OS (Intel). Flash aplikácie môže taktiež využívať aj operačné systémy Linux 5, open SUSE 11, Ubuntu (od verzie 7.10) a operačný systém Solaris 10.

Podpora obrazových formátov

Flash zahŕňa podporu pre formáty GIF, JPEG, PNG a PICT (Adobe Flash Video, 2010).Silverlight podporuje formáty súborov JPEG, GIF, PNG a Bitmap (Microsoft, 2012).

Textová reprezentácia

Fonty písma sú v platforme Flash ukladané ako definície tvaru, čo spôsobuje neschopnosť oddelenia textovej vrstvy od animácie. Klasicky napísaný text vo Flash komponente nebol nikdy priateľský k SEO (Optimalizácia pre vyhľadávače). Neskôr Adobe vytvorilo inovácie, vďaka ktorým môže byť text indexovaný, čím sa spolupráca so SEO uľahčila.

V platforme Silverlight je text generovaný na web serveri ako oddelená časť a je možné k nemu pristupovať zvlášť. Text je ľahko zapisovateľný, pretože nie je skomprimovanou súčasťou, ale je reprezentovaný prostredníctvom XAML.[18]

Výhody platformy Microsoft Silverlight

- Vývoj desktopových aplikácií (Out of Browser)
- Podpora GPU akcelerátorov, podpora prehrávania HD videa
- IIS Smooth Streaming
- Podpora veľkého množstva audio a video formátov
- Podpora DRM (Digital Rights Management)

Nevýhody platformy Microsoft Silverlight

- Slabá trhová penetrácia (inštalovaný iba na 60% počítačoch s prístupom na web)
- Celý vývoj musí byť vykonávaný v prostredí Microsoft Windows

Výhody platformy Flash

- Umožňuje vývoj v oboch operačných systémoch Mac i Windows
- FMS (Flash Media Server) Streaming and Security
- Silná trhová penetrácia (inštalovaný až na 98% počítačoch s prístupom na web)

Nevýhody platformy Flash

- Absencia podpory DRM
- Absencia podpory GPU

1.17 Dátové väzby

Dátové väzby (data binding) sú ďalšie vlastnosti konceptu WPF, ktoré veľmi zefektívňujú vývoj pri práci s dátami. Ponúkajú alternatívu, pomocou ktorej je jednoduchšie menežovať a prezentovať dáta v klientskej aplikácii Silverlightu. Dátová väzba realizuje proces predávania dát z dátového zdroja do cieľa (ovládacie prvky v aplikačnom rozhraní) a v prípade potreby aj odosielanie dát späť do zdroja. Pre dátové väzby sa nielen v Silverlight používa pojem Data Binding. Dátové väzby pomáhajú minimalizovať potrebný zdrojový kód

a tým zefektívniť vývoj aplikácií, avšak nie sú jedinou možnosťou ako previazať ovládacie prvky so zdrojom dát.

Automatické dátové väzby v Silverlight predstavujú proces, ktorý prepojuje ovládací prvok užívateľského rozhrania s entitou dátového zdroja a naopak. Entita obsahuje dáta, ktoré budú predané do ovládacieho prvky, ktorý ich zobrazí. Dáta môžu prúdiť aj opačnou cestou. Teda z ovládacieho prvku do entity. Entita môže mať napríklad väzbu k sade ovládacích prvkov z ktorých každý je prepojený s určitou vlastnosťou entity. [16]

V dátovej väzbe je vždy vyžadovaný dátový zdroj a aspoň jeden cieľ. Cieľom môže byť ovládací prvok, alebo skupina ovládacích prvkov na stránke (alebo v rámci užívateľského rozhrania). Dáta môžu byť medzi zdrojom a cieľom prenášané v jednom alebo v dvoch smeroch. Zdrojom dát sú zvyčajne inštancie objektov. Na vlastnosť dátového zdroja je vždy odkaz v cieli, čo znamená, že cieľový ovládací prvok vie, ku ktorej vlastnosti zdrojovej entity má vytvoriť dátovú väzbu.

1.17.1 Závislostná vlastnosť

Silverlight umožňuje realizovať dátové väzby iba v takom prípade, keď daná vlastnosť cieľového prvku je tzv. závislostná vlastnosť (Dependency Property). Ako príklad sa dá uviesť vlastnosť *Text* ovládacieho prvku *TextBox*, alebo vlastnosť *Fill* prvku *Rectangle*. Obe tieto vlastnosti, a väčšina rôznych vlastností ostatných prvkov, ktoré môžu závisieť od určitých parametrov, túto podmienku spĺňa. Využitie dátovej väzby pre nastavenie vlastnosti *Fill* prvku *Rectangle* je vidieť na nasledujúcom príklade. Ako zdroj dátovej väzby je využitá vlastnosť *FarbaVyplne* objektu pomenovaného ako *ObjektStvorec*.

XAML
```
<Rectangle Fill="{Binding ObjektStvorec.FarbaVyplne} Grid.Column="0"
Grid.Row="0" RadiusX="25" RadiusY="25" Opacity="0.6" StrokeThickness="0" />
```

Popri dátových väzbách patria k výhodám závislostných vlastností aj štýly, animácie a šablóny. Vzhľad ovládacích prvkov sa dá nastavovať pomocou závislostnej vlastnosti *Style*, čo umožňuje dosiahnutie elegantnejšieho vzhľadu a ľahšej zmeny dizajnu aplikácie.

Hodnoty závislostných vlastností ovládacích prvkov sú vnorené prvky, ktoré sú dedené od prvkov nadriadených, tzv. Kontajnerov. Zo zdroja štýlu môže mať napríklad prvok

Canvas (kresliaca plocha) nastavenú vlastnosť *Background* (pozadie) na *White* (biela). Na kresliacu plochu môže byť vložený prvok *Rectangle* a svoju vlastnosť *Background* bude preberať od svojho kontajnerového prvku (v tomto prípade *Canvas*). Pokiaľ by *Canvas* nemal definovaný štýl pre tento parameter, bude mu pridelená hodnota z hierarchicky nadradeného prvku. Ak chceme farbu obdĺžnika meniť, máme možnosť zapísať požadovanú hodnotu do vlastnosti *Background* prvku *Rectangle*, pretože priamo nastavená hodnota vlastnosti má vyššiu prioritu ako hodnota zdedená.

1.17.2 Režim väzieb

Režim väzieb a s ním súvisiaca funkčnosť sú veľmi dôležitou súčasťou dátových väzieb. Väzba ovládacieho prvku určuje, akým spôsobom má byť zdroj dát prepojený s cieľom dátovej väzby. Tu sú dáta určené iba pre čítanie alebo sa môže prostredníctvom ovládacieho prvku určiť, či väzba musí prebiehať v oboch smeroch, takže používatelia môžu upravovať hodnoty ovládacích prvkov a tým aj vlastnosti prepojenej entity (zdroja dátovej väzby). [1]

Tento sa zapisuje pomocou kľúčového slova *Mode* rozšíreného zápisu väzby a určuje či a kedy bude aktualizovaný cieľ alebo zdroj väzby. Východisková hodnota vlastnosti Mode sa líši v závislosti na ovládacom prvku. Existujú tri možné nastavenia režimu väzby: OneTime, OneWay, TwoWay.

One Time

V režime väzby OneTime sú hodnoty odosielané do cieľa iba v prípade, keď je zdrojový objekt nastavený ako zdroj väzby. Ak je napríklad vlastnosť *DataContext* ovládacieho prvku nastavená na objekt, stane sa objekt zdrojom dátovej väzby. Ak k tomu dôjde, obsiahne všetky väzby pre tento zdroj hodnoty z príslušných vlastností zdroja.

Väzba OneTime je ideálna pre väzbu zdrojového objektu s ovládacím prvkom, ktorý bude zobrazovať informácie iba na čítanie a nie je predpoklad vykonávania zmien v zdroji. Pokiaľ by bola vykonaná zmena v zdroji, nová hodnota by pri použití väzby OneTime nebola odoslaná do cieľa.

OneWay

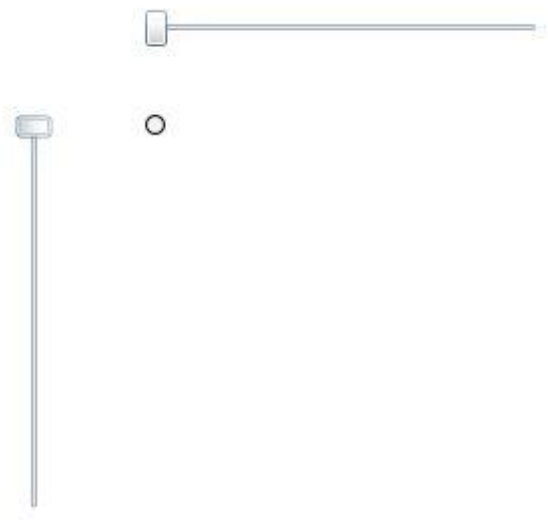
Väzba OneWay odosiela hodnoty zo zdroja do cieľa po vytvorení väzby, zároveň však registruje všetky zmeny v zdrojovom objekte a odosiela ich taktiež do cieľa. V prípade, že

zdrojový objekt implementuje rozhranie InotifyPropertyChanged a jeho settery vlastnosti vyvolajú udalosť PropertyChanged, cieľ väzby zaznamená túto udalosť a obdrží aktualizovanú hodnotu.

Väzba OneWay je vhodná v prípadoch, že užívateľské rozhranie neumožňuje používateľovi upravovať dáta, ale akékoľvek zmeny v zdroji možno odoslať do užívateľského rozhrania.

Two way

Umožňuje rovnaké akcie ako väzba OneWay. Má ale hodnotu v podobe funkcie odosielania dát pozmenených v cieli späť do zdroja. Využíva sa v prípade, že používateľ musí mať možnosť zmeniť dáta v ovládacom prvku a zmeny sa majú prejaviť v zdrojovom objekte.



Obrázok 6 Jednoduchý príklad dátovej väzby

XAML

```
<Grid x:Name="LayoutRoot" Background="White">
    <Ellipse Fill="#FFF4F4F5"  HorizontalAlignment="Left" Height="{Binding
ElementName=Vyska, Path=Value}"
             Margin="161,119,0,0" Stroke="Black" VerticalAlignment="Top"
Width="{Binding
             ElementName=Sirka, Path=Value}"/>
    <Slider x:Name="Sirka" Maximum="200" Minimum="10" HorizontalAlignment="Left"
            Margin="146,290,0,0" VerticalAlignment="Top" Height="23" Width="200"/>
        <Slider x:Name="Vyska" Maximum="200" Minimum="10" Orientation="Vertical"
HorizontalAlignment="Left"
                Margin="107,96,0,0" VerticalAlignment="Top" Height="163"
RenderTransformOrigin="0.5,0.5"/>
</Grid>
```

Na príklade sú znázornené dva elementy typu *Slider* , ktoré sú previazané s parametrami elipsy. Pomocou značky *Binding* je vytvorená väzba medzi *Sliderom* s názvom „Sirka“ a šírkou elipsy. Obdobne je prepojený *Slider* s názvom „Vyska“ s parametrom *Width* (ktorý predstavuje výšku elipsy).

1.17.3 Oznamovanie

Medzi dve kľúčové dátové väzby založené na XAML, patrí InotifyPropertyChanged a InotifyCollectionChanged. Tieto rozhrania poskytujú objektom dátových zdrojov prostriedky k tomu, aby dátové väzby dostali informáciu, že v zdroji došlo k zmene. Kolekcia ObservableCollection<T> implementuje rozhranie InotifyCollectionChanged, ktoré obsahuje triedu kolekcie schopnú oznamovať zmenu položiek v kolekcii.

Či už je režim väzby nastavený na OneWay alebo OneTime, cieľ o aktualizáciu dátového zdroja nie je informovaný, pokiaľ dátový zdroj neimplementuje rozhranie InotifyPropertyChanged. Ani pri použití režimu väzby OneWay by hodnota nebola aktualizovaná. Bez rozhrania InotifyPropertyChanged má režim OneWay rovnaké správanie ako režim OneTime. Pokiaľ nie sú očakávané žiadne zmeny dát v zdroji, je najvýhodnejšie použiť väzbu OneTime, pretože neregistruje udalosti PropertyChanged a poskytuje tak najlepší výkon. [16]

1.17.4 Zoznamy

Silvelirght podporuje nielen dátové väzby medzi jednoduchým ovládacím prvkom (TextBox, TextBlock a iné) a vlastnosťou entity dátového zdroja. Podobne jednoduché je vytvorenie dátovej väzby medzi entitou zoznamu a ovládacím prvkom používateľského rozhrania, napríklad typu ListBox.

Ovládacie prvky založené na zoznamoch môžu tiež využívať oznámenie, pokiaľ sú zviazané s entitami zoznamov vytvorenými pomocou kolekcie ObservableCollection<T> alebo triedami kolekcií, ktoré implementujú rozhranie INotifyCollectionChanged.

1.18 Silverlight a mobilné zariadenia

Silverlight ponúka svoje vymoženosti aj na mobilné zariadenia Windows Phone. Podporované sú nástroje ako napríklad:

- Vysoko kvalitné audio a video za podpory širokej škály kodekov, DRM a IIS Smooth Streaming
- Funkcia Deep Zoom pre pokročilé zobrazovanie obrázkov a zoomovanie.
- Vektorová a Bitmapová grafika a animácie

Silverlight má prístup aj k hardvérovým prvkom mobilných telefónov ako napríklad

- Kamera a mikrofón
- Dotykové senzory
- Pohybové senzory
- Hardvérové komponenty pre podporu videa a grafiky

Silverlight dokáže využiť XNA framework for audio, Media Library Access a má tiež prístup aj k Xbox live. Tieto nástroje sú prístupné pomocou zdrojového kódu a môžu byť jednoducho implementované prostredníctvom pridania špeciálnych knižníc do existujúcich Silverlight aplikácií. [10]

Developer následne dokáže zúžitkovať tieto funkcie priamo pri vývoji mobilnej aplikácie. Organizácie, vyvíjajúce aplikácie takéhoto typu, môžu svoje koncové produkty sprístupniť na Windows Phone Marketplace, odkiaľ si môžu užívatelia zadarmo stiahnuť ich demo verzie alebo kúpiť hotové aplikácie.

Silverlight pre Windows Phone podporuje vstavaný try/buy API systém, pomocou ktorého môže užívateľ konkrétnu aplikáciu najprv bezplatne vyskúšať. V prípade, že je s ňou spokojný, si ju môže zakúpiť. Tento systém zjednodušuje proces transformácie trial verzie na úplnú verziu, ako pre vývojárov tak aj pre užívateľov.[10]

Vývojári majú k dispozícii rovnaké nástroje ako na vývoj aplikácií pre osobné počítače. Podporu pri vývoji, ladení a spúšťaní aplikácie im uľahčuje Windows Phone Emulator, ktorý simuluje prostredie Windows Phone. Týmto odpadá nutnosť vykonávať tieto procesy na reálnom mobilnom zariadení.

1.19 Hlavné vymoženosti platformy Silverlight

1.19.1 Deep Zoom

Deep Zoom je technológia, ktorá bola vyvinutá firmou Microsoft a priniesla nové možnosti interakcie s obrazovými médiami. Slúži na efektívne aranžovanie, a zobrazovanie obrázkov. Užívateľovi umožňuje umiestňovať obrázky do panorámy alebo mozaiky, ktoré následne môže približovať alebo vzďaľovať. Tieto je možné ďalej exportovať ako viacero jednotlivých obrázkov, alebo ako jeden komplexný.

Najsilnejšia vlastnosť technológie Deep Zoom spočíva v efektívnom načítavaní obsahu. Načítava sa vždy len tá časť, ktorá môže byť aktuálne zobrazená na obrazovke. Ďalšie časti sú načítané v prípade, že dôjde k posunu alebo približovaniu obrázku, kedy sa načítajú ich detaily vo väčšom rozlíšení.[14]

Obrazové média sa ukladajú za pomoci troch typov súborov:

Deep Zoom Image (.dzi)

Tento typ má dve časti: DZI súbor (s .dzi alebo .xml príponou) a podsúbor obsahujúci obrazové súbory. Každý priečinok v tomto podsúbore je označený svojím rozlíšením. Vyššie číslo zodpovedá vyššiemu rozlíšeniu. V každom súbore sú obrazové dlaždice, ktoré disponujú uvedeným rozlíšením.

Deep Zoom Collection (.dzc)

Predstavuje kolekciu čísel z .dzi súboru prepojenú a referencovanú prostredníctvom .dzc súboru. V .dzc súbore sú uložené umiestnenia na obrazové súbory, ktorých časti sa dodatočne načítavajú.

Sparse Images (roztrúsené obrázky)

Predstavujú podtriedu súborov .dzi. Tento typ súboru predstavuje množinu izolovaných obrázkov s rôznymi rozlíšeniami, ktoré môžu byť umiestnené v jednom .dzi súbore namiesto viacerých .dzc súboroch.

Efektívnym nástrojom na tvorbu Deep Zoom galérií je nástroj Deep Zoom Composer. Za jeho pomoci môže autor importovať, rozvrhovať a exportovať obrazové médiá.

1.7.1 Pivot Viewer

Toto vylepšenie uľahčuje prácu s veľkými množinami dát prostredníctvom prehľadnej a informatívnej vizualizácie. Vizualizáciou veľkého množstva subjektov naraz môže užívateľ vidieť súvislosti a vzory, ktoré môžu byť inak menej viditeľné. Za pomoci animácií a príťažlivého vzhľadu pôsobia na užívateľa pozitívne i napriek veľkému množstvom dát. Pivot Viewer je vhodný ako na prácu s dátami, tak aj na prezentačné účely.

1.7.2 Smooth Streaming

Silverlight Smooth Streaming je technológia implementovaná v IIS Media Services (Internet Information Services). Umožňuje prehrávanie videa, počas ktorého sa bez akéhokoľvek prerušenia proces prehrávania adaptuje na kvalitu linky medzi serverom a klientským počítačom a na grafický výkon klientského počítača. V praxi to znamená, že video sa distribuuje v takom rozlíšení, aké dokáže linka preniesť. Zjednodušene povedané, ide o technológiu tzv. „hladkého streamovania“, ktorá zaručuje neprerušený obraz bez trhania. Klientská aplikácia si zo serveru sťahuje malé bloky 3-4 sekundových záznamov a podľa okolností (prenosová rýchlosť, grafický výkon) si volí kvalitu, či presnejšie, rozlíšenie. Bloky sú zarovnané a je možné plynule prechádzať medzi rôznymi stupňami kvality.

1.7.3 SketchFlow

SketchFlow je nástroj vo vývojovom prostredí Microsoft Visual Studio, ktorý je prístupný po inštalácii rozšírenia Blend for Visual Studio. SketchFlow je navrhnutý tak, aby podporoval vývoj procesu, od návrhu konceptuálneho dizajnu až po finálnu aplikáciu. Vyniká rýchlym a dynamickým prostredím, ktoré je jednoduché na obsluhu. Uľahčuje komunikáciu s dodávateľmi a odberateľmi a zjednodušuje ich proces spätnej väzby.

S nástrojom SketchFlow dokáže dizajnér rýchlo a ľahko vytvárať interaktívne mapy, ktoré reprezentujú proces vývoja aplikácie. Mapa (projektový prototyp) pozostáva z ľubovoľného množstva virtuálnych plôch, na ktoré je možné črtať, kresliť alebo do nej importovať ľubovoľný element aplikačného rozhrania.

Pri vytváraní prototypu môže vývojár kedykoľvek spustiť projekt a následne ho vidieť v „SketchFlow Player“. Prototypy vytvorené v SketchFlow nie sú len kresby, sú to plne funkčné WPF alebo Silvelright aplikácie. Prostredníctvom jednoduchých modifikácií je

možné prototyp konvertovať na produkčný projekt a ďalšou prácou transformovať prototyp na finálnu aplikáciu.[16]

1.7.4 Perspective 3D

Nástroj perspective 3-D môže vývojár použiť na ľubovoľný prvok aplikačného rozhrania a pomocou neho vytvoriť dojem perspektívnej transformácie. Môže napríklad vytvoriť ilúziu rotácie predmetu alebo obrázok (viď obrázok 6).

Obrázok 7 Jednoduchý príklad Perspective 3-D

XAML

```
<Grid x:Name="LayoutRoot" Background="White">
    <Slider x:Name="sliderX" Minimum="0" Maximum="360" HorizontalAlignment="Left"
            Margin="223,457,0,0" VerticalAlignment="Top" Width="300"/>
    <Slider x:Name="sliderY" Minimum="0" Maximum="360" Orientation="Vertical"
            HorizontalAlignment="Left" Margin="152,129,0,0" VerticalAlignment="Top"
Height="300"/>
    <Slider x:Name="sliderZ" Minimum="0" Maximum="360" Orientation="Vertical"
            HorizontalAlignment="Left" Margin="605,129,0,0" VerticalAlignment="Top"
Height="300"/>
    <Image Source="m_1298360647824.jpg" Margin="255,173,0,0"
HorizontalAlignment="Left"
           VerticalAlignment="Top" Height="226" Width="253">
        <Image.Projection>
            <PlaneProjection RotationX="{Binding ElementName=sliderX, Path=Value}"
                             RotationY="{Binding ElementName=sliderY, Path=Value}"
                             RotationZ="{Binding ElementName=sliderZ, Path=Value}">
            </PlaneProjection>
        </Image.Projection>
    </Image>
</Grid>
```

V príklade sú znázornené tri elementy typu *Slider* a obrázok. Pri posúvaní Sliderov vložený obrázok rotuje po osi X, Y a Z. Každej osi zodpovedá jeden Slider Elementy typu Slider a navzájom sú previazané s PlaneProjection atribútmi vloženého obrázku. Element *sliderX* je previazaný s atribútom *RotationX* , ktorý určuje rotáciu obrázka v po osi „x“. Obdobne sú previazané ďalšie dva slidery ktoré určujú rotáciu po osiach „y“ a „z“.

1.7.5 Štýly a Šablóny

Použitie štýlov značne uľahčuje a urýchľuje prácu v prípadoch, kedy sa na scéne nachádza viacero objektov rovnakého vzhľadu. Jedná sa o akúsi analógiu k technológii CSS, v ktorej sú grafické definície uchovávané na určitom mieste a objekty sa na ne odkazujú. Šablóny sú užitočným nástrojom pre navrhovanie grafického vzhľadu jednotlivých komponentov. S ich pomocou je možné od základov meniť akýkoľvek XAML prvok. Prostredníctvom štýlov a šablón môže vývojár implementovať vizuálne zmeny pre množinu príbuzných elementov bez nutnosti modifikácie ich značiek. Táto separácia kódu a dizajnu spôsobuje rast v dizajnérskom a vývojárskom pracovnom toku.

Cieľ práce a použité metódy

Cieľom mojej práce bolo popísanie teoretických poznatkov o platforme Silverlight a ich následné použitie v praxi. Hlavným cieľom bolo vytvorenie interaktívnej aplikácie – hry. Pri pracovnom postupe som vychádzal z osobných znalostí a skúseností. Na tvorbu aplikačnej logiky bol použitý objektovo – orientovaný jazyk C# a na tvorbu aplikačného rozhrania značkovací jazyk XAML . Na vývoj aplikačnej logiky bolo použité vývojové prostredie Microsoft Visual Studio 2013 a na tvorbu užívateľského rozhrania bolo použité prosterdie Microsoft Expression Blend 4.

Pri tvorbe som postupoval systémovým prístupom. Najprv boli navrhnuté jednotlivé menšie časti ktoré sa postupom tvorby spojili do komplexného prvku. Pri tvorbe funkcií aplikácie bol uplatnený obdobný prístup, najprv boli navrhnuté jednoduché funkcie, neskôr boli vytvorené komplexnejšie, ktoré zabezpečovali interaktivitu medzi prvkami aplikačného rozhrania. Zahrnutá bola aj vymoženosť klávesnicového vstupu, ktorý zvýšil interakciu aplikácie s užívateľom.

2 Výsledky práce

V predchádzajúcich kapitolách boli popísané a vysvetlené prvky, na základe ktorých je možné vytvárať v platforme Silverlight interaktívne aplikácie s animáciami. . Tieto prvky sú taktiež potrebné na vytváranie hier a keďže ich možno Silverlightom realizovať stáva sa týmto vhodným programom na vývoj hier. V súčasnosti existuje mnoho hier vytvorených na platforme Silverlight, od jednoduchých, až po náročné a zložité.

Cieľom tejto diplomovej práce je vytvorenie jednoduchej akčnej hry, ktorej princíp spočíva v ničení mimozemských lodí. Hráč môže disponovať svojou vesmírnou loďou, ktorá bude strieľať rakety. Mimozemské lode budú schopní vypúšťať bomby na raketu a budú prichádzať v nájazdoch. Počet mimozemšťanov v nájazdoch sa bude s pribúdajúcou obtiažnosťou zvyšovať. Mimozemská a vesmírna loď budú postavené oproti sebe a budú strieľať každá opačným smerom. Proces tvorby tejto hry je zdokumentovaný krok za krokom v nasledujúcej podkapitole.

2.1. Proces tvorby hry

Na začiatok musí byť spustené Visual Studio a vytvorená nová Silverlight aplikácia v .NET programovacom jazyku C#. Najskôr je vhodné začať s grafickým rozvrhnutím (Layout) aplikácie. Ako prvé, je pridaný prvok *Canvas* do elementu *Grid* v stránke Page.xaml, ktorá bola vytvorená. Farba jej pozadia je nastavená na čiernu a nazvaná “gameRoot”.

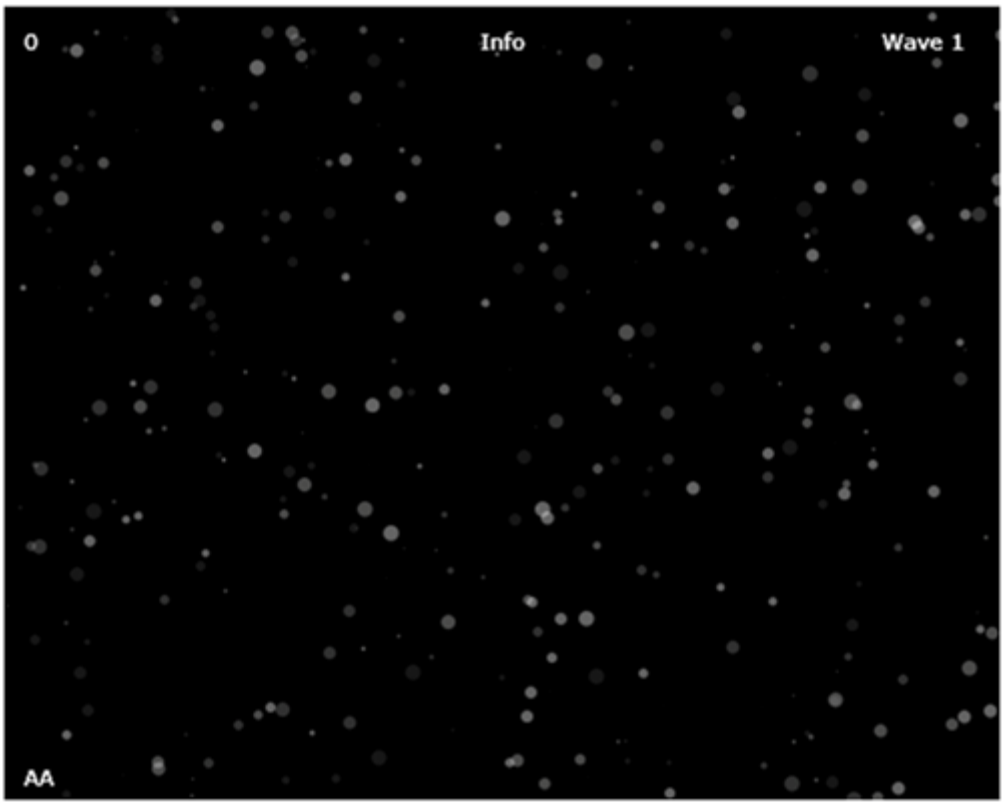
XAML
```
<UserControl x:Class="SimpleShooter.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
   <Grid x:Name="LayoutRoot">
        <Canvas x:Name="gameRoot" Width="500" Height="400" Background="Black" >
        </Canvas>
    </Grid>
```

Ďalším krokom bude pridávanie ovládacích prvkov a informačných elementov do tohto projektu. Na dokončenie herného prostredia budú pridávané užívateľské ovládacie prvky: *Info, LivesRemaining, Score* a *Waveinfo*. Na začiatok pridáme základné elementy *TextBlock* na zobrazovanie informácií o stave hry. Umiestnené sú do *Page.xaml* a každému z nich je priradené x:Name meno. V tomto bode majú ovládacie prvky v sebe umiestnený iba *TextBlock*.

XAML

```
<UserControl x:Class="SimpleShooter.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:SimpleShooter="clr-namespace:SimpleShooter"
    Width="500" Height="400">
    <Grid x:Name="LayoutRoot">
        <Canvas x:Name="gameRoot" Width="500" Height="400" Background="Black">
            <SimpleShooter:RemainingLives x:Name="ctlLives" Canvas.Top="380"
Canvas.Left="10" />
            <SimpleShooter:Score x:Name="ctlScore" Canvas.Top="10"
Canvas.Left="10" />
            <SimpleShooter:WaveInfo x:Name="ctlWaveInfo" Canvas.Left="440"
Canvas.Top="10" />
            <SimpleShooter:Info x:Name="ctlInfo" Canvas.Top="10"/>
        </Canvas>
    </Grid>
</UserControl>
```

Na dokončenie herného prostredia bude pridané hviezdne pole. Bude vytvorená funkcia na generovanie náhodných čísel a ďalšia, ktorá náhodne umiestni elementy *Ellipse* ktoré budú umiestnené na základné plátno. Na túto realizáciu sa bude dediť od triedy *System.Security.Cryptography*.



Obrázok 8 Hviezdne pole (snímka obrazovky)

C#
```
public partial class Page : UserControl
{
    public Page()
    {
        InitializeComponent();

        GenerateStarField(350);
    }

    void GenerateStarField(int numberOfStars)
    {

        for (int i = 0; i < numberOfStars; i++)
        {

            Ellipse star = new Ellipse();
            double size = GetRandInt(10, 800) * .01;
            star.Width = size;
            star.Height = size;
            star.Opacity = GetRandInt(1, 5) * .1;
            star.Fill = new SolidColorBrush(Colors.White);
            int x = GetRandInt(0, (int)Math.Round(gameRoot.Height, 0));
            int y = GetRandInt(0, (int)Math.Round(gameRoot.Width, 0));
            star.SetValue(Canvas.TopProperty, (double)x);
            star.SetValue(Canvas.LeftProperty, (double)y);
            gameRoot.Children.Add(star);
        }
    }

    public int GetRandInt(int min, int max)
    {
        Byte[] rndBytes = new Byte[10];
        RNGCryptoServiceProvider rndC = new RNGCryptoServiceProvider();
        rndC.GetBytes(rndBytes);
        int seed = BitConverter.ToInt32(rndBytes, 0);
        Random rand = new Random(seed);
        return rand.Next(min, max);
    }
```

Bola vytvorená funkcia, ktorá generuje hviezdne pole, nazvaná *GenerateStarField*, ktorá generuje Elipsy. Pri ich generácii sa z ohraničených intervalov náhodne generuje ich veľkosť, poloha a prehľadnosť. Týmto je vytvorené základné prostredie hry a pozadie. Následne bude vytvorená trieda *Sprite*, pomocou ktorej budú animované pohyby prvkov v hre. Pridané budú tiež vektory.

C#

```
public abstract class Sprite
{
    public double Width { get; set; }
    public double Height { get; set; }
    public Vector Velocity { get; set; }
    public Canvas SpriteCanvas { get; set; }
    private Point _position;
    public Point Position
    {
        get
        {
            return _position;
        }
        set
        {
            _position = value;
            SpriteCanvas.SetValue(Canvas.TopProperty,
 _position.Y - (Height / 2));
            SpriteCanvas.SetValue(Canvas.LeftProperty, _position.X - (Width /
2));
        }
    }
    public Sprite(Double width, Double height, Point position)
    {
        Width = width;
        Height = height;

        SpriteCanvas = RenderSpriteCanvas();

        SpriteCanvas.Width = width;
        SpriteCanvas.Height = height;
        Position = position;
    }
    public abstract Canvas RenderSpriteCanvas();
    public Canvas LoadSpriteCanvas(string xamlPath)
    {
        System.IO.Stream s =
this.GetType().Assembly.GetManifestResourceStream(xamlPath);
        return (Canvas)XamlReader.Load(new
System.IO.StreamReader(s).ReadToEnd());
    }
    public virtual void Update(TimeSpan elapsedTime)
    {
        Position = (Position + Velocity * elapsedTime.TotalSeconds);
    }
}
```

Trieda *Sprite* poskytne základ pre animácie entít v hre ako napríklad vesmírna loď, mimozemšťania a projektily. Pre všetky tieto entity musí byť známa ich pozícia a vzhľad. Point (Bod) je využitý na sledovanie pozície a vlastnosti elementu *Canvas* na XAML

zobrazenie každej entity. Konštruktor nastavuje tieto inicializačné parametre a volá metódu *RenderSpriteCanvas*, ktorá bude implementovaná v každej triede, ktorá od nej dedí. Táto metóda umožňuje dediacej triede nastaviť obsah *Canvas* a ovládanie *Sprite* triedy.

Ďalší krok je vytvorenie triedy *Vector*, ktorá pomôže kontrolovať pohyb *Sprite* entity.

C#
```
public struct Vector
{
    public double X;
    public double Y;
    public Vector(double x, double y)
    {
        X = x;
        Y = y;
    }

    public double Length
    {
        get
        {
            return Math.Sqrt(LengthSquared);
        }
    }

    public double LengthSquared
    {
        get
        {
            return X * X + Y * Y;
        }
    }

    public void Normalize()
    {
        double length = Length;
        X /= length;
        Y /= length;
    }

    public static Vector operator -(Vector vector)
    {
        return new Vector(-vector.X, -vector.Y);
    }

    public static Vector operator *(Vector vector, double scalar)
    {
        return new Vector(scalar * vector.X, scalar * vector.Y);
    }

    public static Point operator +(Point point, Vector vector)
    {
        return new Point(point.X + vector.X, point.Y + vector.Y);
    }

    static public Vector CreateVectorFromAngle(double angleInDegrees, double
length)
```

```
    {
        double x = Math.Sin(DegreesToRadians(180 - angleInDegrees)) * length;
        double y = Math.Cos(DegreesToRadians(180 - angleInDegrees)) * length;
        return new Vector(x, y);
    }

    static public double DegreesToRadians(double degrees)
    {
        double radians = ((degrees / 360) * 2 * Math.PI);
        return radians;
    }
}
```

Teraz môže byť implementovaná *Sprite* do novej triedy *Ship* (Loď). Do projektu je pridaná trieda *Ship* a súbor s názvom *Ship.xaml*. Dôležité je uistenie, že vlastnosti tejto triedy budú nastavené na „Embedded Resource“. Teraz bude dedené od tejto triedy *Sprite*.

C#
```
public class Ship : Sprite
{
    public Ship(double width, double height, Point firstPosition)
        : base(width, height, firstPosition)
    {

    }

    public override Canvas RenderSpriteCanvas()
    {
        return LoadSpriteCanvas("SimpleShooter.Sprites.Ship.xaml");
    }
}
```

Keď je trieda ship inštanciovaná, volá konštruktor svojho predka, *Sprite*. Taktiež implementuje *RenderSpriteCanvas* metódu a špecifikuje XAML kód (biely štvorec), ktorý sa nahrá do Sprite Canvas. Teraz môže nasledovať pridanie *Sprite* na hlavnú stránku. V tejto hre bude mať hráč iba jednu loď, ostatné lode budú mimozemské. Pridáme Vlastnosti hlavnej stránky a funkcie budú pridané, ktorá bude inštanciovať loď.

XAML
```
<Canvas x:Name="LayoutRoot" Width="30" Height="30"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" >
    <Rectangle Height="30" Width="30" Fill="White" />
</Canvas>
```

Teraz môže byť volaná metóda *InitializeGame* z konštruktora hlávnej stránky. Po spustení hry je možné si všimnúť biely štvorec, ktorý je umiestnený v ľavom dolnom rohu

a reprezentuje hráčovu loď. Hlavná stránka je definovaná ako plocha 500 x 400 pixelov a metóda *InitializeGame* umiestni hráčovu loď na bod 100 pixelov od ľavého okraja a 300 pixelov od horného okraja *gameRoot* plátna.

```
C#
void InitializeGame()
{
    PlayerShip = new Ship(10, 10, new Point(100, 300));
    gameRoot.Children.Add(PlayerShip.SpriteCanvas);
}
```

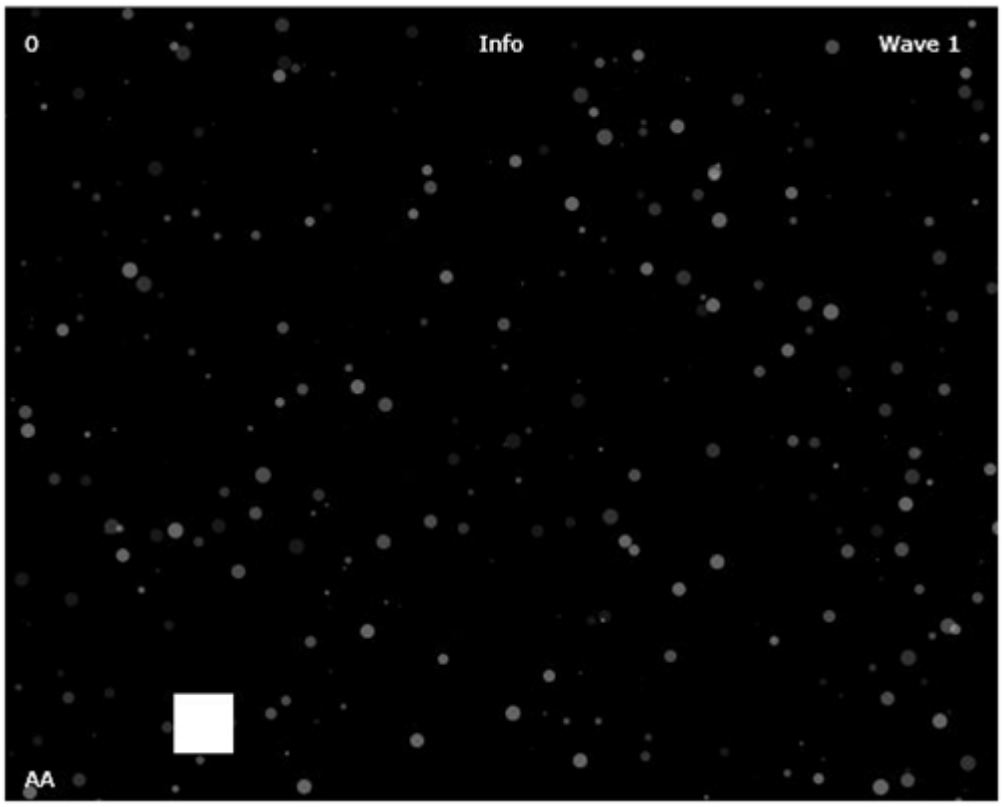
Klávensicový vstup a slučka hry

Následne je možné prvky hry uviesť do pohybu. Na začiatok sú zvolené tlačidlá na klávesnici, ktoré budú uvádzať prvky do pohybu. Následne musia byť klávesy, ktoré boli stlačené, zachytené a musí im byť priradená funkcia. Táto hra je založená na idei konštantnej slučky, pozostáva z opakovania animácií typu *Storyboard*. Trieda vyvolá udalosť a odštartuje vždy nový *Storyboard*. Pre pohyb prvkov a klávesnicový vstup musí byť na stránku pridaná inštanciu triedy *Key Handler* a *GameLoop*. Na tento krok bude metóda *InitializeGame* a konštruktor hlavnej stránky updatovaný. Taktiež bude pridaný aj manipulačný program pre *GameLoop*.

```
C#
public Page()
{
    InitializeComponent();
    keyHandler = new KeyHandler(this);
    GenerateStarField(350);
    InitializeGame();
}

void InitializeGame()
{
    gameLoop = new GameLoop(this);
    gameLoop.Update += new GameLoop.UpdateHandler(gameLoop_Update);

    PlayerShip = new Ship(10, 10, new Point(100, 360));
    gameRoot.Children.Add(PlayerShip.SpriteCanvas);

    gameLoop.Start();
}
```

C#

```
void gameLoop_Update(TimeSpan elapsed)
{
    PlayerShip.Velocity = new Vector(0, 0);
    if (keyHandler.IsKeyPressed(Key.Left))
    {
        PlayerShip.Velocity = Vector.CreateVectorFromAngle(270, 125);
    }
    if (keyHandler.IsKeyPressed(Key.Right))
    {
        PlayerShip.Velocity = Vector.CreateVectorFromAngle(90, 125);
    }
    PlayerShip.Update(elapsed);
}
```



Obrázok 9Herná slučka (snímka obrazovky)

Týmto bola vytvorená funkčná herná slučka. Problém predstavuje fakt, že pohyb s loďou je teraz možný ľubovoľne, teda aj mimo hlavného plátna. Na vyriešenie tohto nedostatku budú pridané *MinX* a *MaxX* vlastnosti do triedy *Ship* a nahradia metódu Update, ktorá sa dedí od triedy *Sprite*. Minimálne a maximálne hodnoty musia byť taktiež pridané v metóde *InitializeGame* stránky po tom, ako bude Ship inštanciovaná.

C#
```
public class Ship : Sprite
{
    public double MaxX { get; set; }
    public double MinX { get; set; }

    public Ship(double width, double height, Point firstPosition)
        : base(width, height, firstPosition)
    {

    }
    public override Canvas RenderSpriteCanvas()
    {
        return LoadSpriteCanvas("SimpleShooter.Sprites.Ship.xaml");
    }
    public override void Update(System.TimeSpan elapsedTime)
    {
        if (Position.X > MaxX)
        {
            Position = new Point(MaxX, Position.Y);
            Velocity = new Vector(0, 0);
        }
        if (Position.X < MinX)
        {
            Position = new Point(MinX, Position.Y);
            Velocity = new Vector(0, 0);
        }
        base.Update(elapsedTime);
    }
}
```

V nasledujúcom kroku budú vytvorené ďalšie entity typu *Sprite*, konkrétne mimozemšťania a projektily. Tieto vznikajú pridaním tried *Alien*, *Missle* a *Bomb* spolu s *Alien.xaml*, *Missle.xaml* a *Bomb.xaml*.

C#
```
public class Bomb : Sprite
{
    public double MaxX { get; set; }
    public double MinX { get; set; }

    public Bomb(double width, double height, Point firstPosition)
        : base(width, height, firstPosition)
    {

    }

    public override Canvas RenderSpriteCanvas()
    {
        return LoadSpriteCanvas("SimpleShooter.Sprites.Bomb.xaml");
    }

    public override void Update(System.TimeSpan elapsedTime)
    {
        base.Update(elapsedTime);
    }
}
```

V tomto bode je potrebné pridať triedu Alien, ktorá sa bude podobať na triedu Ship. Mimozemšťania a Loď budú môcť strieľať jeden na druhého. Čiže Loď bude strieľať smerom zdola nahor a Mimozemšťan smerom zhora nadol. Ako výstrahu môže mimozemšťan použiť bombu.

C#
```
public class Alien : Sprite
{
    public double fireRateMilliseconds = 2000;
    public double fireVelocity = 250;
    public double wayPointMin;
    public double wayPointMax;
    public double speed = 100;
    public bool spawnWait;
    public DateTime spawnComplete;
    public double MaxX { get; set; }
    public double MinX { get; set; }

    public Alien(double width, double height, Point firstPosition)
        : base(width, height, firstPosition)
    {

    }

    public void CheckDirection()
    {
        if (Position.X > wayPointMax)
        {
            Velocity = Vector.CreateVectorFromAngle(270, speed);
        }
        if (Position.X < wayPointMin)
        {
            Velocity = Vector.CreateVectorFromAngle(90, speed);
        }
    }

    public override Canvas RenderSpriteCanvas()
    {
        return LoadSpriteCanvas("SimpleShooter.Sprites.Alien.xaml");
    }

    public override void Update(TimeSpan elapsedTime)
    {
        CheckDirection();
        base.Update(elapsedTime);
    }

    public Bomb Fire()
    {
        Bomb bomb = new Bomb(5, 5, Position);
        bomb.Velocity = Vector.CreateVectorFromAngle(180, fireVelocity);
        return bomb;
    }
}
```

Následne bude do hry pridaná munícia. Tu je potrebné vedieť, kedy sa munícia stretne s elementom *Sprite*. Na tento účel poslúži metóda, ktorá bude detegovať zrážky. Táto metóda je implementovaná do triedy *Sprite*. Každému elementu *Sprite* bude pridaný parameter CollisionRadius, ktorý predstavuje plochu, na ktorej v prípade zasiahnutia muníciou, dôjde ku kolízii.

```
C#
public static bool Collides(Sprite s1, Sprite s2)
{
    Vector v = new Vector((s1.Position.X) - (s2.Position.X), (s1.Position.Y) -
(s2.Position.Y));
    if (s1.CollisionRadius + s2.CollisionRadius > v.Length)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Teraz môže byť herná slučka hlavnej stránky rozšírená o streľbu hlavnej lode, ktorá bude spustená tiež klávesovým vstupom (medzerníkom). Pridaná bude i enumerácia na monitorovanie stavu hry, a podporná triedu *EntityFired*, ktorá bude riadiť strieľanie munície.

```
C#
public enum GameState
{
    Ready = 0,
    Running = 1,
    Paused = 2,
    BetweenWaves = 3,
    GameOver = 4
}
```

C#

```
if (keyHandler.IsKeyPressed(Key.Space))
{
    switch (status)
    {
        case GameState.Ready:
            break;
        case GameState.Running:
            EntityFired(PlayerShip);
            break;
        case GameState.Paused:
            break;
        case GameState.BetweenWaves:
            status = GameState.Running;
            ctlInfo.GameInfo = "";
            StartWave();
            break;
        case GameState.GameOver:
            break;
        default:
            break;
    }
}

void EntityFired(Sprite shooter)
{
    Debug.WriteLine(shooter);
    switch (shooter.ToString())
    {
        case "SimpleShooter.Ship":
            if (missles.Count == 0)
            {
                Missle missle = ((Ship)shooter).Fire();
                missles.Add(missle);
                gameRoot.Children.Add(missle.SpriteCanvas);

            }
            break;
        case "SimpleShooter.Alien":
            Bomb bomb = ((Alien)shooter).Fire();
            bombs.Add(bomb);
            gameRoot.Children.Add(bomb.SpriteCanvas);
            break;
        default:
            break;
    }
}
```

Aby bolo evidovanie stavu hry jednoduchšie, budú pridané verejné vlastnosti ovládačom a bude uchovávaná hodnota o stavoch hry. Napríklad ovládaču *RemainingLives* bude priradená vlastnosť Live s možnosťou nastavenia tejto vlastnosti a tiež bude aktualizovaný *TextBlock*, ktorý bude ukazovať koľko životov hráčovi ostáva. Obdobná operácia bude vykonaná s ostatnými troma ovládačmi.

```
C#
public partial class RemainingLives : UserControl
{
    private int _lives;
    public int Lives
    {
        get { return _lives; }
        set
        {
            _lives = value;
            string livesString = string.Empty;
            for (int i = 0; i < _lives - 1; i++)
            {
                livesString = string.Format("{0}{1}", livesString, "A");
            }
            txtRemainingLives.Text = livesString;
        }
    }

    public RemainingLives()
    {
        InitializeComponent();
    }
}
```

V ďalšom kroku je možné na hlavnú stránku pridať niekoľko ďalších vymožeností. Počas každej hernej slučky musí byť kontrolované, či vystrelená raketa netrafí mimozemšťana, alebo naopak, či vystrelená bomba netrafí hráčovu loď. Hlavná stránka má už zadefinovanú loď, ale mimozemšťania, bomby a rakety musia byť definované ako kolekcie. Ak napríklad tvorca hry začne po jednom prechádzať bombami - ak bomba zasiahne loď alebo odíde z hernej mapy bude ju chcieť odstrániť. Preto bude pridaná zodpovedajúca kolekciu bombám, mimozemšťanom a raketám na sledovanie elementov, ktoré musia byť odstránené. S takýmto prístupom môže herná slučka mazať nepotrebné entity.

```
List<Alien> aliens;
List<Alien> aliensRemove;
List<Alien> alienShooters;
List<Bomb> bombs;
List<Bomb> bombsRemove;
List<Missle> missles;
List<Missle> misslesRemove;
```

Neskôr môže byť pridaná trieda, ktorá bude monitorovať nájazdy mimozemšťanov a tiež ich kolekciu. Každá vlna bude určovať počet mimozemšťanov, ktorí sa zjavia v rovnakom čase, súčet mimozemšťanov umiestnených v jednej vlne, počet mimozemšťanov, ktorí budú zhadzovať bomby a časovú frekvenciu tohto zhadzovania.

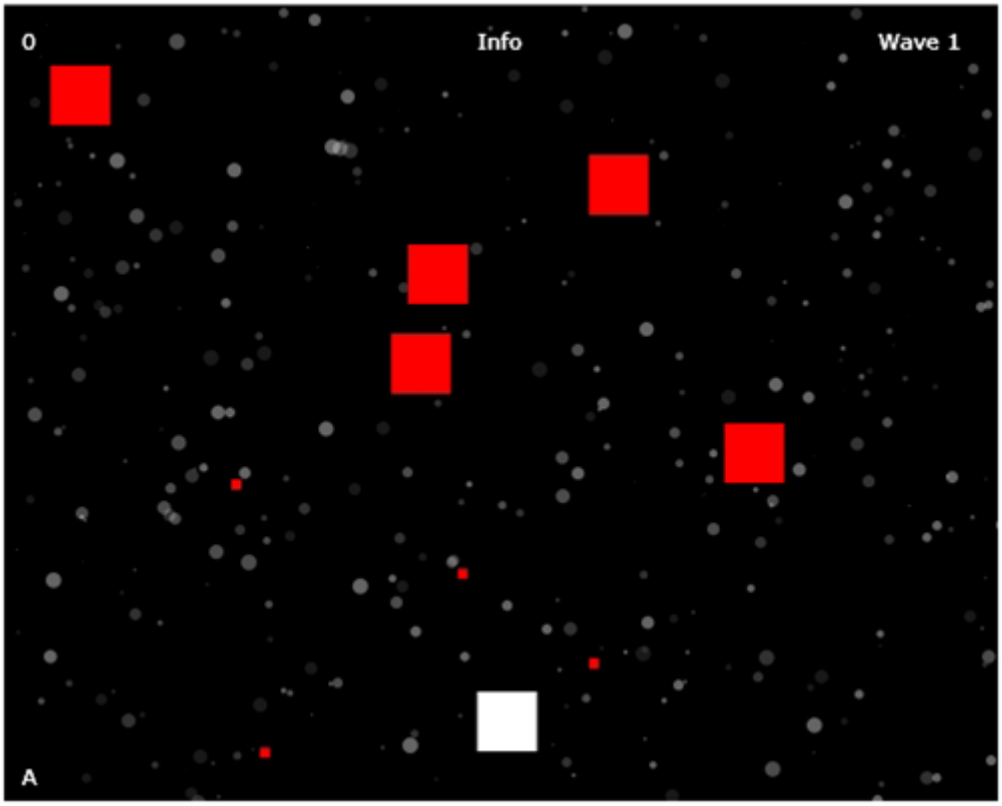
C#
```
public class WaveData
{
    public WaveData(int count, double fireRate, int atOnce, int fireatonce)
    {
        EnemyCount = count;
        fireRateMilliseconds = fireRate;
        enemiesAtOnce = atOnce;
        fireAtOnce = fireatonce;
        waveEmpty = false;
    }
    public int EnemyCount { get; set; }
    public double fireRateMilliseconds { get; set; }
    public int enemiesAtOnce { get; set; }
    public int fireAtOnce { get; set; }
    public bool waveEmpty { get; set; }
}
```

V ďalšom kroku bude vytvorená inicializačná metóda pre hru, ktorá nastaví všetky kolekcie a zvýši sa i obtiažnosť hry. Nakoniec nasleduje kontrola, či ubehlo dostatok času medzi zhodenými bombami. Na záver bude *Sprite* elementom pridaný zodpovedajúci XAML vzhľad.

C#

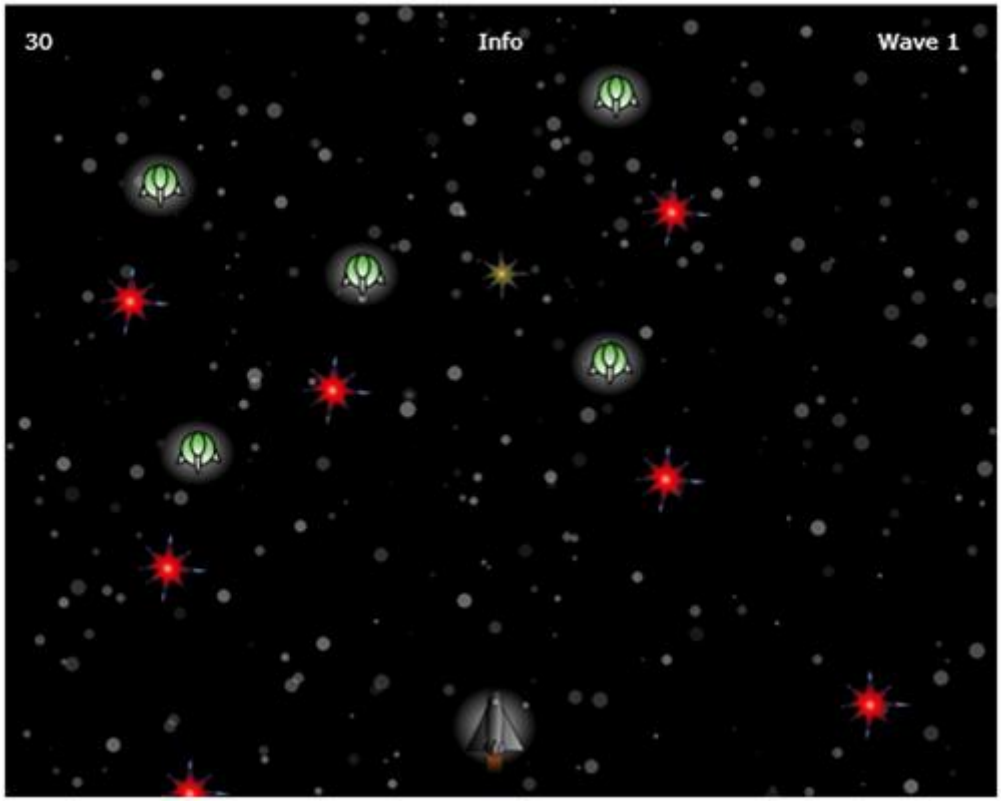
```
void gameLoop_Update(TimeSpan elapsed)
{
    PlayerShip.Velocity = new Vector(0, 0);
    if (keyHandler.IsKeyPressed(Key.Left))
    {
        PlayerShip.Velocity = Vector.CreateVectorFromAngle(270, 125);
    }
    if (keyHandler.IsKeyPressed(Key.Right))
    {
        PlayerShip.Velocity = Vector.CreateVectorFromAngle(90, 125);
    }
    if (keyHandler.IsKeyPressed(Key.Space))
    {
        switch (status)
        {
            case GameState.Ready:
                break;
            case GameState.Running:
                EntityFired(PlayerShip);
                break;
            case GameState.Paused:
                break;
            case GameState.BetweenWaves:
                status = GameState.Running;
                ctlInfo.GameInfo = "";
                StartWave();
                break;
            case GameState.GameOver:
                break;
            default:
                break;
        }
    }
    PlayerShip.Update(elapsed);

    BombLoop(elapsed);
    MissleLoop(elapsed);
    AlienLoop(elapsed);

    foreach (Alien alien in aliensRemove)
    {
        aliens.Remove(alien);
        gameRoot.Children.Remove(alien.SpriteCanvas);
        AlienShot(alien);
    }
    aliensRemove.Clear();
```

```
    foreach (Missle missle in misslesRemove)
    {
        missles.Remove(missle);
        gameRoot.Children.Remove(missle.SpriteCanvas);
    }
    misslesRemove.Clear();

    if (nextShot <= DateTime.Now)
    {
        nextShot =
DateTime.Now.AddMilliseconds(enemyShootMilliseonds).AddMilliseconds(elapsed.Mil
liseconds * -1);

        shotsThisPass = shotsAtOnce;
        if (shotsThisPass > aliens.Count)
        {
            shotsThisPass = aliens.Count;
        }

        if (aliens.Count > 0)
        {
            foreach (Alien alien in aliens)
            {
                alienShooters.Add(alien);
            }
        }

        while (alienShooters.Count > shotsThisPass)
        {
            alienShooters.RemoveAt(GetRandInt(0, alienShooters.Count - 1));
        }

        foreach (Alien alien in alienShooters)
        {
            EntityFired(alien);
        }

        alienShooters.Clear();
    }
}
```



Obrázok 10 Hotová hra bez textúr (snímka obrazovky)

V súčasnej podobe máme k dispozícii hru, ktorej však ešte chýba animácia *sprite* prvkov. Ako nástroj na dotvorenie vizuálnej stránky *sprite* prvkov nam poslúži Expression Blend 4 kde môžme za pomoci elementov na vytváranie užívateľského rozhrania vyrobiť podoby mimozemšťanov, lode, bômb a rakiet. Výsledok je možné pozorovať na obrázku 11.



Obrázok 11 Hotová hra s textúrami (screenshot)

3 Záver

Cieľom tejto diplomovej práce bolo navrhnúť a zostrojiť interaktívnu aplikáciu v platforme Microsoft Silvelright. Na splnenie tohto cieľa bolo potrebné sa zaoberať dvoma čiastkovými cieľmi, ktoré túto prácu rozdelili na dve časti.

Prvým čiastkovým cieľom bolo oboznámenie čitateľa s technologickými aspektami tejto technológie, princíp fungovania aplikácií, popísať typy Silverlight súborov a vývojárske nástroje tejto platformy. V prvej kapitole sa autor venoval týmto aspektom a zostavil tak teoretické základy pre druhú časť tejto diplomovej práce.

Druhým cieľom bolo navrhnutie a vytvorenie interaktívnej aplikácie, v tomto prípade hry, ktorá demonštruje silu a všestrannosť tejto platformy. Praktická časť diplomovej práce ukazuje na príklade hry a dokumentácie procesu vzniku niektoré vlastnosti a výhody platformy Silverlight.

Napriek tomu, že platforma Silverlight je oproti technológii Flash málo rozšírená, ponúka veľa možností, ovládacích prvkov a funkcií, ktoré môžu byť použité na vytváranie či už webových alebo desktopových interaktívnych aplikácií.

Zoznam použitej literatúry

[1] LACKO, L. 2010. *Silverlight - Výukový průvodce tvorbou interaktivních aplikací*, Brno : Computer press, 2010. 453 s. ISBN 978-80-2516-2

[2] PAPA, J. 2009. *Silverlight - datové služby*, Brno:Zoner Press, 2009. 384 s. ISBN: 978-80-7413-041-0

[3] ŠPINAR, D. 2004. *Tvoříme přístupné webové stránky*. Brno: Zoner Press, 2004. 360 s. ISBN 80-86815-11-0

[4] GHODA, A. 2010. *Introducing Silverlight 4,* New York: Apress, 2010. 720 s. ISBN 978-1-4302-2992-6

[5] PARIES, J. 2009. *Silverlight 3*, New York: Apress, 2009. 458 s. ISBN 978-1-4302-2408-2

[6] MICHAIL, A. 2010. *Essential Silverlight 3,* Boston: Pearson Education, 2010 .321 s. ISBN 978-0-321-55416-1

[7] NARAMORE, E. a kol. 2006. *Vytváříme webove aplikace*. Brno: Computer Press, 2006. 813 s. ISBN 80-251-1073-7

[8] PROSISE, J. 2003. *Progamování v Microsoft .NET*. Brno: Computer Press, 2003. 712 s. ISBN 80-7226-879-1

[9] CZERNICKI, B. *Silverlight Hack* [online]. [cit. 5.2.2014]. Dostupné z internetu: <http://silverlighthack.com/post/2008/07/25/Silverlight-20-Concepts-ToBecome-A-Silverlight-Master-%28Series-Part-3-Blend%29.aspx>.

[10] Microsoft msdn. *Vývoj pre telefóny* [online]. [cit 5.3.2014] dostupné na internete:<http://msdn.microsoft.com/sk-sk/ff380145.aspx>

[11] Microsoft msdn. *Walkthrough:Using authentication Service with Silverlight Business Application* [online].[cit 15.3.2014]. dostupné na internete: <http://msdn.microsoft.com/en-us/library/ee942449(v=vs.91).aspx>

[12] Russev R. *Bezpečnosť v aplikáciach Microsoft Silverlight* 2.0. [online]. [cit 10.3.2014] dostupné na internete: <http://www.itnews.sk/tituly/infoware/2009-01-07/c82114-bezpecnost-v-aplikaciach-microsoft-silverlight-2.0>

[11] PUŠ, Petr. Živě.cz. *Poznáváme C# a Microsoft .NET – 1.díl.* [online]. [cit. 18.4.2014]. dostupné na internete: <http://www.zive.cz/clanky/poznavame-c-a-microsoft-net--1dil/sc-3-a-120978/default.aspx>.

[15] Microsoft Silverlight. *Get Started with Silverlight 4 with One Install.* [online]. 2011 [cit. 4.42014] Dostupné na internete: <http://www.silverlight.net/getstarted/>.

[14] Microsoft msdn. *Deep zoom* [online].[cit 18.3.2014]. dostupné na internete: <http://msdn.microsoft.com/en-us/library/cc645050(vs.95).aspx>

[16] Microsoft msdn. *Protoyping with SketchFlow* [online].[cit 5.4.2014]. dostupné na internete: <http://msdn.microsoft.com/en-us/library/ee341458(v=expression.40).aspx>

[17] Microsoft msdn. *Silverlight architecture*, [online].[cit 10.4.2014]. dostupné na internete: <http://msdn.microsoft.com/en-us/library/bb404713(v=vs.95).aspx>

[18] Smashingmagazine.com. *Flash vs. Silverlight* [Online]. [cit. 8.4.2014]. Dostupné na internete: <http://www.smashingmagazine.com/tag/flash/ >

[19] Adobe. [Online] [cit. 25.3.2014]. Dostupné na: < http://www.adobe.com/sk >

[20] Silverlight 3.0. [Online]. [cit. 18.3.2014]. Dostupné na internete: <http://blog.aspnet.sk/knihy/archive/2009/11/05/silverlight-3-0-objekty-prevytvorenie prezentacnej-vrstvy.aspx>

[21] VOŘÍŠEK, L. *Microsoft představil Silverlight 5! Má co nabídnout?.* [online] [cit. 8.3.2014]. Dostupné na internete: <http://pctuning.tyden.cz/index.php?option=com_content&view=article&id=19478&catid=1&Itemid=57>.

Prílohy

Ako prílohy sú poskytnuté projekty

Databinding – demonštruje prepájanie väzieb medzi XAML prvkami

Jednoducho sa spúšťa sa prostredníctvom *DataBindingTestPage.html* súboru ktorý sa nachádza v priečinku \DataBinding\ DataBinding \Bin\Debug

polohovaniePrvkov– demonštruje polohovanie prvkov XAML

Jednoducho sa spúšťa prostredníctvom *polohovaniePrvkovTestPage.html* súboru ktorý sa nachádza v priečinku \polohovaniePrvkov\polohovaniePrvkov\Bin\Debug.

Perspective3D– demonštruje prepájanie 3D polohovanie prvkov XAML

Jednoducho sa spúšťa sa prostredníctvom *Perspective3D.html* súboru ktorý sa nachádza v priečinku \Perspective3D\Perspective3D\Bin\Debug.

Simple Shooter – hra vytvorená v platforme Silverlight

Spúšťa sa prostredníctvom *TestPage.html* súboru ktorý sa nachádza v priečinku \SimpleShooter\SimpleShooter\Bin\Debug

Pre spustenie ukážok je nutné nainštalovať Silverlight plugin z oficialnej stránky <http://www.microsoft.com/getsilverlight/get-started/install/default.aspx#instruct>