Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Multimediální hra v prostředí Windows Phone

Plzeň, 2012 Martin Hora

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 2. srpna 2012

Martin Hora

Abstract

The purpose of this work *Multimedia game in Windows Phone environment* is to explore possibilities of developing applications for Windows Phone 7 mobile platform. The work is mainly focused on creating multimedia and network applications. The aim of the work is to develop a network game for both mobile and desktop platform featuring multimedia elements. The network client will be designed in the way to be reused in other projects.

Obsah

1 Úvod

Cílem práce je prozkoumat a popsat vývoj aplikací v prostředí Windows Phone 7 se zaměřením na multimediální a síťové možnosti platformy.

V praktické části práce je realizována multimediální síťová hra pro mobilní i desktopové prostředí. Při vytváření aplikace jsem se soustředil zejména na znovupoužitelnost síťového rozhraní, které bude sloužit jako základ pro síťovou komunikaci v dalších projektech. S pomocí tohoto rozhraní bude možné vytvářet další síťové hry v reálném čase nebo hry tahové. Popsané síťové rozhraní bylo vytvořeno pro mobilní i desktopové prostředí. Jako vlastní hra byla zvolena hra ping pong, tedy hra v reálném čase.

Mobilní klient demonstruje mimo použití síťové komunikace také práci s multimediálními prvky, práci s dotykovými gesty, která slouží pro ovládání hry, a použití lokálního úložiště aplikace, které zde slouží pro ukládání herních voleb. Mobilní klient dále zahrnuje práci s herním menu, tedy s položkami, které lze vybírat dotykem.

Práce je členěna na na teoretickou a praktickou část. Teoretická část zahrnuje kapitoly *Platforma Windows Phone*, *Základní prvky Windows Phone aplikací* a *XNA framework*. Hlavní kapitolou je praktická část, která představuje programátorskou dokumentaci.

V kapitole *Platforma Windows Phone* je stručně popsán vývoj samotného systému a nástroje, které slouží pro vývoj aplikací. Je zde zmíněno, pomocí jakých frameworků lze aplikace modelovat.

Kapitola *Základní prvky Windows Phone aplikací* se zabývá standardními nástroji a postupy, které mohou být při vývoji aplikace použity. Kapitola je doplněna o ukázky použití jednotlivých nástrojů ve formě zdrojových kódů.

Další kapitola *XNA framework* se již konkrétněji věnuje XNA frameworku, tedy frameworku určený pro vývoj her. Tento framework byl také použit při realizaci vlastní hry. Kapitola popisuje strukturu základního projektu v tomto frameworku a její jednotlivé součásti.

Praktická část práce popisuje realizaci aplikace pro mobilní i desktopové prostředí. Rozebírá, jaké metodiky jsem při vytváření aplikace volil. Součástí této kapitoly je i popis obecného síťového rozhraní.

2 Platforma Windows Phone

2.1 Historie systému Windows Phone 7

Systém Windows Phone 7 je vyvinutý firmou Microsoft a byl představen v druhé polovině roku 2010. [1] Navazuje na mobilní operační systém Windows Mobile, jehož poslední verze byla 6.5. S tímto systémem však není dále kompatibilní a jedná se tedy o zcela novou platformu. Cílovou skupinu představují běžní uživatelé. Systém tedy není zaměřený, jako jeho předchůdce na business aplikace. [2]

Původní záměr bylo vytvořit novou velkou aktualizaci pro Windows Mobile s kódovým označením "Photon". Práce na aktualizaci začala v roce 2005. Vývoj aktualizace byl však pomalý a nakonec byl vývoj aktualizace ukončen. [3] V roce 2008 započaly práce na zcela novém operačním systému [4], který měl být vydán v roce 2009 jako Windows Phone, nakonec byl ale vydán až v polovině roku 2010. [5]

Windows Phone 7 se liší od Windows Mobile především kompletně přebudovaným uživatelským prostředím Metro, které je založené na dlaždicích a "hubech". Huby představují aplikace, které zahrnují několik společně souvisejících služeb systému do jedné aplikace. Jednoduše řečeno, díky nim je možné najít v systému vše na jednom místě. V systému jsou také integrovány služby Microsoftu jako Microsoft Office, XBox Live a Windows Hotmail. Aplikace pro Windows Phone jsou oficiálně dostupné přes *Marketplace*.

Další verze systému nese označení Windows Phone 8 a vydání systému se předpokládá na podzim 2012. [9] Mezi nejvýraznější změny, které nová verze přinese, patří podpora vícejádrových procesorů, paměťových karet, technologie NFC, nové varianty rozlišení displeje a předělaná úvodní stránka s přehledem aplikací. [10] Žádné ze stávajících zařízeních však pravděpodobně nebude na tuto verzi aktualizováno. Pro stávající zařízení je připravena verze 7.8, která bude zahrnovat přepracovanou plochu s dlaždicemi. [11] V tabulce 2.1 je uveden přehled významných aktualizací systému.

Číslo verze	Označení	Vlastnosti verze
7	*NoDo*	Mezi hlavní vylepšení aktualizace patří podpora funkce pro kopírování textu, vkládání textu a rychlejší práce s aplikacemi a hrami. [6]
7.1	*Mango*, *Windows Phone 7.5*	Jedná se o první větší aktualizaci Windows Phone, která přinesla zhruba 500 vylepšení. Mezi výrazné změny v této verzi patří multitasking, podpora více jazyků (včetně češtiny), podpora HTML5, nový internetový prohlížeč Internet Explorer 9 nebo úsporný režim baterie. Oficiální označení této verze je 7.1, nicméně je označována také jako *Windows Phone 7.5*. [7]
7.5	*Refresh*	Aktualizace snížila hardwarové nároky na zařízení a je možné ji provozovat na zařízení jen s 256 MB RAM. Dále umožňuje do MMS zpráv přidat několik multimediálních příloh. Také známa pod neoficiálním názvem *Tango*. [8]

Tabulka 2.1: Významnější aktualizace systému

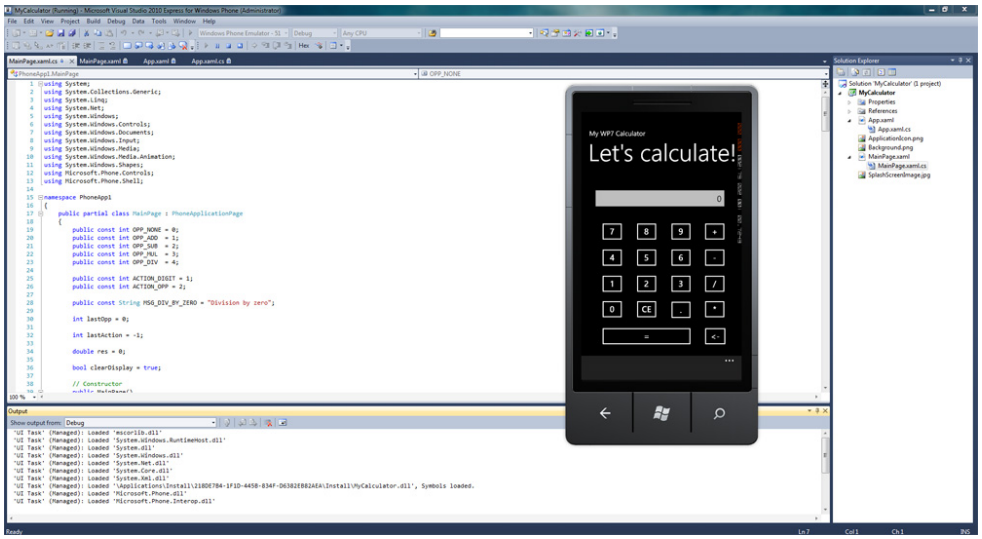
2.2 Nástroje pro vývoj aplikací Windows Phone 7

Pro vytváření a ladění aplikací pro Windows Phone 7 se využívají nástroje sady *Windows Phone SDK*. Tato sada zahrnuje vývojové prostředí *Visual Studio 2010 Express for Windows Phone*, viz. obr. 2.1. Dále emulátor pro testování aplikací *Windows Phone Emulator* a nástroj pro měření výkonnostních charakteristik aplikace *Windows Phone Performance Analysis Tool*, který je součástí vývojového prostředí.

Tyto nástroje jsou k dispozici zdarma. Stažené mohou být zde – *http://www.microsoft.com/en-us/download/details.aspx?id=27570*.

2.2.1 Visual Studio 2010 for Windows Phone

Aplikace se vytvářejí v jazyce C# nebo Visual Basic. Při vytváření nového projektu si lze také vybrat z několika předpřipravených projektových šablon pro oba typy jazyků. Takové šablony představují například standardní aplikaci Windows Phone, hru pro Windows Phone, aplikaci spojující XNA a Silverlight frameworky nebo aplikaci běžící na pozadí a periodicky vykonávající určitou úlohu.



Obrázek 2.1: Ukázka vývojového prostředí se spuštěným emulátorem

2.2.2 Návrh standardní Windows Phone aplikace - XAML

Design UI ve Windows Phone Application se vytváří pomocí jazyka XAML, který je založený na XML, a který je všeobecně určený pro popis grafického rozhraní v aplikacích firmy Microsoft. Kód XAML může být psán ručně, kdy v kódu definujeme jednotlivé elementy a pomocí souřadnic je umisťujeme do UI. Druhou možností je vytvářet design UI prostřednictvím grafického prostředí, kdy se jednotlivé prvky umisťují na plochu. Grafické prostředí pro vytváření UI je součástí *Microsoft Visual Studia 2010 Express for Windows Phone*.

Příklad definice návrhu aplikace pomocí XAML (jen část kódu definující titulek, nadpis):

```
<StackPanel x:Name="TitlePanel" Grid.Row="0"
   Margin="12,17,0,28">
    <TextBlock x:Name="ApplicationTitle" Text="My WP7
       Calculator" Style="{StaticResource
       PhoneTextNormalStyle}"/>
    <TextBlock x:Name="PageTitle" Text="Let's calculate!"
       Margin="9,-7,0,0" Style="{StaticResource
       PhoneTextTitle1Style}"/>
</StackPanel>
```

2.2.3 Windows Phone Emulator - testování aplikací

Součástí *Visual Studio 2010 for Windows Phone* je také emulátor, který se používá pro testování a ladění aplikací. Emulátor simuluje prostředí Windows Phone. Pro vývoj není potřeba vlastnit fyzické zařízení s Windows Phone.

Na pravé straně emulátoru se nachází panel s možnostmi.

Možnosti emulátoru:

- ukončení nebo minimalizace emulátoru,
- otáčení emulátoru,
- roztažení velikosti emulátoru,
- procentuální změna velikosti emulátoru,
- další možnosti.

Pokročilé možnosti emulátoru:

- rozhraní pro testování akcelerátoru,
- rozhraní pro testování lokace,
- možnost sejmout a uložit aktuální snímek displeje v emulátoru.

2.2.4 Frameworky pro vývoj aplikací

Pro vývoj aplikací nabízí platforma Windows Phone 7 dva frameworky:

- **Silverlight** - používá se pro klasické XAML aplikace, v případě, že je třeba využít prvky založené na HTML prohlížeči, pro přehrávání videí, pro využití základních ovládacích prvků systému.
- **XNA** - používá se spíše pro vývoj herních aplikací s podporou 2D a 3D grafiky, různých grafických efektů, textur a animací.

Samozřejmě je možné tyto dva frameworky v případě potřeby kombinovat. Takovým příkladem může být hra, která bude napsána v XNA a menu ke hře napsané v Silverlightu.

3 Základní prvky Windows Phone aplikací

Kapitola se věnuje několika základním prvkům, které aplikace pro Windows Phone mohou využívat. Při realizaci bylo čerpáno z *Microsoft Developer Network* [12], což se vztahuje i na zdrojové kódy, kterými je text doprovázen.

3.1 Stránky a navigace v aplikaci

Aplikace pro Windows Phone mohou být rozčleněny do stránek. Jedna stránka zahrnuje obsah, který uživatel telefonu vidí právě na displeji. Stránky se chovají jako běžné webové stránky, tzn. můžeme je mezi sebou prolinkovat.

3.1.1 Vytvoření nové stránky

Nová stránka se vytvoří ve vývojovém prostředí kliknutím pravého tlačítka na název projektu, zvolením *Add -> New Item* a výběrem buď *Portrait Page* pro stránku umístěnou na výšku nebo *Landscape Page* pro stránku umístěnou na šířku.

3.1.2 Přepínání mezi stránkami

Na stránky se můžeme v aplikaci odkazovat pomocí odkazů specifikovaných názvem stránky.

Mějme například vytvořenou stránku *MainPage.xaml*, pokud se z této stránky chceme odkázat na stránku *SubMenuPage.xaml*, vytvoříme na stránce *MainPage.xaml* tlačítko *ToSubMenu* s následující událostí:

```
private void Button ToSubMenu_Click(object sender,
   RoutedEventArgs e)
{
    NavigationService.Navigate(new
       Uri("/SubMenuPage.xaml", UriKind.Relative));
}
```

3.2 Spouštění integrovaných aplikací systému

Pokud ve Windows Phone aplikaci potřebujeme spustit integrovanou aplikaci systému, jako např. webový prohlížeč, telefonní seznam, e-mailového klienta, použijeme nástroje nazvané Launchers nebo Choosers z jmenného prostoru *Microsoft.Phone.Tasks*.

Launchers i Choosers slouží pro spuštění Windows Phone aplikace s tím rozdílem, že Chooser navíc vrací výsledek akce, který jsme v dané aplikaci provedli. Například vrátí vybraný kontakt z telefonního seznamu.

3.2.1 Stručný popis použití Launcheru

Na tomto příkladě je popsáno v několika bodech vytvoření a použití Launcheru, který bude odesílat na telefonní číslo SMS zprávu s daným textem. Odeslání bude probíhat tak, že se spustí akce nové zprávy, kde bude předvyplněné číslo příjemce a text zprávy. Zpráva se odešle potvrzením.

1. Vytvoření instance Launcheru

```
SmsComposeTask smsComposeTask = new SmsComposeTask();
```

2. Nastavení parametrů instanci Launcheru

```
smsComposeTask.To = "123456789"; // tel. cislo
smsComposeTask.Body = "Zkousim SMS task.";
```

3. Spuštění metody ***Show*** instance Launcheru

```
smsComposeTask.Show();
```

3.2.2 Stručný popis použití Chooseru

Uvedený příklad demonstruje případ, kdy se spustí telefonní seznam, uživatel zvolí kontakt a bude s ním dále pracovat. Kontakt se zde pouze zobrazí.

1. Vytvoří se instance Chooseru - bude vytvořena globálně

```
PhoneNumberChooserTask phoneNumberChooserTask;
```

2. Specifikace callback metody Chooseru, která se zavolá po jeho skončení. V tomto případě ***cTask_Completed***.

```
phoneNumberChooserTask = new PhoneNumberChooserTask();
phoneNumberChooserTask.Completed += new
   EventHandler<PhoneNumberResult>(cTask_Completed);
```

3. Vytvoříme callback metodu ***cTask_Completed***, která bude zobrazovat výsledek po ukončení Chooseru. V tomto případě je výsledkem objekt **PhoneNumberResult** obsahující data o zvoleném kontaktu.

```
void cTask_Completed(object sender, PhoneNumberResult
   e)
{
    if (e.TaskResult == TaskResult.OK)
    {
        MessageBox.Show("Zvolili jste kontakt " +
           e.DisplayName + " s telefonnim cislem " +
           e.PhoneNumber);
    }
}
```

4. Zavoláme metodu ***Show*** instance Chooseru, zachytáváme případnou výjimku

```
try
{
    phoneNumberChooserTask.Show();
}
catch (System.InvalidOperationException ex)
{
    MessageBox.Show("Vyskytla se chyba.");
}
```

3.3 Síťování

3.3.1 Zjišťování stavu a dostupnosti připojení

Ke zjištění, zda je v přístroji povolena Wi-Fi nebo datové připojení přes operátora slouží třídy v namespace *Microsoft.Phone.Net.NetworkInformation*. Konkrétně k tomuto účelu slouží třída **DeviceNetworkInformation**.

Složky třídy DeviceNetworkInformation

- *CellularMobileOperator* - jméno mobilního operátora.
- *IsCellularDataEnabled* - určuje, zda je povolena síť operátora.
- *IsCellularDataRoamingEnabled* - určuje, zda je povolen roaming.
- *IsNetworkAvailable* - určuje, zda je v telefonu povoleno nějaké datové připojení, např. GPRS, 3G, atd.
- *IsWiFiEnabled* - určuje, zda je povolena Wi-Fi.

Příklad použití třídy **DeviceNetworkInformation** *zobrazuje po kliknutí na tlačítko button1 informace. Zobrazuje, zda je aktivní připojení přes operátora nebo je zapnutý roaming a Wi-Fi:*

```
private void button1_Click(object sender, RoutedEventArgs
   e)
{
    System.Text.StringBuilder sb = new
       System.Text.StringBuilder();

    sb.Append("Povolena sit operatora?:  ");
    sb.AppendLine(DeviceNetworkInformation.
       IsCellularDataEnabled.ToString());

    sb.Append("Povolen roaming?:  ");

    sb.AppendLine(DeviceNetworkInformation.
       IsCellularDataRoamingEnabled.ToString());

    sb.Append("Povolena Wi-Fi?: ");
    sb.AppendLine(DeviceNetworkInformation.
       IsWiFiEnabled.ToString());

    MessageBox.Show(sb.ToString());
}
```

3.3.2 Komunikace mezi zařízeními

Windows Phone podporuje od verze 7.1 (*Mango*) vytváření socketů, takže lze vytvořit aplikaci na bázi klient - server, kde si mohou klienti se serverem vyměňovat zprávy. Sockety mohou používat buď spojovou službu TCP nebo nespojovou UDP. K vytváření socketů slouží jmenný prostor *System.Net.Sockets*.

Další variantou síťové komunikace mezi zařízeními je použití multicastu, který systém také podporuje.

3.3.3 Webové služby a HTTP

Jde o třídy, jejichž prostřednictvím lze přistupovat k webovým službám a vytvářet webové požadavky:

- **HttpWebRequest** - třída reprezentující HTTP požadavek, umožňující odesílat hlavičky (headers). Je odvozena od abstraktní třídy požadavku **WebRequest**.
- **WebClient** - obsahuje metody pro zasílání a přijímání dat z různých URI.

Příklad využití třídy WebClient ke stažení obsahu ze zadaného URI

V tomto příkladě se po kliknutí na tlačítko button1 pomocí handleru ***webClient_Download*** *zobrazí obsah webové stránky http://msdn.microsoft.com.*

```
private void webClient_Download(object sender,
    DownloadStringCompletedEventArgs e)
{
    if (e.Error != null)  {
        Deployment.Current.Dispatcher.BeginInvoke(() =>
        {
            MessageBox.Show(e.Error.Message);
        });
    }
    else
    {
        MessageBox.Show(e.Result);
    }
}

private void button1_Click(object sender, RoutedEventArgs
   e)
{
    WebClient webClient = new WebClient();
    webClient.DownloadStringCompleted += new
        DownloadStringCompletedEventHandler(
           webClient_Download);
    webClient.DownloadStringAsync(new
       System.Uri("http://msdn.microsoft.com"));
}
```

3.4 Způsob ukládání dat na zařízení

V systému Windows Phone nemají aplikace povolený přímý přístup k souborovému systému, ale pro každou aplikaci je vytvořeno speciální úložiště *Isolated Storage*. K tomuto úložišti má přístup jen daná aplikace. Takové omezení je zřízeno kvůli bezpečnosti.

Aplikace Windows Phone mohou přistupovat k lokálnímu úložišti různými způsoby:

- Ke složkám a souborům se přistupuje pomocí třídy **IsolatedStorageFile** (*System.IO.IsolatedStorage*).
- K nastavením, která jsou reprezentovány formou klíč - hodnota se přistupuje přes třídu **IsolatedStorageSettings** (*System.IO.IsolatedStorage*).
- K lokální databázi systému se přistupuje pomocí LINQ to SQL (*System.Data.Linq*) rozhraní.

3.4.1 Složky v Isolated Storage

Do následujících složek jsou ukládána aplikační data:

- **Shared/Media** - sem ukládají aplikace obrázky alba, které se zobrazují při přehrávání zvuku na pozadí.
- **Shared/ShellContent** - zde jsou uloženy dlaždice aplikací.
- **Shared/Transfers** - aplikace může do složky ukládat data, i když aplikace neběží na popředí.

Ukázka použití Isolated Storage

Na tomto příkladu je ukázáno, jak vytvořit v Isolated Storage novou složku, zapsat do ni textový soubor a obsah tohoto textového souboru následně přečíst a zobrazit.

```
IsolatedStorageFile myStore =
   IsolatedStorageFile.GetUserStoreForApplication();
myStore.CreateDirectory("NovaSlozka");
// Zapis do souboru
using (var isoFileStream = new IsolatedStorageFileStream(
        "NovaSlozka\\soubor.txt", FileMode.OpenOrCreate,
           myStore))
{
    using (var isoFileWriter = new
       StreamWriter(isoFileStream))
    {
        isoFileWriter.WriteLine("Ukazka pouziti Isolated
           Storage!");
    }
}
// Cteni ze souboru
try {
    using (var isoFileStream = new
       IsolatedStorageFileStream(
            "NovaSlozka\\soubor.txt", FileMode.Open,
               myStore))
    {
        using (var isoFileReader = new
           StreamReader(isoFileStream))
        {
            MessageBox.Show(isoFileReader.ReadLine());
        }
    }
}
catch {
    MessageBox.Show("Soubor nenalezen.");
}
```

3.5 Senzory

Systém podporuje několik senzorů, se kterými je možné komunikovat a získávat z nich data, se kterými lze dále pracovat. U některých senzorů je nutné zjistit, jestli je daný model zařízení podporuje.

3.5.1 Typy senzorů

- **Akcelerometr** (třída **Accelerometer**)
  Akcelerometr měří síly působící na zařízení, které se poté využijí k výpočtu směru pohybujícího se zařízení.

  Pro testování akcelerometru se používá nástroj *Accelerometer Sensor Simulator*, který se otevře spuštěním aplikace v emulátoru a zvolením menu *Další nástroje*.

- **Kompas** (třída **Compass**)
  Kompas je používán k určení úhlu, pod kterým je zařízení otáčeno vzhledem k severnímu magnetickému pólu. Kompas není přítomen ve všech Windows Phone zařízeních, a proto je nutné při vývoji aplikace kontrolovat, zda je senzor k dispozici.

- **Gyroskop** (třída **Gyroscope**)
  Gyroskop se používá k určení rychlosti rotace zařízení ve všech směrech. Hodnoty, které poskytuje gyroskop se používají k určení směru zařízení v prostoru. Stejně jako v případě kompasu je důležité zjišťovat, zda je senzor v zařízení přítomen.

- **Zpřesnění výpočtu informace o pohybu a poloze** (třída **Motion**)
  Třída Motion provádí výpočty a přepočty získaných dat z již zmíněných senzorů a poskytuje informaci o aktuálním poloze zařízení. Třída používá dvě konfigurace senzorů. První konfigurace používá senzory akcelerometr a kompas. Druhá přidává ještě gyroskop. Opět je nutné kontrolovat, jestli jsou dané senzory v zařízení k dispozici.

3.6 Multitasking

Při spuštění více úloh může na popředí běžet jen jedna úloha, avšak ostatní úlohy mohou vykonávat další akce, i když jsou v pozadí.

Před verzí Windows Phone 7.1 byla úloha, která byla opuštěna automaticky ukončena. Systém se však změnil s příchodem verze 7.1. V této verzi se po přepnutí úlohy, úloha uspí a její aktuální stav se zapíše do paměti, takže úlohu lze opět okamžitě vyvolat zpět.

Akce, které mohou běžet na pozadí:

- **Přenos souborů** - může se jednat o download i upload přes HTTP.
- **Hudba na pozadí**.
- **Naplánované oznámení** - na bázi alarmů.
- **Naplánované úlohy**.

3.7 Přístup ke kontaktům a kalendáři

Windows Phone poskytuje prostředky pro přístup k uživatelským kontaktům a kalendáři. K tomu slouží jmenný prostor *Microsoft.Phone.UserData*. Příklady tříd pro práci s uživatelskými daty:

- **Account** - obsahuje údaje o uživatelských účtech jako např. Microsoft Outlook.
- **Appointment** - třída obsahující informace o záznamu v kalendáři.
- **Appointments** - třída poskytující metody k manipulaci se záznamy v kalendáři.
- **Contact (-Address, CompanyInformation, EmailAddress, PhoneNumber)** - analogicky jako **Appointment** obsahuje informace o kontaktu v telefonním seznamu.
- **Contacts** - třída pro práci s kontakty.

3.8 Práce s parametry zařízení

Pokud je potřeba zjistit hardwarové parametry přístroje, použije se třída **DeviceStatus**.

Složky třídy DeviceStatus:

- *ApplicationCurrentMemoryUsage* - informace, kolik paměti zabírají spuštěné aplikace v bytech.
- *ApplicationMemoryUsageLimit* - kolik paměti může aplikace alokovat v bytech.
- *ApplicationPeakMemoryUsage* - paměťové maximum aplikace.
- *DeviceFirmwareVersion* - verze firmwaru.
- *DeviceHardwareVersion* - verze hardwaru.
- *DeviceManufacturer* - výrobce zařízení.
- *DeviceName* - název zařízení.
- *DeviceTotalMemory* - velikost RAM paměti zařízení v bytech.
- *IsKeyboardDeployed* - zda je vysunuta klávesnice.
- *IsKeyboardPresent* - zda zařízení obsahuje hardwarovou klávesnici.
- *PowerSource* - zda je zařízení připojeno k nabíječce.

3.9 Lokace zařízení

Windows Phone mohou pomocí GPS, Wi-Fi nebo podle BTS poskytnout údaje o umístění zařízení. Volba metody, kterou chceme pro zjištění umístění použít závisí na tom, jak přesně chceme určit naši polohu a na využití baterie. Pro účely zjišťování polohy slouží jmenný prostor *System.Device.Location*.

3.10 Používání médií

Pro práci s médii slouží následující prostředky:

- Třída **MediaPlayerLauncher**, která umožňuje přehrávat zvuk nebo video. Používá se v XNA aplikacích.
- Třída **MediaElement** API pro přehrávání zvuku a videí, ovšem nabízí lepší možnosti pro úpravu uživatelského rozhraní. Používá se v Silverlight aplikacích.
- **XNA Game Studio** - umožňuje přidávání zvukových efektů do aplikací.
- **MediaStreamSource** - pro účely streamování videa.
- Namespace *Microsoft.Phone.BackgroundAudio* umožňující přehrávání zvuku (hudby), když není daná aplikace v popředí.

3.11 Vytváření rozbalovacího menu

Ve Windows Phone aplikaci lze vytvořit rozbalovací menu, zobrazené ve spodní části displeje.

Existují 3 velikosti této lišty:

- Mini velikost, kde jsou vidět jen 3 tečky umístěné vpravo.
- Standardní velikost, kde jsou vidět i ikony akcí.
- Rozšířená velikost i s popisem akcí pod ikonami.

3.11.1 Vytvoření menu

Menu lze vytvořit buď přes XAML nebo přímo v kódu aplikace použitím třídy **ApplicationBar**, nacházející se v namespace *Microsoft.Phone.Shell*. Pokud bychom chtěli dynamicky měnit popisky za chodu aplikace, vytvoříme menu přímo v kódu.

Složky třídy ApplicationBar:

- *Mode* - specifikuje jaká výchozí velikost menu bude použita.
- *Opacity* - průhlednost menu.
- *BackgroundColor* - barva pozadí menu.
- *ForegroundColor* - barva ikonek a popisků v menu.
- *IsMenuEnabled* - určuje, zda lze menu používat.
- *IsVisible* - určuje, zda je menu viditelné.

4 XNA framework

4.1 Vytvoření projektu aplikace XNA aplikace

Pro vytvoření aplikace pro Windows Phone, která bude používat XNA framework, zvolíme ve vývojovém prostředí *File -> New Project*. V otevřeném dialogovém okně zvolíme v levé části programovací jazyk - buď Visual Basic nebo C#. V pravé části vybereme možnost *Windows Phone Game*. Zvolíme také název naší aplikace.

4.2 Součásti XNA projektu

XNA projekt zahrnuje dvě části. První z nich obsahuje zdrojové kódy projektu, druhá část obsahuje multimediální obsah. V této části se nacházejí např. obrázky nebo zvuky použité ve hře. Pokud chceme do multimediálního obsahu přidat obsah, klikneme v *Solution Exploreru* pravým tlačítkem na úvodník obsahové části a zvolíme *Add -> Existing Item*. V otevřeném dialogovém okně vybereme příslušný soubor.

4.3 Struktura XNA projektu

Strukturu XNA projektu tvoří hlavní třída, která je oděděná od třídy *Microsoft.Xna.Framework.Game*.

4.3.1 Důležité metody třídy Game

Konstruktor
Zde probíhá vytvoření instance třídy **GraphicsDeviceManager**, která má na starosti grafickou stránku v zařízení. Lze například nakonfigurovat, zda se grafická plocha roztáhne na celou šířku displeje.

Initialize
V metodě ***Initialize*** nastavíme počáteční hodnoty proměnným. Zde se ještě však nepracuje s grafickým obsahem.

Příklad metody **Initialize***:*

```
protected override void Initialize()
{
     // Povolena gesta pro dotykovy displeje
    TouchPanel.EnabledGestures = GestureType.FreeDrag |
       GestureType.Tap;
    base.Initialize();
}
```

LoadContent
V metodě ***LoadContent*** se vytvoří instance třídy **SpriteBatch**, což je třída vykreslující načtené grafické textury pomocí metody ***Draw***.

Vytvoření instance třídy **SpriteBatch** *pro další příklady:*

```
Spritebatch spriteBatch = new SpriteBatch(GraphicsDevice);
```

Parametr *GraphicsDevice* je složka třídy **Game** vracející instanci třídy **GraphicsDevice** (jmenný prostor *Microsoft.Xna.Framework.Graphics*). Třídu používá **SpriteBatch** k vykreslování objektů.

Dále se v metodě inicializuje multimediální obsah, který máme připravený v obsahové části projektu. Načtení se děje pomocí metody ***Content.Load***. Jejím parametrem je název multimediálního souboru z obsahové části. Název souboru se uvádí bez přípony. Proto se žádné multimediální soubory nesmí jmenovat stejně, aby nedocházelo ke konfliktu názvů.

Příklad načtení připraveného obrázku bg.png:

```
Texture2D background = Content.Load<Texture2D>("bg");
```

Mimo běžné multimediální obsahy jako grafické prvky nebo zvuky lze také načíst písma. Nejdříve je však nutné vytvořit soubor s definicí písma. Soubor se vytvoří kliknutím pravého tlačítka na úvodník obsahové části a volbou *Add -> New Item*. Následně v dialogovém okně vybereme ze seznamu možnost *Sprite Font*. Do obsahové části bude přidán soubor s příponou *spritefont*. Jedná se o soubor ve formátu XML, který definuje vlastnosti písma jako použitý font, velikost, řez atd. Soubor upravíme podle toho, jaké chceme použít písmo.
Příklad načtení písma definovaného v souboru SpriteFont1.sprintefont:

```
SpriteFont font = Content.Load<SpriteFont>("SpriteFont1");
```

Update
Metoda ***Update*** běží v cyklu a obstarává aktualizaci stavu hry. Umožňuje například měnit souřadnice grafických objektů, zpracovává dotyková gesta a vyhodnocuje je. V této metodě se nemění grafický obsah scény, jen se aktualizují potřebné proměnné.

Draw
Metoda překreslující obsah scény. Zde se již pracuje s připraveným grafickým obsahem, který se vykresluje se zadanými parametry, jako například souřadnice objektů nebo jejich barva. Slouží i k vykreslování textových prvků hry jako hlášky nebo menu.
Příklad vykreslení textu pomocí připraveného fontu z minulého příkladu:

```
spriteBatch.DrawString(font, "Textovy vypis", new
   Vector2(20, 20), Color.White);
```

Vector2 je struktura o dvou složkách *X* a *Y*, které určují souřadnice grafických objektů.
Color je struktura obsahující reprezentace barev.

5 Praktická část

5.1 Cíle práce

- Mobilní klient
- Desktopový klient
- Síťová komunikace
- Použití multimediálních a interaktivních prvků

5.1.1 Mobilní klient

Aplikace pro mobilní zařízení představuje multimediální a síťovou hru ping pong napsanou pomocí XNA frameworku. Hraní probíhá tak, že jeden z hráčů založí hru a vyčká, než se k němu připojí další hráč. Poté může zakládající hráč začít hru. Hra se ovládá pohybem pálky, na mobilním zařízení je to dotykem a přetažením pálky. Hrát hru lze, pokud jsou zařízení připojena ke stejné Wi-Fi síti. Pro ukládání herních předvoleb se využívá lokální úložiště aplikace.

5.1.2 Desktopový klient

Desktopový klient je téměř totožný s mobilním. Liší se však v použití nástrojů pro síťovou komunikaci přes UDP multicast, v ovládání hry, které probíhá přes klávenici a ve způsobu ukládání herních předvoleb.

5.1.3 Síťová komunikace

Pro síťovou komunikaci je využíván UDP multicast. Zařízení se přidávají do multicastové skupiny a pak mohou přijímat a zasílat zprávy ostatním členům skupiny datagramové pakety. Síťová část aplikace mobilního i desktopového klienta byla navržena tak, aby mohla sloužit jako podklad pro vývoj dalších síťových her. Hry mohou být jak v reálném čase, tak tahové.

5.1.4 Použití multimediálních a interaktivních prvků

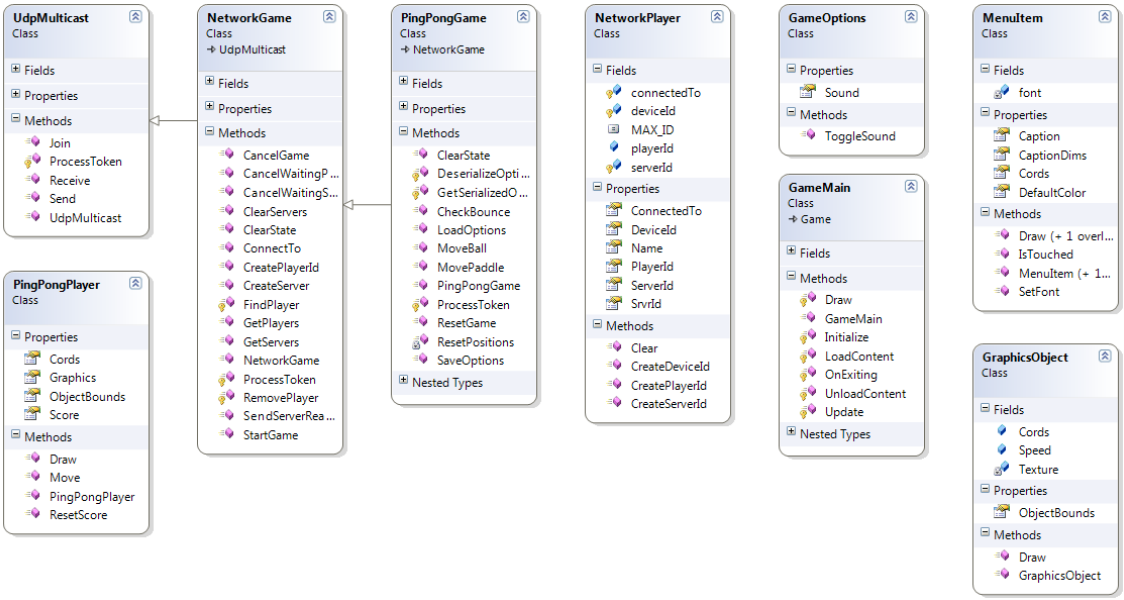
Mobilní klient demonstruje použití multimediálních prvků jako jsou grafické elementy, zvuky nebo písma. Dále zahrnuje použití dotykových gest. Ovládání aplikace probíhá právě přes dotyková gesta. Byly vytvořeny také nástroje, které zjednodušují vytváření menu s volitelnými položkami přes dotyková gesta.

5.2 Struktura aplikace a přehled tříd v aplikaci

Zdrojový kód aplikace je zapouzdřený ve jmenném prostoru *PingPongMulti* v mobilní aplikaci a *PingPong* v desktopové aplikaci. Objektový návrh aplikace je stejný jak pro mobilního, tak desktopového klienta.V tabulce 5.1 je uveden přehled jednotlivých tříd.

Třída	Popis
GraphicsObject	Vlastnosti 2D grafických objektů. Což jsou souřadnice objektu, textura objektu a rychlost objektu pro pohybující se objekty.
GameOptions	Obsahuje složky reprezentující herní volby.
UdpMulticast	Zahrnuje základní síťové komunikační metody pro komunikaci přes multicast.
NetworkGame	Třída oděděná od třídy **UdpMulticast**, jedná se o obecné herní síťové rozhraní s metodami pro vytvoření hry a připojení se ke hře.
NetworkPlayer	Reprezentace hráče v síti, součást obecného síťového rozhraní.
PingPongGame	Třída oděděná od třídy **NetworkGame**. Třída již obsahuje herní logiku pro konkrétní hru - tedy ping pong.
PingPongPlayer	Představuje hráče konkrétní hry, tedy ping pongu. Obsahuje informaci o skóre hráče a texturu hráče.
MenuItem	Položka na dotykovém displeji, kterou lze zvolit "tap"gestem (kliknutím na displej).
GameMain	Hlavní třída odvozená od *Microsoft.Xna.Framework.Game*. Třída obsahuje obecnou herní logiku pro XNA hry.
Program	Obsahuje metodu *Main*.

Tabulka 5.1: Přehled tříd mobilní a desktopové aplikace



Obrázek 5.1: UML diagram mobilní a desktopové aplikace

5.3 Reprezentace grafických objektů ve hře

Grafické prvky hry jsou v aplikaci reprezentovány pomocí třídy **GraphicsObject**. Třída obsahuje základní vlastnosti objektu jako textura, souřadnice objektu a rychlost objektu. Rychlost objektu představuje dvojrozměrný vektor, který určuje o kolik bodů se bude objekt posouvat v metodě **Update**. Dále ještě třída zahrnuje metodu ***ObjectBounds***, která zjednodušuje detekci kolizí s jinými objekty.

Ukázka metod **LoadContent**, **Draw** a vykreslení objektu pomocí třídy **GraphicsObject**:

```
SpriteBatch spriteBatch;
Texture2D background;
GraphicsObject background;

. . .

protected override void LoadContent()
{
    spriteBatch = new SpriteBatch(GraphicsDevice);

    // nacteni textury
    bgTexture = Content.Load<Texture2D>("bg");
    background = new GraphicsObject(bgTexture,
       Vector2.Zero);
}

protected override void Draw(GameTime gameTime)
{
    // Barva pozadi nastavena na cernou
    GraphicsDevice.Clear(Color.Black);

    spriteBatch.Begin();

    // Vykresleni pozadi
    background.Draw(spriteBatch);

    spriteBatch.End();

    base.Draw(gameTime);
}
```

5.4 Herní volby

Volby ve hře, jako přepínání zvuku jsou zapouzdřeny ve třídě **GameOptions**, která obsahuje gettery a settery daných vlastností.

5.5 Síťová komunikace ve hře

Pro komunikaci mezi zařízeními slouží v herním prostředí UDP multicast. Multicast rozesílá všem účastníkům multicastové skupiny datagramový paket. Rozsah IP adres pro multicastovou skupinu je 224.0.0.0 – 239.255.255.255. Hodnota portu může nabývat hodnot 1024 – 65535. Lze také nastavit, že odeslaný paket přijme i jeho odesílatel.

Při použití multicastu musí být zařízení připojena na stejné Wi-Fi síti. Na rozdíl od protokolu TCP není ale potřeba žádného desktopového zařízení (serveru), přes který by komunikace procházela. Spojení přes TCP protokol v aplikacích pro Windows Phone 7 totiž zatím neumožňuje vytvořit server. Zařízení se může chovat pouze jako klient. A k propojení zařízení je potřeba desktopové zařízení, které by reprezentovalo server.

Při použití multicastu mohou mobilní zařízení komunikovat rovnou mezi sebou.

Při realizaci konkrétní hry, ping pongu, což je hra pro 2 hráče by mohl být také použitý UDP unicast. Multicast byl zvolen s ohledem na realizaci obecného síťového rozhraní. Pomocí tohoto rozhraní bude možné realizovat hry různého typu, např. pro několik hráčů. V tomto případě se jeví použití UDP multicastu jako lepší volba.

5.5.1 Realizace multicastového rozhraní ve Windows Phone

Multicastová komunikace je ve Windows Phone 7 realizována pomocí třídy **System.Net.Sockets.UdpAnySourceMulticastClient**. Při vytváření instance třídy se v konstruktoru jako první parametr určí IP adresa multicastové skupiny. Druhým parametrem je port, na kterém bude komunikace probíhat. V aplikaci je konkrétně zvolen port 52364.

Pro zahájení komunikace ve skupině je nejdříve nutné se ke skupině přidat. Teprve poté můžeme zasílat a přijímat zprávy. Zprávy se odesílají po bytech, proto je potřeba odesílanou zprávu nejdříve převést na pole bytů. Zpráva je poté přijata také v bytech, podle potřeby lze poté přijatou zprávu převést na **string**.

Pro asynchronní odesílání bytových zpráv se používá metoda **BeginSendToGroup**. Prvním parametrem metody je bytové pole, které se posílá. Druhý parametr je offset, od kterého se posílá. Třetí parametr je délka zasílaného pole. Čtvrtý parametr je callback funkce dokončující odesílání dat pomocí metody **EndSendToGroup**. Pátým parametrem je objekt, který se předává callback funkci jako parametr.

Asynchronní příjem zpráv se zajišťuje metodou **BeginReceiveFromGroup**. Parametry jsou analogické k odesílací metodě. Prvním řetězec, do kterého se zapíší přijatá data. Druhý parametr je offset. Třetí parametr je délka přijatých dat. Čtvrtým parametrem je callback funkce dokončující příjem dat pomocí metody **EndReceiveFromGroup**. Pátým parametrem je objekt, který se předává callback funkci jako parametr.

Příklady použití třídy **UdpAnySourceMulticastClient** *pro asynchronní přidání klienta do skupiny:*

```
bool IsJoined = false;

. . .

public void Join()
{
    if (IsJoined)
    {
        return;
    }

    client = new UdpAnySourceMulticastClient(
       IPAddress.Parse(groupAddress), groupPort);

    // Asynchronni pripojeni k multicastove skupine
    client.BeginJoinGroup(
        result => {
            // Dokonceni asynchronniho pripojeni
            Client.EndJoinGroup(result);

            // Pripojeno
            IsJoined = true;

            // Zacneme prijimat zpravy
            Receive();
    }, null);
}
```

5.5.2 Třída UdpMulticast

Třída obsahuje základní metody pro síťovou komunikaci přes multicast mezi zařízeními.

Důležité složky třídy:

- **UdpAnySourceMulticastClient** *client* - síťový klient, který vykonává asynchronní operace pro přidání zařízení do skupiny, zaslání a příjem zpráv. Po přidání vyšle speciální zprávu signalizující, že se do skupiny přidalo další zařízení.
- **string** *groupAddress* - IP adresa skupiny.
- **int** *groupPort* - port, na kterém bude probíhat komunikace.
- **byte[]** *buffer* - buffer, kam se ukládají přijaté zprávy.

Metody třídy:

- **void Join()** - vytvoří instanci **UdpAnySourceMulticastClient** a přidá klienta do multicastové skupiny.
- **void Send(string** *message***)** - asynchronně odešle zprávu členům multicastové skupiny.
- **void Receive()** - asynchronně přijme od skupiny zprávu a dále ji zpracovává pomocí metody ***ProcessToken***. Metoda je virtuální a její tělo je definované v poděděných třídách.

5.5.3 Obecné herní síťové rozhraní

Třída **NetworkGame** je oděděná od třídy **UdpMulticast**. Rozšiřuje tak základní prostředky pro multicastovou komunikaci tak, aby reprezentovala herní síťové rozhraní pro hry založené na bázi server - klient. Jedná se o třídu, která řídí síťovou komunikaci. Pro přehlednost byla vytvořena ještě třída **NetworkPlayer**, která zahrnuje atributy hráče v síti. S touto třídou pracuje třída **NetworkGame**.

V modelu komunikace nefiguruje žádný reálný server (jako v případě TCP protokolu), který by akceptoval připojení. Nicméně jako server je určeno jedno zařízení, které založí hru. Ostatní hráči se budou chovat jako klienti a mohou se připojit k zařízení, které založilo hru.

Důležité složky třídy NetworkGame:

- **NetworkPlayer** *Player* - zapouzdřuje základní vlastnosti síťového hráče.
- **int** *MinPlayersCount* - minimální počet hráčů ve hře.
- **int** *MaxPlayersCount* - maximální počet hráčů ve hře.
- **List<int>** *Servers* - seznam ID zařízeních, která založila hru.
- **List<NetworkPlayer>** *Players* - seznam hráčů čekající na potvrzení od konkrétního serveru.
- **List<NetworkPlayer>** *PlayersActive* - seznamu aktivních hráčů ve hře.

Důležité metody třídy NetworkGame:

- **void ProcessToken(string** *message***)** - přepisuje nadřazenou virtuální metodu. Zpracovává přijaté zprávy a rozhoduje, jaká data se odešlou dále, popřípadě nastavuje složky třídy. Třída **UdpMulticast** ihned po přidání zařízení do skupiny odesílá notifikační zprávu, že bylo přidáno zařízení. Zpráva je zde zpracována a zařízení se přiřadí ID (složka *DeviceId*).
- **void CreateServer()** - třída vytvoří z daného zařízení server (přiřadí mu *ServerId*), ke kterému se mohou ostatní připojit.
- **void GetPlayers()** - odešle své *ServerId* a poté naplní seznam *Players* ID hráčů, kteří vybrali tento server.

- **void GetServers()** - naplní seznam *Servers* ID serverů, které čekají na připojení hráčů.
- **void CreatePlayerId()** - vytvoří hráči *PlayerId*.
- **void ConnectTo(int** *serverId***)** - přiřadí hráče (klienta) k serveru.
- **void CancelWaitingServer()** - server přeruší čekání na hráče. Všem členům skupiny zašle své ID a to se odstraní ze seznamu *Servers*.
- **void CancelWaitingPlayer()** - hráč přeruší čekání na server, odešle zprávu, ke kterému byl připojen serveru a své ID, které se odstraní ze seznamu *Players*.
- **void StartGame()** - server zahájí hru tak, že do seznamu *PlayersActive* vloží nejdříve informaci o svém hráči a poté i o ostatních připojených hráčů. Seznam se odešle ostatním hráčům.
- **void SendServerReady()** - odešle klientům zprávu, že všichni ostatní klienti jsou připraveni a hra může začít.
- **void CancelGame()** - slouží pro ukončení aktivní hry. Pokud je hra ukončena na serveru, odešle se ID serveru. Všem hráčům, kteří jsou k tomuto serveru připojeni se hra ukončí. V případě, že ze hry odešel hráč, odešle se ID serveru, k jakému byl připojen a jeho *PlayerId*. Na serveru se ukončí hra v případě, že počet hráčů klesne pod minimální hodnotu (*MinPlayersCount*).
- **void ClearState()** - nastaví složky třídy na počáteční hodnoty.

5.5.4 Reprezentace hráče v síťovém rozhraní

Pro účel reprezentace hráče slouží třída **NetworkPlayer**, ta odděluje vlastní atributy hráče od řídící logiky třídy **NetworkGame**.
Důležité složky třídy NetworkPlayer:

- **int** *DeviceId* - obecné ID (reprezentováno číslem) zařízení.
- **int** *ServerId* - ID zařízení (reprezentováno číslem), které založilo hru.
- **int** *PlayerId* - ID hráče (reprezentováno číslem), hráč dostane přidělené své ID poté, co buď založí hru nebo se k nějaké připojí.
- **int** *ConnectedTo* - ID zařízení, ke kterému je hráč připojen.
- **int** *SrvrId* - v případě serveru vrátí *ServerId*, v případě klienta *ConnectedTo*.
- **string** *Name* - jméno hráče.

5.5.5 Popis logiky síťové hry

Pro zahájení síťové hry je nejprve nutné, aby jakýkoli člen skupiny založil hru pomocí metody **CreateServer**. Pak je na serveru nutné zavolat **GetPlayers**, která zjišťuje, jestli si nějaký hráč nevybral náš server. Seznam hráčů, kteří tak učinili se nachází v seznamu *Players*. V době, kdy server čeká na hráče, lze čekání ukončit zavoláním ***CancelWaitingServer***.

Na straně klienta se pro zjištění dostupných serverů zavolá metoda ***GetServers***, která zjistí, jaké servery čekají na hráče. Výběr serveru provedeme metodou ***ConnectTo***. V době, kdy hráč čeká na server, lze čekání ukončit zavoláním ***CancelWaitingPlayer***.

Na straně serveru zahájíme hru pomocí metody ***StartGame*** a následně ***SendServerReady***.

5.5.6 Vrstva konkrétní hry

Logiku vlastní hry, tedy ping pong, reprezentuje třída **PingPongGame**. Ta je poděděna od třídy **NetworkGame** a konkretizuje tím vlastnosti hry. Atributy jednotlivých hráčů jsou zahrnuty ve třídě **PingPongPlayer**.

Důležité složky třídy PingPongGame

- **PingPongPlayer** *playerOne, playerTwo* - objekty reprezentující 2 hráče ve hře.
- **GraphicsObject** *ball* - objekt ping pongového míčku, během hry mu bude měněna rychlost.
- **Vector2** *DefaultSpeed* - počáteční rychlost míčku, dvojrozměrný vektor určující směr míčku a rychlost posouvání míčku v herní metodě **Update**.
- **int** *MaxScore* - skóre, po jehož dosáhnutí se hra ukončí.

Důležité metody třídy PingPongGame

- **void ProcessToken(string** *message***)** - rozšiřuje nadřazenou metodu o další, již konkretizované akce, které se dějí po zpracování přijaté zprávy. Konkrétně obstarává pohyb pálek a uvedení hry do počátečního stavu (vynulování skóre).
- **void MovePaddle(float** *X***)** - pohybuje hráčovou pálkou, odešle *ServerId* (nebo v případě klienta *ConnectedTo*), *PlayerId* a parametr *X*, který určuje o kolik a na jakou stranu se má pálka pohnout. Po přijetí těchto informací se zkontroluje, jestli zpráva došla na správný server a podle *PlayerId* se pohne správným hráčem.
- **void MoveBall()** - hýbe pálkou tak, že přičítá k aktuálním souřadnicím míčku vektor rychlosti.
- **void ResetGame(PlayerTypes** *servingPlayer***, bool** *increaseScore***)** - náhodně nastaví výchozí pozice hráčů, parametr *servingPlayer* určuje, který hráč podává, *increaseScore* určí, jestli se tomuto hráči má zvednout skóre. Odesílá *ServerId* (nebo v případě klienta *ConnectedTo*), informaci, který hráč podává a informaci, jestli se zvedne skóre. Po přijetí na správný server se vygenerují náhodné pozice a ty se opět odešlou.

- **void ResetPositions(PlayerTypes** *servingPlayer*, **bool** *increaseScore*) - metoda je zavolána po přijetí informací odeslané metodou ***ResetGame***. Metoda nastaví hráčům nové souřadnice, nastaví podávajícího hráče a popřípadě navýší skóre.
- **void CheckBounce()** - metoda detekující kolize míčku a zdí. Při detekci je míček změnou směru vektoru rychlosti odražen. Kolize se detekuje pomocí metody ***Intersects*** třídy **Rectangle**. Metoda vrací *true*, pokud se dva objekty typu **Rectangle** protínají.
- **void SaveOptions()** - uloží do *Isolated Storage* herní volby (nastavení zvuku) a jméno hráče. Uložení probíhá tak, že se údaje převedou na řetězec, který se uloží do textového souboru.
- **void LoadOptions()** - načte předvolby hry z textového souboru v *Isolated Storage*. Pokud je aplikace spuštěna poprvé, vytvoří se textový soubor s výchozími předvolbami.

Důležité složky třídy PingPongPlayer

- **GraphicsObject** *Graphics* - textura hráče, tedy pálky.
- **int** *Score* - skóre hráče.

Příklad detekce kolize míčku s hráčem a následné přehrání zvuku:

```
GraphicsObject Ball;
PlayerOne Ball;
GameOptions Options;
SoundEffect PaddleHit;

. . .

public void CheckBounce()
{
    if (Ball.ObjectBounds.Intersects(
       PlayerOne.ObjectBounds))
    {
        Ball.Speed.Y *= -1; // Zmeni se smer micku
        Ball.Cords += Ball.Speed;
        if (Options.Sound) // Pokud je zvuk povolen
        {
            PaddleHit.Play(); // Prehrani zvuku
        }
    }
}
```

5.5.7 Popis použití třídy PingPongGame

Pokud potřebujeme ve hře pohnout pálkou, zaznamenáme si z pohybového gesta změnu souřadnice *X* a tu předáme metodě ***MovePaddle***.

Pro pohyb míčku bude v metodě ***Update*** volána metoda ***MoveBall***, která k aktuální pozici míčku přičítá vektor rychlosti míčku.

Metoda *ResetGame* se zavolá při zahájení hry nebo v případě, že hráč nestihne odpálit míček (nedojde ke kolizi objektu *playerOne* nebo *playerTwo* s objektem *ball*).

5.5.8 Formát vyměňovaných zpráv

V tabulce 5.2 je uvedený formát vyměňovaných zpráv v rámci síťové komunikace.

Formát zprávy	**Význam**
GET_PLAYERS: *ServerId*	Server *ServerId* se dotazuje na hráče, kteří jsou k němu připojení.
SERVER_CANCEL_WAIT: *ServerId*	Server *ServerId* přerušuje čekání na hráče.
SERVER_CANCEL_GAME: *ServerId*	Server *ServerId* ukončil běžící hru.
START_GAME: *ServerId*/*Players*	Server *ServerId* započal hru a všem ostatním účastníkům odeslal seznam hráčů *Players* ve tvaru řetězce.
SERVER_READY: *ServerId*	Server *ServerId* odeslal zprávu, že hra může začít.
GET_SERVERS	Klient se dotazuje na čekající servery.
PLAYER_CANCEL_WAIT: *ConnectedTo*/*PlayerId*	Klient *PlayerId* připojený k serveru *ConnectedTo* přerušil čekání na server.
PLAYER_CANCEL_GAME: *ConnectedTo*/*PlayerId*	Klient *PlayerId* připojený k serveru *ConnectedTo* ukončil běžící hru.
SERVER_ID: *ServerId*	Server *ServerId* odpovídá na požadavek čekajících hráčů na seznam serverů a zasílá své ID.
PLAYER_IN: *ConnectedTo*/*PlayerId*	Klient *PlayerId* připojený k serveru *ConnectedTo* potvrdí, že je připraven ke hře.
SERVER_QUEUE_PLAYER_ID: *ConnectedTo*/*PlayerId*/*Name*	Klient *PlayerId* připojený k serveru *ConnectedTo* odpovídá na požadavek serveru na seznam připojených hráčů a odesílá také své jméno *Name*.
MOVE: *SrvrId*/*PlayerId*/*X*	Pohyb hráče *PlayerId* na serveru *SrvrId* o *X* pixelů.
RESET: *SrvrId*/*PlayerType*/*increaseScore*	Požadavek na přenastavení pozic hráčům, hráč *PlayerType* (označení jednoho ze dvou hráčů) je podávajícím hráčem, *increaseScore* určuje, zda se tomuto hráči zvýší skóre.
DEFAULT_POSITION: *SrvrId*/*PlayerType*/*increaseScore*/ *pos1*/*pos2*	Dokončení nastavení pozic *pos1* a *pos2* hráčům.

Tabulka 5.2: Přehled vyměňovaných zpráv v rámci síťové komunikace

5.6 Práce s grafickým menu a vybíratelnými položkami

Ve hře se také vyskytují různá menu a vybíratelné položky dotekem na displej. Vytváření těchto položek, zjednodušuje třída **MenuItem**, která představuje jednu položku, kterou lze ve hře vybrat dotykem.

Složky třídy MenuItem:

- **string** *Caption* - text položky, určuje se v konstruktoru.
- **Vector2** *Cords* - souřadnice položky.
- **SpriteFont2** *font* - písmo položky.
- **Color2** *DefaultColor* - výchozí barva textu.
- **Vector2** *CaptionDims* - rozměry položky.

Zjištění, zda byla položka vybrána se děje pomocí metody:

- **bool IsTouched(GestureSample** *gesture***)** - pomocí parametru *gesture*, který byl získán z "tap"gesta se prověří, zda byla daná položka vybrána. Prověření funguje tak, že ze souřadnic a rozměru položky se vytvoří objekty typu **Rectangle**, tedy obdélníky představující plochu, kterou položka zabírá. Poté se pouze porovná metodou ***Contains*** třídy **Rectangle**, zda plocha obsahuje souřadnice dotyku, které obsahuje parametr *gesture*.

Ukázka funkčnosti metody **IsTouched***:*

```
// text polozky
string caption = "OPTIONS";
// odchyti se dotykove gesto
GestureSample gesture = TouchPanel.ReadGesture();
// souradnice polozky
Vector2 cords = new Vector2(50, 50);
// nacteme pismo
SpriteFont font = Content.Load<SpriteFont>("SpriteFont1");
// rozmer polozky
Vector2 dimensions = font.MeasureString(caption);
// plocha, kterou polozka zabira
Rectangle bounds = new Rectangle((int)cords.X,
   (int)cords.Y, (int)dimensions.X, (int)dimensions.Y);
if (bounds.Contains((int)gesture.Position.X,
   (int)gesture.Position.Y)) {
    // polozka byla vybrana
 }
```

5.7 Obecná herní logika hry

Zbývající logika hry se nachází v hlavní třídě **GameMain**, především v metodách ***Update*** a ***Draw***. Logika určuje a podmiňuje, jaké akce se budou ve hře dít. V této třídě je umístěno i herní menu. Síťová komunikace a konkrétní herní logika je zapouzdřena ve třídě **PingPongGame**.

5.7.1 Popis fungování herní logiky

Při inicializaci v metodě ***Initialize*** se nastaví dostupná dotyková gesta. V našem případě přetažení a "kliknutí na displej". Vytvoří se instance herní třídy **PingPongGame**, síťové parametry se nastaví v konstruktoru. V kontruktoru se také vytvoří instance herních voleb *GameOptions* a nastaví se počáteční rychlost a směr míčku.

Metoda ***LoadContent*** načte všechny obsahové prvky hry a vytvoří z nich grafické objekty.

Hra je rozčleněna na několik částí. Jednotlivé části jsou popsány výčtem **GameState**, který obsahuje hodnoty (části hry):

- *Menu* - úvodní herní menu.
- *Options* - volby hry.
- *FindServers* - část hry, kde se vybírá dostupný server.
- *WaitingForServer* - v této části se čeká na potvrzení od vybraného serveru, že hra může začít.
- *WaitingForPlayer* - server čeká na připojení hráčů.
- *Game* - stav, kdy se hraje hra.
- *GameOver* - stav, kdy jeden z hráčů vyhrál - dosáhl požadovaného skóre.

Složka *gameState* obsahuje aktuální část hry, která se vykresluje. To znamená, že v metodě ***Update*** se pracuje jen s těmi objekty a proměnnými, podle toho v jaké části hry se momentálně nacházíme. Stejně tak v metodě ***Draw*** se vykresluje příslušná část hry. Pro skok do jiné části hry stačí v metodě ***Update*** přiřadit složce *gameState* jinou hodnotu z výčtu **GameState**.

Těsně před ukončením aplikace se zavolá přetížená metoda ***OnExiting***, která zajistí, že se podle aktuálního stavu hry zavolají metody pro ukončení čekání hráče na server, pro ukončení čekání serveru na hráče nebo pro ukončení probíhající hry.

Popis jednotlivých částí hry

Menu
Pro reprezentaci menu slouží seznam **List<MenuItem>** *mainMenu*. Do tohoto seznamu jsou při inicializaci vloženy položky hlavního menu (instance třídy **MenuItem**).

- Metoda ***Draw***
  Položky se v metodě ***Draw*** vykreslí, přičemž se zaznamenají souřadnice jednotlivých položek do dvojrozměrných vektorů. Vektory se pak používají při zjišťování, která položka byla dotykem vybrána.

- Metoda ***Update***
  Probíhá detekce dotykového gesta "klepnutí na displej". Dále se zjistí, jestli jsme vybrali dotykem některou z položek a přeskočíme podle toho na jinou část hry. Zjišťování, zda byla vybrána položka menu probíhá s pomocí zaznamenaných souřadnic položek z metody ***Draw***. Způsob výběru položek probíhá pomocí třídy **MenuItem**, která obsahuje text položky, souřadnice položky a je schopna spočítat rozměry textu položky.

Příklad detekce gesta a zjištění, zda byla položka vybrána dotykem:

```
MenuItem newGameItem = new MenuItem("NEW GAME");

. . .

while (TouchPanel.IsGestureAvailable)
{
    GestureSample gesture = TouchPanel.ReadGesture();

    // porovnani, zda se jedna o "tap" (klikaci) gesto
    if (gesture.GestureType == GestureType.Tap)
    {
        if (newGameItem.isTouched(gesture))
        {
            gameState = GameState.WaitingForPlayer;
        }
    }
}
```

Options
Funkčnost této části, co se týče používání dotykového gesta je totožná s částí *Menu*. Po kliku na některou z možností se aktualizují složky *PingPongGame.Options*, s kterými se pracuje v metodě ***Update***. Také se zde nastavuje jméno hráče pomocí softwarové klávesnice. Po změně volby zapnutí nebo vypnutí zvuku či změně jména se volby ukládají do lokálního úložiště aplikace *Isolated Storage* pomocí metody ***SaveOptions*** třídy **PingPongGame**. Uložené volby jsou načtené při spuštění aplikace pomocí metody ***LoadOptions***. Pokud je aplikace spuštěna poprvé, soubor s uloženým nastavením v *Isolated Storage* ještě neexistuje. Je proto vytvořen s výchozím nastavením předvoleb.

Příklad zobrazení softwarové klávesnice a uložení nové hodnoty:

```
string playerName;

. . .

Guide.BeginShowKeyboardInput(
    PlayerIndex.One,
    // titulek dialogoveho okna
    "Player name",
    // popis textoveho pole
    "Please insert your name",
    // promenna, kam se ulozi hodnota z klavesnice
    playerName,
    // callback funkce spustena po ukonceni klavesnice
    result => {
            if (result.IsCompleted)
            {
                string keyboardResult =
                   Guide.EndShowKeyboardInput(result);
                if (null != keyboardResult) {
                    // ulozime novou hodnotu hracova jmena
                    playerName = keyboardResult;
                }
            }
        },
    null
);
```

FindServers
V části **FindServers** probíhá výpis zařízeních, které založily hru.

- Metoda ***Draw***
  Vykreslí obsah seznamu *Servers* třídy **PingPongGame**. Také vykreslí text *CANCEL*, kterým se lze vrátit do menu.

- Metoda ***Update***
  Hráč se přidá k multicastové skupině a zavolá metodu ***GetServers*** třídy **PingPongGame**, která naplní seznam *Servers*. Zjišťuje se, zda hráč nevybral dotykem některý server. Pokud ano, zavolá se metoda ***ConnectTo*** třídy **PingPongGame**, která přiřadí hráče k danému serveru. Dále se stav hry změní na *WaitingForServer*, tedy čekání na potvrzení od serveru.

Pokud hráč zvolí při čekání odchod zpět do menu, volá se metoda ***ClearState*** třídy **PingPongGame**.

WaitingForServer
Zde čekají hráči, kteří vybrali server na jeho potvrzení, že hra může začít.

- Metoda ***Draw***
  V metodě ***Draw*** se vykreslí obsah seznamu *Players* třídy **PingPongGame**. Jedná se o seznam hráčů, kteří vybrali náš server. Také se kontroluje podmínka, zda je připojen dostatek hráčů. V případě, že ano, vykreslí se text, že hra může být započata.

- Metoda ***Update***
  Zde se hráči přiřadí jeho *PlayerId* a čeká se, až server zavolá metodu ***SendServerReady*** třídy **PingPongGame**. V případě volby odchodu do menu se zavolají metody ***CancelWaitingPlayer*** a ***ClearState*** třídy **PingPongGame**. První z nich způsobí, že na serveru se aktualizuje seznam čekajících hráčů. To znamená, že dojde k odstranění tohoto hráče ze seznamu.

WaitingForPlayers
Část hry **WaitingForPlayers** založí server a poté čeká na hráče. Když je připojen dostatek hráčů, může server začít hru.

- Metoda ***Update***
  Zavoláním metody ***CreateServer*** třídy **PingPongGame** se založí server. Poté se serveru přiřadí *PlayerId*. Když je vytvořený server, zavolá se metoda ***GetPlayers*** třídy **PingPongGame**, která zjistí, jací hráči vybrali náš server a naplní seznam *Players*.

  Po tom, co se potvrdí start hry se všem hráčům připojeným k tomuto serveru odešle seznam hráčů (metodou ***StartGame***). A ti odešlou serveru odpověď, že jsou připraveni. Server zkontroluje, že obdržel tolik potvrzení, kolik je hráčů. Následně odešle hráčům, že hra je připravena (metodou ***SendServerReady***). Na obou stranách se hra přemístí do části *Game*.

  Pokud server přeruší čekání na hráče, všem připojeným hráčům se odešle notifikační zpráva. Tím se klientům i serveru automaticky změní část hry na *Menu*.

Game
Část, ve které probíhá vlastní hra.

- Metoda ***Draw***
  Metoda vykresluje všechny grafické objekty na scéně.

- Metoda ***Update***
  Zde je zajištěno, že díky zachycování dotykových gest táhnutí lze pohybovat pálkami. Při dotyku se porovnávají souřadnice a pokud hráč táhne za svoji pálku, odesílá se na zařízení informace o nové poloze pálky.

  Pohyb míčku je zajištěn přičítáním vektoru rychlosti k jeho stávající poloze:

```
Ball.Cords += Ball.Speed;
```

  Pokud je zvolen odchod ze hry, v případě serveru se odešle všem klientům *ServerId*. Klienti se, stejně jako server, vrátí zpět do menu. Když hru ukončí klient, zašle se *ServerId* (ID serveru, ke kterému je připojen) a jeho *PlayerId*. Klient ukončí hru a vrátí se do menu. Po přijetí zprávy na serveru je hráč odebrán ze seznamu aktivních hráčů. V případě, že počet hráčů klesl pod minimum, vrací se i server do menu.

5.8 Popis odlišností u desktopové aplikace

Již bylo zmíněno, že desktopová verze aplikace se od té mobilní liší jen v použitých prostředcích pro síťovou komunikaci, ve způsobu ovládání aplikace a v použitém úložišti pro ukládání předvoleb hry. Jinak zůstává objektový model aplikace nezměněný.

5.8.1 Prostředky pro síťovou komunikaci

Základní síťová komunikace je také zapouzdřena ve třídě **UdpMulticast**, která obsahuje totožné metody. K UDP komunikaci zde však slouží třída **Socket**. Pro přidání zařízení do UDP multicastové skupiny nastavíme socket jako datagramový, navážeme ho na port, přes který budeme komunikovat a přiřadíme mu multicastovou adresu. Pro asynchronní zasílání zpráv do skupiny se používají metody **BeginSendTo** a **EndSendTo**. Pro přijímání zpráv pak **BeginReceiveFrom** a **EndReceiveFrom**.

Příklad použití třídy **Socket** k přidání zařízení do multicastové skupiny:

```
bool IsJoined = false;

. . .

// Vytvoreni datagramoveho socketu pro komunikaci
Socket sock = new Socket(AddressFamily.InterNetwork,
   SocketType.Dgram, ProtocolType.Udp);

EndPoint localEP = (EndPoint)new IPEndPoint(IPAddress.Any,
   groupPort);

// Navazani socketu na port multicastove skupiny
sock.Bind(localEP);

MulticastOption multicastOption = new
   MulticastOption(IPAddress.Parse(groupAddress));

// Nastaveni adresy multicastove skupiny
sock.SetSocketOption(SocketOptionLevel.IP,
   SocketOptionName.AddMembership, multicastOption);

// Jsme pripojeni ke skupine
IsJoined = true;
```

5.8.2 Ovládání hry pomocí klávesnice

Výběr položek v menu v desktopovém klientovi probíhá stisknutím klávesnice, která reprezentuje danou položku. Kterou klávesnici je nutné stisknout, je v položce vyznačeno pomocí závorek. Např. *(N)EW GAME*.

Pro zachycení stisknutých kláves využíváme dvě proměnné currentKeyboardState a previousKeyboardState typu **KeyboardState** (jmenný prostor *Microsoft.Xna.Framework.Input*).

První z nich určuje aktuální stav klávesnice, druhá předchozí stav. Stav je zachycován v metodě ***U*pdate** následujícím způsobem:

```
// Predchozi stav klavesnice
previousKeyboardState = currentKeyboardState;

// Aktualni stav klavesnice
currentKeyboardState = Keyboard.GetState();
```

Test, zda byla stisknuta příslušná klávesnice:

```
// Klavesa "N"
if (currentKeyboardState.IsKeyDown(Keys.N))
{
    gameState = GameState.WaitingForPlayer;
}
```

V části hry, kde se mění jméno hráče vypadá práce zachytávání a identifikace kláves následujícím způsobem:

```
// Zaznamenavame stisknute klavesy
Keys[] pressedKeys = currentKeyboardState.GetPressedKeys();

// Prochazime a identifikujeme klavesy
foreach (Keys key in pressedKeys)
{
    if (previousKeyboardState.IsKeyUp(key))
    {
        // Navrat do menu bez ulozeni
        if (key == Keys.Escape)
        {
            . . .
        }
        // Navrat do menu s ulozenim
        else if (key == Keys.Enter)
        {
            . . .
        }
        // Pismena a cislice
        else if ((key >= Keys.A && key <= Keys.Z) || (key
           >= Keys.D0 && key <= Keys.D9))
        {
            . . .
        }
    }
}
```

5.8.3 Načítání a ukládání herní předvoleb

U herních předvoleb se ukládání liší od mobilní aplikace v použitém úložném prostoru. U mobilní aplikace se data ukládají od *Isolated Storage*, v případě desktopové aplikace standardně na disk.

5.9 Testování aplikace

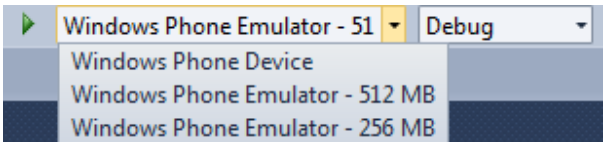
5.9.1 Testovací prostředí

Vývojové prostředí *Microsoft Visual Studio 2010* (*Windows Phone SDK* minimálně verze 7.1.1) umožňuje spustit dva emulátory na jednom počítači paralelně. V této verzi totiž přibyla možnost testovat také aplikace pro novou verzi systému *Windows Phone 7.5*, u které jsou snížené hardwarové nároky na zařízení. Aplikace byla tedy testována pomocí dvou emulátorů. První emulátor byl určený pro standardní zařízení s 512 MB RAM, druhý určený pro zařízení s 256 MB RAM.

Pro spuštění aplikace v debug módu se nejdříve zvolí emulátor příslušného zařízení, viz obr. 5.2. To se děje ve standardní vývojové liště. Aby byla lišta zobrazena, zaškrtneme *View -> Toolbars -> Standard*. Volíme mezi zařízeními *Windows Phone Emulator - 512 MB* a *Windows Phone Emulator - 256 MB*. Vedle volby zařízení se nachází volby *Debug* a *Release*. První z voleb spustí emulátor ve standardním debug režimu. Druhou volbu volíme v případě, že chceme sestavit XAP balíček. XAP balíček slouží pro nahrání a publikaci aplikace na *Marketplace*.

Nebyl nalezen způsob, jak spustit 2 emulátory v režimu *Debug* (ladění). Proto v tomto případě bylo postupováno tak, že nejdříve byl zvolen emulátor jednoho zařízení a pravým kliknutím na projekt zvolím *Deploy*. Volba *Deploy* spustí emulátor bez možnosti ladění. Dále byl vybrán emulátor druhého zařízení a tento emulátor již můžeme spustit v režimu *Debug*.

Aplikaci byla také lokálně testována s jedním emulátorem a desktopovou aplikací.



Obrázek 5.2: Část standardní lišty sloužící pro přepínání mezi emulátory

5.9.2 Testované scénáře

1. **Ukončení čekajícího serveru na klienty**

   Na straně klienta se server odstraní ze seznamu dostupných serverů. Je nutné vyčkat na nový server.

2. **Ukončení serveru, na jehož potvrzení čeká klient**

   Čekající klient je vrácen do hlavního menu.

3. **Ukončení čekání klienta na potvrzení serveru, že hra může začít**

   Hráč je na serveru smazán ze seznamu čekajících hráčů a je nutné počkat na nového hráče, aby mohla hra začít.

4. **Ukončení hry při jejím běhu**

   Druhé zařízení zareaguje tak, že se vrátí do hlavního menu.

6 Závěr

V práci *Multimediální hra se síťovými prvky v prostředí Windows Phone* se zkoumají možnosti vývoje aplikací pro mobilní platformu Windows Phone 7. Cílem práce bylo vytvořit multimediální síťovou hru pro více hráčů, pro mobilní i desktopové prostředí. Ohled byl kladen především na využití multimediálních a síťových prvků.

Co se týče obsahu mé práce, nejdříve byly uvedeny základní nástroje pro vývoj aplikací pro Windows Phone 7. Mezi tyto nástroje patří vývojové prostředí *Visual Studio 2010 Express for Windows Phone* a emulátor aplikací *Windows Phone Phone*. Emulátor je možné využít v případě, že nemáme k dispozici fyzické zařízení se systémem Windows Phone 7. Dále byl zmíněn jazyk *XAML*, který slouží pro grafický návrh aplikací. Také byly zmíněny frameworky *Silverlight* a *XNA*. První z nich se používá pro aplikace nenáročné na grafiku. Pro tyto účely slouží naopak druhý framework, který je specializován především na vývoj grafických her.

Poté byly uvedeny základní prvky aplikací pro Windows Phone 7 a popis, jak se s danými prvky pracuje. Mezi tyto prvky patří například navigace v aplikaci, práce se sítí, práce se senzory nebo způsob, jakým lze číst kontakty v telefonu. Tato část zahrnuje také ukázky použití prvků a ukázkové zdrojové kódy.

Další část práce se týká přímo frameworku *XNA*. Zde je popsána především struktura frameworku a jsou rozebrány hlavní řídící metody hry.

V následující části je popsána realizace vlastní aplikace, tedy síťové hry. Hra byla vytvořena pro mobilní i pro desktopové prostředí. Obě verze hry byly napsány v jazyce *C#*, pomocí frameworku *XNA*. Při realizaci vlastní hry byla zvolena hra v reálném čase, konkrétně ping pong. Hra funguje tak, že jeden z hráčů založí hru a druhý hráč se k němu připojí, poté může zakládající hráč započít hru.

V mobilní aplikaci je ukázáno použití základních multimediálních prvků jako obrázky nebo zvuky. Bylo také předvedeno načtení a použití písma pro vykreslování hlášek ve hře. Aplikace také zahrnuje použití dotykových gest. Pro práci s položkami, které lze vybrat dotykem byla vytvořena speciální třída, ulehčující výběr položek např. v menu. Síťová komunikace je řešena pomocí UDP multicastu. Při použití tohoto způsobu síťové komunikace musí být hráči připojeni na stejné Wi-Fi síti. Hráči mohou hrát hru jen s pomocí mobilních zařízení. Odpadá tedy potřeba serveru, který by běžel na desktopovém zařízení. Herní síťový klient byl jak pro mobilní prostředí, tak pro desktopové navržen tak, aby bylo možné s jeho pomocí vyvíjet další síťové hry, v reálném čase nebo tahové. Bylo předvedeno také použití lokálního úložiště aplikace *Isolated Storage* pro ukládání herních předvoleb. Při testování aplikace bylo využívánol dvou paralelně běžících emulátorů.

Desktopová verze aplikace je téměř totožná s mobilní verzí. Liší se v použití základních síťových prostředků pro komunikaci, způsobem ovládání a úložištěm pro herní předvolby.

Přehled zkratek

ID	*Identifikátor*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UI	*User Interface*
XAML	*Extensible Application Markup Language*
XML	*Extensible Markup Language*

Tabulka 6.1: Přehled zkratek použitých v práci

Literatura

[1] PAUL, Ian. *PCWorld – Europe to Get Windows Phone 7 Oct. 21, Ahead of US, says Report* [online]. 2010 [cit. 2012-08-02]. Dostupné na WWW: <http://www.pcworld.com/article/206235/europe_to_get_windows_phone_7_oct_21_ahead_of_us_says_report.html>.

[2] BRIGHT, Peter. *Arstechnica – Windows Phone 7 Series in the Enterprise: not all good news* [online]. 2010 [cit. 2012-06-19]. Dostupné na WWW: <http://arstechnica.com/information-technology/2010/03/windows-phone-7-series-in-the-enterprise-not-all-good-news>.

[3] HERRMAN, John. *Gizmodo – What Windows Phone 7 Could Have Been* [online]. 2010 [cit. 2012-06-19]. Dostupné na WWW: <http://gizmodo.com/5480387/what-windows-phone-7-could-have-been>.

[4] MINIMAN, Brandon. *Pocketnow – Thoughts on Windows Phone 7 Series* [online]. 2010 [cit. 2012-06-19]. Dostupné na WWW: <http://pocketnow.com/thought/thoughts-on-windows-phone-7-series-btw-photon-is-dead>.

[5] *Mobiletechworld – Steve Ballmer wishes Windows Mobile 7 had already launched* [online]. 2009 [cit. 2012-06-19]. Dostupné na WWW: <http://www.mobiletechworld.com/2009/09/24/steve-ballmer-wishes-windows-mobile-7-had-already-launched-but-they-screwed-up>.

[6] *Microsoft – Windows Phone update history* [online]. c2012 [cit. 2012-08-01]. Dostupné na WWW: <http://www.microsoft.com/windowsphone/en-us/howto/wp7/basics/update-history.aspx>.

[7] MOLEN, Brad. *Engadget – Windows Phone 7.5 Mango in-depth preview* [online]. 2011 [cit. 2012-08-03]. Dostupné na WWW: <http://www.engadget.com/2011/06/27/windows-phone-7-5-mango-in-depth-preview-video>.

[8] GILSON, David. *All About Windows Phone – Exploring Windows Phone 7.5 Refresh* [online]. 2012 [cit. 2012-08-03]. Dostupné na WWW: <http://allaboutwindowsphone.com/features/item/15351_Exploring_Windows_75_Refresh.php>.

[9] BRANSCOMBE, Mary. *Techradar – Windows Phone 8 release date and latest details* [online]. 2012 [cit. 2012-08-02]. Dostupné na WWW: <http://www.techradar.com/news/phone-and-communications/mobile-phones/windows-phone-8-release-date-and-latest-details-1065086>.

[10] BELFIORE, Joe. *Windows Phone Blog – Announcing Windows Phone 8* [online]. 2012 [cit. 2012-08-02]. Dostupné na WWW: <http://windowsteamblog.com/windows_phone/b/windowsphone/archive/2012/06/20/announcing-windows-phone-8.aspx>.

[11] MILES, Stuart. *Pocket-lint – What's new in Windows Phone 7.8?* [online]. 2012 [cit. 2012-08-05]. Dostupné na WWW: <http://www.pocket-lint.com/news/46232/windows-phone-7-8-new-features>.

[12] *Windows Phone Development* [online]. c2012 [cit. 2012-01-06]. Dostupné na WWW: <http://msdn.microsoft.com/en-us/library/ff402535(v=vs.92)>.

[13] *App Hub – Windows Phone and Xbox LIVE Indie Games Development* [online]. c2012. Dostupné na WWW: <http://create.msdn.com/en-US>.

[14] *GARVIS Solutions – Windows Phone* [online]. c2012 [cit. 2012-08-07]. Dostupné na WWW: <http://wp7.garvis.cz/tema-zaklady.html>.

[15] Charles Petzold – *Programming Windows Phone*, 2010, ISBN: 978-0-7356-4335-2.

Seznam obrázků

Seznam tabulek

Přílohy

Příloha 1 – Uživatelská příručka

Mobilní aplikaci můžeme spustit ve vývojovém studiu v emulátoru. Pokud vlastní uživatel vývojářský účet, může testovat aplikaci na fyzickém zařízení nahráním aplikace (balíčku XAP) do zařízení.

Desktopová aplikace se spouští souborem */PingPong/PingPong/bin/x86/Debug/PingPong.exe*, pokud se nacházíme v kořenové složce projektu.

Po spuštění hry se zobrazí menu s položkami, které lze na mobilním zařízení vybrat dotykem. V desktopové verzi aplikace se položky vybírají písmeny. Jaké písmeno reprezentuje příslušnou položky je značeno závorkami, např. (N)EW GAME.

- **NEW GAME** - založí novou hru.
- **JOIN GAME** - připojí se ke stávající hře.
- **OPTIONS** - volby hry.

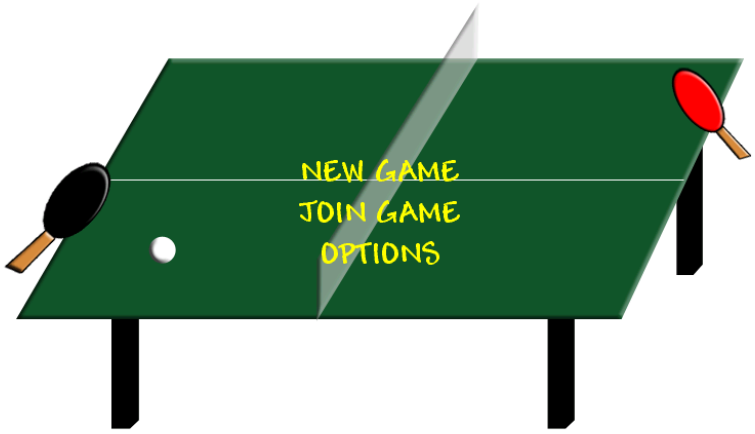
V případě založení nové hry počkáme, až se na displeji zobrazí jméno druhého hráče, který se připojil. Poté můžeme spustit hru volbou **START GAME**.

Pokud se připojujeme ke hře, vyčkáme na zobrazení serverů, jeden zvolíme a počkáme na zahájení hry.

V mobilním prostředí se hra ovládá přetahováním pálek horizontálním směrem. V desktopové verzi se pálky ovládají šipkami doprava a doleva. V případě, že nestačí odehrát míček včas, jeho oponent získává 1 bod. Hra končí v okamžiku, kdy jeden z hráčů dosáhl 10ti bodů.

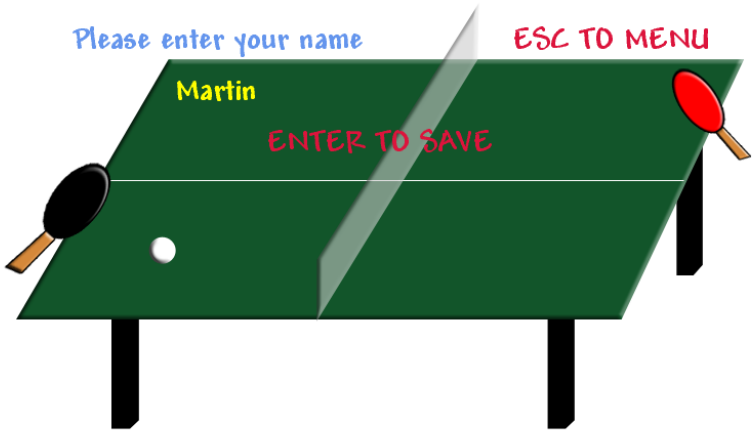
Ve volbách hry můžeme výběrem položky **SOUND** vypnout nebo zapnout zvuk. Můžeme také nastavit hráči jméno výběrem **PLAYER NAME**.

Snímky různých fází hry



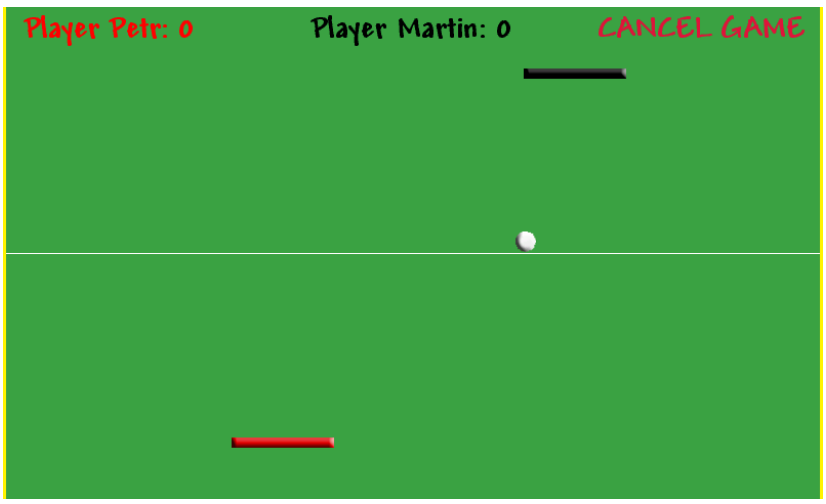
Obrázek 6.1: Menu hry - mobilní aplikace

Obrázek 6.2: Možnosti hry - mobilní aplikace



Obrázek 6.3: Zadávání jména hráče - desktopová aplikace

Obrázek 6.4: Server připraven na start hry - desktopová aplikace



Obrázek 6.5: Průběh hry - mobilní aplikace

Příloha 2 – Publikace mobilní aplikace na Marketplace

Následující informace o publikaci aplikace na *Marketplace* byly čerpané ze zdroje [14]. Máme-li připravenou a otestovanou aplikaci, můžeme ji publikovat na *Marketplace*, odkud bude k dospozici ke stažení. Pro publikaci aplikací na *Marketplace* je nutné mít založený vývojářský účet, který si lze vytvořit na webu APP HUB - *http://create.msdn.com*. Za získání účtu je nutné platit poplatek, který činí 99 USD na rok. Pro studenty, kteří využíjí program *DreamSpark (www.dreamspark.com)* je však účet zdarma. Aplikace musí splňovat několik požadavků tak, aby v poslední fázi publikace prošla certifikací.

Příprava aplikace k publikaci

V aplikaci je třeba mít nejdříve připraveny:

- **Ikonu aplikace** - ikona zobrazená v seznamu aplikací s rozměrem 62x62 pixelů. Používá se formát PNG a nachází se v kořenovém adresáři aplikace. V této práci má název *PhoneGameThumb.png*.

- **Dlaždici aplikace** - větší ikona o rozměrech 173x173 pixelů. Ikona je zobrazena na úvodní obrazovce v případě, že si ji tam uživatel umístí pro rychlý přístup. Používá se formát PNG. V této práci má název *Background.png*.

- **Startovací obrázek** - jedná se o obrázek, který se zobrazí při startu aplikace. Některé aplikace se mohou při startu načítat déle, proto je vhodné zobrazit při startu tento obrázek. Jeho název je *SplashScreenImage.jpg* a pokud se nachází v kořenovém adresáři, zobrazí se při startu. Délka startu aplikace je jeden z technických požadavků na certifikaci aplikace, délka nesmí přesáhnout 5 sekund.

- **Manifest aplikace** - soubor *WMAppManifest.xml*, který se nachází ve složce *Properties* v kořenovém adresáři. Jeho obsahem je XML specifikace pro nasazení aplikace. Obsahuje parametr *ProductId*, což je jednoznačný identifikátor aplikace na Marketplace. Dále obsahuje specifikaci ikon, které se v aplikaci použijí. Další částí manifestu je část *Capabilities*, jež určuje, jaké vlastnosti telefonu aplikace využívá. Jedná se např. o senzory, GPS nebo síťování. Je možné ponechat seznam všech možností s tím, že opravdu využité vlastnosti se zjistí při certifikaci.

Pro nasazení aplikace je také důležité určit lokalizaci aplikace. Aplikace může být lokalizována do více jazyků, ale je třeba nastavit jeden tzv. neutrální jazyk - základní jazyk celé aplikace. Jazyk lze ve Visual Studiu nastavit kliknutím pravého tlačítka na projekt a zvolením *Properties -> Application -> Assembly Information...*.

Nyní je potřeba vytvořit XAP balíček, který se zkompiluje v režimu *Release*. Ve vývojovém studiu se tedy nastaví jako výstupní konfigurace možnost *Release* a poté se aplikace zkompiluje. Výsledný balíček se bude nacházet v adresáři **Bin/Release/*NazevAplikace.xap***.

Pro úspěšné nahrání aplikace na *Marketplace* je také třeba věnovat pozornost dalším doplňkovým zdrojům. Tyto zdroje se týkají vizuálních prvků a slouží zejména pro propagaci aplikace. Vyžadované grafické prvky:

- **Malý obrázek pro mobilní Marketplace** - obrázek použitý v mobilní aplikaci Marketplace, formát obrázku je PNG, rozměr 99x99 pixelů.

- **Velký obrázek pro PC Marketplace** - obrázek použitý v aplikaci Zune, formát obrázku je PNG, rozměr 200x200 pixelů.

- **Snímky z aplikace** - jedná se o obrázky z reálného používání apliakce, sloužící jako ukázka. U každé aplikace musí být nahrán minimálně 1, maximálně 8. Pro tvorbu těchto snímků lze využít *Windows Phone Emulator*.

Publikační proces
Kroky vedoucí k publikaci aplikace:

- Přihlášení po vývojářský účtem na *http://create.msdn.com* a zvolením *Submit a new app!*.

- V prvním kroku se nahraje balíček XAP. Poté se aplikaci nastaví další údaje jako její jméno nebo verze. Také lze určit, jestli se má aplikace publikovat na veřejný *Marketplace*, tzn. že aplikace bude dostupná pro kohokoliv nebo jestli bude aplikace umístěna na testovací *Marketplace*. Tato možnost umožní aplikaci nabídnout po dobu 90 dní vybraným testerům. Nakonec je také možnost určit technické výjimky, které nesplňují certifikaci, ale přesto chcete aplikaci zveřejnit.

- V dalším kroku probíhá specifikace dalších údajů jako např. popis aplikace nebo její klíčová slova. Nahrávají se zde také doplňkové zdroje.

- V následujícím kroku se určuje cena aplikace (aplikace může být však zdarma). Také se zde určuje, pro který geografický region bude aplikace určena.

- Posledním krokem je zvolení, co se s aplikací stane po certifikaci, jestli ji chcete publikovat ručně nebo se tak má stát automaticky. Můžete zde také zanechat poznámky pro testery.

Po dokončení jednotlivých kroků je zahájen certifikační proces, skládající se z následujících požadavků:

- Požadavky na aplikaci - vyžadované vlastnosti a funkce aplikace, zde se bere ohled na technické řešení, obchodní model a obsah aplikace.

- Požadavky na obsah - určují pravidla pro používání ochranných známek, vulgárních výrazů, skryté reklamy či reklamy na speciální typy odvětví a zboží.

- Požadavky na publikaci aplikací - zabývají se především obsahem balíku XAP a doplňkových zdrojů.

- Požadavky technického řešení - nejvíce ovlivňující vývojáře, zabývající se způsobem využívání zdrojů, interakce s uživatelem, atd.

- Doplňkové požadavky - další dodatečné požadavky nebo doporučení pro zlepšení efektivity a bezpečnosti aplikace.

Po úspěšné certifikaci aplikace bude vývojář informován e-mailem a aplikace umístěna na *Marketplace* buď automaticky nebo tak učiní vývojář ručně. Na vývojářském účtu je možné sledovat statistiky stahování aplikace.