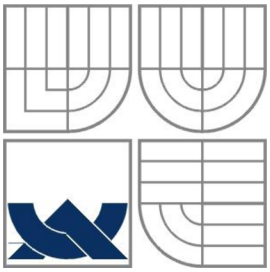
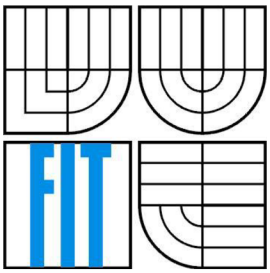


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

HRA MRAVENCI PRO MOBILNÍ ZAŘÍZENÍ

ANTZ GAME FOR MOBILE DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

PETR HOVORKA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ZBYNĚK KŘIVKA, PH.D.

BRNO 2009

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2008/2009

Zadání bakalářské práce

Řešitel: **Hovorka Petr**
Obor: Informační technologie
Téma: **Hra Mravenci pro mobilní zařízení**
Kategorie: Uživatelská rozhraní

Pokyny:

1. Seznamte se s různými typy mobilních zařízení a korespondujícími vývojovými platformami (jazyky, vývojová prostředí, přenositelnost).
2. Navrhněte multiplatformní realizaci hry Mravenci (včetně užitelského rozhraní, hratelnou offline, inspirujte se existující hrou pro stolní počítače) jak pro jednodnoho, tak pro více hráčů.
3. Hru pro vybranou množinu mobilních zařízení implementujte a řádně otestujte. Navrhněte budoucí rozšíření.
4. Na základě získaných zkušeností klasifikujte a diskutujte konkrétní i obecné problémy provázející vývoj mobilních aplikací a jejich možná řešení.

Literatura:

- Lacko, L.: *Programujeme mobilní aplikace ve Visual Studiu .NET*, Computer Press, 2004.
- Tidwell, J.: *Designing Interfaces*, 2005.
- *Webové stránky hry Mravenci*, http://www.gemtree.cz/program.htm, [online].

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Křivka Zbyněk, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2008
Datum odevzdání: 20. května 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Bakalářská práce popisuje problematiku vývoje her pro mobilní zařízení. Obsahuje popis dnes nejvíce používaných platforem mobilních zařízení a snaží se vybrat jednu z nich, vhodnou pro vývoj jednoduché karetní hry Mravenci. Dále popisuje návrh a implementaci této hry na platformě Windows Mobile v programovacím jazyce C#. Podstatná část práce se zabývá tvorbou uživatelského rozhraní. V závěru jsou diskutována možná budoucí rozšíření hry a obecné problémy provázející vývoj her pro mobilní zařízení.

Klíčová slova

Mravenci, hra, mobilní zařízení, platforma, vývoj, Windows Mobile, C#.

Abstract

This Bachelor's thesis describes the basics of game development for mobile devices. It contains a description of common mobile platforms and tries to choose one of them, suitable for developing of simple card game Antz. Next it describes design and implementation of this game on platform Windows Mobile and programming language C#. Major part of this document dwells on graphical user interface development. At the end it discusses future extensions of the game and common problems of game development for mobile devices.

Keywords

Antz, game, mobile device, platform, development, Windows Mobile, C#.

Citace

Hovorka Petr: Hra Mravenci pro mobilní zařízení. Brno, 2009, bakalářská práce, FIT VUT v Brně.

Hra Mravenci pro mobilní zařízení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Zbyňka Křivky, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Petr Hovorka
20. května 2009

Poděkování

Tímto bych chtěl poděkovat mému vedoucímu Ing. Zbyňku Křivkovi, Ph.D., který mi poskytl cenné rady během mé práce.

Obsah

Úvod

V této práci se autor snaží o rozebrání problematiky vývoje her pro mobilní zařízení. Díky neustálému technickému pokroku se na trhu objevuje stále větší počet mobilních zařízení, která slouží jako univerzální komunikační prostředky a usnadňují lidem řešení každodenních situací. Díky relativně nízkým cenám těchto zařízení je jejich vlastnictví dnes celkem běžnou záležitostí. Téměř každý člověk dnes vlastní obyčejný mobilní telefon nebo komunikátor, schopný vykonávat nejen základní funkce, jako je telefonování, organizování času a čtení elektronické pošty, ale také umožňuje spouštět pokročilejší aplikace.

Mobilní zařízení jsou tedy určená k neustálému nošení u svého majitele a právě na tato zařízení se zaměřil i trh počítačových her. Proto se v této práci autor věnuje vývoji her pro mobilní zařízení z pohledu programátora počítačových aplikací a nabízí řešení několika problémů souvisejících s jejich vývojem.

V práci autor uvádí příklad různých univerzálních vývojových platforem pro mobilní zařízení a zaměřuje se zejména na platformu Windows Mobile.

Dále se zabývá návrhem a implementací hry Mravenci od Ing. Miroslava Němečka, původně navržené pro osobní počítač se systémem Windows.

Nakonec se autor snaží shrnout problémy související s vývojem her a diskutuje možné další rozšíření hry.

V práci se vyskytují názvy některých objektových tříd, metod, vlastností a objektů implementovaných ve hře. Tyto názvy jsou odlišené použitím fontu `psacího stroje`.

1 Srovnání platforem pro vývoj mobilních aplikací

V této kapitole se seznámíme s pojmy *mobilní platforma*, *mobilní zařízení* a přiblížíme si nejpoužívanější platformy pro vývoj mobilních aplikací v dnešní době.

1.1 Mobilní platformy

Mobilní platformou se označuje souhrn hardwarových a softwarových prostředků, použitých pro vývoj a provoz mobilních aplikací.

Hardwarovým prostředkem se rozumí přenosné, většinou víceúčelové, elektronické zařízení kompaktních rozměrů. Může to být prostý mobilní telefon, „chytrý“ mobilní telefon s operačním systémem, PDA[1], herní konzola, kompaktní počítač třídy PC, nebo nějaké jiné zařízení schopné spouštět uživatelské programy.

Softwarovým prostředkem chápejme nějaký programový základ, operační systém či vývojové prostředí potřebné pro realizaci a provoz mobilní aplikace.

1.2 Výčet nejrozšířenějších mobilních platforem

V této kapitole uvedu příklad tří nejvíce rozšířených platforem použitelných pro vývoj a provoz mobilních aplikací.

1.2.1 Java ME (Micro Edition)

Velice rozšířená platforma pro provoz mobilních aplikací převážně na mobilních telefonech. Programy se kompilují do kódu, který běží ve virtuálním stroji Java uvnitř mobilního zařízení. Virtuální stroj představuje abstrakční vrstvu mezi programem a fyzickým zařízením. Je tak zajištěna dobrá přenositelnost programů mezi zařízeními.

Java ME se dělí na několik profilů podle zaměření a nároků na výkon. Pro nás je asi nejdůležitější profil MIDP[2]. Je to profil navržený pro mobilní telefony. Obsahuje základní knihovny pro práci s grafickým uživatelským rozhraním a knihovny pro 2D grafiku.

---

[1] Personal Digital Asistent - osobní digitální pomocník; malý kapesní počítač.

[2] Mobile Information Device Profile je aplikační rozhraní J2ME, které definuje, jakým způsobem softwarové aplikace spolupracují s mobilními telefony.

Dále existuje mnoho rozšíření, která obsahují rozhraní pro bezdrátovou komunikaci, grafické knihovny, multimediální knihovny, rozhraní pro webové služby, GPS[3] navigaci a jiné. Tato rozšíření jsou ovšem specifická pro konkrétní modely mobilních zařízení a programy se tak musejí kompilovat zvlášť pro každý model. To znesnadňuje vývoj pokročilých aplikací pro tuto platformu.

Vývojových prostředí je několik. Ta nejpoužívanější jsou Netbeans a Eclipse. Jsou to multiplatformní vývojové kity psané v jazyce Java. K jejich spuštění je nutné mít nainstalované Java Runtime Environment (běhové prostředí Java).

1.2.2 Symbian

Platforma vévodící trhu mobilních telefonů. Symbian je operační systém převládající na mobilních telefonech značky Nokia, Samsung, Sony Ericsson a Motorola. Je neustále vyvíjen a těší se silné podpoře ze strany firmy Nokia a několika menších výrobců mobilních telefonů. Umožňuje provoz širokého spektra programů a aplikací díky kvalitnímu API[4] [3].

Hlavním programovacím jazykem je C++, ale díky široké škále knihoven je možno spouštět i programy napsané v jazyce Python, Java, Perl, Simkin a OPL. Programy se kompilují do spustitelné verze pro operační systém Symbian, nebo jsou, v případě skriptovacích jazyků, interpretovány příslušnými interprety.

Jako vývojové prostředí je nabízen balík nástrojů Carbide.c++. Je dodáván v několika edicích, placených i zdarma. Carbide.c++ je určen pro systém Windows. Aplikace lze vyvíjet i v alternativních vývojových balících, nebo přímo v textovém editoru s kompilací přes program make. Zásuvný modul Carbide.c++ lze použít v prostředí Microsoft Visual Studio a pro Apple OS X existuje balík XCode IDE. V nedávné době také přibyla podpora knihoven uživatelského rozhraní Qt. Jsou to multiplatformní knihovny pro tvorbu uživatelských rozhraní, které jsou podporované na široké škále platforem.

1.2.3 Windows Mobile

Druhá nejrozšířenější platforma pro vývoj a provoz mobilních aplikací. Stejně jako Symbian je Windows Mobile operačním systémem. Na rozdíl od Symbianu se však Windows Mobile nespecializuje pouze na mobilní telefony, ale najdeme jej i na zařízeních typu PDA, komunikátorech nebo na specializovaných průmyslových zařízeních pro monitorování a sběr dat v terénu. Zařízení Windows Mobile se dají rozdělit do dvou kategorií - Pocket PC a Smartphone.

---

[3] Global Positioning System je vojenský polohový družicový systém provozovaný Ministerstvem obrany Spojených států amerických.

[4] API je zkratka anglických slov **application programming interface,** což znamená rozhraní pro programování aplikací.

Pocket PC je zařízení určené především pro kancelářskou práci v terénu. Základem je dotyková obrazovka, několik ovládacích tlačítek a někdy je součástí zařízení i zabudovaná klávesnice.

Smartphone, neboli tzv. „chytrý“ mobilní telefon, je zařízení, jehož základem je uživatelské rozhraní přizpůsobené pro ovládání zabudovanou klávesnicí. Obrazovka je v drtivé většině bez dotykového ovládání, klávesnice číselná nebo znaková, zabudovaná do čela přístroje jako u mobilního telefonu. Ovládací prvky na obrazovce jsou přizpůsobeny pro ovládání pomocí klávesnice a někdy je přístroj vybaven i vysouvací rozšířenou klávesnici pro pohodlné vkládání textu.

Windows Mobile má propracované API a lze používat i specializované knihovny pro práci s multimédii, internetem, bezdrátovou komunikací a GPS navigací. Existuje i specializovaná podpora trojrozměrné akcelerované grafiky. Programování aplikací pro Windows Mobile se velice přibližuje k programování aplikací pro systém Windows na osobních počítačích. Díky rozhraní .NET Compact Framework je možno používat stejné programovací postupy známé ze „stolních“ aplikací.

Vývojové prostředí pro operační systém Windows Mobile se nazývá Visual Studio a je dodáváno společností Microsoft. Visual Studio je určeno pro operační systém Windows a dodává se v placených edicích i zdarma. [6]

2 Výběr cílové platformy

Při výběru cílové platformy se zaměříme na přenositelnost aplikace mezi co největším počtem mobilních zařízení. Dalším kritériem výběru je dostupnost a kvalita vývojového prostředí a nástroje na ladění aplikace (debugger). Pokud by nás zajímala pouze přenositelnost aplikace, nejlepší volbou by byla kombinace programovacího jazyka C++, knihovny pro uživatelské rozhraní Qt a některé z volně šiřitelných vývojových prostředí.

Podle předešlých zkušeností a také na doporučení vedoucího práce však autor ke své práci zvolil programovací jazyk C#, knihovny .NET Compact Framework a vývojové prostředí Visual Studio 2008. Debugger a nástroje pro komunikaci s mobilním zařízením lze získat z vývojového balíku Windows Mobile 5/6.5 Standard a Professional. Autorem v praxi osvědčený postup je střídavé použití různých verzí knihoven pro kategorie Smartphone i PocketPC, což je užitečné zejména při testování kompatibility kódu. Je tím myšlena kompilace kódu pro určitou verzi platformy Windows Mobile a jeho testování na jiné platformě. Tímto postupem se dá lehce ověřit, zda bude výsledný zkompilovaný kód spustitelný bez problémů na celé množině cílových zařízení. Podrobný výčet použitých nástrojů je uveden v příloze na disku DVD.

2.1 Parametry zařízení na platformě Windows Mobile

Při návrhu hry pro mobilní zařízení je potřeba zjistit několik důležitých parametrů o cílové platformě. Jsou to parametry obrazovky a možnosti zobrazení grafických ovládacích prvků, parametry vstupních zařízení, možnosti zvukového výstupu, a na konec možnosti konektivity pro hru dvou hráčů na dvou mobilních zařízeních.

2.2 Parametry obrazovky

Velice důležitým parametrem pro grafickou reprezentaci hry na obrazovce mobilního zařízení je její rozlišení, poměr stran a možnost změny orientace obrazovky. Tyto parametry jsou na dnešních mobilních zařízeních velice různorodé.

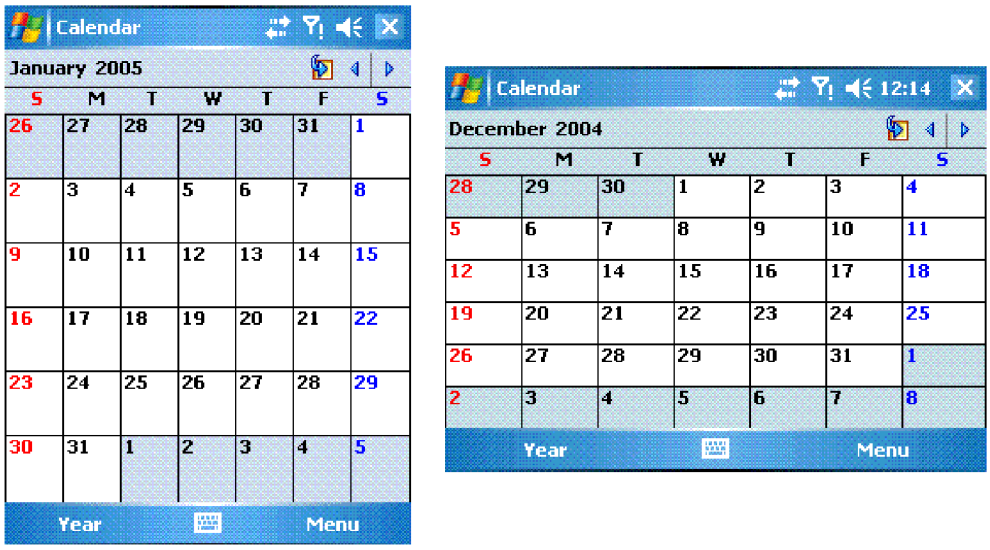
Pokud chceme dodržet dostatečnou kompatibilitu, musí být hra schopna zobrazit grafické prvky jak na nejmenším, tak na největším možném rozlišení obrazovky. Jako nejmenší rozlišení je uvažována hrací plocha o rozměrech 176 x 220 obrazových bodů, největší potom plocha o rozměrech 480 x 800 obrazových bodů. Jsou to rozlišení převzatá ze specifikací dnes prodávaných komunikátorů s operačním systémem Windows Mobile (dříve Pocket PC) v různých verzích, z literatury [4]

a on-line dokumentace [5], [6]. Poměr stran se většinou odvíjí od rozlišení obrazovky a bývá 3:4, 4:5, 1:1 nebo jiné. To je důležité kvůli uspořádání jednotlivých prvků na obrazovce. Seznam běžných rozlišení je uveden v Tab. 2.1.

Poměr stran 5:3	Poměr stran 5:4	Poměr stran 4:3	Poměr stran 1:1
800x480	176x220	320x240	240x240
		640x480	320x320
			480x480

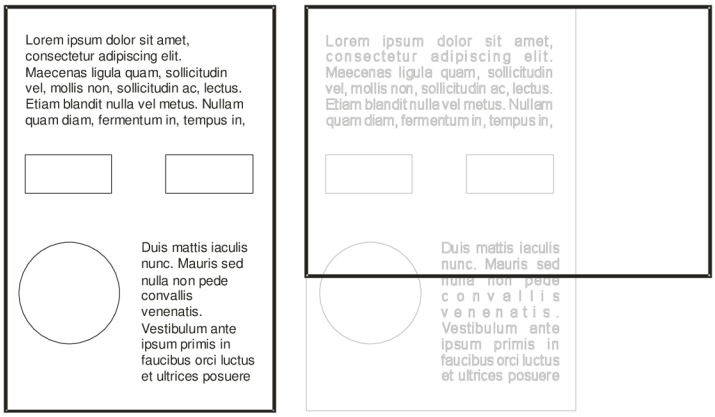
Tab. 2.1: Seznam dnes běžně používaných rozlišení obrazovek mobilních zařízení. Rozlišení jsou uvedena v počtech obrazových bodů v horizontálním a vertikálním směru.

Další vlastností operačního systému Windows Mobile je možnost změny orientace obrazovky. Mnoho mobilních zařízení mění orientaci obrazovky například při vysunutí klávesnice nebo při změně pozice zařízení v prostoru. Pokud by tato událost nebyla brána v úvahu, obraz by mohl být po této změně deformovaný nebo by se nezobrazil celý. Otočení obrazu je totiž jen v částečné režii operačního systému. Aplikaci je při této události pouze oznámena změna orientace obrazovky a je změněn souřadnicový systém tak, že počátek souřadnic je vždy v levém horním rohu obrazovky. Proto mohou následně některé prvky okna zasahovat mimo obrazovou plochu a musejí být aplikací znovu umístěny na vhodnou pozici. V souvislosti s orientací obrazovky se budou používat anglické termíny pro označení orientace obrazovky. Termínem *portrait* bude označena orientace obrazovky na výšku, termínem *landscape* orientace obrazovky na délku. Režimy zobrazení jsou ilustrované na Obr. 2.1.



Obr. 2.1: Režimy orientace obrazovky. Vlevo je zobrazen režim portrait, vpravo režim landscape.

Problém změny orientace obrazovky je ilustrován na Obr. 2.2. Vlevo je obrazovka v režimu portrait s korektně zobrazeným obsahem. Vpravo je obrazovka po změně orientace v režimu landscape se stejným obsahem, avšak bez úpravy na novou orientaci obrazovky. Obsah je viditelně oříznutý a nevykrývá celou plochu obrazovky.



Obr. 2.2: Ilustrace problému zobrazení při změně orientace obrazovky. Vlevo je obrazovka s orientací portrait, vpravo s orientací landscape.

2.3 Možnosti vstupních zařízení

Vstupní zařízení jsou velice důležitá z hlediska ovládání programu. Uživatelské rozhraní musí být přizpůsobeno jednak pro ovládání dotykovou obrazovkou, tak i integrovanou klávesnicí. Pokud má mobilní zařízení vestavěnou hardwarovou klávesnici, obvykle je to buď klávesnice plnohodnotná (Obr. 2.3), nebo numerická jako v mobilním telefonu (Obr. 2.4).

Obr. 2.3: Mobilní zařízení s plnohodnotnou klávesnicí (qwerty)

Obr. 2.4: Mobilní zařízení s numerickou klávesnicí

Při ovládání klávesnicí je vhodné rozmístit ovládací prvky programu do tvaru reprezentujícího umístění tlačítek na klávesnici. Pokud je například standardní klávesnice umístěna dole, podél delší strany obrazovky, ovládací prvky na obrazovce je vhodné umístit na stejnou stranu obrazovky.

Na tento problém se autor zaměřil při návrhu uživatelského rozhraní hry. Ovládání dotykovou obrazovkou by mělo brát ohled nejen na ovládání perem, ale také na možnost ovládání prstem. To znamená zvětšit velikost ovládacích prvků programu na úkor informačních ikon.

2.4 Možnosti zvukového výstupu

Platforma Windows Mobile ve verzi 5 spolu s knihovnami .NET Compact Framework neobsahují nativní podporu zvukového výstupu. V předešlé verzi 2003 této platformy byl zvukový výstup řešen pomocí rozhraní DirectSound. To však bylo ve verzi 5 odstraněno a až ve verzi 6 bylo implementováno jednoduché API pro přehrávání běžně podporovaných formátů, pro které jsou v mobilním zařízení dostupné kodeky[5] (MP3, WMA a MID). Z důvodu zachování kompatibility je proto nutné pro přehrávání zvuku použít systémové volání z knihovny *CoreDll.DLL* umístěné v systému. Přesný popis tohoto volání lze nalézt v dokumentaci v [6].

2.5 Možnosti konektivity

Pokud chceme hru navrhnout s možností hry více hráčů na více zařízeních, je nutné zjistit možnosti spojení hráčů mezi sebou. V dnešní době jsou již mobilní zařízení natolik vyspělá, že i levnější modely disponují bezdrátovými technologiemi Bluetooth a připojením k síti internet. Je proto možné se později zabývat realizací propojení mobilních zařízení přes tyto komunikační kanály.

2.5.1 Bluetooth

Bluetooth je bezdrátová technologie původně vyvinutá jako náhrada za kabely při propojování zařízení jako jsou mobilní telefony, headsety a počítače. Od té doby se Bluetooth rozvinul v bezdrátový standard pro připojení elektronických přístrojů. S touto technologií nejen, že nejsou potřeba kabely k propojení zařízení, ale spojení probíhá automaticky a bez nutnosti instalovat softwarové ovladače. [1][2]

Technologie bluetooth přímo vybízí k použití při hraní her více hráčů. Navázat spojení mezi mobilními zařízeními je většinou jednoduché a spojení představuje spolehlivý komunikační kanál na vzdálenost několika metrů. Navíc Bluetooth dnes najdeme téměř na každém mobilním zařízení. Proto se dá dobře využít při návrhu hry pro více hráčů.

---

[5] Kodek, složenina z počátečních slabik slov „**ko**dér a **dek**odér“; převzato z anglického **codec** analogického původu. Je to zařízení nebo počítačový program, který dokáže transformovat datový proud nebo signál.

2.5.2 Připojení k internetu

Existuje mnoho technologií pro mobilní připojení k síti internet. Nejběžnější jsou v dnešní době 2.5G-3G mobilní sítě provozované operátory mobilních telefonních sítí a sítě WiFi standardu IEEE 802.11a/b/g/n [1].

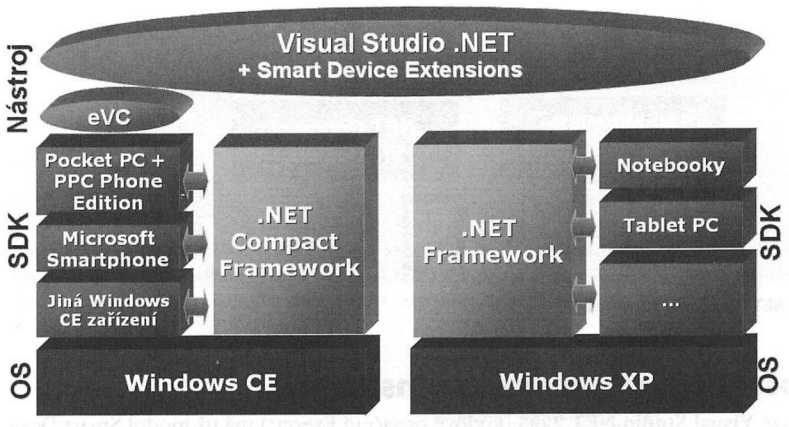
Mobilní zařízení se systémem Windows Mobile dokáže tyto technologie připojení využívat, transparentně mezi nimi přepínat a udržet tak stálé připojení k síti internet. Proto lze navrhnout internetový model hry více hráčů, kdy hru řídí vzdálený server přístupný na síti internet.

2.6 Technologická platforma .NET Compact Framework 2.0

Následující popis je citovaný z [4]. Kniha pochází z roku 2003 a některé údaje v ní uvedené jsou lehce zastaralé. Jako zevrubný popis .NET Compact Frameworku však jistě dobře poslouží.

Technologická platforma .NET Compact Framework umožňuje efektivní vývoj aplikací. Laicky bychom mohli říci, že se jedná o analogii .NET Frameworku pro desktopové PC a servery. Vysvětlit vztah mezi technologickými platformami .NET Framework a .NET Compact Framework pro našince odchovaného na množinové matematice nebude žádný problém. Stačí jediná věta: .Net Compact Framework je podmnožinou .NET Frameworku.

Slovo podmnožina znamená, že .NET Compact Framework neobsahuje všechny funkce velkého .NET Frameworku, ale pouze ty, u kterých je možné dodržet striktní kompatibilitu v rámci možností (především zobrazovacích) mobilního zařízení.



Obr. 2.5: Srovnání architektur .NET Framework a .NET Compact Framework převzaté z [4].

„Rozhraní mezi těmito dvěma softwarovými platformami je však z hlediska posuzování, zda se jedná o mobilní aplikaci či nikoliv, poměrně neostré. Za mobilní zařízení přeci považujeme také

notebooky a nejnovější počítače řady Tablet PC, a to i přesto, že na nich mohou dokonce běžet serverové operační systémy, jako jsou Windows 2000 Server nebo modernější Windows 2003 Server.

Zatímco .NET Framework se používá u operačních systémů pro servery, desktopové počítače a Tablet PC, .NET Compact Framework se používá u platforem Pocket PC, Smartphone či různých Windows CE, resp. u zařízení Windows Powered.

Výrazný rozdíl je také v požadavcích na úložnou kapacitu počítače. Zatímco .NET Compact Framework se spokojí se dvěma megabajty úložného prostrou v operační paměti (pevnými disky přístroje této třídy nedisponují), „velký" .NET Framework potřebuje více než 43 MB diskové kapacity.

Srovnání obou architektur je ilustrováno na Obr. 2.5.

3 Návrh hry Mravenci pro mobilní zařízení

V této kapitole bude vysvětlena podstata hry a základní struktura programového kódu. Autor popíše jednotlivé objektové třídy programu, jejich vzájemné vazby a nakonec popíšu implementaci vlastních prvků grafického uživatelského rozhraní.

3.1 Hra Mravenci

Hra Mravenci je tahová karetní hra dvou hráčů. Jejím původním autorem je Ing. Miroslav Němeček ze společnosti Gemtree Software s.r.o.. Hra je určena pro operační systém Windows a je nabízena na stránkách firmy jako freeware. Toto je popis hry citovaný přímo z původního programu:

> *Nikdo z černých ani červených mravenců si již nepamatuje, kdo a kdy začal jejich odvěkou válku o mraveniště „U dvou smrků“. Aby věčným soubojům učinili konec, rozhodli se postavit hrady, s jejichž pomocí by získali nadvládu nad celým územím.*
>
> *Cílem hry je postavit hrad o výšce 100, nebo zničit hrad soupeře. Levým tlačítkem myši lze vybrat kartu, pravé tlačítko kartu odloží. Každá karta spotřebuje určité množství surovin (číslo vpravo nahoře). Rychlost tvorby surovin závisí na množství týmů.*

Obr. 3.1: Ukázka herní plochy hry Mravenci od Ing. Miroslava Němečka.

Hraje se s třemi druhy karet. Jsou to karty stavební, útočné a kouzelnické. Karet je od každého druhu 10 a liší se od sebe cenou surovin a akcí, kterou ve hře provádějí.

Stavební karty staví buď hrad, nebo hradbu. Jejich cena udává kolik cihel bude tato karta stát a akce karty určuje, co se bude stavět a o kolik jednotek.

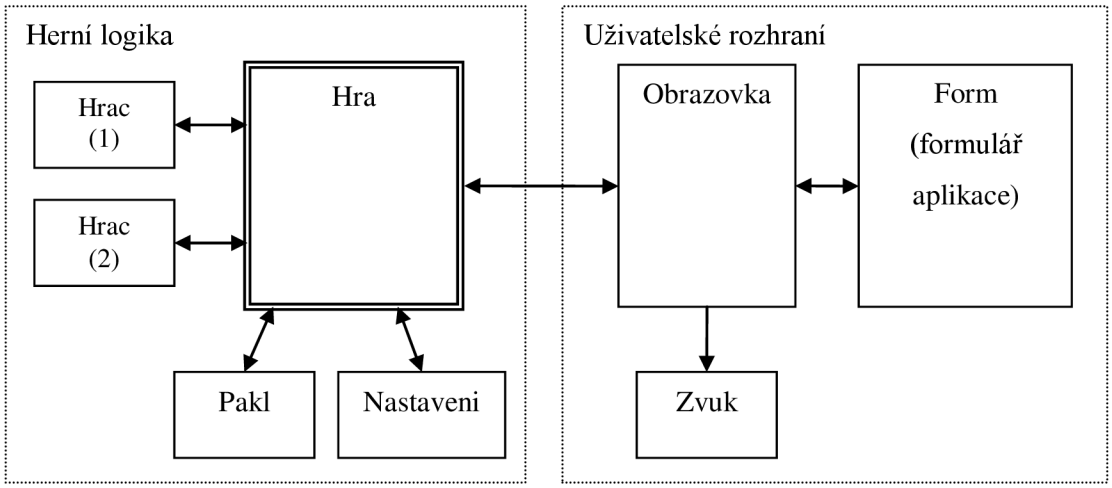
Útočné karty jsou o něco zajímavější. Ty levnější pouze útočí na hrad a hradbu soupeře, dražší naopak můžou nejen bořit hrad soupeře, ale například převádět zásoby nebo je ničit (karta *Zloděj*).

Kouzelnické karty umějí vykouzlit zásoby, ničit je soupeřovi, zaklínat soupeře (snížit všechny jeho prostředky o 1) a nakonec přičarovat či odčarovat podstatnou část hradu.

3.2 Objektový model hry Mravenci

Na začátku vývoje hry byla snaha navrhnout funkční a jednoduchou objektovou strukturu, vhodnou pro tento typ hry s možností budoucích rozšíření funkcionality bez významnějších zásahů do kódu. Programovací jazyk C# k tomuto nabízí ideální prostředek – třídy.

Na Obr. 3.2 jsou znázorněny objektové třídy a šipky nastiňující tok informací mezi nimi. Aby bylo možné zajistit dostatečnou flexibilitu programového kódu, objektové třídy byly rozděleny na dvě části. První část tvoří třídy starající se o herní logiku a druhé se starají o obsluhu uživatelského rozhraní. Toto rozdělení usnadňuje případnou budoucí implementaci hry pro jiný typ zařízení s případnými jinými možnostmi uživatelského vstupu a výstupu.



Obr. 3.2: Zjednodušený objektový model hry Mravenci.

Hlavní řídicí třídou herní logiky je třída `Hra`. Ta obsahuje nejdůležitější obslužné rutiny a zajišťuje fungování celé hry. Pracuje se dvěma instancemi třídy `Hrac`, které uchovávají informace o hráčích, jejich kartách a stavech. O rozdávání a míchání karet se stará třída `Pakl` a o nastavování parametrů hry třída `Nastaveni`.

Správu uživatelského rozhraní má na starosti třída `Obrazovka`. Ta zajišťuje vykreslování stavu hry na obrazovku, přehrávání animací a zvuků a zpracovává vstupy od uživatele. Vstupem se rozumí interakce s ovládacími prvky programu, jako jsou hrací karty, nabídka programu nebo vstup z klávesnice zařízení.

Jak probíhá hra z pohledu jednotlivých objektů? Ve vstupním bodu programu jsou vytvořeny instance tříd `Hrac`, `Obrazovka` a `Nastaveni`. Ty jsou použity pro inicializaci třídy `Hra`, která zastává roli řadiče hry.

Herní řadič určuje hráče na tahu, zajišťuje hráčům spojení s třídou `Pakl`, generující karty, a provádí změny stavových hodnot hráčů při zahrání karty. Dále předává informace o stavu hry třídě `Obrazovka`. Té jsou na základě herních událostí předávány informace o hráčích, včetně pokynů pro aktualizaci herní plochy.

3.3 Třída Hrac

Třídu `Hrac` byla vytvořena z důvodu potřeby uchovávat informace o každém hráči zvlášť. Tato třída vychází z rozhraní `IHrac`, zajišťující jednotný přístup k objektům hráčů a umožňuje budoucí rozšíření o nové typy hráčů. Například bude velice jednoduché vytvořit třídu hráče komunikujícího vzdáleně přes Bluetooth nebo síť Internet. Zatím si vystačíme s běžným typem hráče, ovládajícím hru přes obrazovku či klávesnici mobilního zařízení.

3.3.1 Rozhraní IHrac

Rozhraní `IHrac` neposkytuje žádné veřejné metody. Zprostředkovává přístup k vlastnostem jako je jméno hráče, aktuální seznam karet hráčem vlastněných, výška hradu a zdi a nakonec stav surovin a poddaných. Navíc je zde ještě jedna vlastnost s názvem `JeNaTahu`, která zjišťuje, jak její název napovídá, zda je hráč na tahu.

3.3.2 Umělá inteligence

Přímo v třídě `Hrac` je implementována umělá inteligence. Privátní metoda `TahPocitace()` obsahuje hodnotící funkci, která v případě potřeby vybere nejlepší kartu, kterou je možné v danou chvíli zahrát nebo odložit. Tato metoda je volána v případě, že je typ hráče nastaven na hodnotu `Pocitac` výčtového typu `tHrac` a je-li hráč na tahu.

Jelikož hra Mravenci je typem hry s neurčitostí, kdy dopředu nelze odhadnout tah protihráče, není použita žádná z metod umělé inteligence pro hraní her (popisy metod jsou k nalezení v [7]). Protihráči si totiž navzájem „nevidí“ do karet a tak nelze efektivně odhadnout protihráčův tah. Naopak byla snaha o minimalizaci nároků na výpočet tahu. Hra je totiž určena pro nepříliš výpočetně

zdatná mobilní zařízení a vysoké nároky na výpočetní výkon by jistě snižovaly jejich výdrž na baterie.

3.3.3 Algoritmus hodnotící funkce umělé inteligence

Do těla hodnotící funkce byl implementován následující algoritmus:

1. Pokud má hráč k dispozici alespoň jednu stavební kartu, kterou může v danou chvíli zahrát, vyber z těchto karet tu s nejvyšší cenou a zahraj ji. Pokud žádnou stavební kartu zahrát nelze, pokračuj bodem 2.
2. Pokud má hráč k dispozici alespoň jednu kartu, kterou může v danou chvíli zahrát, vyber z těchto karet tu s nejvyšší cenou a zahraj ji. Pokud žádnou kartu zahrát nemůže, pokračuj bodem 3.
3. Z karet, které má hráč k dispozici, vyber tu s největší cenou a odlož ji.

Tento algoritmus používá původní verze hry Mravenci od Ing. Miroslava Němečka. Je to algoritmus velmi efektivní a začátečník má zprvu veliké potíže nad počítačem vyhrát. Autora proto napadlo algoritmus obohatit o tři různé úrovně. Nejjednodušší úroveň byla pojmenována „Larva“, prostřední „Dělník“ a nejtěžší „Voják“. Právě nejtěžší úroveň používá výše zmíněný algoritmus, lehčí úrovně obsahují různá omezení.

U nejjednodušší úrovně bylo zvoleno maximální omezení při výběru karty a to na karty s cenou nejvýše 5. Z hratelných karet se navíc vybírá ta nejnižší a všechny karty vyšší než 5 funkce vyřadí a případně takové karty nechá odložit.

U prostřední úrovně už tak silné omezení není a pracuje se s výše zmíněným algoritmem počínaje bodem 2. Nejsou proto upřednostňovány karty stavební a protihráč s umělou inteligencí se proto nesnaží co nejrychleji postavit hrad.

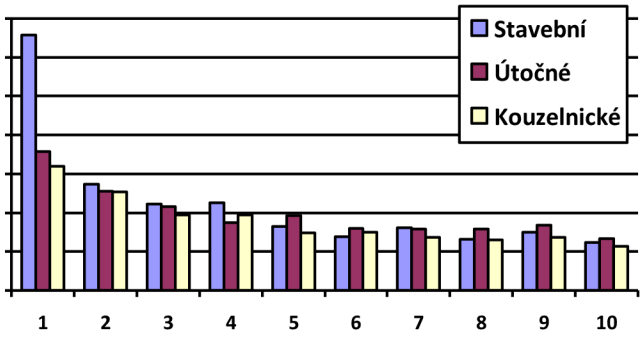
Třída `Hrac` s rozhraním `IHrac` tvoří „černou skříňku“, která nezávisle na implementaci poskytuje přístup k vlastnostem hráče. V případě hráče typu počítač navíc obsahuje umělou inteligenci využitou při výběru tažené karty.

3.4 Třída Pakl

Třída `Pakl` byla navržena na zjednodušení procesu rozdávání karet. Autorův původní záměr byl na začátku hry „zamíchat“ karty a rozdávat je střídavě hráčům. Odložené karty by se vracely zpět do balíčku a znovu rozdávaly hráčům. Bohužel se později ukázalo, že pravidla hry způsobovala opětovné rozdávání stejných karet hráčům, kteří kartu už jednou vlastnili. Nebylo tak možné ve většině kombinací rozdaných karet zajistit vyrovnaný souboj obou hráčů. Proto byl celý proces rozdávání

karet zjednodušen na pouhé generování karet, kdy už jednou zahrané nebo odložené karty jsou zahozeny.

Karty jsou generované funkcí, která zvyšuje pravděpodobnost výskytu méně hodnotných karet a „vysoké“ karty se generují méně často. Toto rozložení ilustruje histogram na Obr. 3.3.



Obr. 3.3: Histogram četnosti karet generovaných třídou Pakl.
Histogram pochází ze vzorku 300 generovaných karet.
Čísla 1-10 označují hodnotu karet.

Nejvíce se přitom generují stavební karty nižší hodnoty. Funkce je sestavena podle původního algoritmu Ing. Miroslava Němečka, obsaženého ve volně přístupném zdrojovém kódu originální hry Mravenci.

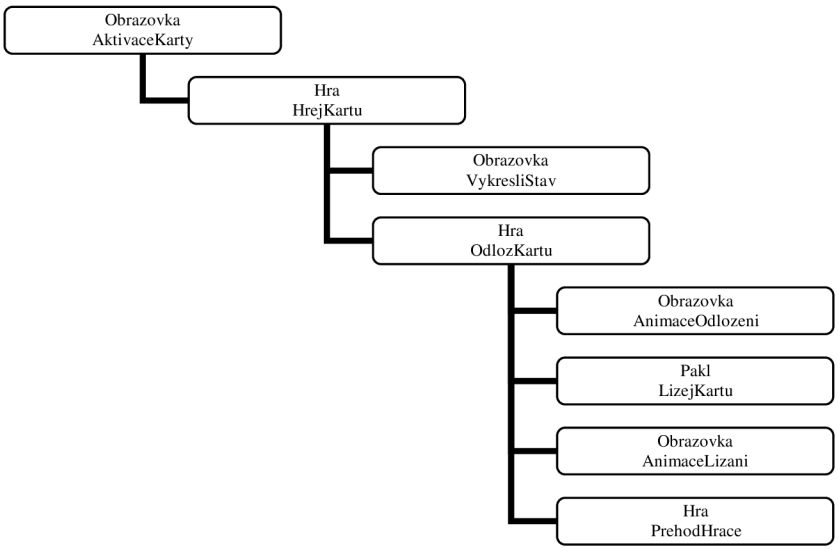
3.5 Třída Hra

Třída `Hra` tvoří společně s třídou `Obrazovka` základ herního jádra. Tím je myšlen fakt, že řídí průběh hry, vypočítává nové hodnoty stavů při zahrání karty, střídá hráče na tahu a komunikuje s třídou `Obrazovka`. Třída `Hra` obsahuje několik nejdůležitějších vlastností a metod, které stojí za to uvést a popsat tak princip celého programu.

Spuštění hry probíhá inicializací třídy Hra pomocí instancí tříd `Hrac`, `Obrazovka` a `Nastaveni`. Při inicializaci se odkazy na tyto objekty uloží do privátních proměnných a prvnímu ze dvou hráčů se nastaví vlastnost `JeNaTahu` na hodnotu `true`. Výše zmíněné odkazy jsou přístupné i zvenčí pomocí veřejných vlastností. Jakýkoliv jiný objekt tak může komunikovat skrze třídu `Hra` s jiným objektem v programu. Tohoto propojení je hojně využito při řízení běhu hry napříč celým programovým kódem. Když je inicializace u konce, čeká se na aktivaci karty, čili na volání od třídy `Obrazovka`. Právě tato třída má na starosti vstup od uživatele a spouští řídicí metody.

Na obrázku (Obr. 3.4) je uvedeno zjednodušené znázornění posloupnosti metod volaných třídou `Obrazovka` při aktivaci hrací karty uživatelem. Názvy metod jsou zčásti smyšlené a spíše

zastupují několik metod skutečných. Povšimněte si postupné střídání metod tříd `Obrazovka` a `Hra`, kterým je naznačena jejich vzájemná interakce.



Obr. 3.4: Posloupnost volaných metod, následujících po aktivaci karty uživatelem. Třída metody je vždy v horní části popisu, název metody v části dolní.

Proces zahrání karty řídí metoda `HrejKartu()`. Vstupním parametrem je index konkrétní karty hráče na tahu. Metoda ve svém těle přistupuje k vlastnosti `HracNaTahu` třídy `Hrac`, která je odkazem na objekt hráče, který právě hraje. Díky tomu, že jde o objekt, může metoda přistupovat ke všem jeho vlastnostem a pracovat tak se stavem hráčových prostředků. Může tak zjistit hodnotu vlastnosti `MuzeZahrat` objektu `HracNaTahu`, ze které zjistí, zda má hráč dostatek prostředků k zahrání karty. Pokud je prostředků dostatek, provede se na základě parametrů karty úprava prostředků hráče a případně i protihráče.

Hracích karet jsou ve hře Mravenci, jak už bylo zmíněno výše, tři druhy. Karty stavební, útočné a kouzelnické. Na zahrání útočné karty je zde zvláštní metoda `Utok()`. Princip útočné karty spočívá v boření protihráčova hradu a přednostně zdi. Metoda `Utok()` tedy v první řadě sníží příslušnou výšku zdi protihráče o hodnotu útoku karty. Pokud je tato hodnota větší, než výška zdi protihráče, je zbytek hodnoty odečten z výšky protihráčova hradu.

Po zahrání karty už zbývá jen kontrola, zda hráč na tahu nedosáhl výšky hradu 100, nebo úplně nezbořil protihráčův hrad. V obou případech by se jednalo o vítězství hráče a hra by skončila.

3.6 Třída Obrazovka a grafické uživatelské rozhraní

Třída `Obrazovka`, jak její název napovídá, se stará o vykreslování grafického uživatelského rozhraní hry na obrazovku mobilního zařízení. Její činnost není zcela autonomní, ale základní pokyny pro vykreslování jsou předávány z třídy `Hra`.

Třída má na starosti tyto úkoly:

- Vytvoření a rozmístění prvků uživatelského rozhraní na obrazovce mobilního zařízení
- Zpracování vstupu uživatele a předání zpráv třídě `Hra`
- Vykreslení aktuálního stavu hry na požádání
- Zobrazení animace karet

3.6.1 Vytvoření a rozmístění prvků uživatelského rozhraní

Jelikož obrazovky mobilních zařízení můžou mít různá rozlišení a poměr stran (viz. kapitola 2.2), musela být vytvořena metoda, která by zajistila správné zobrazení všech objektů na obrazovce. Výsledný obraz musí hlavně působit přirozeně a veškeré textové popisky musejí být čitelné. Proto byl s pomocí návodu v [5] vytvořen následující postup přizpůsobení cílové obrazovce.

Byla použita funkce pro zjištění orientace obrazovky. Tato funkce zjišťuje orientaci na základě rozlišení obrazovky. Vypočítává poměr mezi horizontálním a vertikálním rozlišením. Dle tohoto poměru funkce vrací tři různé režimy - Portrait, Landscape a Simple. Režim Portrait má poměr stran menší než 0,8, což značí obrazovku orientovanou na výšku. Režim Landscape značí obrazovku orientovanou na šířku a hodnota poměru stran je větší než 1,3. V ostatních případech funkce vrací režim Simple, což znamená poměr stran větší než 0,8 a menší než 1,3.

Režim	Poměr stran (x:y)
Portrait	< 0,8
Simple	0,8 - 1,3
Landscape	> 1,3

Tab. 3.1: Režimy orientace obrazovky dle funkce uvedené v [5].

Na základě zjištěné orientace obrazovky se pokračuje v umístění jednotlivých prvků uživatelského rozhraní na plochu obrazovky. Zatím není důležité skutečné rozlišení obrazovky, ale pouze její orientace. Pro každou orientaci bylo zvoleno referenční rozlišení, pro které se prvky na obrazovku rozmísťují. Poté je vypočten poměr mezi referenčním a skutečným rozlišením obrazovky a všechny

pozice a velikosti jednotlivých prvků se tímto poměrem vynásobí. Díky tomu vypadá rozhraní hry stejně na obrazovkách s rozlišením 240x320 i 480x640.

3.6.2 Zpracování vstupu uživatele a předávání zpráv třídě Hra

Uživatelské vstupy se zpracovávají pomocí skupiny metod třídy `Obrazovka`. Takováto metoda se označuje anglickým termínem handler a je volána vždy při vykonání určité akce uživatelem, například při zmáčknutí tlačítka mobilního zařízení, nebo při klepnutí na hrací kartu na obrazovce (v případě dotykové obrazovky).

Každý ovládací prvek, nebo jejich skupina, má přiřazený svůj handler. Uvnitř je obsažen kód, který například zjišťuje, která hrací karta byla aktivována a na základě tohoto zjištění se tato informace předá dále třídě `Hra`, která provede další kroky vedoucí k zahrání karty. Prvky uživatelského rozhraní, které zpracovávají uživatelský vstup, jsou znázorněné na Obr. 3.5. Jsou to hrací karty, nabídka programu a tlačítka mobilního zařízení.

Mezi hracími kartami se dá pohybovat buď pomocí navigačních kláves, nebo jejich označením prstem v případě zařízení s dotykovou obrazovkou. Při označení prstem se karta ihned aktivuje, což se v případě zařízení Smartphone bez dotykové obrazovky provádí potvrzovacím tlačítkem navigačních kláves (skupina tlačítek č. 3 na Obr. 3.5). Nabídku programu lze otevřít pomocí levého a pravého kontextového tlačítka mobilního zařízení (č. 4 na Obr. 3.5), pohybuje se v něm směrovými navigačními klávesami.

Obr. 3.5: Vstupní ovládací prvky uživatelského rozhraní.
1. hrací karty, 2. nabídka programu, 3. navigační tlačítka mobilního zařízení, 4. kontextová tlačítka

3.6.3 Vykreslení aktuálního stavu hry na požádání

Třída `Obrazovka` obsahuje dvě metody, které provádějí vykreslení aktuálního herního stavu na obrazovku. Jsou to metody `VykresliStav()` a `VykresliKarty()`.

Metoda `VykresliStav()` provádí vykreslení aktuálních hodnot prostředků obou hráčů a vykreslení hradů a hradeb. Metoda `VykresliKraty()` provádí vykreslení hracích karet hráče, který je právě na tahu. Pokud je na tahu hráč typu člověk, jsou hrací karty normálně zobrazené. Pokud je naopak na tahu hráč jiného typu, například počítač, karty se zobrazí lícovou stranou nahoru a nezobrazí svou hodnotu. Proto v případě hry dvou hráčů na jednom mobilním zařízení si budou hráči vidět do karet, ale pokud bude hrát jen jeden hráč proti počítači, jeho karty neuvidí.

3.6.4 Animace karet

Správná hra, která chce zaujmout hráče, by dle mého názoru měla používat animace k vizuální prezentaci akcí, které se během hry dějí. Nejjednodušším způsobem, jak tyto animace vytvořit, je používat kombinaci kreslení obrázků a měnění jejich pozice na obrazovce. K tomu, aby byla animace zřetelná je taktéž za potřebí animaci vhodně krokovat a použít vhodné prodlevy mezi kroky.

Implementaci takové animace je možné si představit jako posloupnost následujících kroků:

1. Vykreslíme na obrazovku snímek animace
2. Přerušíme běh programu na určitou dobu a nezapomeneme před přerušením zajistit, aby se obrázek opravdu vykreslil (k přerušení běhu programu totiž může dojít v místě, kdy obrázek ještě nebyl vykreslen na obrazovku)
3. Smažeme předcházející obrázek z obrazovky a na jeho místo vykreslíme obrázek dalšího kroku animace. Můžeme taktéž změnit jeho pozici k vyvolání dojmu pohybu obrázku.
4. Opět zajistíme důsledné vykreslení obrázku a přerušíme běh programu
5. Tento postup opakujeme pro všechny jednotlivé snímky animace

Konkrétní implementace animace byla provedena způsobem popsaným v následujícím odstavci.

Animace se sice zobrazuje vždy po aktivaci karty a zdánlivě probíhá celá najednou, ale v jejím průběhu se provádí také jiné metody, starající se o logickou část tahu. Proto byla animační část kódu rozdělena do tří nových metod `AnimujKartu()`, `AnimujLizani()` a `AnimujOdlozeni()`.

Metoda `AnimujKartu()` provádí animaci pohybu karty při jejím zahrání. Karta se při animaci přesune v několika krocích do středu hracího pole mezi hrady. Animace je rozdělena na několik kroků a mezi každým krokem pohybu je určitá prodleva.

Metoda `AnimujLizani()` je velice podobná, avšak její průběh je opačný. Ve středu hrací plochy mezi hrady se objeví nová líznutá karta, která se přesune na prázdné místo po tažené kartě.

Metoda `AnimujOdlozeni()` probíhá stejně, jako metoda `AnimujKartu()`, avšak s viditelnou lícovou stranou karty.

Mimo výše zmíněné metody se provádí ještě animace stavových ikon při zahrání karty. Tato animace je řízena z metody `VykresliStav()` na základě instanční proměnné *animace* třídy Obrazovka.

3.7 Vykreslování ikon a obrázků

Cílová platforma byla vybrána s ohledem na kvalitu programovacího jazyka, vývojových nástrojů a s přihlédnutím k osobním zkušenostem autora, které s platformou má. Vývojové knihovny .NET Compact Framework jsou, jak je psáno v kapitole 2.6, redukovanou verzí knihoven .NET Framework. Nese to s sebou však omezení, na které autor narazil při použití ovládacích prvků, které vykreslují grafické ikony a obrázky.

Při programování stavových ikon bylo třeba zajistit, aby každá ikona zobrazila obrázek prostředku (cihly, zbraně, krystaly…) a u tohoto obrázku číselnou hodnotu. Původně k tomu byly použity standardní komponenty z knihovny .NET Compact Framework – `Label` a `PictureBox`.

Komponenta `Label` je určena k zobrazování textových popisků, komponenta `PictureBox` zobrazuje obrázky. Na rozdíl od plné verze knihoven .NET Framework však žádná ze standardních komponent nedokáže pracovat s průhledným pozadím. Každá komponenta má vždy tvar obdélníku vyplněného určitou barvou pozadí a v něm se vykresluje obsah – text, obrázek, tlačítko apod. Proto při kombinaci obrázku a textu byl vždy obrázek překrytý pozadím textu, nebo obráceně. Proto bylo přikročeno k vytvoření vlastních komponent. Bylo využito dědičnosti tříd k vytvoření vlastních komponent `StateBox`, `CardBox` a `HradBox`.

3.7.1 Implementace vlastní komponenty StateBox

Komponenta `StateBox` je rozšířením standardní komponenty `PictureBox`. Dědí všechny její vlastnosti a metody. Navíc byla přidána možnost přímého vykreslení textu do bitové mapy obrázku. `PictureBox` vykresluje bitovou mapu, která je nastavena v jeho vlastnosti Image. Autor pro svůj účel vytvořil nové vlastnosti `ImageX`, `Text` a `Animace`. Dále pomocné metody `Renderuj()` a `AnimujZmenu()`. Komponenta `PictureBox` disponuje zajímavou vlastností, tzv. dvojitým bufferem. Je to vyrovnávací paměť použitá při jejím vykreslování na obrazovku, která jistým způsobem odstraňuje problikávání komponenty při vykreslování. Pokud by byl přímo nahrazen kód starající se o vykreslování pozadí a bitmapy komponenty vlastním kódem, o tuto optimalizaci

vykreslování by se přišlo. Proto byla implementována vlastnost `ImageX`, do které se předává původní bitová mapa k vykreslení.

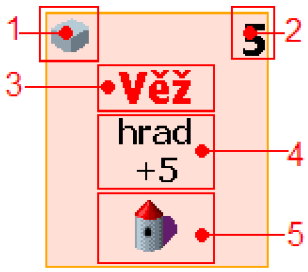
S bitovou mapou se následně pracuje v metodě `Renderuj()`. Pomocí vestavěných funkcí pro práci s grafikou se vytvoří prázdná bitová mapa o rozměrech prvku `StateBox`. Ta je poté vyplněna barvou pozadí komponenty, přes toto pozadí je vykreslen obrázek `ImageX` a nakonec ještě vypsán text. Tímto způsobem se generuje hotová bitová mapa, která se uloží do vlastnosti `Image` rodičovské třídy `PictureBox`. Ta se následně vykreslí standardním způsobem včetně všech optimalizací, o které se tak dále není třeba starat.

Metoda `Renderuj()` se volá při každé změně některé z vlastností `ImageX`, `Text` nebo `Animace`. Aby se předešlo zbytečnému generování bitové mapy, před voláním metody je provedena kontrolu, zda se nově zadané vlastnosti liší od předchozích. V případě že se vlastnosti shodují, generování se neprovede. Tato optimalizace byla zavedena na základě znatelně pomalého vykreslování komponenty během hry.

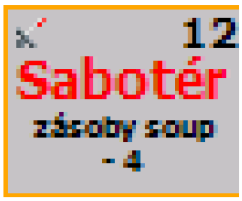
3.7.2 Implementace vlastní komponenty CardBox

Komponenta `CardBox` má velice podobnou implementaci jako `StateBox`. Rozdíl je v možnosti vytvoření obrázku hrací karty přesně podle vstupních parametrů. Každá hrací karta se skládá z několika částí (viz. Obr. 3.6).

Z důvodu zachování dobré čitelnosti karet na malé obrazovce mobilního zařízení bylo hned od začátku vyloučeno vykreslování rubové strany karty pouze pomocí bitmapového obrázku. Text byl totiž díky absenci interpolace bitmap v použité verzi knihoven .NET Compact Framework nečitelný. Na některých mobilních zařízeních s malým rozlišením obrazovky navíc nebyl ani zřetelný obrázek karty a už vůbec ne ikona. Autor se tedy rozhodl ke generování obrázku celé karty podle parametrů vlastními silami.



Obr. 3.6: Části hrací karty.
1. Ikona prostředku, 2. Cena karty, 3. Název karty, 4. Popis chování karty, 5. Obrázek karty



Obr. 3.7: Hrací karta optimalizovaná pro malé rozlišení

Díky zkušenostem z implementace prvku `StateBox` také použil stejné techniky generování bitmapy. Jak je vidět na Obr. 3.6, hrací karta má 5 částí, které se vykreslují jedna po druhé. Opět byla k tomuto účelu implementována metoda `Renderuj()`.

Ikona prostředků a cena karty se vykreslují jako první. Jejich výška je 20% z celkové výšky karty. Název karty se vykresluje o řádek níž. Šířka a výška názvu karty se vypočítává na základě počtu slov v názvu tak, aby byl text vždy čitelný a využíval maximum vyhrazené plochy karty. Obdobně je tomu u popisu chování karty. Obrázek karty je zarovnaný se spodním okrajem a jeho rozměr je dán zbylým místem od popisu chování karty směrem dolů. Zároveň se autor snaží udržet optimální poměr mezi obrázkem a textovými popisky.

Při výpočtu velikosti obrázku karty se také zjišťuje, zda obrázek není příliš malý a nezřetelný. Jeho minimální velikost musí být alespoň jedna třetina šířky spodního okraje karty. V případě menšího rozměru se obrázek karty nevykreslí a volné místo se použije pro textový popis karty. Je tak zaručeno, že i na velmi malé obrazovce bude karta čitelná. Příklad takto optimalizované hrací karty je na Obr. 3.7. Takováto karta se zobrazí na zařízení s rozlišením obrazovky 240x240 obrazových bodů, kde je nedostatek volného prostoru určeného pro hrací karty. Karta je sice široká a klasické hrací kartě se příliš nepodobá, avšak její informační hodnota zůstala stejná.

3.7.3 Implementace vlastní komponenty HradBox

Pro účel vykreslování mravenčích hradů a zdí autor vytvořil komponentu `HradBox`. Je to komponenta založená na poněkud odlišném principu, než předešlé komponenty `StavBox` a `CardBox`. Jejím účelem je zobrazovat obrázek hradu nebo zdi proměnné výšky, rostoucí od svého základu směrem vzhůru.

Původně autor zamýšlel tuto komponentu postavit na stejném základu jako komponenty předešlé. Neustálé generování bitové mapy by však bylo zbytečné a jen by zatěžovalo výpočetní prostředky mobilního zařízení. Po celou dobu hry se totiž používá ten samý obrázek hradu nebo zdi a není tak potřeba žádnou část obrázku měnit.

K vytvoření dojmu hradu rostoucího do výšky je využito původní komponenty `Panel` a v ní umístěný `PictureBox`, který pouze mění pozici. Komponenta `Panel` je kontejner s definovanými rozměry, do kterého je možné umístit jiné libovolné objekty. Tyto objekty mohou mít uvnitř libovolnou pozici, avšak viditelná je pouze jejich část, která se nachází uvnitř obdélníku daného rozměry panelu.

Dovnitř panelu byla tedy umístěna komponenta `PictureBox`, která vykresluje samotný obrázek hradu. Ta se vysouvá v panelu od jeho spodní hrany směrem nahoru, čímž se postupně odkrývá a vytváří dojem, že hrad roste.

3.7.4 Problém při implementaci průhledných obrázků

Jak je uvedeno v kapitole 2.6, platforma .NET Compact Framework neposkytuje možnost vykreslování průhledných obrázků standardním způsobem. Tato možnost je sice obsažená v plné verzi platformy pro osobní počítače, ale z důvodů výkonnostních ji ve verzi Compact nelze použít.

Při samotné implementaci hry průhledné obrázky nebyly potřeba, ale jistě by byla grafická reprezentace hry zajímavější při použití hracích karet se zakulacenými okraji. Toto omezení autorovi také zprvu znemožnilo zobrazit na pozadí hrací plochy obrázek. Ikony hradů a zdí totiž pokrývají většinu volné hrací plochy a protože je jejich tvar stále stejný (provádí se pouze kreslení obrázku uvnitř komponenty `HradBox`), docházelo k překrývání obrázku na pozadí bílým pozadím těchto komponent. Problém je ilustrován na Obr. 3.8. Autor se rozhodl pokusně tuto chybějící implementaci doplnit.

Obr. 3.8: Ilustrace chybějící implementace průhlednosti obrázků.
Vlevo je obrazovka zařízení bez implementace průhlednosti komponent, vpravo je průhlednost naimplementovaná.

Aby bylo možné obejít onu nemožnost zprůhlednit pozadí jakékoliv komponenty, musely být rozšířeny třídy použitých komponent o možnost vykreslení libovolného obrázku na jejich pozadí. Ke každé vybrané třídě komponent byla implementována vlastnost `Pozadi`. Do této vlastnosti je předán odkaz na zvolený obrázek pozadí celé hrací plochy, nikoliv jen jeho části určené pro konkrétní

komponentu. Komponenta samotná si tak na základě známých údajů o svém umístění na obrazovce vybere jen tu část obrázku pozadí, kterou překrývá plochou svého kontejneru.

Tato implementace má ale jednu nevýhodu a to tu, že na mobilních zařízeních s pomalejším procesorem klesá výkon vykreslování a například pohyb karet při animaci je znatelně pomalejší.

Z tohoto důvodu přibyla možnost nastavení, zda zobrazovat pozadí či nikoliv. Toto nastavení je implicitně vypnuto a dá se zapnout v okně nastavení.

3.8 Třída Nastaveni

Aby bylo možné uchovávat nastavení parametrů hry i v době, kdy hra není spuštěna, je potřeba zvolit některý z dostupných způsobů jejich ukládání.

V podstatě je na výběr ze dvou variant. Tou první je ukládání parametrů do souboru v adresáři hry v nějakém textovém nebo binárním formátu, případně ve formě XML. Tato varianta se jeví jako univerzálnější z pohledu budoucí možné implementace hry na jiné platformě.

Druhou možností je ukládání nastavení do systémových registrů operačního systému. Toto není příliš univerzální způsob, ale autor jej zvolil z důvodu snadnější implementace na zvolené platformě.

Aby zde byla možnost budoucí implementace ukládání nastavení jiným způsobem, byla navržena třída `Nastaveni`, která systém ukládání nastavení zapouzdřuje.

Třída `Nastaveni` má veřejně přístupné vlastnosti, které umožňují načítat nebo ukládat nastavení programu. Tyto vlastnosti jsou pojmenované `Obtiznost`, `Protihrac`, `Audio` a `Pozadi`.

Jsou to celočíselné proměnné vracející index příslušné volby. Při prvním spuštění hry, kdy nastavení nelze načíst, jelikož ještě nebylo hrou uloženo, se použijí výchozí hodnoty definované přímo v programu.

Komplexní přístup k nastavení je implementován ve zvláštním dialogovém okně, které je možné aktivovat přes nabídku programu.

3.9 Třída Zvuk

Aby se dosáhlo oživení průběhu hry, byla navržena také třída starající se o přehrávání zvukových efektů. Jak je napsáno v kapitole 2.4, v operačním systému je přítomné rozhraní pro přehrávání zvukových souborů a proudů. V [6] byl vyhledán vzorový kód pro použití tohoto rozhraní. Z tohoto vzorového kódu byly použity metody `PlaySync()` a `PlayAsync()`.

Metody umožňují přehrát zvukový soubor synchronně nebo asynchronně. Tyto dva způsoby přehrávání se od sebe liší tím, že při volání metody `PlaySync()` je přehrán zvukový soubor a po dobu tohoto přehrávání je pozastavena činnost programu. Lze tak jednoduše počkat na konec přehrávání a teprve poté se věnovat vykonání dalších příkazů programu. Metoda `PlayAsync()`

naopak umožňuje přehrát zvuk a již během přehrávání lze ve vykonávání programu pokračovat. Asynchronní přehrávání se tak hodí v případě, že je potřeba přehrávat nějaké zvuky na pozadí hry. Například doprovodnou hudbu nebo zvuky dokreslující atmosféru hry.

3.10 Načítání bitových map obrázků

Aby bylo vůbec možné ve hře zobrazovat obrázky, musejí se do programu buď nějakým způsobem načíst, nebo obrázky vygenerovat vlastními silami v průběhu programu. Druhá varianta je náročnější a její použití se hodí zejména v případě jednodušších her, kde není potřeba složitější grafika. Příkladem může být například hra piškvorky, kdy se grafika vykresluje pomocí přímek a kruhů. Zde evidentně není potřeba načítat obrázky nějakým způsobem dříve vytvořené a uložené ve formě bitových map. Pro hry, kde je potřeba použít předem vytvořené obrázky je tedy nutné je na mobilní zařízení uložit a při spuštění programu je načíst.

Vývojové prostředí Microsoft Visual Studio nabízí možnost správy obrázků. Avšak ve výsledku jsou po kompilaci programu součástí výsledného spustitelného souboru. Tuto možnost je vhodné použít při ukládání menších obrázků, jako je například ikona programu. Pro účel hry Mravenci byl proto zvolen způsob uložení obrázků v souborovém systému mobilního zařízení v adresáři spolu se spustitelným souborem hry. Obrázky jsou uložené ve formě bitových map ve formátu BMP.

Načítání bitových map se provádí během inicializace programu. Původně se obrázky načítaly v průběhu hry vždy, když bylo potřeba obrázek zobrazit. Autor vycházel z předpokladu, že mobilní zařízení používá stejnou fyzickou paměť pro operační paměť i úložný prostor souborového systému. Je-li tomu tak či nikoliv, přímé načítání souborů bylo velmi pomalé.

Byla proto implementována vyrovnávací paměť. Všechny obrázky použité ve hře se načítají během inicializace programu do této vyrovnávací paměti, kde zůstanou po celou dobu běhu programu. Při použití v grafických komponentách se obrázky načítají z vyrovnávací paměti, což je mnohem rychlejší.

Načítání obrázků do vyrovnávací paměti trvá během inicializace programu určitou dobu. Aby bylo zřejmé, že se během této doby něco děje, bylo vytvořeno spouštěcí okno hry, kdy se postupně zobrazují informace o průběhu inicializace hry. Povaha těchto informací by pro uživatele hry byla ve většině případů nesrozumitelná či nepodstatná, proto se místo technických údajů zobrazují hlášky „Plním sklady…“, „Míchám maltu…“ a jiné.

4 Závěr

Během práce se autorovi podařilo prozkoumat problematiku vývoje her pro mobilní zařízení. Nezabýval se pokročilejšími postupy při tvorbě grafického uživatelského rozhraní za použití specializovaných knihoven. Spíše se snažil využít standardních prostředků vývojové platformy. Podařilo se mu tak navrhnout a implementovat funkční karetní hru Mravenci.

Autor provedl testování funkčnosti hry na vlastních mobilních zařízeních Samsung i600 a HP iPAQ hx2410. Na obou zařízeních byl chod hry plynulý a to i přes různé verze platformy Windows Mobile na každém z nich.

Za vlastní přínos autor považuje obejití některých omezení standardních komponent .NET Compact Framework, na která narazil během implementace hry. Zejména je to chybějící implementace průhlednosti komponent. Autorovo řešení je uvedeno v kapitole 3.6.

Další omezení, na která autor narazil, se týkají kompatibility mezi jednotlivými verzemi platformy Windows Mobile. Tato platforma prochází neustálým vývojem a čas od času se najde problém s nekompatibilní implementací různých částí aplikací na různých verzích platformy. Příkladem může být přehrávání zvukových souborů uvedené v kapitole 3.9.

Jako možná budoucí rozšíření hry by autor určitě doporučil návrh a implementaci hry dvou hráčů na různých mobilních zařízeních. Použil by k tomu síťové rozhraní s použitím herního serveru umístěného na internetu, nebo rozhraní Bluetooth. Bylo by tak možné pořádat herní turnaje a stavět žebříčky nejlepších hráčů.

V rámci rozšíření funkcionality o hru přes internet by se mohlo využít projektu další bakalářské práce zadané v tomto roce, zpracované na téma „Webová varianta hry Mravenci“.

Dále by bylo jistě zajímavé nabídnout výslednou hru volně ke stažení na internetu a za příplatek nabízet výše zmíněné rozšíření hry po síti. Jednalo by se jistě o zajímavý obchodní model.

Literatura

[1] CAMPBELL, Andrew T., SCHWARTZ, Mischa. ACM SIGCOMM computer communication review. In *SIGCOMM Comput. Commun. Rev.*. New York, NY, USA : ACM, 2001. ACM SIGCOMM computer communication review. s. 20-24. ISSN 0146-4833.

[2] DIDELES, Myra. Bluetooth: a technical overview. In *Crossroads*. New York, NY, USA : ACM, 2003. s. 11-18. Dostupný z WWW: <http://doi.acm.org/10.1145/904080.904083>. ISSN 1528-4972.

[3] HARRISON, Richard , NORTHAM, Phil. *Symbian OS C++ for Mobile Phones*. New York, NY, USA : John Wiley & Sons, Inc., 2003. 826 s. ISBN 0470856114.

[4] LACKO, Luboslav. *Programujeme mobilní aplikace ve Visual Studiu .NET*. Brno : Computer Press, 2004. 480 s., CD-ROM. ISBN 80-251-0176-2.

[5] *Step by Step: Developing Orientation-Aware and Resolution-Aware Windows Mobile-based Applications in Native Code* [online]. Microsoft Corporation, c2009 [cit. 2009-05-15]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/aa454895.aspx>.

[6] *Windows Mobile Developer Center* [online]. Microsoft Corporation, c2009 [cit. 2009-04-08]. Dostupný z WWW: <http://msdn.microsoft.com/cs-cz/windowsmobile/default(en-us).aspx>.

[7] Zbořil, František V., *Základy umělé inteligence :IZU*. Brno : Fakulta informačních technologií, 2008. 142 s.

Seznam obrázků

Seznam příloh

A. Obsah přiloženého disku DVD

DVD:\

\	aplikace	
	\ dokumentace	-programová dokumentace
	\ instalátor	-instalační program hry pro mobilní zařízení
	\ zdrojové soubory	-zdrojové soubory pro MS Visual Studio 2008
	\ instalace.txt	-návod na instalaci hry na mobilní zařízení
\	technická zpráva	-technická zpráva ve formátu PDF
\	nástroje	-instalační soubory nástrojů použitých při vývoji