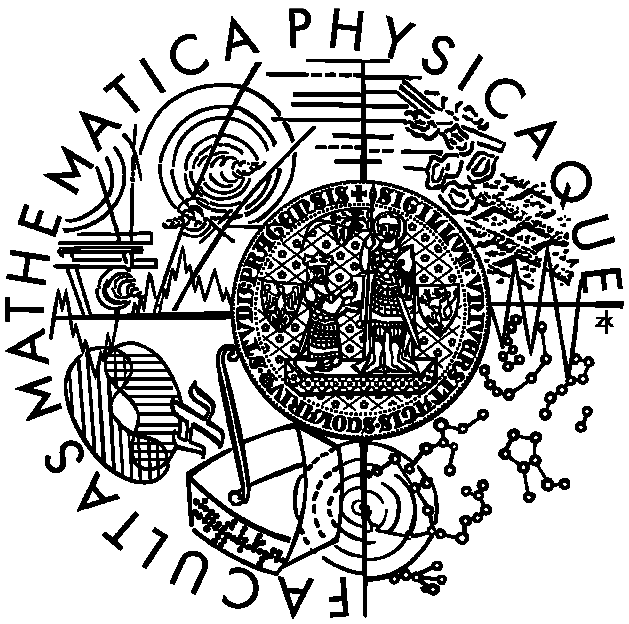
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta



DIPLOMOVÁ PRÁCE

Ivan Vacula

Multiplatformní prostředí pro vývoj mobilních her

Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. David Bednárek, Ph.D.

Studijní program: Informatika, Softwarové systémy

Ďakujem vedúcemu práce, doktorovi Davidovi Bednárkovi za mnoho užitočných rád, ochotu a pomocnú ruku počas celej tvorby tejto práce.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne Ivan Vacula

Obsah

Abstrakt

Názov práce: Multiplatformní prostředí pro vývoj mobilních her
Autor: Ivan Vacula
Katedra: Katedra softwarového inženýrství
Vedúci diplomovej práce: RNDr. David Bednárek, Ph.D.
E-mail vedúceho: bednarek@ksi.mff.cuni.cz
Abstrakt:
Inteligentné mobilné zariadenia sú v dnešnej dobe čoraz viac obľúbené. Spolu s ich rozšírením prichádza aj problém s vývojom aplikácií na niekoľko platforiem súčasne. Jednotlivé mobilné operačné systémy ako aj samotné zariadenia sa od seba zásadne líšia a pre multiplatformný vývoj je potrebný systém abstrahujúci rozdiely medzi nimi. Ešte o niečo zložitejšia je situácia pri vývoji hier, ktoré vyžadujú čo najlepšie využitie prostriedkov zariadenia, s ohľadom na rýchlosť, odozvu, grafické spracovanie a podobne. Cieľom tejto práce je porovnať existujúce riešenia pre multiplatformný vývoj mobilných hier, potom navrhnúť a implementovať vlastný systém. Systém musí zabezpečiť jednotný vývoj hier pre niekoľko operačných systémov zároveň. Súčasťou práce je implementácia systému pre dve diametrálne odlišné mobilné platformy a implementácia demonštračnej hry.
Kľúčové slová: multiplatformný, mobil, hra, vývoj

Title: Multi-platform environment for mobile-game development
Author: Ivan Vacula
Department: Department of Software Engineering
Supervisor: RNDr. David Bednárek, Ph.D.
Supervisor's e-mail address: bednarek@ksi.mff.cuni.cz
Abstract:
Smart mobile devices are getting more and more popular these days. But there comes a problem with their expansion as well. Mobile operating systems differ from each other, as do mobile devices themselves, so development across multiple platforms is not simple. Situation is even worse when it comes to games. These need to use limited resources of a mobile device as much as possible, taking into account quick response to user actions, graphical execution et cetera. The first goal of this thesis is to compare existing solutions for multi-platform mobile game development. The second goal is to design a new system. This system must ensure targeting of multiple platforms during mobile game development. Included in this thesis is implementation of the system on two vastly different operating systems as well as implementation of a demonstration game.
Keywords: multi-platform, mobile, smartphone, game, development

1. Úvod

1.1. Motivácia

V dnešnej dobe zažívajú mobilné zariadenia skutočný rozmach. Momentálne pripadá celosvetovo na každého človeka takmer jeden mobilný telefón[1]. Zďaleka najdynamickejšie rozvíjajúcim sa sektorom sú inteligentné zariadenia s otvorenými operačnými systémami [1]. Ich možnosti a hardwarový výkon sú dnes v podstate na úrovni stolných počítačov spred niekoľkých rokov.

1.1.1. Porovnanie výkonu

Pre lepšiu predstavu o výkone dnešných mobilných zariadení uvediem porovnanie so stolnými počítačmi. Veľmi rozšírenou architektúrou procesorov pre mobilné zariadenia je v dnešnej dobe ARMv7. Tento dosahuje teoretický výkon až 0,9 MIPS / 1 MHz [2]. Pre porovnanie, procesor Intel Pentium III pre stolné počítače dosahuje 1,6 MIPS / 1 MHz [3]. Intel Pentium III pracujúci na frekvencii 600 MHz, ktorý bol vydaný koncom roku 1999 je z tohto hľadiska výkonovo približne na rovnakej úrovni ako 1 GHz ARMv7 procesory z roku 2009 [4].

Keď sa pozrieme na veľkosť operačnej pamäte, je rozdiel ešte menší. Dnešné mobilné zariadenia majú k dispozícii približne 512 MB. To odpovedá priemernému stolnému počítaču z obdobia okolo roku 2005.

1.1.2. Operačné systémy

Na rozdiel od stolných počítačov, kde z hľadiska rozšírenia operačných systémov jednoznačne dominuje Microsoft so svojimi Windows, nie je situácia v mobilnom svete zďaleka taká jednoznačná. Dynamický rozvoj mobilných zariadení a ich operačných systémov a tvrdá konkurencia majú za následok pomerne veľkú fragmentáciu trhu. Ako vidno v tabuľke 1, v súčasnosti existuje aspoň 5 významných mobilných platforiem, ktoré medzi sebou súperia o trhový podiel.

Tabuľka 1: Prehľad dominantných operačných systémov

Názov	Uvedený	Trhový podiel [5]
Symbian	2001	41%
BlackBerry OS	1999	18%
Android	2008	17%
iOS	2007	14%
Windows Mobile	2000	5%

[1] Celosvetovo 4,6 miliardy mobilných telefónov koncom roka 2009 [74].

Súčasne s týmto rozmachom preto prišiel aj problém pre vývojárov aplikácií na tieto zariadenia. Snahou je samozrejme podporovať svojou aplikáciou čo najväčší počet platforiem a tak mať možnosť pokryť čo najväčšiu časť trhu.

Vývoj multiplatformných aplikácii nie je pritom jednoduchý, keďže vývojové platformy pre jednotlivé zariadenia sa od seba značne líšia. Už na úrovni podporovaných programovacích jazykov nenájdeme ani jeden, ktorý by sa dal použiť na každej platforme. Niektoré platformy sú založené na jazyku kompilovanom do natívneho kódu ako Symbian C++ na Symbian OS, C++ na Windows Mobile alebo Objective-C na iOS. Iné sú zas založené na interpretovaných jazykoch, napríklad BlackBerry OS na jazyku a platforme Java alebo Windows Phone 7 na platforme .NET (a poťažmo jazyku C#). Kompletný prehľad vývojových platforiem pre jednotlivé operačné systémy sa nachádza v tabuľke 2.

Tabuľka 2: Vývojové platformy hlavných operačných systémov

Názov	Vývojová platforma	Natívne aplikácie
Symbian	Symbian C++, Java ME, Flash Lite, Python, .NET Compact Framework [6]	Áno
BlackBerry OS	Java, Web applications	Nie
Android	Java, Native Development Kit - C	Áno
iOS	Objective-C, Web applications	Áno
Windows Mobile	C++, .NET Compact Framework	Áno
Windows Phone 7	Silverlight, XNA	Nie
Palm webOS	Web applications, C a C++	Áno

Rozdielne vývojové platformy však nie sú jediným problémom pri snahe o súbežný vývoj aplikácií pre viacero operačných systémov. Už samotné zariadenia sa od seba značne líšia. Rozdiely existujú v rozlíšeniach displejov, výkonoch a možnostiach procesorov alebo medzi princípmi ovládania, ktoré môže byť založené na dotykovej obrazovke, fyzických tlačidlách, prípadne kombinácii obidvoch. Porovnanie charakteristík zariadení jednotlivých platforiem je v tabuľke 3.

Tabuľka 3: Charakteristiky zariadení

Názov	Edícia / verzia	Architektúra CPU	Dotykový displej	Rozlíšenia[2]
Symbian	Symbian^1	ARM	Áno / Nie	800x352, 320x240, 640x360
RIM	4.7+	ARM	Áno / Nie	480x360, 480x320, 400x360, 320x240
Android	2.x	ARM	Áno	854x480, 800x480, 720x480, 480x320,400x240, 320x240
iOS	3.x+	ARM	Áno	960x640, 480x320
Windows Mobile	Professional 6.x, Classic 6.x	ARM	Áno	800x480, 640x480, 320x320, 400x240, 320x240
	Standard 6.x	ARM	Nie	320x320, 400x240, 320x240
Windows Phone 7	-	ARM	Áno	800x480
Palm webOS	1.0+	ARM	Áno	320x480

Kým situácia je aspoň čiastočne riešiteľná pre bežné aplikácie formou robustných aplikačných platforiem (podrobnejšie rozobraných v druhej kapitole), pre hry je situácia oveľa zložitejšia. Na rozdiel od bežných aplikácií očakáva používateľ od hier veľmi dobrú odozvu a bohatšiu grafickú reprezentáciu. Jednotné prostredie musí teda riešiť nie len problémy s odlišnosťami mobilných platforiem, ale musí byť aj dostatočne výkonné a schopné využiť prostriedky daného zariadenia.

1.2. Ciele práce

Prvým cieľom tejto práce bolo preskúmať existujúce riešenia pre multiplatformný vývoj hier pre mobilné zariadenia. V prehľade sú zahrnuté existujúce prostredia pre vývoj hier, okrajovo je preskúmaná aj možnosť použitia riešení, ktoré nie sú pre hry špeciálne určené.

Ťažiskom práce bol návrh riešenia, ktoré by umožňovalo vývoj mobilných hier tak, aby bola hra spustiteľná na ľubovoľnej mobilnej platforme bez toho, aby sa vývojár hry musel zaoberať špecifikami jednotlivých operačných systémov. Systém je určený pre jednoduchšie hry s dvojrozmernou grafikou založenou na 2D textúrach a umožňuje okrem grafiky jednoduchú prácu aj s textom, zvukmi, používateľskými vstupmi a rozhraním.

Systém, ktorý je v tejto práci navrhnutý som nazval MPME – Multi-platform mobile engine. Základnými cieľmi pri návrhu systému boli:

- **Portabilita popisu hry** – Každá hra postavená na MPME musí byť bez akýchkoľvek zmien spustiteľná na implementácii MPME pre ľubovoľnú platformu[3].

[2] Uvádzam iba bežné rozlíšenia používané v zariadeniach, nie všetky rozlíšenia podporované operačným systémom.

[3] Za predpokladu, že je hra postavená tak, že vie korektne pracovať s displejom orientovaným na šírku aj na výšku, že korektne podporuje všetky profily používateľských vstupov a podobne.

- **Konzistentné správanie rozhrania** – Úzko spojené s prvým bodom. Rozhranie MPME musí byť dobre definované a správať sa rovnako v každej implementácii. Musí abstrahovať prostriedky zariadenia do jednotnej vrstvy.
- **Popis celej hry v skriptovacom jazyku** – Takisto súvisí s prvým bodom. Hra postavená nad MPME by mala byť takmer kompletne popísaná skriptovacím jazykom. V jazyku natívnom pre danú platformu bude musieť byť napísané len minimum nutné pre spustenie aplikácie.
- **Jednoduchosť implementácie** – Implementovať MPME na ľubovoľnú mobilnú platformu musí byť čo najjednoduchšie. Návrh MPME musí byť "tenký".

Pretože neexistuje jazyk, ktorý by bol priamo kompilovateľný pre každú platformu, hry postavené na systéme MPME budú musieť byt popísané nejakým interpretovaným skriptovacím jazykom. S ohľadom na tieto predpoklady som vybral pre popis hier skriptovací jazyk Lua.

Uskutočniteľnosť navrhnutého riešenia je demonštrovaná implementáciou pre dve odlišné mobilné platformy a implementáciou vzorovej hry. Pre pilotnú implementáciu som si vybral platformy Windows Mobile 6 Standard a Windows Phone 7. Napriek tomu, že by sa podľa názvu mohlo zdať, že ide o podobné operačné systémy, opak je pravdou.

Implementácia systému pre Windows Mobile 6 Standard je napísaná v jazyku C++ a preložená do natívneho kódu zariadenia. Zariadenia s týmto operačným systémom majú displeje s nízkym rozlíšením, ktoré navyše nie sú dotykové. Pri implementácii som okrem štandardných vývojových nástrojov a knižníc pre danú platformu použil knižnicu GapiDraw pre prácu s grafikou. Okrem toho som využil aj LuaCE, implementáciu virtuálneho stroja skriptovacieho jazyka Lua pre platformu Windows CE (a teda poťažmo aj Windows Mobile 6 Standard).

Windows Phone 7 neumožňuje preklad do natívneho kódu zariadenia a celá implementácia je založená na platforme .NET a jazyku C#. Práve z tohto dôvodu bolo nutné pre túto platformu implementovať aj virtuálny stroj pre skriptovací jazyk Lua. Štandardným nástrojom pre vývoj hier pre operačný systém Windows Phone 7 je XNA Game Studio, ktoré som taktiež pri implementácii využil. Zariadenia s Windows Phone 7 disponujú len minimálnym počtom fyzických tlačidiel a majú dotykové obrazovky s relatívne vysokým rozlíšením.

Vzorová hra, implementovaná ako ukážka použiteľnosti navrhnutého a implementovaného riešenia je jednoduchá ťahová strategická hra pre dvoch hráčov. V hre sa snaží hráč poraziť toho druhého pomocou lietadiel, ktoré v jednotlivých ťahoch ovláda. Prostredie hry je zobrazené v 2D grafike z nadhľadu. Pri implementácii hry som použil voľne dostupné grafické a zvukové elementy.

1.3. Štruktúra práce

Zostávajúca časť tejto práce je rozdelená nasledovne. V druhej kapitole sa nachádza prehľad aktuálnych mobilných platforiem. Do prehľadu som zahrnul operačné systémy, ktoré sú momentálne dominantné alebo majú potenciál stať sa v blízkej dobe významnými. Ďalej nasleduje prehľad existujúcich systémov riešiacich problematiku vývoja multiplatformných mobilných hier. Analytická kapitola je zakončená návrhom môjho riešenia.

V tretej kapitole je popísaný navrhovaný systém, jeho architektúra a podrobne aj jeho súčasti a rozhrania.

Štvrtá kapitola obsahuje popis implementácie systému na vybrané mobilné platformy, popis možnej implementácie pre ďalšie platformy ako aj popis implementácie vzorovej hry. Po tejto kapitole nasleduje záver práce a prílohy.

2. Analýza

2.1. Mobilné platformy

Ako som uviedol v úvode, existuje niekoľko operačných systémov, s ktorými je nutné pri vývoji aplikácií pre mobilné zariadenia počítať. V tejto kapitole ďalej nasleduje ich prehľad, spolu s krátkou históriou.

2.1.1. Symbian

Symbian je momentálne najrozšírenejší operačný systém inteligentných mobilov. V súčasnosti je open source a o vývoj tejto platformy sa stará spoločnosť Nokia. Jeho korene siahajú až do roku 1984, kedy bola uvedená platforma Psion Organiser ako prvý vreckový počítač na svete, vtedy ešte založený na 8-bitovej architektúre. V roku 1989 vydal Psion prvé zariadenie postavené na 16-bitovom operačnom systéme EPOC16. O 8 rokov neskôr bola predstavená 32-bitová platforma EPOC32, ktorá už je priamym predchodcom systému Symbian určeného pre mobily [7] [8 s. 2-3].

V roku 1998 bolo vytvorené konzorcium firiem Psion, Ericsson, Motorola a Nokia a operačný systém EPOC bol premenovaný na Symbian. Nokia 9210 Communicator z roku 2001 bola prvým telefónom s otvoreným OS Symbian. Tento operačný systém postupne získaval na obľube a v roku 2006 bol predaný stomiliónty telefón so Symbianom [7] [8 s. 2-3].

V roku 2008 odkúpila operačný systém Nokia, bola vytvorená Symbian Foundation a zo Symbianu sa postupne stala open source platforma. Napriek tomu, že je Symbian stále najrozšírenejším mobilným operačným systémom, svoj podiel postupne stráca. Nokia sa snaží o oživenie a inováciu platformy pomocou novej generácie Symbian^3 [7] [8 s. 2-3].

Na operačný systém Symbian je možné vyvíjať aplikácie v niekoľkých vývojových prostrediach a jazykoch, primárne však pomocou C++, Java ME a Flash Lite [8].

2.1.2. BlackBerry OS

BlackBerry OS bol počas celej svojej existencie doménou spoločnosti Research in Motion, ktorá ho vyvíjala a ako jediná vyrábala zariadenia na tejto platforme. V súčasnosti sa teší obľube hlavne vo firemnom prostredí. Napriek veľkému podielu trhu rovnako ako Symbian postupne stráca [5].

Začiatok BlackBerry OS sa niesol v znamení zariadenia BlackBerry 850 z roku 1998. Išlo o inteligentnejší pager rozšírený o možnosť prijímať a odosielať e-maily. Po veľkom úspechu uviedol RIM na trh o dva roky neskôr radu 857/957, ktorá sa už podobala na dnešné zariadenia BlackBerry. Z hľadiska operačného systému však nepriniesla žiadne veľké zmeny. Tie prišli až v roku 2002, keď začal BlackBerry OS plne podporovať vývoj aplikácii na platforme J2ME. Od tohto míľniku prešiel postupne BlackBerry OS evolučnými a kozmetickými zmenami ako sú postupná podpora pre farebné obrazovky, rôzne

komunikačné rozhrania a dotykové obrazovky. S verziou 6.0 umožnil v roku 2010 RIM vývoj aplikácií založených na webových štandardoch [9] [10] [11] [12].

2.1.3. Android

Android je jedným z dvojice relatívne nových mobilných operačných systémov, ktoré v súčasnosti naberajú na popularite.

V roku 2005 odkúpila spoločnosť Google firmu Android, Inc. spolu s jej operačným systémom s rovnakým názvom. O dva roky neskôr bolo vytvorené konzorcium Open Handset Alliance, združujúce mobilných operátorov, výrobcov mobilných zariadení a ich komponentov [13].

Koncom roku 2008 bol uvedený na trh prvý telefón s OS Android. Systém umožňuje vývoj aplikácií na platforme JDK a čiastočne podporuje aj vývoj v natívnom kóde [13] [14].

2.1.4. iOS

iOS je druhým mladým mobilným operačným systémom. iOS od začiatku vyvíja spoločnosť Apple, ktorá je zároveň jediným producentom mobilných zariadení na tejto platforme. Prvým mobilom postaveným na tejto platforme bol v roku 2007 iPhone. Aktuálne je dostupná už štvrtá verzia tohto operačného systému [15].

V súčasnosti je možné vyvíjať na iOS aplikácie ako natívne prostredníctvom iOS SDK založenom na jazyku Objective-C alebo ako webové aplikácie. iOS SDK je dostupný iba pre Mac OS X [16] [17] [18 s. 10-11].

2.1.5. Windows Mobile

Windows Mobile je posledným z trojice klasických mobilných operačných systémov. Napriek svojej kedysi dominantnej pozícii zaujíma dnes už iba relatívne malú časť trhu.

V roku 2000 predstavil Microsoft operačný systém Pocket PC 2000. Tento systém bol založený na jadre Windows CE 3.0 a čiastočne vychádzal z predošlej mobilnej platformy Microsoftu Palm-size PC. Verzia Pocket PC 2002 prvý krát priniesla podporu aj pre inteligentné telefóny bez dotykovej obrazovky. Významnú zmenu predstavovala verzia Windows Mobile 5 predstavená v roku 2005. Tá okrem značne zmeneného používateľského rozhrania priniesla aj podporu pre aplikácie postavené na .NET Compact Framework. Dovtedy boli podporované iba natívne aplikácie založené prevažne na jazykoch C a C++ [19] [20] [21].

Súčasná verzia Windows Mobile 6.5.3 založená na jadre Windows CE 5.2 je pravdepodobne aj poslednou, keďže Microsoft predstavil v roku 2010 operačný systém Windows Phone 7.

2.1.6. Windows Phone 7

Vo februári 2010 predstavil Microsoft nový operačný systém Windows Phone 7. Tento sa od platformy Windows Mobile radikálne líši. Okrem používateľského rozhrania, ktoré nemá

s minulou platformou prakticky nič spoločné je rozdielne aj jadro systému. Ten je postavený na Windows CE 6.0 a vyvíjať aplikácie na neho je už možné iba skrz platformu .NET, konkrétne Silverlight pre štandardné aplikácie a XNA pre hry [22].

Prvé zariadenia so systémom Windows Phone 7 sa začali predávať na prelome októbra a novembra 2010.

2.1.7. Palm webOS

Spoločnosť Palm uviedla na trh v roku 2009 nový operačný systém Palm webOS. Systém webOS je postavený na linuxovom jadre a vývoj aplikácií na neho je možný pomocou SDK pre webové aplikácie na báze JavaScriptu alebo PDK pre natívne aplikácie v jazykoch C a C++ [18]. Koncom roku 2010 odkúpila Palm korporácia Hewlett-Packard a bol vydaný webOS 2.0 [23].

2.2. Existujúce riešenia

Nápad na vytvorenie platformy, na ktorej by bolo možné vyvíjať aplikácie a špeciálne hry pre viacero mobilných platforiem zároveň nie je nový. Existuje niekoľko riešení. V tejto kapitole sa budem snažiť pokryť prehľadom najpoužívanejšie z nich a vysvetliť, prečo väčšina z nich neodpovedá mojej predstave o realizácii platformy pre multiplatformný vývoj mobilných hier.

Najväčšie problémy týchto systémov sú podpora iba niekoľkých platforiem (Unity, Corona), licenčné poplatky dané tým, že ide o komerčné riešenia (Unity, Corona, EDGELIB) alebo to, že nie sú priamo určené pre vývoj hier (PhoneGap, MoSync).

2.2.1. Unity

Unity je prostredie pre vývoj hier, ktoré nie je zamerané iba na mobilné platformy. Vo verzii 3 podporuje vývoj hier pre PC, konzoly Nintendo Wii, Xbox 360, PlayStation 3 a mobilné operačné systémy iOS a Android. Platforma Unity pozostáva z grafického vývojového prostredia pre vývoj hier a z implementácie herného enginu pre jednotlivé platformy. Vývojové prostredie je dostupné pre stolové počítače PC aj Mac [24 s. xii], [25 s. 1-2]. Hry je možné popisovať pomocou skriptov v jazykoch JavaScript, C# alebo Boo (dialekt jazyka Python) [25 s. 16] [26 s. 80].

Veľkou devízou Unity je možnosť vývoja hier, ktoré sú plne 3D. Bohužiaľ Unity momentálne podporuje z iba mobilných platforiem iba iOS a čiastočne Android. Plná podpora pre Android je stále vo vývoji [27]. Práve preto, že je Unity špecializované na vývoj 3D hier, dá sa len ťažko predpokladať rýchle rozšírenie na ďalšie platformy. Unity je navyše komerčné riešenie, v bezplatnej verzii nie je vývoj pre mobilné platformy možný. Výška licenčných poplatkov nie je zanedbateľná [28].

2.2.2. Corona

Corona je vývojová platforma spoločnosti Ansca Mobile určená špeciálne pre vývoj mobilných hier. V súčasnosti podporuje vývoj pre iOS a Android. Vývojové prostredie je

dostupné len pre Mac OS X [29]. Corona umožňuje vývoj kompletne v skriptovacom jazyku Lua a okrem iných obsahuje rozhrania pre 2D grafiku, simuláciu fyziky, či sieťovú komunikáciu [30].

Medzi nevýhody systému Corona patrí podpora iba dvoch mobilných operačných systémov, dostupnosť vývojového prostredia iba pre Mac OS X a licenčné poplatky [31].

2.2.3. EDGELIB

EDGELIB je multiplatformný systém pre vývoj mobilných aplikácií, predovšetkým hier. Systém vyvíja holandská spoločnosť Elements Interactive. EDGELIB zabezpečuje jednotné rozhranie pre vývoj aplikácií pre iOS, Windows Mobile a Symbian. Rozhranie obsahuje okrem iného podporu pre 2D a 3D grafiku, sieťovú komunikáciu alebo prácu so súbormi [32].

Hry využívajúce EDGELIB musia byť vyvinuté v C++, čo automaticky diskvalifikuje mobilné platformy, ktoré nepodporujú preklad aplikácií z tohto jazyka. Okrem toho ide rovnako ako v prípade predchádzajúcich prostredí o komerčné riešenie s relatívne vysokými licenčnými poplatkami [33].

2.2.4. PhoneGap

PhoneGap je open source platforma pre vývoj mobilných aplikácií vyvíjaná spoločnosťou Nitobi. V súčasnosti podporuje simultánny vývoj pre mobilné operačné systémy iOS, Android, BlackBerry OS, Symbian a webOS. Bohužiaľ PhoneGap nie je určený priamo pre vývoj hier a vývoj v ňom prebieha na báze webových technológií ako HTML, CSS a JavaScript [34] [35]. I keď PhoneGap umožňuje prístup k niektorým prostriedkom hostiteľského zariadenia, napríklad podporu pre akcelerovanú grafiku neobsahuje [36].

Systém PhoneGap by pravdepodobne bolo možné využiť na vývoj veľmi jednoduchých multiplatformných mobilných hier. Na vývoj robustnejších hier sa však nehodí, kvôli absencii rozhraní pre prácu s grafikou a použitým vývojovým technológiám.

2.2.5. MoSync

MoSync je vývojová platforma pre vývoj aplikácií pre mobilné operačné systémy iOS, Android, Moblin, Windows Mobile a Symbian. Vývojové prostredie je založené na prostredí Eclipse, vývoj prebieha v jazykoch C, C++ s podporou pre skriptovacie jazyky [37].

MoSync je open source a je zdarma pre vývoj aplikácií, ktoré sú taktiež open source [38]. Bohužiaľ rovnako ako PhoneGap nie je ani MoSync určený primárne pre vývoj hier. Na rozdiel od prvej menovanej platformy je MoSync postavený na jazyku kompilovanom do natívneho kódu zariadenia a obsahuje aspoň základné knižnice pre prácu s grafikou, zvukmi a podobne [39].

2.2.6. ScummVM

I keď nejde o vývojovú platformu, rád by som spomenul ešte ScummVM, pretože je z hľadiska vývoja pre viacero platforiem zaujímavý.

Scumm (Script Creation Utility for Maniac Mansion) je skriptovací jazyk a systém vyvíjaný spoločnosťou LucasArts a používaný v rokoch 1987 až 1998. Na tejto platforme bolo postavených množstvo známych hier, ako napríklad Maniac Mansion, Sam & Max Hit the Road, či séria Monkey Island od LucasArts, séria Goblins od Coktel Vision alebo série Quest for Glory a Leisure Suit Larry od firmy Sierra [40].

ScummVM je virtuálny stroj pre hry postavené na platforme Scumm. V súčasnosti existuje verzia pre veľké množstvo platforiem, okrem iných aj mobilných ako Android, iOS, Windows Mobile, Maemo, Symbian, PalmOS [41] [42]. Pri spúšťaní hier pod ScummVM pri tom nie je potrebné herné súbory nijak modifikovať [43].

I keď Scumm nie je z hľadiska vývoja nových hier v dnešnej dobe už zaujímavý, ScummVM ukazuje, že vývoj hier fungujúcich na niekoľkých mobilných platformách zároveň je reálny.

2.3. Výber skriptovacieho jazyka

Keďže neexistuje jazyk, ktorý by sa dal priamo kompilovať do natívneho kódu každého mobilného operačného systému, bude musieť byť hra postavená na MPME popísaná nejakým skriptovacím jazykom. Keďže vytváranie kompilátoru pre tento jazyk na každú platformu zvlášť by bolo pomerne náročné, bude tento jazyk nejakým spôsobom interpretovaný.

Aby bol skriptovací jazyk vhodný pre účely použitia v MPME, musí spĺňať niekoľko podmienok:

- **Rýchlosť** – Keďže mobilné zariadenia disponujú limitovanými prostriedkami ako množstvo pamäte a výkon procesora, musí byť jazyk dostatočne efektívny. Pretože volania medzi skriptovacím prostredím a natívnym prostredím MPME budú pomerne časté, musia byť vykonávané s minimálnou réžiou.
- **Portabilita** – Jazyk musí byť nezávislý na platforme, na ktorej beží. Zároveň musia existovať prostriedky na to, aby bolo možné jazyk jednoducho začleniť do vývojových prostredí rozličných mobilných platforiem.
- **Popis dát** – Návrh MPME nepredpokladá žiadny dedikovaný mechanizmus na popis dát. Skriptovací jazyk musí byť teda vhodný aj na tento účel.

S ohľadom na tieto požiadavky nasleduje prehľad vybraných skriptovacích jazykov.

2.3.1. Python

Python začal vyvíjať koncom 80-tych rokov Guido van Rossum v výskumnom centre CWI [44]. Python podporuje niekoľko programátorských paradigiem a štýlov, ako klasické štruktúrované programovanie, objektovo orientované programovanie a čiastočne aj funkcionálne programovanie [45].

Originálna implementácia jazyka Python obsahuje kompilátor a virtuálny stroj a je napísaná v jazyku C. Taktiež existujú implementácie pre platformy Java a .NET [46]. Licencia pre

Python umožňuje jeho využitie pre ľubovoľný účel [47]. Dokumentácia a dostupná literatúra k jazyku sa na veľmi vysokej úrovni.

Jazyk Python je veľmi sofistikovaný a komplexný, čo je jeho výhodou, ale aj nevýhodou. Štandardná knižnica napríklad obsahuje množstvo rozhraní pre podporu vývoja webových aplikácií [48], ktoré by pre vývoj mobilných hier boli úplne zbytočné. Originálna implementácia Python 3.1.3, čo je v čase písania tejto práce aktuálna verzia, obsahuje 1,5 MB zdrojového kódu. Implementácia virtuálneho stroja na platformy, ktoré nepodporuje by bola pomerne časovo náročná. Preto som pri výbere preferoval jednoduchší skriptovací jazyk.

2.3.2. Ruby

Jazyk Ruby začal vyvíjať v polovici 90-tych rokov Yukihiro Matsumoto. Jazyk Ruby je dynamický, reflektívny, objektovo orientovaný a čiastočne funkcionálny [49].

Originálna implementácia jazyka interpretuje zdrojový kód Ruby priamo, bez kompilácie do medzikódu. Je napísaná v jazyku C a dostupná pod licenciou GPL alebo Ruby licenciou, ktorá s obmedzeniami umožňuje využitie jazyka pre ľubovoľný účel. Dostupné sú aj implementácie JRuby pre platformu Java a IronRuby pre platformu .NET [50] [51] [49 s. 11-12].

Kompletná špecifikácia jazyka neexistuje, považuje sa za ňu originálna implementácia. O jazyku bolo vydané množstvo publikácií [52].

Ruby je, rovnako ako Python, veľmi silný a sofistikovaný jazyk. Pre použitie v mobilnom prostredí by bol príliš komplikovaný. Rovnako implementácia pre nové platformy by bola časovo náročná, zdrojový kód v jazyku C k originálnej implementácii má nezanedbateľných 20 MB.

2.3.3. GameMonkey Script

GameMonkey Script je skriptovací jazyk vyvíjaný Matthewom Riekom a Gregom Douglasom, určený špeciálne pre použitie v prostredí hier. Má syntax podobnú jazyku C a pôvodne bol inšpirovaný jazykom Lua [53 s. 165, 166] [54].

Oproti jazyku Lua ponúka GameMonkey Script jednoduchší prístup natívnych (C, C++) funkcií k skriptovým a naopak. Ďalšou výhodou je podpora pre celé aj reálne čísla. Zdrojový kód pre kompilátor aj virtuálny stroj je voľne dostupný a použiteľný pre ľubovoľné účely pod licenciou MIT [53 s. 165, 166] [54].

Dvoma veľkými nevýhodami tohto jazyka sú viazanosť na hosťujúcu platformu v jazyku C alebo C++ a všeobecne málo dostupnú dokumentáciu alebo publikácie.

2.3.4. Lua

Vývoj jazyka Lua začal v roku 1993 na Pontifícia Universidade Católica do Rio de Janeiro [55 s. xiii] [56 s. 310]. V súčasnosti je použitie jazyka Lua v prostredí vývoja hier veľmi

rozšírené. Používajú ho vysoko-rozpočtové hry ako Crysis, World of Warcraft, či Baldur's Gate no aj menšie hry nezávislých štúdií ako napríklad Aquaria [57] [58].

Jazyk Lua je spolu so základnou implementáciou kompilátora a virtuálneho stroja poskytovaný úplne zadarmo na ľubovoľné účely pod licenciou MIT [59]. Základná implementácia je navyše napísaná s ohľadom na maximálnu portabilitu v čistom ANSI C a preto je možné priamo ju využiť v platformách, ktoré umožňujú kompiláciu niektorého z jazykov odvodených od C [60 s. 3].

Virtuálny stroj Lua nie je, na rozdiel od mnohých iných virtuálnych strojov, založený na zásobníku, ale na registroch. Vďaka tomu je veľmi efektívny pri prístupe k lokálnym premenným, keďže nepotrebuje obvyklé metódy zásobníkových strojov *push* a *pop* [60]. Registrové virtuálne stroje sú podľa [61] významne rýchlejšie ako zásobníkové virtuálne stroje.

Lua bola navrhnutá okrem iného aj pre účel definície dát [62 s. 2-3] a často sa pre tento účelu aj využíva [60 s. 3] [55 s. 4].

Dostupná dokumentácia a literatúra je na veľmi vysokej úrovni, na mnoho zdrojov sa odkazujem v tejto práci. Využiteľná implementácia virtuálneho stroja Lua už existuje pre mnoho rôznych platforiem [63]. Navyše implementácia virtuálneho stroja pre ďalšie platformy je spomedzi ostatných spomínaných jazykov najjednoduchšia. Zdrojový kód k originálnej implementácii Lua 5.1 má sympatických 440 kB.

S ohľadom na všetky hľadiská považujem jazyk Lua za najvhodnejšieho kandidáta pre skriptovací jazyk v prostredí MPME. Systém MPME predpokladá využitie verzie jazyka 5.1, ktorá je v čase písania tejto práce aktuálnou verziou. Vlastný virtuálny stroj pre tento jazyk som implementoval v rámci implementácie MPME pre Windows Phone 7. Implementačné detaily jazyka sú popísané v kapitole 4.

2.4. Požiadavky na systém

Ako som spomenul v úvode, systém MPME bol navrhnutý pre hry s dvojrozmernou grafikou. Rovnako podporuje aj jednoduchú prácu s textom, zvukmi, používateľskými vstupmi a používateľským rozhraním.

Pri štúdiu mobilných hier a existujúcich prostredí pre ich vývoj som došiel k záveru, že pri návrhu MPME bolo potrebné s ohľadom na typické požiadavky hier a jednoduchosť ich vývoja brať do úvahy nasledujúce prvky:

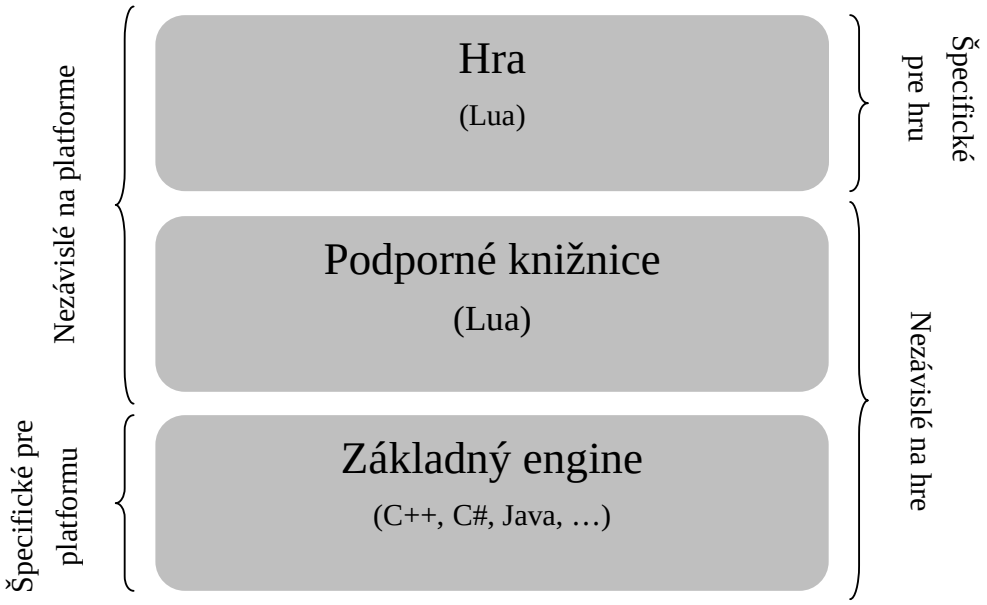
- **Virtuálny stroj Lua** – Systém musí podporovať interpretáciu jazyka Lua, resp. Lua bytecode. Navyše bude rozhranie systému dostupné herným Lua skriptom.

- **Manažment obsahu** – Systémy súborov jednotlivých cieľových platforiem sa môžu značne líšiť, preto bude súčasťou návrhu MPME aj návrh pre jednotný prístup k obsahu hry ako grafické prvky, zvukové efekty, Lua skripty a podobne.
- **Časovanie** – Systém zabezpečuje vhodné časovanie pre hru, aby nebolo potrebné jeho implementáciu riešiť na úrovni skriptov.
- **Vstup používateľa** – MPME musí kontrolovať vstupy používateľa a jednotne ich predávať herných skriptom na spracovanie.
- **Grafika** – Systém navrhuje jednotný systém pre prácu s dvojrozmernou grafikou a jednoduchou manipuláciou s grafickými elementmi.
- **Text** – Aby nebolo nutné v rámci skriptov zložito implementovať vykresľovanie textov, ich vykresľovanie podporuje už priamo MPME.
- **Zvuky** – Súčasťou návrhu je jednoduché rozhranie pre prehrávanie zvukových efektov.
- **Používateľské rozhranie** – Aby nebolo potrebné vyvíjať prostriedky pre vytváranie používateľského rozhrania v každej hre osobitne, je súčasťou návrhu MPME aj rozhranie pre jednoduché vytváranie UI.

3. Architektúra

Architektúra hry založenej na platforme MPME pozostáva z troch vrstiev, ako schematicky znázorňuje obrázok 1.

Obrázok 1: Vrstvy architektúry



Prvou vrstvou je základná implementácia enginu MPME. To, ako je táto vrstva implementovaná závisí od platformy, na ktorú sa implementuje, jej rozhranie je však pre každú hru postavenú na MPME rovnaké. Táto vrstva ma za úlohu spúšťať Lua skripty vo virtuálnom stroji a zároveň týmto skriptom poskytnúť množinu funkcií v rámci rozhrania. Táto vrstva zabezpečuje aj časovanie aplikácie. Ďalej sa budem na túto vrstvu odvolávať ako na **základnú vrstvu**.

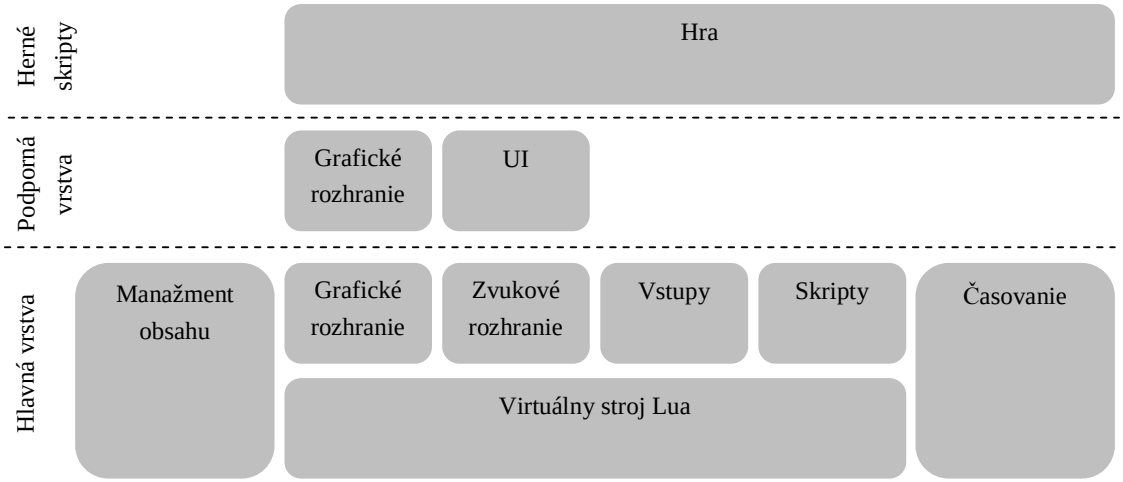
Druhou vrstvou sú podporné knižnice. Tie sú taktiež rovnaké pre každú hru a pretože sú v jazyku Lua, sú nezávislé aj na platforme. Funkcie, ktoré tieto knižnice popisujú rozširujú základné funkcie poskytované prvou vrstvou. Účelom tejto vrstvy je zamedziť nutnosti implementácie funkcií rozhrania, ktoré nie sú priamo závislé na konkrétnom operačnom systéme v prvej vrstve. Ďalej sa budem na túto vrstvu odvolávať ako na **podpornú vrstvu**. Podporná vrstva je transparentná, skripty z hernej vrstvy môžu používať všetky funkcie základnej vrstvy rovnako ako podpornej.

Treťou vrstvou sú samotné herné skripty. Tie sú nezávislé na platforme, ale špecifické pre každú konkrétnu hru. Herné skripty využívajú funkcie rozhrania, ktoré poskytujú prvé dve

vrstvy a zároveň sami definujú požadované herné funkcie. Ďalej sa budem na túto vrstvu odvolávať ako na **vrstvu herných skriptov** alebo zjednodušene ako na **herné skripty**.

Štruktúru architektúry spolu s jednotlivými rozhraniami zobrazuje obrázok 2. Jednotlivé súčasti sú podrobne rozobraté v nasledujúcich podkapitolách.

Obrázok 2: Architektúra a rozhrania



3.1. Virtuálny stroj Lua

Keďže všetky hry nad MPME budú popísané skriptovacím jazykom Lua, základnou požiadavkou je možnosť interpretácie tohto jazyka prostredníctvom virtuálneho stroja. Tento stroj musí podporovať jazyk Lua kompletne, s výnimkou v podobe štandardných funkcií. Niektoré štandardné funkcie totiž umožňujú zasahovať do konkrétnej implementácie virtuálneho stroja a s inou než originálnou implementáciou by nemuseli fungovať rovnako, prípadne by ich implementácia bola zložitá. Takéto funkcie nemajú pre potreby vývoja hier využitie a systém ich nemusí podporovať. Kompletný zoznam vyžadovaných štandardných Lua funkcií sa nachádza v prílohe A.

Okrem štandardných Lua funkcií implementácia MPME poskytuje prostredníctvom virtuálneho stroja Lua aj funkcie pre vývoj hier.

To, či sa budú na zariadení interpretovať zdrojové skripty Lua alebo skompilovaný Lua bytecode určujú jednotlivé implementácie MPME. Moja implementácia pre Windows Mobile 6 Standard interpretuje zdrojové skripty Lua. Implementácia pre Windows Phone 7 interpretuje Lua bytecode, do ktorého sú kompilované Lua skripty počas kompilácie celej aplikácie založenej na MPME.

3.2. Manažment obsahu

Systémy súborov alebo odkazov na prostriedky aplikácie jednotlivých cieľových platforiem sa môžu značne líšiť, preto ich MPME abstrahuje. Pri odkazovaní sa na konkrétny prostriedok hry používa hra systém nezávislý na platforme. Systém podporuje nasledujúce typy prostriedkov:

- **Textúry** – 2D obrázky s podporou čiastočnej priehľadnosti, vo formáte 24-bitový PNG.
- **Zvuky** – Zvukové efekty vo formáte WAV.
- **Skripty** – Skripty Lua, do projektu prikladané ako zdrojový kód (neskompilované).

Štandardná cesta k prostriedku sa skladá z imaginárnych adresárov oddelených lomkou. Na začiatku reťazca sa lomka neuvádza. Po poslednom imaginárnom adresáre nasleduje názov imaginárneho súboru. Tento sa uvádza bez prípony. Cesta k prostriedku aplikácie teda môže vyzerať napríklad nasledovne:

AirBattles.Content/Textures/Aircraft/Airplane1

Všetky funkcie rozhrania, ktoré pracujú s prostriedkami aplikácie očakávajú cestu k obsahu v tomto formáte. To ako sú jednotlivé zdroje uložené v projekte a na zariadení určuje konkrétna implementácia MPME. Konkrétny prostriedok musí byť samozrejme na rozličných implementáciách dostupný pod rovnakou cestou.

3.3. Časovanie

Jednou zo základných funkcií, ktoré musí implementácia MPME zabezpečiť je korektné časovanie. Hry postavené na MPME budú bežať v konštantnej počte krokov za sekundu. Implicitná je hodnota 25 krokov za sekundu. To dáva aplikácii v každom kroku 40 milisekúnd na výpočty a vykreslenie. Fixné tempo som zvolil z toho dôvodu, aby sa predišlo počítaniu s uplynutým časom. Výhodou je, že sa tak značne zníži počet aritmetických výpočtov, ktoré by boli potrebné pre výpočet variabilnej simulácie v každom kroku. Rovnako to uľahčí vývoj hier na MPME, keďže herný vývojár sa časovaním nemusí nijako zaoberať. Nevýhodou zas, že ak kroky budú trvať viac ako je čas rezervovaný pre každý krok, hra sa spomalí.

Hra môže kedykoľvek zmeniť tempo pomocou funkcie uvedenej v tabuľke 4.

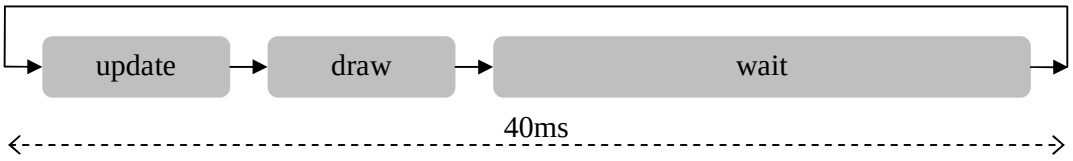
Tabuľka 4: Funkcia pre zmenu časovania

Názov	Popis
engine.changeFPS(fps)	Zmení limit na počet krokov za sekundu.

Keďže aritmetické výpočty na mobilných zariadeniach sú stále relatívne drahé, považujem to za dobrý kompromis. Využitie fixného tempa odporúčajú aj [8 s. 46-49] a [9 s. 139].

Každá implementácia MPME musí teda v hlavnej slučke zabezpečiť, že ak výpočty a vykresľovanie zaberú menej ako čas vyhradený pre každý krok (implicitne 40 milisekúnd), tak zvyšok tohto intervalu strávi aplikácia v nečinnosti. Diagram hlavnej slučky implementácie je znázornený na obrázku 3.

Obrázok 3: Hlavná slučka MPME



3.4. Používateľské vstupy

Implementácia MPME musí zabezpečiť zachytávanie vstupov používateľa a predať ich na spracovanie herným skriptom. Každé mobilné zariadenie je v dnešnej dobe vybavené dotykovou obrazovkou alebo fyzickými tlačidlami, prípadne obidvoma. Systém predpokladá využitie dvoch profilov pre používateľský vstup.

Prvým profilom je profil pre dotykovú obrazovku. Aby zariadenie podporovalo tento profil, musí podporovať registrovanie aspoň jedného dotyku na obrazovke. Ak zariadenie podporuje viacero simultánnych dotykových bodov, tieto sa ignorujú a použije sa iba prvý z nich. Údaje o vstupoch na dotykovom displeju sa v každom kroku predávajú herným skriptom pomocou funkcie *update* (viac v kapitole Herné skripty), prostredníctvom tabuľky *touch*, ktorej hodnoty sú popísané v tabuľke 5.

Tabuľka 5: Hodnoty tabuľky *touch*

Hodnota	Význam
touch.active	Určuje, či je registrovaný dotykový bod na zariadení, teda či sa používateľ dotýka displeja. Výraz má vždy booleovskú hodnotu.
touch.x	Určuje súradnicu X registrovaného dotyku. Hodnota má zmysel, iba ak je dotyk registrovaný. Výraz má vždy číselnú hodnotu.
touch.y	Určuje súradnicu Y registrovaného dotyku. Hodnota má zmysel, iba ak je dotyk registrovaný. Výraz má vždy číselnú hodnotu.

Druhým profilom je profil pre fyzické tlačidlá. Aby zariadenie tento profil podporovalo, musí disponovať štvorsmerným krížom, potvrdzovacím tlačidlom a aspoň štyrmi ďalšími tlačidlami. Údaje o vstupoch sa predávajú herným skriptom prostredníctvom tabuľky *keypad*, ktorej hodnoty sú popísané v tabuľke 6.

Tabuľka 6: Hodnoty tabuľky *keypad*

Hodnota	Význam
keypad.left	Určuje, či je stlačený smerový kríž zariadenia smerom doľava. Výraz má vždy booleovskú hodnotu.

Hodnota	Význam
keypad.right	Určuje, či je stlačený smerový kríž zariadenia smerom doprava. Výraz má vždy booleovskú hodnotu.
keypad.up	Určuje, či je stlačený smerový kríž zariadenia smerom hore. Výraz má vždy booleovskú hodnotu.
keypad.down	Určuje, či je stlačený smerový kríž zariadenia smerom dole. Výraz má vždy booleovskú hodnotu.
keypad.enter	Určuje, či je stlačené hlavné potvrdzovacie tlačidlo zariadenia. Výraz má vždy booleovskú hodnotu.
keypad.action1	Určuje, či je stlačené tlačidlo pre prvú akciu. Odporúča sa využitie ľavého kontextového tlačidla, ak takým zariadenie disponuje. Výraz má vždy booleovskú hodnotu.
keypad.action2	Určuje, či je stlačené tlačidlo pre druhú akciu. Odporúča sa využitie pravého kontextového tlačidla, ak takým zariadenie disponuje. Výraz má vždy booleovskú hodnotu.
keypad.action3	Určuje, či je stlačené tlačidlo pre tretiu akciu. Odporúča sa využitie tlačidla späť, ak takým zariadenie disponuje. Výraz má vždy booleovskú hodnotu.
keypad.action4	Určuje, či je stlačené tlačidlo pre štvrtú akciu. Odporúča sa využitie tlačidla domov, ak takým zariadenie disponuje. Výraz má vždy booleovskú hodnotu.

Zariadenie musí z týchto profilov podporovať minimálne jeden, môže však podporovať obidva zároveň. Počas celého behu aplikácie musia byť podporované rovnaké profily. Aby mohol herný skript zistiť, aké profily zariadenie podporuje a podľa toho prípadne prispôsobiť používateľské rozhranie, odhaľuje implementácia MPME funkcie podľa tabuľky 7.

Tabuľka 7: Funkcie rozhrania pre interakciu s používateľom

Názov	Popis
device.hasTouchscreen()	Vracia hodnotu TRUE, ak zariadenie plne podporuje profil pre ovládanie dotykovým displejom, inak vracia FALSE.
device.hasKeypad()	Vracia hodnotu TRUE, ak zariadenie plne podporuje profil pre ovládanie fyzickými tlačidlami, inak vracia FALSE.

3.5. Grafické rozhranie

Jednou z hlavných úloh MPME je umožniť jednotný prístup ku grafickému rozhraniu zariadenia. Systém umožňuje manipuláciu s 2D grafikou. Základným grafickým prvkom je textúra. Tá môže byť polopriesvitná a pred vykreslením na displej je možné ju transformovať.

Keďže rôzne mobilné zariadenia disponujú rôznymi rozlíšeniami displejov, bude podporná vrstva poskytovať virtuálny display. Tento display má garantované minimálne rozlíšenie 640×480 (resp. 480×640) obrazových bodov. Pre pevne dané rozlíšenie som sa rozhodol

preto, aby bolo možné jednotlivé implementácie graficky optimalizovať. Hra teda taktiež môže počítať s týmto minimálnym rozlíšením a rozvrhnúť podľa toho zobrazovanú scénu.

Virtuálny displej zabezpečujú funkcie podpornej vrstvy. Idea je taká, že ak bude mať fyzická obrazovka zariadenia natívne rozlíšenie nižšie ako 640×480 (resp. 480×640) obrazových bodov, všetky grafické objekty sa budú vykresľovať v polovičnej veľkosti. MPME teda podporuje dva grafické profily – plný (full-size) a polovičný (half-size).

Základným grafickým prvkom je v MPME textúra. Na textúry sa odkazuje štandardnou cestou k prostriedku, ako je uvedené v kapitole o manažmente obsahu. MPME umožňuje vykresľovanie textúr spolu s ich rotáciou a zmenšovaním, resp. zväčšovaním. Všetky textúry, ktoré MPME spracováva musia byť vo formáte PNG. MPME podporuje textúry s alfa zložkou pre polopriesvitnosť.

Zoznam funkcií grafického rozhrania je v tabuľke 8. Ich podrobný popis, spolu s presne definovaným správaním sa nachádza v prílohe B.

Tabuľka 8: Prehľad funkcií grafického rozhrania

Názov	Popis
engine.clearScreen(r, g, b)	Prekreslí celý displej zvolenou farbou.
engine.loadTexture(name)	Načíta textúru.
engine.getTextureNativeResolution(texture)	Vráti natívne rozlíšenie textúry.
engine.getTextureResolution(texture)	Vráti virtuálne rozlíšenie textúry podľa aktuálneho grafického profilu.
engine.drawTextureSimple(texture, x, y)	Vykreslí textúru bez zmenšovania/zväčšovania a rotácie.
engine.drawTextureResampled(texture, x, y, angle, scale)	Vykreslí textúru so zmenšovaním/zväčšovaním a rotáciou.
engine.drawTexture(texture, x, y [, angle [, scale]])	Vykreslí textúru (voliteľne) so zmenšovaním/zväčšovaním a (voliteľne) rotáciou. Pri tom berie do úvahy grafický profil a podľa neho vykresľuje textúru v 50% alebo 100% mierke.
device.getNativeResolution()	Vráti natívne rozlíšenie displeja s ohľadom na jeho aktuálnu rotáciu.
device.getResolution()	Vráti virtuálne rozlíšenie displeja s ohľadom na jeho aktuálnu rotáciu a grafický profil.

3.6. Text

Úzko spojené s grafikou je aj vykresľovanie textu. Pre zjednodušenie použitia v hre zavádza MPME tri preddefinované fonty na vykresľovanie textu.

Presnú vizualizáciu každého fontu určujú jednotlivé implementácie systému, no kvôli uľahčeniu manipulácie s umiestnením textu zavádza MPME dva koncepty. Prvým je pevná

výška textu vykreslených jednotlivými fontmi a druhým možnosť získania šírky textu v danom fonte pomocou funkcie. Horizontálny presah textu takto môže vývojár hry jednoducho identifikovať a text jednoducho zalomiť, ale pri vertikálnom presahu by takúto možnosť nemal. Z tohto dôvodu som sa rozhodol pre pevnú výšku a variabilnú šírku.

Základné fonty majú výšku vykresľovaného textu zadefinovanú v tabuľke 9.

Tabuľka 9: Zoznam podporovaných fontov

Font	Výška
fonts.normal	20 virtuálnych obrazových bodov
fonts.big	28 virtuálnych obrazových bodov
fonts.small	13 virtuálnych obrazových bodov

Za virtuálne obrazové body sa považujú obrazové body virtuálneho displeja. Šírka znakov vo fontoch nie je pevne definovaná, rovnako nie je definované ani či ide o fonty s pevnou alebo premenlivou šírkou znakov.

V tabuľke 10 nasleduje zoznam funkcií pre zobrazovanie textu. Ich podrobný popis, spolu s presne definovaným správaním sa nachádza v prílohe B.

Tabuľka 10: Prehľad funkcií pre prácu s textom

Názov	Popis
fonts.normal	Font pre text strednej veľkosti.
fonts.big	Font pre veľký text.
fonts.small	Font pre malý text.
engine.drawTextSimple(text, font, x, y, r, g, b)	Vykreslí text daným fontom a zvolenou farbou na pozícií zadanej reálnymi obrazovkovými koordinátami.
engine.drawText(text, font, x, y, r, g, b)	Vykreslí text daným fontom a zvolenou farbou na pozícií zadanej virtuálnymi obrazovkovými koordinátami.
engine.getNativeTextWidth(text, font)	Vráti šírku textu vykresleného daným fontom v reálnych obrazovkových bodoch.
engine.getTextWidth(text, font)	Vráti šírku textu vykresleného daným fontom vo virtuálnych obrazovkových bodoch.
engine.getFontHeight(font)	Vráti výšku ľubovoľného textu vykresleného daným fontom vo virtuálnych obrazovkových bodoch.

3.7. Zvukové rozhranie

MPME obsahuje jednoduché rozhranie pre prácu so zvukmi. Rozhranie je určené hlavne na prehrávanie krátkych zvukových efektov počas hry alebo v rámci používateľského rozhrania. Na zvuky sa v hre odkazuje štandardnými cestami k prostriedkom aplikácie podľa popisu v kapitole o manažmente obsahu.

V tabuľke 11 sa nachádza prehľad funkcií zvukového rozhrania. Podrobný popis a presne definované správanie týchto funkcií sa nachádza v prílohe B.

Tabuľka 11: Prehľad funkcií zvukového rozhrania

Názov	Popis
engine.loadSound(name)	Načíta zvukový efekt.
engine.playSound(sound)	Prehrá poskytnutý zvukový efekt.

3.8. Používateľské rozhranie

Podporná vrstva MPME obsahuje rozhranie pre rýchle vytváranie obrazoviek používateľského rozhrania. Prehľad objektov (tabuliek) pre prácu s používateľským rozhraním sa nachádza v tabuľke 12.

Tabuľka 12: Prehľad tabuliek používateľského rozhrania

Názov	Popis
ui.window	Reprezentuje okno používateľského rozhrania.
ui.button	Reprezentuje tlačidlo v okne.
ui.label	Reprezentuje popis v okne (štítok).

Základným prvkom používateľského rozhrania je okno. Do toho je možné umiestniť tlačidlá a štítky. Všetky dcérske elementy okno automaticky umiestni pod seba. Rovnako zabezpečuje aj správne ošetrenie používateľských vstupov.

Každý prvok obsahuje funkciu *new*, ktorá sa používa na vytváranie novej inštancie daného prvku. Dcérske prvky okna sa umiestňujú do hodnoty (tabuľky) *elements* okna.

Obrazovky menu demonštračnej hry Air Battles sú vytvorené pomocou tohto systému.

3.9. Herné skripty

Herné skripty musia implementovať tri základné funkcie uvedené v tabuľke 13.

Tabuľka 13: Základné funkcie vyžadované v herných skriptoch

Názov	Popis
loadContent()	Hra načíta potrebné prostriedky.
update(touch, keypad)	Hra spracuje používateľské vstupy a vykoná jeden herný krok.
draw()	Hra vykreslí aktuálnu scénu.

Funkciu *loadContent* volá MPME iba raz a to po základnej inicializácii. V tejto funkcií má hra možnosť načítať základné prostriedky, ktoré počas svojho behu vyžaduje.

Funkciu *update* sa volá v každom kroku. Tejto funkcii predáva implementácia MPME stav používateľských vstupov vo formáte špecifikovanom v podkapitole 3.4. V tejto funkcií hra implementuje výpočty pre uskutočnenie ďalšieho herného kroku a ošetruje používateľské vstupy.

Funkcia *draw* je volaná v každom kroku po funkcii *update*. V tejto funkcii má hra implementovať vykresľovanie aktuálnej scény.

Aby nebolo nutné mať logiku celej hry iba v hlavnom hernom skripte, obsahuje každá implementácia MPME aj funkciu pre načítanie a vykonanie skriptu. Táto funkcia je uvedená v tabuľke 14. Na vstupe očakáva cestu k skriptu v štandardnom formáte pre odkazovanie na prostriedky hry. Túto funkciu je možné využiť aj na načítavanie skriptov, ktoré obsahujú herné dáta.

Tabuľka 14: Funkcia pre spúšťanie skriptov

Názov	Popis
engine.runScript(name)	Načíta a okamžite vykoná herný skript.

3.10. Cieľové mobilné zariadenie

Aby bolo možné MPME implementovať podľa navrhnutého popisu, musí cieľové zariadenie spĺňať nasledujúce minimálne požiadavky:

- Rozlíšenie displeja aspoň 320×240, resp. 240×320 obrazových bodov.
- Dotykový displej alebo tlačidlá pokrývajúce profil pre ovládanie klávesnicou.

Tieto podmienky nepovažujem za veľmi reštriktívne. Spĺňa ich drvivá väčšina mobilných zariadení aktuálnej aj predošlej generácie, návrh MPME teda z toho hľadiska na multiplatformnosti neutrpí.

4. Implementácia

Na vzorovú implementáciu MPME som si vybral 2 platformy, ktoré sa od seba maximálne odlišujú a budú dobre demonštrovať použiteľnosť navrhnutého riešenia. Ide o platformy Windows Phone 7 a Windows Mobile 6 Standard. Napriek tomu, že by sa podľa názvu mohlo zdať, že ide o veľmi podobné operačné systémy, opak je pravdou.

Windows Phone 7 je nový operačný systém uvedený na trh spoločnosťou Microsoft v roku 2010. Zariadenia s týmto operačným systémom musia mať dotykový displej s rozlíšením 480×800 obrazových bodov, výkonný 1 GHz procesor a len minimálny počet tlačidiel (6). Na Windows Phone 7 je možné vyvíjať aplikácie iba pomocou vývojových platforiem Silverlight a XNA, vývoj natívnych aplikácií nie je možný [64 s. 2-5] [22 s. 2-7].

Windows Mobile 6 je starší, ale ešte stále používaný operačný systém, rovnako od spoločnosti Microsoft. Vo verzii Standard ide o zariadenia bez dotykového displeja, ovládané obvykle klasickou telefónnou klávesnicou. Rozlíšenia displejov sú nižšie, zvyčajne okolo 240×320 obrazových bodov [65] [21 s. 7-9].

Taktiež je súčasťou tejto práce aj implementácia demonštračnej hry Air Battles. Vďaka jednotlivým implementáciám je možné túto hru spustiť v podstate na piatich platformách:

- Zariadenie s Windows Phone 7
- Zariadenie s Windows Mobile 6 Standard
- Emulátor Windows Phone 7
- Emulátor Windows Mobile 6 Standard
- PC s OS Windows

V nasledujúcich kapitolách popíšem, ako som implementoval MPME na platformách Windows Phone 7 a Windows Mobile 6 Standard a načrtnem, ako by bolo možné implementovať MPME na ďalšie dve populárne platformy – Android a Symbian. Ďalej popíšem implementáciu hry Air Battles.

Informácie a používaní projektov, ktoré sú súčasťou tejto práce sa nachádzajú v prílohe E.

4.1. Windows Phone 7

Implementácia MPME pre operačný systém Windows Phone 7 je napísaná v jazyku C# a postavená na platforme XNA, čo je štandardná vývojová platforma pre Windows Phone 7 hry [22]. Pri implementácii MPME pre Windows Phone 7 bolo potrebné implementovať virtuálny stroj Lua pre túto platformu. Pre jednotlivé rozhrania, ako grafické, zvukové a používateľské vstupy som využil prostriedky, ktoré ponúka XNA.

Taktiež na uloženie obsahu (prostriedkov) hry som zvolil postup, ktorý umožňuje XNA a pre všetky prostriedky je vytvorený XNA Content Project.

4.1.1. Virtuálny stroj Lua

Pretože platforma Windows Phone 7 neumožňuje kompilovať aplikácie do natívneho kódu a neumožňuje vývoj v jazykoch C alebo C++ [22 s. 2-4], nebolo možné použiť v implementácii MPME pre Windows Phone 7 originálnu implementáciu virtuálneho stroja Lua. Rozhodol som sa preto implementovať vlastný virtuálny stroj v jazyku C#.

Štandardne sa pri spúšťaní Lua skriptov najprv zdrojové súbory skompilujú do Lua bytecode a až ten sa spúšťa na samotnom virtuálnom stroji Lua [62 s. 6]. Pretože lexikálna a syntaktická analýza sú časovo náročné, nie je vhodné kompilovať Lua skripty do Lua bytecode počas behu aplikácie, ale výhodnejšie je ich predkompilovať počas kompilácie celej aplikácie.

Keďže by implementácia kompilátora bola navyše pracná a pomerne zbytočná, rozhodol som sa, že budem implementovať iba samotný virtuálny stroj Lua, ktorý bude vedieť interpretovať Lua bytecode.

Štandardný kompilátor Lua kompiluje zdrojové skripty do Lua bytecode, ktorý pozostáva z inštrukcií celkom tridsiatich siedmich typov [66 s. 4]. Pre každú funkciu definovanú v zdrojovom kóde sa vytvára jej prototyp, pričom sa za samostatnú funkciu považuje kód na najvyššej úrovni (tzv. globálna funkcia). Virtuálny stroj, ktorý som implementoval podporuje všetky tieto typy inštrukcií.

Jazyk Lua má dynamické typovanie, všetky premenné sú vždy typu Variant a môžu nadobúdať ľubovoľné hodnoty týchto typov [55 s. 9-17]:

- **Nil** – Typ a zároveň hodnota, jej hlavným účelom je byť rozdielna od všetkých iných hodnôt.
- **Boolean** – Logické hodnoty *true* / *false*.
- **Number** – Reálne číslo, jediný číselný typ, ktorý Lua štandardne podporuje [55 s. 10].
- **String** – Textový reťazec.
- **Table** – Tabuľka, jediný štruktúrovaný dátový typ, ktorý Lua pozná. Ide o asociatívne pole s dynamickou veľkosťou, indexované ľubovoľnou hodnotou (okrem nil) a obsahujúce ľubovoľnú hodnotu.
- **Function** – Funkcie sú v jazyku Lua hodnotami.
- **Userdata, coroutines** – Hodnoty z hosťujúceho prostredia jazyka.

Pri implementácii dátového typu Variant som preferoval rýchlosť pred úsporou pamäte. Trieda reprezentujúca dátový typ Variant obsahuje polia pre hodnoty boolean a number a referencie na hodnoty string, table, function a userdata. Okrem toho ešte obsahuje pole, ktoré určuje aký typ hodnoty aktuálne trieda uchováva. Pre toto riešenie som sa rozhodol pretože platforma .NET nepozná obdobu mechanizmu *union* z jazykov C, C++. Variant nebolo možné implementovať ani pomocou dedičnosti, kde by existovala jedna základná

trieda pre Variant a hodnoty jednotlivých typov premenných by držali odvodené triedy. Totiž v prípade, že by do inštancie niektorej z odvodených tried bola priradená hodnota iného typu, musela by byť vytvorená nová inštancia triedy odpovedajúcej danému typu.

Pri takejto implementácii typu Variant odpadáva réžia spojená s virtuálnymi funkciami alebo pretypovaním. V najhoršom prípade, teda ak je premenná typu nil, je v takejto implementácii nevyužitých približne 25 bajtov – 1 bajt pre boolean, 8 bajtov pre double a 16 bajtov na referencie (.NET framework presnú veľkosť referencií neuvádza, predpokladám 4 bajty).

Jediný štruktúrovaný typ jazyka Lua, tabuľku, som implementoval rovnako, ako v originálnej implementácii jazyka. Trieda reprezentujúca tento typ obsahuje mechanizmus pre uchovávanie hodnôt indexovaných číslom ako v poli a ostatných hodnôt uchovávaných v slovníku s hešovanými kľúčmi [60 s. 6-8]. V mojej implementácii zastupujú tieto dva mechanizmy inštancie tried *ArrayList* a *Dictionary*.

Princíp registrácie CLR[4] funkcií pre prostredie jazyka Lua som zvolil značne iný, než aký je v originálnej implementácii virtuálneho stroja Lua. Pri registrácii využívam reflexiu platformy .NET. Pre registráciu funkcie stačí zavolať metódu *RegisterFunction* triedy *VirtualMachine* s požadovaným názvom funkcie pre prostredie Lua, názvom CLR metódy a inštanciou CLR objektu, resp. referenciou na typ v prípade statických metód. O konverziu vstupných a výstupných parametrov sa stará samotný virtuálny stroj.

4.1.2. Integrácia Lua skriptov

Pre jednoduchú integráciu Lua skriptov do obsahu hry, ktorý obsahuje XNA Content Project slúži knižnica *Mpme.ContentPipeline*. Táto obsahuje triedy potrebné na kompiláciu Lua skriptov do Lua bytecode počas kompilácie aplikácie.

Pre využitie tejto knižnice ju stačí pridať ako referenciu k projektu XNA Content Project.

4.1.3. Využitie implementácie pre vývoj hier

Potrebné vývojové nástroje pre vývoj hier postavených na implementácii MPME pre Windows Phone 7 sú uvedené v prílohe E. Pre vytvorenie novej hry sú potrebné nasledovné kroky:

1. Vytvorenie projektu typu XNA Game Studio 4.0 – Windows Phone Game.
2. Do projektu pridať referencie na knižnice *Mpme.Engine* a *Mpme.LuaVirtualMachine*.
3. Do XNA Content Project, ktorý je pre hru automaticky vytvorený pridať referenciu na knižnicu *Mpme.ContentPipeline*.
4. V tomto projekte vytvoriť nový súbor (skript) s príponou *lua*.
5. Pridať existujúci projekt *Mpme.Content*.
6. Zmeniť hlavnú triedu novej hry tak, aby dedila od triedy *Mpme.Engine.Game*.

[4] CLR – Common Language Runtime, prostredie .NET framework.

7. Implementovať abstraktnú vlastnosť tejto triedy *GameScriptName* tak, aby vracala názov hlavného herného skriptu.

Pre zjednodušenie je na priloženom CD projekt *EmptyGame*, ktorý je podľa týchto krokov vytvorený a obsahuje prázdnu hru. Táto hra obsahuje iba jednoduchý algoritmus, ktorý vykresľuje na displej počet sekúnd od spustenia hry. Tento projekt je možné použiť ako základ pre vývoj novej hry.

4.2. Windows

Pre demonštráciu rozšíriteľnosti aj za hranice mobilných zariadení som implementoval MPME aj pre OS Windows. Implementácia je založená na XNA a zdieľa väčšinu kódu s implementáciou pre Windows Phone 7. Prakticky totožné sú aj princípy použitia tejto implementácie pre vývoj hier nad MPME.

Jediným rozdielom oproti implementácii pre Windows Phone 7 je, že implementácia pre Windows podporuje ovládanie pomocou klávesnice aj myši, podporuje teda obidva profily MPME pre používateľské rozhranie.

4.3. Windows Mobile 6 Standard

Implementácia MPME pre Windows Mobile 6 Standard je napísaná v jazyku C++ a kompilovaná do natívneho kódu zariadenia. Využil som v nej dve knižnice tretích strán.

Prvou je LuaCE, verzia originálnej implementácie virtuálneho stroja Lua pre Windows CE. Operačný systém Windows Mobile 6 Standard je založený na jadre Windows CE, preto na ňom táto verzia funguje. Táto knižnica je dostupná, rovnako ako originálna implementácia Lua, pod licenciou MIT. Dostupné zdrojové kódy som umiestnil do samostatného projektu a kompilujú sa do dynamickej knižnice (DLL), ktorú MPME využíva.

Druhou knižnicou je GapiDraw. Táto umožňuje jednoduchú prácu s grafikou. Vybral som si ju, pretože uľahčuje prácu s načítavaním, transformáciou a vykresľovaním textúr a bitmapových fontov. Navyše niektoré zariadenia s Windows Mobile podporujú hardwarovo akcelerovanú grafiku a niektoré nie, GapiDraw využíva dostupné prostriedky všetkých zariadení [67]. Knižnica je dostupná zadarmo pre nekomerčné účely [68].

4.3.1. Herný obsah

SDK pre Windows Mobile nepodporuje projekty dedikované pre herný obsah alebo iný podobný mechanizmus na jednoduché nasadzovanie herného obsahu spolu s celým projektom počas vývoja.

Pre tento účel som využil konfiguráciu projektu, konkrétne možnosť *Project properties → Configuration properties → Deployment → Additional files*, kam zadávam všetky prostriedky hry.

4.3.2. Využitie implementácie pre vývoj hier

Potrebné vývojové nástroje pre vývoj hier postavených na implementácii MPME pre Windows Mobile 6 Standard sú uvedené v prílohe E. Pre vytvorenie novej hry sú potrebné nasledovné kroky:

1. Vytvoriť prázdny projekt typu *Windows Application* pre Windows Mobile 6 Standard.
2. Do adresára projektu skopírovať súbor *Game.h*, ktorý je súčasťou balíčka MPME, vložiť ho do projektu.
3. Pridať odkaz na *Mpme.lib* súbor v nastaveniach projektu *Property pages → Configuration properties → Linker → Input → Additional dependencies*. Zároveň pridať adresár, v ktorom sa tento súbor nachádza do *Property pages → Configuration properties → Linker → General → Additional library directories.*
4. Vytvoriť nový *Resource File* v projekte.
5. Do tohto súboru pridať novú ikonu, ktorá bude reprezentovať ikonu aplikácie. Identifikátor ikony sa predáva aj funkcii *StartGame*.
6. Vytvoriť nový C++ zdrojový súbor pre vstupný pod aplikácie.
7. Do tohto súboru vložiť odkaz na hlavičkový súbor *Game.h*, pridať vstupný bod aplikácie, funkciu *WinMain*. Obsah tejto funkcie má byť volanie funkcie *StartGame*.
8. Aby nebolo nutné manuálne kopírovať na zariadenie alebo emulátor knižnice a prostriedky využívané aplikáciou, je tieto potrebné nastaviť v *Project properties → Configuration properties → Deployment → Additional files*. Sú potrebné nasledovné súbory:
    - *Mpme.dll*
    - *GapiDraw.dll*
    - *LuaCE.dll*
    - *Mpme.Content\Scripts\Engine.lua*
    - *Mpme.Content\Scripts\UI\Window.lua*
    - *Mpme.Content\Scripts\UI\Button.lua*
    - *Mpme.Content\Scripts\UI\Label.lua*
    - Všetky ostatné prostriedky konkrétnej hry

Rovnako ako v prípade implementácie pre Windows Phone 7, je na priloženom CD projekt s prázdnou hrou *EmptyGame* pre Windows Mobile 6 Standard. Tento projekt je založený uvedeným postupom a je možné ho využiť na tvorbu nových hier.

4.4. Android

I keď na Android je možné vyvíjať natívne aplikácie v jazyku C, primárnou vývojovou platformou je Java. Preto v tejto kapitole načrtnem možnú implementáciu MPME v Jave.

4.4.1. Virtuálny stroj Lua

Existujú aspoň 4 implementácie virtuálneho stroja Lua pre platformu Java [63].

- **Kahlua** – Implementácia virtuálneho stroja a kompilátora Lua 5.1, vrátane štandardných knižníc. Určená pre platformu J2ME, dostupná pod licenciou MIT [69].
- **Mochalua** – Konverzia virtuálneho stroja Lua 5.1, vrátane štandardných knižníc. Určená pre mobilnú platformu J2ME. Dostupná pod licenciou MIT [70].
- **Luaj** – Implementácia virtuálneho stroja Lua 5.1, vrátane štandardných knižníc. Určená pre platformy J2ME a J2SE. Dostupná pod licenciou MIT [71].
- **Jill** – Implementácia virtuálneho stroja Lua 5.1, čiastočne so štandardnými knižnicami. Určená pre platformu J2ME. Dostupná pod licenciou MIT [72].

Implementácia MPME na operačnom systéme Android by využila niektorú z dostupných implementácií virtuálneho stroja Lua, prípadne implementovala vlastnú ako v prípade Windows Phone 7.

4.4.2. Grafické rozhranie

Android SDK obsahuje rozhranie pre prácu s dvojrozmernou grafikou v balíčku *android.graphics.drawable*. Pre implementáciu vykresľovania a transformácie jednotlivých textúr predpokladám využitie tried *BitmapDrawable* a *RotateDrawable* [73].

4.4.3. Text

Vykresľovanie textu umožňuje trieda *android.graphics.Canvas* obsiahnutá v Android SDK, spolu s definíciou použitého fontu pomocou triedy *android.graphics.Paint*.

4.4.4. Zvukové rozhranie

Pre prehrávanie jednoduchých zvukových efektov predpokladám využitie triedy z Android SDK *android.media.MediaPlayer*.

4.4.5. Používateľské vstupy

Všetky Android zariadenia majú dotykové obrazovky, implementácia MPME na Android by teda podporovala profil dotykového displeja. Používateľský vstup je možné spracovať v udalostiach *onClick*, *onTouch* a podobne triedy *android.view.View*.

4.5. Symbian

Pre operačný systém Symbian predpokladám implementáciu MPME v jazyku C++ pomocou štandardného Symbian SDK, kompilovanú do natívneho kódu zariadenia.

4.5.1. Virtuálny stroj Lua

Keďže je implementácia realizovaná v jazyku C++, predpokladám využitie originálnej implementácie virtuálneho stroja Lua, ktorá je v jazyku ANSI C.

4.5.2. Grafické rozhranie

Pre dekódovanie obrázkov z formátu PNG je potrebné použiť *Image Conversion Library*, ktorá je súčasťou štandardných knižníc. Po dekódovaní sa inštancia triedy *CFbsBitmap* vykresľuje na obrazovku zariadenia pomocou metódy *CWindowGc::BitBlt* [8 s. 99-111].

4.5.3. Text

Implementácia vykresľovania textu by využívala bitmapové fonty, kde by všetky znaku v danom fonte boli uložené v obrázku a na obrazovku sa podľa potreby vykresľovali časti tohto obrázku. Tento mechanizmus odporúča [8 s. 111-112] pre vyšší výkon aplikácie.

4.5.4. Zvukové rozhranie

Pre prehrávanie zvukových efektov poskytujú knižnice OS Symbian triedu *CMdaAudioPlayerUtility*. Na načítanie zvukového súboru slúži metóda *OpenFileL* a na prehrávanie metóda *Play* tejto triedy [8 s. 128-133].

4.5.5. Používateľské vstupy

Niektoré zariadenia s OS Symbian disponujú dotykovým displejom, iné klávesnicou. Implementácia MPME by teda obsahovala obidva profily interakcie s používateľom.

Odchytávanie udalostí kláves sa registruje pomocou metódy *CCoeAppUI::AddToStackL*. Samotné udalosti klávesnice sú dostupné po prepísaní metódy *CCoeControl::OfferKeyEventL* [8 s. 49-52].

Prepísaním metódy *CCoeControl::HandlePointerEventL* sa ošetruje vstup z dotykovej obrazovky [8 s. 52-53].

4.6. Demonštračná hra

Hra Air Battles, ktorá je súčasťou tejto práce demonštruje použiteľnosť navrhnutej platformy a funkčnosť implementácií MPME. Hra bola navrhnutá a vyvinutá iba pre potreby tejto práce a nebol v nej kladený žiadny dôraz na hrateľnosť alebo audiovizuálnu kvalitu. Hra podporuje iba zobrazenie na displejoch na výšku.

Air Battles je jednoduchá ťahová strategická hra pre dvoch hráčov. V hre sa hráč snaží poraziť súpera pomocou lietadiel, ktoré má k dispozícii a ktoré v jednotlivých ťahoch ovláda. Hra je koncipovaná v dvojrozmernej grafickej podobe s nadhľadu. Podrobný popis hry sa nachádza v dizajn dokumente v prílohe C.

Hra je v plnom rozsahu implementovaná nad systémom MPME a je spustiteľná na všetkých implementáciách MPME bez nutnosti akýchkoľvek zmien. V hre som použil voľne dostupné grafické a zvukové elementy, ich zoznam sa nachádza v kapitole 7.

Postup pre spúšťanie hry na jednotlivých platformách je detailne popísaný v prílohe E.

5. Záver

Operačných systémov pre mobilné zariadenia existuje celá rada. Na rozdiel od operačných systémov stolných počítačov nie je ani jedna platforma dominantná a je preto nutné vyvíjať aplikácie na viacerých platformách zároveň.

Z hľadiska vývoja aplikácií sa od seba tieto platformy značne líšia a nie je jednoduché vyvíjať aplikácie na niekoľko platforiem zároveň. Platformy sa navzájom líšia podporovanými vývojovými prostrediami, jazykmi a rozhraniami. Líšia sa od seba aj jednotlivé zariadenia a to hlavne displejmi, spôsobom interakcie s používateľom, ale aj výkonmi procesorov a množstvom pamäte.

Pre hry je problém ešte o niečo málo komplikovanejší, keďže pri vývoji hier je zväčša potrebný rýchly prístup k prostriedkom zariadenia, ako je grafické, zvukové alebo používateľské rozhranie. I keď existujú vývojové systémy, ktoré umožňujú súčasný vývoj pre viacero platforiem, doposiaľ žiadny z nich neposkytoval unifikovaný a rýchly prístup k týmto rozhraniam pre všetky alebo aspoň väčšinu majoritných mobilných operačných systémov. Navyše ide väčšinou o komerčné a navyše drahé riešenia.

Systém MPME, ktorý je v tejto práci popísaný, je navrhnutý tak, aby umožňoval jednoduchý vývoj hier na viacero platforiem, potenciálne na všetky mobilné. Poskytuje jednotné rozhranie pre prístup k základným prostriedkom každého zariadenia. Základná idea systému je to, aby pri vývoji hry nebolo potrebné písať prakticky žiadny kód na natívnej platforme pre daný operačný systém. Navyše bolo mojou snahou navrhnúť MPME tak, aby nebol na úrovni implementácie príliš komplexný a dal sa tak jednoducho implementovať na ľubovoľnú mobilnú platformu.

To, že je navrhované riešenie funkčné demonštruje implementácia MPME na dvoch odlišných mobilných platformách. Prvou je platforma Windows Mobile 6 Standard, reprezentujúca zariadenia s nízkym rozlíšením displeja, ovládaním pomocou klávesnice a prekladom implementácie do natívneho kódu zariadenia. Druhou platformou je Windows Phone 7, reprezentujúca zariadenia s dotykovými obrazovkami s vysokým rozlíšením a prekladom implementácie do medzikódu platformy .NET.

Použiteľnosť navrhnutého riešenia dokazuje demonštračná hra Air Battles. Zdrojové súbory a ostatný obsah demonštračnej hry je úplne totožný pre spustenie na všetkých platformách. Popis hry sa teda podarilo dokonalo oddeliť od hosťujúceho prostredia MPME. Pre spustenie na jednotlivých platformách sa líši iba implementácia MPME. Navrhovanou implementáciou na ďalších platformách by bolo možné hru bez akýchkoľvek zmien spúšťať aj na zariadeniach s inými operačnými systémami.

5.1. Rozšíriteľnosť systému

Návrh MPME nie je uzavretý a bolo by možné ho rozšíriť o ďalšie funkcie. Grafické rozhranie podporuje iba základné transformácie textúr. Ďalšími krokmi by bolo nastavovanie transparentnosti pri vykresľovaní, disproporčné naťahovanie textúr, vykresľovanie iba časti textúry a podobne.

Zvukové rozhranie je navrhnuté pomerne jednoducho, rozšíriť by ho bolo možné možnosti jednoduchej transformácie zvuku (výška, hlasitosť, tempo) a o možnosť kontroly jednotlivých zvukových inštancií (zastavovanie, opakovanie).

Keďže mnoho dnešných zariadení podporuje akcelerometer, ktorý je často používaný aj ako ovládací prvok pri hrách, bolo by možné v MPME zaviesť tretí profil pre používateľský vstup pomocou akcelerometra.

MPME nemá žiadnu podporu pre sieťovú komunikáciu. Rozšírenie by mohlo zahŕňať podporu pre webové služby alebo nízkoúrovňovú komunikáciu prostredníctvom internetu. Ďalej by bolo možné rozšíriť systém o možnosť priamej komunikácie dvoch zariadení prostredníctvom Wi-Fi alebo Bluetooth.

Aj keď bolo prostredie MPME navrhnuté za účelom vývoja hier pre mobilné zariadenia, jeho možnosti tu nekončia. Vzhľadom k minimálnym požiadavkám na hosťujúcu platformu a jednoduchosti implementácie by bolo možné rozšíriť MPME aj iné zariadenia, i keď to nebolo primárnym zámerom návrhu. Možnosť implementácie na stolné počítače bola demonštrovaná implementáciou pre OS Windows a implementácia na ďalšie operačné systémy stolných počítačov, či herné konzoly ako Xbox alebo PlayStation je určite reálna.

6. Bibliografia

1. **Deloitte.** Evolution of smart phone market | Deloitte Telecommunications Predictions 2009. [Online] 2009. [Dátum: 14. November 2010.] http://www.deloitte.co.uk/TMTPredictions/telecommunications/Smartphones-clever-in-downturn.cfm.

2. **ARM.** Dhrystone and MIPs performance of ARM processors. [Online] [Dátum: 2. December 2010.] http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka3885.html.

3. **Longbottom, Roy.** PC CPU Performance Comparisons. [Online] August 2010. [Dátum: 2. December 2010.] http://www.roylongbottom.org.uk/cpuspeed.htm.

4. **Wikipedia.** Pentium III - Coppermine. [Online] [Dátum: 2. December 2010.] http://en.wikipedia.org/wiki/Pentium_III#Coppermine.

5. **Gartner.** Gartner Says Worldwide Mobile Device Sales Grew 13.8 Percent in Second Quarter of 2010, But Competition Drove Prices Down. [Online] 12. August 2010. [Dátum: 14. November 2010.] http://www.gartner.com/it/page.jsp?id=1421013.

6. **Symbian Developer Community.** Multi-language programming - An overview. [Online] [Dátum: 24. November 2010.] http://developer.symbian.org/wiki/Multi-Language_Programming_-_An_Overview.

7. **Symbian.** Important dates in Symbian History. [Online] [Dátum: 24. November 2010.] http://www.symbian.org/about-us/history-symbian.

8. **Chehimi, Fadi, a iní.** *Games on Symbian OS: A Handbook for Mobile Development.* s.l. : Wiley, 2008. ISBN: 978-0470998045.

9. **Hamer, Carol a Davison, Andrew.** *Learn BlackBerry Games Development.* s.l. : Apress, 2010. ISBN: 978-1430227182.

10. **Halevy, Ronen.** The History of RIM & the BlackBerry Smartphone, Part 1: The Origins. [Online] 12. Február 2009. [Dátum: 24. November 2010.] http://www.berryreview.com/2009/02/12/the-history-of-rim-the-blackberry-smartphone-part-1-the-origins/.

11. —. The History of RIM & the BlackBerry Smartphone, Part 2: The Black & White Years. [Online] 17. Február 2009. [Dátum: 24. November 2010.] http://www.berryreview.com/2009/02/17/the-history-of-rim-the-blackberry-smartphone-part-2-the-black-white-years/.

12. —. The History of RIM & the BlackBerry Smartphone, Part 3: The Evolution Of Color. [Online] 16. Marec 2009. [Dátum: 24. November 2010.] http://www.berryreview.com/2009/03/16/the-history-of-rim-the-blackberry-smartphone-part-3-the-evolution-of-color/.

13. **Roth, Daniel.** Google's Open Source Android OS Will Free the Wireless Web. [Online] 23. Jún 2008. [Dátum: 24. November 2010.] http://www.wired.com/techbiz/media/magazine/16-07/ff_android?currentPage=all.

14. **Silva, Vladimir.** *Pro Android Games.* s.l. : Apress, 2009. ISBN: 978-1430226475.

15. **Wikipedia.** iOS (Apple). [Online] [Dátum: 24. November 2010.] http://en.wikipedia.org/wiki/IOS_%28Apple%29.

16. **Daley, Michael.** *Learning iOS Game Programming.* s.l. : Addison-Wesley Professional, 2010. ISBN: 978-0321699428.

17. **Barney, Lee S.** *Developing Hybrid Applications for the iPhone.* s.l. : Addison-Wesley Professional. ISBN13: 978-0321604163.

18. **Zammetti, Frank.** *Practical Palm Pre WebOS Projects.* s.l. : Apress, 2009. ISBN: 978-1430226741.

19. **Henry, Neil a Goss, Tricia.** The History of Windows Mobile. [Online] 31. December 2008. [Dátum: 24. November 2010.] http://www.brighthub.com/computing/windows-platform/articles/1295.aspx.

20. **Wikipedia.** Windows Mobile. [Online] [Dátum: 24. November 2010.] http://en.wikipedia.org/wiki/Windows_Mobile.

21. **Dawes, Adam.** *Windows Mobile Game Development.* s.l. : Apress, 2010. ISBN: 978-1430229285.

22. **Petzold, Charles.** *Programming Windows Phone 7.* s.l. : Microsoft Press, 2010. ISBN: 978-0735643352.

23. **Topolsky, Joshua.** webOS 2.0 review. [Online] 19. October 2010. [Dátum: 24. November 2010.] http://www.engadget.com/2010/10/19/webos-2-0-review/.

24. **McDermott, Wes.** *Creating 3D Game Art for the iPhone with Unity.* s.l. : Focal Press, 2010. ISBN: 978-0240815633.

25. **Goldstone, Will.** *Unity Game Development Essentials.* s.l. : Packt Publishing, 2009. ISBN: 978-1847198181.

26. **Creighton, Ryan Henson.** *Unity 3D Game Development by Example.* s.l. : Packt Publishing, 2010. ISBN: 978-1849690546.

27. **Unity Technologies.** UNITY: Android. [Online] [Dátum: 16. November 2010.] http://unity3d.com/unity/publishing/android.

28. **Unity.** Buy Unity 3. [Online] [Dátum: 4. December 2010.] https://store.unity3d.com/shop/.

29. **Ansca Mobile.** Corona FAQ. [Online] [Dátum: 24. November 2010.] http://www.anscamobile.com/corona/faq/.

30. —. Corona API Reference. [Online] [Dátum: 24. November 2010.] http://developer.anscamobile.com/resources/apis/.

31. —. Download Corona. [Online] [Dátum: 4. December 2010.] https://developer.anscamobile.com/user/register?destination=downloads/game-edition.

32. **Elements Interactive.** EDGELIB Features. [Online] [Dátum: 27. November 2010.] http://www.edgelib.com/index.php?node=features.

33. —. EDGELIB License. [Online] [Dátum: 4. December 2010.] http://www.edgelib.com/index.php?node=license.

34. **Stark, Jonathan.** *Building iPhone Apps with HTML, CSS, and JavaScript.* s.l. : O'Reilly Media, 2010. ISBN: 978-0596805784.

35. **Nitobi.** PhoneGap - About. [Online] [Dátum: 27. November 2010.] http://www.phonegap.com/about.

36. —. PhoneGap API Documentation. [Online] [Dátum: 27. November 2010.] http://www.phonegap.com/download.

37. **MoSync.** MoSync - Feature/Platform Support. [Online] [Dátum: 27. November 2010.] http://www.mosync.com/documentation/manualpages/featureplatform-matrix.

38. —. MoSync - Pricing and Subscriptions. [Online] [Dátum: 27. November 2010.] http://www.mosync.com/mosync-dual-licence-model.

39. —. MoSync API Reference. [Online] [Dátum: 27. November 2010.] http://www.mosync.com/files/imports/doxygen/latest/html/index.html.

40. **ScummVM.** Compatibility list. [Online] [Dátum: 24. November 2010.] http://www.scummvm.org/compatibility/.

41. —. Downloads. [Online] [Dátum: 24. November 2010.] http://www.scummvm.org/downloads/.

42. **ScummVM Wiki.** Platforms. [Online] [Dátum: 24. November 2010.] http://wiki.scummvm.org/index.php/Platforms.

43. —. Datafiles. [Online] [Dátum: 24. November 2010.] http://wiki.scummvm.org/index.php/Datafiles.

44. **Rossum, Guido van.** A Brief Timeline of Python. [Online] 20. Január 2009. [Dátum: 2. December 2010.] http://python-history.blogspot.com/2009/01/brief-timeline-of-python.html.

45. **Beazley, David M.** *Python Essential Reference.* s.l. : Addison-Wesley Professional, 2009. ISBN: 978-0672329784.

46. **Python.** Python Programming Language. [Online] [Dátum: 2. December 2010.] http://www.python.org/.

47. **Python Software Foundation.** History and Licence. [Online] 2010. [Dátum: 2. December 2010.] http://docs.python.org/license.html.

48. **Phillips, Dusty.** *Python 3 Object Oriented Programming.* s.l. : Packt Publishing, 2010. ISBN: 978-1849511261.

49. **Flanagan, David a Matsumoto, Yukihiro.** *The Ruby Programming Language.* s.l. : O'Reilly Media, 2008. ISBN: 978-0596516178.

50. **Ruby.** Download Ruby. [Online] [Dátum: 3. December 2010.] http://www.ruby-lang.org/en/downloads/.

51. —. Ruby License. [Online] [Dátum: 3. December 2010.] http://www.ruby-lang.org/en/about/license.txt.

52. **Wikipedia.** Ruby (programming language). [Online] [Dátum: 3. December 2010.] http://en.wikipedia.org/wiki/Ruby_%28programming_language%29.

53. **Hattan, John.** *Beginning Game Programming: A GameDev.net Collection.* s.l. : Course Technology PTR, 2009. ISBN: 978-1598638059.

54. **Riek, Matthew a Douglas, Greg.** GameMonkey Script. [Online] [Dátum: 2. December 2010.] http://www.somedude.net/gamemonkey/.

55. **Ierusalimschy, Roberto.** *Programming in Lua.* s.l. : Lua.org, 2006. ISBN: 978-8590379829.

56. **McShaffry, Mike.** *Game Coding Complete, Third Edition.* s.l. : Charles River Media, 2009. ISBN: 978-1584506805.

57. **Wikipedia.** Lua-scripted video games. [Online] [Dátum: 23. November 2010.] http://en.wikipedia.org/wiki/Category:Lua-scripted_video_games.

58. **Lua Wiki.** Lua Uses. [Online] [Dátum: 23. November 2010.] http://lua-users.org/wiki/LuaUses.

59. **Lua.org.** Lua: Licence. [Online] 20. November 2010. [Dátum: 23. November 2010.] http://www.lua.org/license.html.

60. **Ierusalimschy, Roberto, de Figueiredo, Luiz Henrique a Celes, Waldemar.** The Implementation of Lua 5.0. *Journal of Universal Computer Science.* 2005, Zv. 11, #7.

61. **Shi, Yunhe, a iní.** Virtual Machine Showdown: Stack Versus Registers. [Online] [Dátum: 23. November 2010.] https://www.usenix.org/events/vee05/full_papers/p153-yunhe.pdf.

62. **Glasberg, Mark Stroetzel, Bresler, Jim a Cho, Yongmin 'Kevin'.** The Lua Architecture. [Online] [Dátum: 21. November 2010.] pgl.yoyo.org/lua/docs/luaarchitecture.doc.

63. **Lua Users.** Lua Implementation. [Online] [Dátum: 23. November 2010.] http://lua-users.org/wiki/LuaImplementations.

64. **Randolph, Nick a Fairbairn, Christopher.** *Professional Windows Phone 7 Application Development: Building Applications and Games Using Visual Studio, Silverlight, and XNA.* s.l. : Wrox, 2010. ISBN: 978-0470891667.

65. **Saffitz, Michael.** Going Places: Adaptable Apps for Windows Mobile. *MSDN Magazine.* [Online] Jún 2008. [Dátum: 23. November 2010.] http://msdn.microsoft.com/en-us/windowsmobile/bb264320.aspx.

66. **Man, Kein-Hong.** A No-Frills Introduction to Lua 5.1 VM Instructions. [Online] 13. Marec 2006. [Dátum: 23. November 2010.] http://luaforge.net/docman/view.php/83/98/ANoFrillsIntroToLua51VMInstructions.pdf.

67. **Develant Technologies.** GapiDraw - Feature List. [Online] [Dátum: 4. December 2010.] http://www.gapidraw.com/features.php.

68. —. GapiDraw - Download GapiDraw. [Online] [Dátum: 4. December 2010.] http://www.gapidraw.com/download.php.

69. **Kahlua.** Summary. [Online] [Dátum: 23. November 2010.] http://code.google.com/p/kahlua/.

70. **Mochalua.** About Mochalua. [Online] [Dátum: 23. November 2010.] http://code.google.com/p/mochalua/.

71. **Luaj.** Summary. [Online] [Dátum: 23. November 2010.] http://sourceforge.net/projects/luaj/.

72. **Jillcode.** Summary. [Online] [Dátum: 23. November 2010.] http://code.google.com/p/jillcode/.

73. **Android.com.** Android Developers - 2D graphics. [Online] [Dátum: 5. December 2010.] http://developer.android.com/guide/topics/graphics/2d-graphics.html.

74. **Wikipedia.** Mobile phone - Wikipedia, the free encyclopedia. [Online] [Dátum: 14. November 2010.] http://en.wikipedia.org/wiki/Mobile_phone.

7. Použité knižnice, kód a obsah

Okrem štandardných vývojových prostriedkov som pri implementácii MPME a demonštračnej hry použil nasledujúce knižnice, zdrojový kód a obsah:

- **LuaCE**[5] – Verzia originálnej implementácie Lua pre Windows CE. Dostupná pre ľubovoľný účel pod licenciou MIT.
- **Lua compiler**[6] – Štandardný kompilátor Lua kompilovaný pre OS Windows. Dostupný pre ľubovoľné účely pod licenciou MIT.
- **GapiDraw**[7] – Grafická knižnica pre Windows Mobile. Skúšobná verzia dostupná pre nekomerčné účely.
- **SpriteLib**[8] – Kolekcia grafických elementov. Dostupná pre nekomerčné účely pod licenciou GPL.
- **YoYo Games Resource Packs**[9] – Kolekcie grafických elementov. Dostupné pre nekomerčné účely.
- **PacDV free sound effects**[10] – Kolekcia zvukových efektov. Dostupné pre ľubovoľné účely s výnimkou priameho predaja.

[5] Dostupná na adrese https://github.com/ynezz/luace
[6] Dostupný na adrese http://code.google.com/p/luaforwindows/
[7] Dostupná na adrese http://www.gapidraw.com/download.php
[8] Dostupná na adrese http://www.flyingyogi.com/fun/spritelib.html
[9] Dostupné na adrese http://www.yoyogames.com/make/resources
[10] Dostupná na adrese http://www.pacdv.com/sounds/index.html

Príloha A. Podporované štandardné funkcie Lua

Virtuálny stroj Lua v rámci implementácie MPME musí pre každú platformu podporovať podmnožinu štandardných funkcií Lua. Kompletný zoznam nasleduje v tabuľkách 15 až 23.

Tabuľka 15: Základné funkcie

Funkcia	Implementovaná	Dôvod
assert (v [, message])	Áno	
collectgarbage (opt [, arg])	Nie	Garbage collection môže byť silne závislý od platformy, navyše jeho priame volanie nie je v hernom prostredí zväčša potrebné.
dofile (filename)	Nie	Používa sa funkcia engine.runScript, ktorá funguje korektne v rámci manažmentu obsahu.
error (message [, level])	Áno	
_G	Nie	Môže ovplyvňovať konkrétnu implementáciu, využitie je veľmi úzke.
getfenv ([f])	Nie	Môže byť závislé od konkrétnej implementácie, využitie je veľmi úzke.
getmetatable (object)	Áno	
ipairs (t)	Áno	
load (func [, chunkname])	Nie	Používa sa funkcia engine.runScript, ktorá funguje korektne v rámci manažmentu obsahu.
loadfile ([filename])	Nie	Používa sa funkcia engine.runScript, ktorá funguje korektne v rámci manažmentu obsahu.
loadstring (string [, chunkname])	Nie	Používa sa funkcia engine.runScript, ktorá funguje korektne v rámci manažmentu obsahu.
next (table [, index])	Áno	
pairs (t)	Áno	
pcall (f, arg1, ···)	Áno	
print (···)	Áno	Nevypisuje text do okna aplikácie, ale do ladiaceho výstupu.
rawequal (v1, v2)	Áno	
rawget (table, index)	Áno	
rawset (table, index, value)	Áno	
select (index, ···)	Áno	
setfenv (f, table)	Nie	Môže byť závislé od konkrétnej implementácie, využitie je veľmi úzke.
setmetatable (table, metatable)	Áno	

Funkcia	Implementovaná	Dôvod
tonumber (e [, base])	Áno	
tostring (e)	Áno	
type (v)	Áno	
unpack (list [, i [, j]])	Áno	
_VERSION	Áno	
xpcall (f, err)	Áno	

Tabuľka 16: Manipulácia s korutinami

Funkcia	Implementovaná	Dôvod
coroutine.create (f)	Nie	Vlákna (korutiny) nie sú podporované.
coroutine.resume (co [, val1, ···])	Nie	Vlákna (korutiny) nie sú podporované.
coroutine.running ()	Nie	Vlákna (korutiny) nie sú podporované.
coroutine.status (co)	Nie	Vlákna (korutiny) nie sú podporované.
coroutine.wrap (f)	Nie	Vlákna (korutiny) nie sú podporované.
coroutine.yield (···)	Nie	Vlákna (korutiny) nie sú podporované.

Tabuľka 17: Moduly

Funkcia	Implementovaná	Dôvod
module (name [, ···])	Nie	Moduly (balíčky) nie sú podporované.
require (modname)	Nie	Moduly (balíčky) nie sú podporované.
package.cpath	Nie	Moduly (balíčky) nie sú podporované.
package.loaded	Nie	Moduly (balíčky) nie sú podporované.
package.loaders	Nie	Moduly (balíčky) nie sú podporované.
package.loadlib (libname, funcname)	Nie	Moduly (balíčky) nie sú podporované.
package.path	Nie	Moduly (balíčky) nie sú podporované.
package.preload	Nie	Moduly (balíčky) nie sú podporované.
package.seeall (module)	Nie	Moduly (balíčky) nie sú podporované.

Tabuľka 18: Manipulácia s reťazcami

Funkcia	Implementovaná	Dôvod
string.byte (s [, i [, j]])	Áno	
string.char (···)	Áno	
string.dump (function)	Nie	Závislé od implementácie virtuálneho stroja, bez podpory funkcie *loadstring* zbytočné.
string.find (s, pattern [, init [, plain]])	Áno	

Funkcia	Implementovaná	Dôvod
string.format (formatstring, ···)	Áno	
string.gmatch (s, pattern)	Áno	
string.gsub (s, pattern, repl [, n])	Áno	
string.len (s)	Áno	
string.lower (s)	Áno	
string.match (s, pattern [, init])	Áno	
string.rep (s, n)	Áno	
string.reverse (s)	Áno	
string.sub (s, i [, j])	Áno	
string.upper (s)	Áno	

Tabuľka 19: Manipulácia s tabuľkami

Funkcia	Implementovaná	Dôvod
table.concat (table [, sep [, i [, j]]])	Áno	
table.insert (table, [pos,] value)	Áno	
table.maxn (table)	Áno	
table.remove (table [, pos])	Áno	
table.sort (table [, comp])	Áno	

Tabuľka 20: Matematické funkcie

Funkcia	Implementovaná	Dôvod
math.abs (x)	Áno	
math.acos (x)	Áno	
math.asin (x)	Áno	
math.atan (x)	Áno	
math.atan2 (y, x)	Áno	
math.ceil (x)	Áno	
math.cos (x)	Áno	
math.cosh (x)	Áno	
math.deg (x)	Áno	
math.exp (x)	Áno	
math.floor (x)	Áno	
math.fmod (x, y)	Áno	
math.frexp (x)	Nie	Funkcia má úzke využitie.
math.huge	Áno	
math.ldexp (m, e)	Nie	Funkcia má úzke využitie.
math.log (x)	Áno	

Funkcia	Implementovaná	Dôvod
math.log10 (x)	Áno	
math.max (x, ···)	Áno	
math.min (x, ···)	Áno	
math.modf (x)	Áno	
math.pi	Áno	
math.pow (x, y)	Áno	
math.rad (x)	Áno	
math.random ([m [, n]])	Áno	
math.randomseed (x)	Áno	
math.sin (x)	Áno	
math.sinh (x)	Áno	
math.sqrt (x)	Áno	
math.tan (x)	Áno	
math.tanh (x)	Áno	

Tabuľka 21: Práca so súbormi

Funkcia	Implementovaná	Dôvod
io.close ([file])	Nie	Charakter práce so súbormi je závislý od operačného systému.
io.flush ()	Nie	Charakter práce so súbormi je závislý od operačného systému.
io.input ([file])	Nie	Charakter práce so súbormi je závislý od operačného systému.
io.lines ([filename])	Nie	Charakter práce so súbormi je závislý od operačného systému.
io.open (filename [, mode])	Nie	Charakter práce so súbormi je závislý od operačného systému.
io.output ([file])	Nie	Charakter práce so súbormi je závislý od operačného systému.
io.popen (prog [, mode])	Nie	Charakter práce so súbormi je závislý od operačného systému.
io.read (···)	Nie	Charakter práce so súbormi je závislý od operačného systému.
io.tmpfile ()	Nie	Charakter práce so súbormi je závislý od operačného systému.
io.type (obj)	Nie	Charakter práce so súbormi je závislý od operačného systému.
io.write (···)	Nie	Charakter práce so súbormi je závislý od operačného systému.
file:close ()	Nie	Charakter práce so súbormi je závislý od

Funkcia	Implementovaná	Dôvod
		operačného systému.
file:flush ()	Nie	Charakter práce so súbormi je závislý od operačného systému.
file:lines ()	Nie	Charakter práce so súbormi je závislý od operačného systému.
file:read (···)	Nie	Charakter práce so súbormi je závislý od operačného systému.
file:seek ([whence] [, offset])	Nie	Charakter práce so súbormi je závislý od operačného systému.
file:setvbuf (mode [, size])	Nie	Charakter práce so súbormi je závislý od operačného systému.
file:write (···)	Nie	Charakter práce so súbormi je závislý od operačného systému.

Tabuľka 22: Funkcie operačného systému

Funkcia	Implementovaná	Dôvod
os.clock ()	Nie	Môže byť závislé od implementácie.
os.date ([format [, time]])	Nie	Môže byť závislé od implementácie.
os.difftime (t2, t1)	Nie	Môže byť závislé od implementácie.
os.execute ([command])	Nie	Systém neumožňuje spúšťanie externých aplikácií.
os.exit ([code])	Nie	Používa sa štandardná funkcia exit(), návratové hodnoty aplikácie môžu byť závislé od operačného systému.
os.getenv (varname)	Nie	Môže byť závislé od implementácie.
os.remove (filename)	Nie	Charakter práce so súbormi je závislý od operačného systému.
os.rename (oldname, newname)	Nie	Charakter práce so súbormi je závislý od operačného systému.
os.setlocale (locale [, category])	Nie	Môže byť závislé od implementácie.
os.time ([table])	Nie	Môže byť závislé od implementácie.
os.tmpname ()	Nie	Charakter práce so súbormi je závislý od operačného systému.

Tabuľka 23: Debug knižnica

Funkcia	Implementovaná	Dôvod
debug.debug ()	Nie	Interaktívny režim nie je podporovaný, bol by zložitý pre implementáciu na mobilnom zariadení.
debug.getfenv (o)	Nie	Môže byť závislé od konkrétnej

Funkcia	Implementovaná	Dôvod
		implementácie, využitie je veľmi úzke.
debug.gethook ([thread])	Nie	Môže byť závislé od konkrétnej implementácie, využitie je veľmi úzke.
debug.getinfo ([thread,] function [, what])	Nie	Vlákna (korutiny) nie sú podporované.
debug.getlocal ([thread,] level, local)	Nie	Môže byť závislé od konkrétnej implementácie, využitie je veľmi úzke.
debug.getmetatable (object)	Áno	
debug.getregistry ()	Nie	Môže byť závislé od konkrétnej implementácie, využitie je veľmi úzke.
debug.getupvalue (func, up)	Nie	Môže byť závislé od konkrétnej implementácie, využitie je veľmi úzke.
debug.setfenv (object, table)	Nie	Môže byť závislé od konkrétnej implementácie, využitie je veľmi úzke.
debug.sethook ([thread,] hook, mask [, count])	Nie	Môže byť závislé od konkrétnej implementácie, využitie je veľmi úzke.
debug.setlocal ([thread,] level, local, value)	Nie	Môže byť závislé od konkrétnej implementácie, využitie je veľmi úzke.
debug.setmetatable (object, table)	Áno	
debug.setupvalue (func, up, value)	Nie	Môže byť závislé od konkrétnej implementácie, využitie je veľmi úzke.
debug.traceback ([thread,] [message] [, level])	Nie	Môže byť závislé od konkrétnej implementácie, využitie je veľmi úzke.

Príloha B. Funkcie rozhrania MPME

I. Funkcia pre zmenu časovania

Funkcia	engine.changeFPS(fps)
Parametre	fps – Požadovaný maximálny počet krokov za sekundu. Musí byť kladné číslo.
Návratové hodnoty	Žiadne
Popis	Funkcia zmení časovanie tak, aby bolo od ďalšieho kroku dosiahnutých maximálne zadaný limit pre počet krokov za sekundu. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Základná vrstva

II. Funkcie grafického rozhrania

Funkcia	engine.clearScreen(r, g, b)
Parametre	r – Farba, červená zložka. Musí byť číslo v rozsahu 0-255 (vrátane). g – Farba, zelená zložka. Musí byť číslo v rozsahu 0-255 (vrátane). b – Farba, modrá zložka. Musí byť číslo v rozsahu 0-255 (vrátane).
Návratové hodnoty	Žiadne
Popis	Funkcia prekreslí celý displej zariadenia požadovanou farbou. Ak je niektorý parameter väčší ako 255, použije sa hodnota 255. Ak je niektorý parameter menší ako 0, použije sa 0. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Základná vrstva

Funkcia	texture = engine.loadTexture(name)
Parametre	name – Cesta k textúre v štandardnom formáte MPME pre obsah hry.
Návratové hodnoty	texture – Načítaná textúra, použiteľná pri volaní funkcií pre vykresľovanie textúr. Dátový typ tejto hodnoty nie je definovaný a môže byť závislí od implementácie. Pre potreby herných skriptov sa považuje za hodnotu typu userdata.
Popis	Funkcia načíta textúru z herného obsahu do pamäte a vráti referenciu na textúru (presný návratový typ závisí od implementácie). Ak textúra neexistuje, vráti hodnotu nil.
Implementuje	Základná vrstva

Funkcia	width, height = engine.getTextureNativeResolution(texture)
Parametre	texture – Textúra, hodnota získaná volaním funkcie engine.loadTexture.
Návratové hodnoty	width – Reálna šírka textúry v obrazových bodoch. height – Reálna výška textúry v obrazových bodoch.
Popis	Funkcia vráti reálnu výšku a šírku textúry. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Základná vrstva

Funkcia	width, height = engine.getTextureResolution(texture)
Parametre	texture – Textúra, hodnota získaná volaním funkcie engine.loadTexture.
Návratové hodnoty	width – Šírka textúry vo virtuálnych obrazových bodoch. height – Výška textúry vo virtuálnych obrazových bodoch.
Popis	Funkcia vráti výšku a šírku textúry vo virtuálnych obrazových bodoch. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Podporná vrstva

Funkcia	engine.drawTextureSimple(texture, x, y)
Parametre	texture – Textúra, hodnota získaná volaním funkcie engine.loadTexture. x – Pozícia, na ktorej sa má textúra vykresliť, súradnica X. y – Pozícia, na ktorej sa má textúra vykresliť, súradnica Y.
Návratové hodnoty	Žiadne
Popis	Funkcia vykreslí textúru na daných súradniciach bez rotácie, zväčšovania alebo zmenšovania a bez ohľadu na aktuálny grafický profil. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Základná vrstva

Funkcia	engine.drawTextureResampled(texture, x, y, angle, scale)
Parametre	texture – Textúra, hodnota získaná volaním funkcie engine.loadTexture. x – Pozícia, na ktorej sa má textúra vykresliť, súradnica X. y – Pozícia, na ktorej sa má textúra vykresliť, súradnica Y. angle – Uhol, o ktorý sa má textúra pri vykresľovaní otočiť,

	v stupňoch. scale – Faktor zmenšenia alebo zväčšenia textúry. Pri hodnote 1 sa textúra vykreslí v pôvodnej veľkosti.
Návratové hodnoty	Žiadne
Popis	Funkcia vykreslí textúru na daných súradniciach s rotáciou, zväčšením alebo zmenšením a bez ohľadu na aktuálny grafický profil. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Základná vrstva

Funkcia	engine.drawTexture(texture, x, y [, angle [, scale]])
Parametre	texture – Textúra, hodnota získaná volaním funkcie engine.loadTexture. x – Pozícia, na ktorej sa má textúra vykresliť, súradnica X. y – Pozícia, na ktorej sa má textúra vykresliť, súradnica Y. angle – Uhol, o ktorý sa má textúra pri vykresľovaní otočiť, v stupňoch, nepovinný parameter. scale – Faktor zmenšenia alebo zväčšenia textúry. Pri hodnote 1 sa textúra vykreslí v pôvodnej veľkosti. Nepovinný parameter.
Návratové hodnoty	Žiadne
Popis	Funkcia vykreslí textúru na daných súradniciach s voliteľnou rotáciou, zväčšením alebo zmenšením s ohľadom na aktuálny grafický profil. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Podporná vrstva

Funkcia	width, height = device.getNativeResolution()
Parametre	Žiadne
Návratové hodnoty	width – Reálna šírka displeja zariadenia v obrazových bodoch. height – Reálna výška displeja zariadenia v obrazových bodoch.
Popis	Vráti natívne rozlíšenie displeja zariadenia v obrazových bodoch.
Implementuje	Základná vrstva

Funkcia	width, height = device.getResolution()
Parametre	Žiadne
Návratové hodnoty	width – Šírka displeja zariadenia vo virtuálnych obrazových bodoch. height – Výška displeja zariadenia vo virtuálnych obrazových

	bodoch.
Popis	Vráti virtuálne rozlíšenie displeja zariadenia v obrazových bodoch.
Implementuje	Základná vrstva

III. Funkcie pre vykresľovanie textu

Funkcia	engine.drawTextSimple(text, font, x, y, r, g, b)
Parametre	text – Text, ktorý sa má vykresliť. font – Font, ktorý sa má pri vykresľovaní použiť. Jedna z hodnôt *fonts.big*, *fonts.normal*, *fonts.small*. x – Pozícia v reálnych obrazových bodoch, na ktorú sa má text vykresliť, súradnica X. y – Pozícia v reálnych obrazových bodoch, na ktorú sa má text vykresliť, súradnica Y. r – Farba písma, červená zložka. g – Farba písma, zelená zložka. b – Farba písma, modrá zložka.
Návratové hodnoty	Žiadne
Popis	Vykreslí text požadovaným fontom a farbou na danú pozíciu. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Základná vrstva

Funkcia	engine.drawText(text, font, x, y, r, g, b)
Parametre	text – Text, ktorý sa má vykresliť. font – Font, ktorý sa má pri vykresľovaní použiť. Jedna z hodnôt *fonts.big*, *fonts.normal*, *fonts.small*. x – Pozícia vo virtuálnych obrazových bodoch, na ktorú sa má text vykresliť, súradnica X. y – Pozícia vo virtuálnych obrazových bodoch, na ktorú sa má text vykresliť, súradnica Y. r – Farba písma, červená zložka. g – Farba písma, zelená zložka. b – Farba písma, modrá zložka.
Návratové hodnoty	Žiadne
Popis	Vykreslí text požadovaným fontom a farbou na danú pozíciu. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Podporná vrstva

Funkcia	width = engine.getNativeTextWidth(text, font)

Parametre	text – Text, ktorého šírka sa má zistiť. font – Font, ktorý sa má pri výpočte použiť. Jedna z hodnôt *fonts.big*, *fonts.normal*, *fonts.small*.
Návratové hodnoty	width – Šírka textu v reálnych obrazových bodoch.
Popis	Zistí šírku textu pri vykreslení požadovaným fontom v reálnych obrazových bodoch zariadenia. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Základná vrstva

Funkcia	width = engine.getTextWidth(text, font)
Parametre	text – Text, ktorého šírka sa má zistiť. font – Font, ktorý sa má pri výpočte použiť. Jedna z hodnôt *fonts.big*, *fonts.normal*, *fonts.small*.
Návratové hodnoty	width – Šírka textu vo virtuálnych obrazových bodoch.
Popis	Zistí šírku textu pri vykreslení požadovaným fontom vo virtuálnych obrazových bodoch. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Podporná vrstva

Funkcia	height = engine.getFontHeight(font)
Parametre	font – Font, ktorého výška sa má zistiť. Jedna z hodnôt *fonts.big*, *fonts.normal*, *fonts.small*.
Návratové hodnoty	height – Výška ľubovoľného textu v danom fonte vo virtuálnych obrazových bodoch.
Popis	Zistí výšku ľubovoľného textu v danom fonte vo virtuálnych obrazových bodoch. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Podporná vrstva

IV. Funkcie zvukového rozhrania

Funkcia	sound = engine.loadSound(name)
Parametre	name – Cesta k zvukovému efektu v štandardnom formáte MPME pre obsah hry.
Návratové hodnoty	sound – Načítaný zvukový efekt, použiteľný pri volaní funkcií pre prehrávanie zvukových efektov. Dátový typ tejto hodnoty nie je definovaný a môže byť závislí od implementácie. Pre potreby herných skriptov sa považuje za hodnotu typu userdata.

Popis	Načíta zvukový efekt z obsahu hry a vráti objekt, ktorý daný zvukový efekt reprezentuje. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Základná vrstva

Funkcia	engine.playSound(sound)
Parametre	sound – Zvukový efekt, ktorý sa má prehrať.
Návratové hodnoty	Žiadne
Popis	Prehrá asynchrónne zvukový efekt. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Základná vrstva

V. Funkcia pre prácu so skriptami

Funkcia	engine.runScript(name)
Parametre	name – Cesta k skriptu v štandardnom formáte MPME pre obsah hry.
Návratové hodnoty	Žiadne
Popis	Načíta a vykoná skript zo zadanej cesty. Ak funkcia dostane nesprávne parametre (iný počet parametrov, nesprávny typ alebo rozsah parametrov), vyhodí chybu.
Implementuje	Základná vrstva

Príloha C. Dizajn dokument hry Air Battles

I. Úvod

Abstrakt

Air Battles je ťahová strategická hra z fiktívneho prostredia inšpirovaného druhou svetovou vojnou určená pre dvoch hráčov. Hráči medzi sebou bojujú pomocou lietadiel, ktoré ovládajú.

Platforma

Hra je koncipovaná pre mobilné zariadenia, s ohľadom na viacero rozličných typov a platforiem.

II. Herné mechanizmy

Air Battles je ťahová strategická hra z pohľadu zhora (top-down). Každý hráč má k dispozícii niekoľko lietadiel, ktoré počas svojho ťahu priamo ovláda.

Cieľ hry

Cieľom hry je zničiť všetky súperove lietadlá a zároveň zabrániť zničeniu všetkých vlastných lietadiel. Každá partia hry končí v okamihu, keď jeden z hráčov nemá k dispozícii už žiadne lietadlá. Druhý hráč je v takom prípade víťazom hry.

Herná fáza

Počas hlavnej hernej fázy, taktickej bitky, sa hráči striedajú v jednotlivých ťahoch. Všetky vybrané lietadlá začínajú v tejto fáze na svojich štandardných pozíciách na mape. V každom ťahu hráč určí povely pre jednotlivé jednotky. Po zadaní povelu ho jednotka okamžite vykoná. Po zadaní povelov všetkým jednotkám nasleduje ťah druhého hráča.

Povely jednotiek

Počas svojho ťahu môže dávať hráč jednotkám dva typy povelov:

- **Pohybový povel** – určuje pohyb jednotky.
- **Útočný povel** – určuje použitie zbraňových systémov jednotky.

V jednom ťahu môže jednotka vykonať iba jeden povel z každej kategórie.

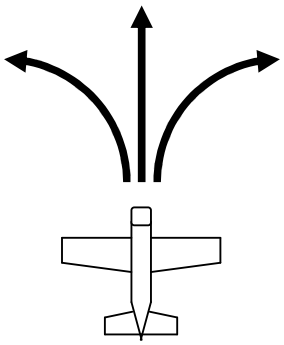
Zmena pozície

Základným pohybovým povelom jednotky je povel k zmene pozície. To, kam sa jednotka v danom ťahu posunie určujú smer a vzdialenosť.

Smer pohybu je ovplyvnený agilnosťou a to tak, že agilnosť určuje polomer najmenšej kružnice, ktorú je schopná jednotka opísať. Ohraničuje teda smer, ktorým sa jednotka môže presúvať. Smer jednotky môže byť ľubovoľne na stupnici *maximálne doľava – maximálne doprava*. V strede tejto stupnice je smer *rovno*. Jednotka sa vždy za jedno kolo posunie po

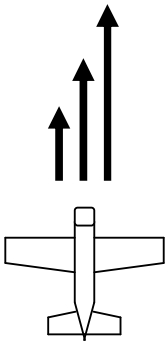
úsečke (ak je smer rovno) alebo kružnicovom oblúku (ak je smer iný ako rovno) tak, ako zobrazuje obrázok 4.

Obrázok 4: Smer lietadla



Vzdialenosť je ovplyvnená rýchlosťou a to tak, že minimálna a maximálna rýchlosť určujú minimálnu a maximálnu vzdialenosť, o ktorú sa jednotka určeným smerom pohne. Rýchlosť jednotky môže byť ľubovoľne na stupnici *minimálna vzdialenosť – maximálna vzdialenosť*, tak ako zobrazuje obrázok 5.

Obrázok 5: Rýchlosť lietadla



Smer spolu so vzdialenosťou určujú výslednú pozíciu a rotáciu jednotky.

Tabuľka 24: Povel k zmene pozície

Povel k zmene pozície	
Kategória	Pohybový povel
Atribúty	Smer, vzdialenosť
Vstupné podmienky	Jednotka v danom ťahu ešte nevykonala pohybový povel
Výstupné podmienky	Nasleduje povel na útok (ak sú v dosahu súperove jednotky)

Útok

Jednotkám so zbraňovými systémami môže hráč vydať povel na útok. Všetky zbraňové systémy majú neobmedzenú muníciu a môžu byť použité každé kolo po pohybovom povele.

Zbraňový systém je zbraňový systém jednotky, ak ním disponuje.

Cieľ vyberie hráč podľa dosahu zbraňového systému zo zoznamu súperových jednotiek.

Tabuľka 25: Povel na útok

Povel na útok	
Kategória	Útočný povel
Atribúty	Zbraňový systém, cieľ
Vstupné podmienky	Jednotka je po vykonaní pohybového povelu, cieľ je v dosahu zbraňového systému.
Výstupné podmienky	Žiadne

Stavy lietadiel

Každé lietadlo sa v danom ťahu nachádza v práve jednom zo stavov:

- **Vo vzduchu** – štandardný stav lietadla, lietadlo musí v každom ťahu vykonať aspoň pohybový povel.
- **Zostrelené** – lietadlo bolo zostrelené a nie je možné ho nijak používať.

Zbraňové systémy

Všetky zbraňové systémy v hre majú niekoľko vlastností:

- **Sila (power)** – určuje efektívnosť zbrane proti vzdušným cieľom. Táto je nezávislá od vzdialenosti jednotky a cieľa alebo ich vzájomnej pozície.
- **Dostrel a smer (range and cone of fire)** – smer, ktorým zbraň umožňuje strieľať a jej dostrel.

MG III machinegun

Zbraň používaná lietadlom De Leech, popis sa nachádza v tabuľke 26.

Tabuľka 26: Mk III machinegun

Mk III machinegun	
Sila proti vzdušným cieľom	190
Smer a dostrel	48000 dopredu v rozmedzí 50°

MK 109 cannon

Zbraň používaná lietadlom Hawker Cyclone, popis sa nachádza v tabuľke 27.

Tabuľka 27: MK 109 cannon

MK 109 cannon	
Sila proti vzdušným cieľom	160
Smer a dostrel	45000 dopredu v rozmedzí 45°

Typy jednotiek

Obidvaja hráči majú k dispozícii rovnaké typy jednotiek. Rozdiely medzi jednotkami obidvoch strán sú len vizuálne. V každej partii dostanú k dispozícii obidvaja hráči rovnaké typy jednotiek v rovnakých počtoch.

Každá jednotka má sadu vlastností, od ktorých sa odvíja štýl jej použitia, výdrž apod. Jednotlivé vlastnosti sú:

- **Pancier (armor)** – určuje koľko daná jednotka vydrží zásahov, kým je nenávratne zničená. U jednotiek sa rozlišuje maximálny a aktuálny pancier. Každá jednotka začína s hodnotou aktuálneho panciera na úrovni maximálneho. Po utŕžení zásahov sa hodnota aktuálneho panciera znižuje. Keď dosiahne hodnotu 0 (alebo nižšiu), jednotka je nenávratne zničená a ďalej sa do hry nezapája.
- **Rýchlosť (speed)** – maximálna rýchlosť, ktorou sa dokáže jednotka pohybovať, teda maximálna vzdialenosť, ktorú dokáže za jeden ťah prekonať. U lietadiel ďalej existuje minimálna rýchlosť, ktorá sa priamo odvíja od maximálnej (lietadlo vo vzduchu počas ťahu nemôže ostať na jednom mieste).
- **Agilnosť (agility)** – určuje schopnosť jednotky vykonávať prudké manévre. Čím je agilnosť vyššia, tým menší je polomer kružnice, ktorú dokáže jednotka pri pohybovom povele opísať.
- **Zbraňový systém (weapon system)** – zbraňový systém, ktorým jednotka disponuje. Viac v podkapitole Zbraňové systémy.

Hawker Cyclone

Vlastnosti útočná stíhačky Hawker Cyclone sú popísané v tabuľke 28.

Tabuľka 28: Hawker Cyclone

Hawker Cyclone	
Pancier	240
Rýchlosť (minimálna / maximálna)	5 / 11
Agilnosť	10
Zbraňový systém	MK 109 cannon

De Leech

Vlastnosti ľahkého bombardéru De Leech sú popísané v tabuľke 29.

Tabuľka 29: De Leech

De Leech	
Pancier	380
Rýchlosť (minimálna / maximálna)	3 / 9
Agilnosť	7
Zbraňové systémy	MG III machinegun

Mapy

Prostredie, v ktorom sa hra odohráva je more. Mapy sú vyskladané zo štvorcov v štvorcovej mriežke. Jednotlivé štvorce mriežky sa od seba odlišujú iba vizuálne. Jednotky sa pohybujú voľne po mape, nie sú pevne viazané na jeden štvorec.

K dispozícii je jediná preddefinovaná mapa. Mapy nie je možné meniť alebo náhodne generovať.

III. Vizuálny štýl

Celá hra sa odohráva v pohľade zhora (top-down view). Jednotlivé vizuálne elementy sa nesnažia byť realistické, skôr ide o vtipnú kreslenú grafiku.

Jednotky

Vzhľad jednotiek je inšpirovaný jednotkami druhej svetovej vojny. Jednotky obidvoch strán vyzerajú rovnako, sú rozlíšené iba základovou farbou:

- **Zelená farba** – pre jednotky prvého hráča.
- **Červená farba** – pre jednotky druhého hráča.

Používateľské rozhranie

Používateľské rozhranie je kontrastné, skôr tmavé. Dominantná farba je čierna. Všetky prvky používateľského rozhrania majú ostré hrany a pravé uhly.

IV. Audio štýl

Keďže ide o hru určenú na mobilné zariadenia, na zvukovú stránku hry nie je kladený príliš veľký dôraz. Hra neobsahuje žiadnu hudbu. Obsahuje jednoduché zvukové efekty pri používaní prvkov používateľského rozhrania a jednotlivých akcií jednotiek v hre.

V. Ovládanie

Ovládanie hry je koncipované tak, aby vyhovovalo rozličným mobilným platformám. Hra podporuje dva profily – profil dotykového ovládania a profil fyzických tlačidiel.

Dotykové ovládanie

Predpokladá využitie na zariadeniach s dotykovou obrazovkou. Tento profil nevyžaduje okrem dotykovej obrazovky žiadne fyzické tlačidlá.

Pohybom dotyku po obrazovke hráč mení pozíciu na mape. Počas zadávania pohybového povelu sú na obrazovke tlačidlá pre zadávanie smeru a rýchlosti a tlačidlo na potvrdenie povelu.

Počas zadávania útočného povelu je možné dotykom na cieľ zaútočiť. Na obrazovke sa nachádza aj tlačidlo pre neútočenie na žiadny z dostupných cieľov.

Počas celej hernej fázy je na obrazovke tlačidlo pre návrat do menu.

Ovládanie fyzickými tlačidlami

Predpokladá využitie na zariadeniach bez dotykovej obrazovky, s klávesmi. Tento profil nevyžaduje dotykovú obrazovku.

Počas zadávania pohybového povelu zadáva sa má hráč možnosť akčným tlačidlom prepínať medzi režimom ovládania smeru a rýchlosti lietadla a pohybom po mape. Ovládanie smeru a rýchlosti alebo ovládanie pohybu po mape sa vykonáva pomocou smerového tlačidla. Hráč potvrdzuje pohybový povel potvrdzovacím tlačidlom.

Počas zadávania útočného povelu má hráč možnosť vybrať smerovým tlačidlom cieľ alebo vydať akčným tlačidlom povel na neútočenie. Po vybratí cieľa ho zadá potvrdzovacím tlačidlom.

Príloha D. Obsah priloženého CD

Prehľad obsahu CD, ktoré je súčasťou tejto práce sa nachádza v tabuľke 30.

Tabuľka 30: Obsah priloženého CD

Popis	Cesta
Implementácia MPME pre platformy Windows Phone 7, Windows Mobile 6 Standard a OS Windows (projekty pre Visual Studio 2008, resp. 2010)	*source\wp7\Mpme* *source\wm6s\Mpme* *source\win32\Mpme*
Zdieľaný obsah hry Air Battles a MPME, spoločný pre všetky platformy	*source\shared*
Visual Studio 2008 (resp. 2010) projekty pre hru Air Battles pre jednotlivé platformy	*source\wp7\AirBattles* *source\wm6s\AirBattles* *source\win32\AirBattles*
Visual Studio 2008 (resp. 2010) projekty s prázdnou hrou Empty Game pre jednotlivé platformy	*source\wp7\EmptyGame* *source\wm6s\EmptyGame* *source\win32\EmptyGame*
Inštalátor hry Air Battles vo formáte XAP pre zariadenie alebo emulátor Windows Phone 7	*binaries\wp7*
Inštalátor hry Air Battles vo formáte CAB pre zariadenie Windows Mobile 6 Standard	*binaries\wm6s*
Inštalátor hry Air Battles pre OS Windows	*binaries\win32*
Dokumentácia kódu implementácie MPME pre Windows Phone 7 a OS Windows	*docs\documentation\C#*
Dokumentácia kódu implementácie MPME pre Windows Mobile 6 Standard	*docs\documentation\C++*
Dokumentácia kódu implementácie hry Air Battles	*docs\documentation\Lua*
Text tejto diplomovej práce v elektronickej podobe	*docs*

Príloha E. Inštalácia a používanie projektov

Súčasťou tejto práce sú aj projekty s implementáciou MPME a demonštračnou hrou. Pre používanie projektov je potrebný nasledujúci softvér:

- Visual Studio 2008 vo verzii Professional alebo vyššej (ďalej iba VS2008).
- Visual Studio 2010[11] vo verzii Professional alebo vyššej (ďalej iba VS2010).
- Windows Mobile 6 Professional and Standard Software Development Kits Refresh[12].
- Windows Mobile 6.1 Standard Emulator Images (USA)[13].
- Windows Phone Developer Tools[14].

I. Spúšťanie aplikácie na Windows Phone 7 emulátore

Pre spúšťanie Air Battles na emulátore bez otváranie projektu vo VS2010 je možné použiť nástroj *Application Deployment*, ktorý je súčasťou štandardnej inštalácie Windows Phone 7 SDK. Odkaz na nástroj sa nachádza v ponuke Štart v zložke *Windows Phone Developer Tools*. Pri inštalácii je potrebné vybrať ako cieľ *Windows Phone 7 Emulator* a XAP súbor z adresára *binaries\wp7*. Po inštalácii je možné spustiť hru z hlavnej ponuky programov v emulátore.

Projekt je možné spúšťať na emulátore aj prostredníctvom VS2010. Po otvorení projektu z adresára *source\wp7\AirBattles* na priloženom DVD je potrebné vybrať ako cieľové zariadenie v pravom hornom rohu okna *Windows Phone 7 Emulator* a projekt spustiť príkazom *Start Without Debugging* z menu *Debug* (klávesová skratka CTRL+F5).

II. Spúšťanie aplikácie na Windows Phone 7 zariadení

Spustenie Air Battles na zariadení je možné rovnakými spôsobmi ako spustenie na emulátore. Jediným rozdielom je, že sa pri inštalácii alebo spúšťaní vyberie ako cieľ *Windows Phone 7 Device*.

Zariadenie musí byť odomknuté pre vývoj aplikácii pomocou nástroja *Windows Phone Developer Registration*, ktorý je taktiež súčasťou inštalácie Windows Phone 7 SDK.

III. Spúšťanie aplikácie na Windows Mobile 6 Standard emulátore

Pre spustenie projektu na Windows Mobile 6 Standard emulátore je potrebné otvoriť projekt z adresára *source\wm6s\AirBattles* vo VS2008. Po otvorení je nutné vybrať projekt *AirBattles*

---

[11] SDK pre Windows Mobile 6 nepodporuje Visual Studio 2010 a SDK pre Windows Phone 7 na druhej strane podporuje iba Visual Studio 2010. Je preto potrebné mať obidve verzie.

[12] Možné stiahnuť z http://www.microsoft.com/downloads/en/details.aspx?FamilyID=06111a3a-a651-4745-88ef-3d48091a390b&displaylang=en#SystemRequirements.

[13] Možné stiahnuť z http://www.microsoft.com/downloads/en/details.aspx?FamilyID=3d6f581e-c093-4b15-ab0c-a2ce5bffdb47.

[14] Možné stiahnuť z http://www.microsoft.com/downloads/en/details.aspx?FamilyID=04704acf-a63a-4f97-952c-8b51b34b00ce&displaylang=en

zo zoznamu projektov, *USA Windows Mobile 6.1 Standard QVGA Emulator*[15] ako cieľové zariadenie v pravom hornom rohu okna a spustiť pomocou príkazu *Start Without Debugging* z menu *Debug* (klávesová skratka CTRL+F5).

IV. Spúšťanie aplikácie na Windows Mobile 6 Standard zariadení

Pre inštaláciu na mobilnom zariadení je možné použiť inštalačný CAB súbor pripravený v adresári *binaries\wm6s*. Tento súbor je potrebné skopírovať do zariadenia a následne spustiť. Po dokončení inštalácie sa hra spustí pomocou odkazu v ponuke Štart.

Spúšťať aplikáciu na zariadení je možné rovnako ako na emulátore aj pomocou VS2008. Najprv je potrebné zariadenie pripojiť k PC pomocou USB s použitím aplikácie *ActiveSync* (pre Windows XP), resp. *Windows Mobile Device Center* (pre Windows Vista a 7). Ďalšie kroky sú totožné ako v prípade spúšťania na emulátore, akurát sa ako cieľové zariadenie vyberie *Windows Mobile 6 Standard Device*.

V. Spúšťanie aplikácie na Windows PC

Inštalátor Air Battles sa nachádza na priloženom DVD v adresári *binaries\win32*. Inštalátor automaticky stiahne a nainštaluje prípadné prerekvizity. Po dokončení inštalácie sa hra spustí.

Spúšťať aplikáciu je možné aj priamo z projektu *source\win32\AirBattles* otvoreného vo VS2010 pomocou príkazu *Start Without Debugging* z menu *Debug* (klávesová skratka CTRL+F5).

[15] Emulátory staršie než vo verzii 6.1 obsahujú chybu s funkciou *SizeOfResource* (http://www.1-script.com/forums/SizeofResource-problem-on-WM-5-0-based-Smartphone-article3366--15.htm) a projekt na nich nemusí fungovať správne.