VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

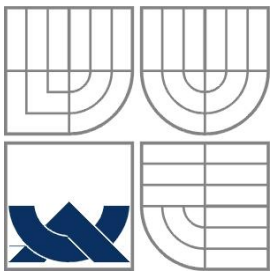
LOGICKÉ SÍŤOVÉ HRY PRO ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

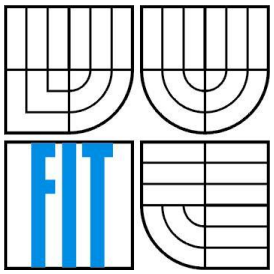
AUTOR PRÁCE
AUTHOR

MICHAL KUBÍK

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

LOGICKÉ SÍŤOVÉ HRY PRO ANDROID

LOGICAL NETWORK GAMES FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL KUBÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN KOUŘIL
SUPERVISOR

BRNO 2012

Abstrakt

Cílem této práce byla implementace aplikace pro hraní logických her, která bude spustitelná na zařízeních se systémem android. Práce zabývá systémem Android a možnostmi tvorby a publikace aplikací pro tento systém. Popsány jsou také známé algoritmy, používané pro implementaci umělé inteligence pro hraní logických her. Řeší se zde i způsoby bezdrátové komunikace mezi přenosnými zařízeními. V implementační části této práce je popsáno grafické uživatelské rozhraní aplikace, řešení umělé inteligence a síťové komunikace.

Abstract

The main goal of this bachelor's thesis was the implementation of application for playing logical games. This work deals with Android platform and the possibilities of creation and deployment of applications for this platform. The known algorithms that are used for implementation of an artificial intelligence are described in the work. The ways of wireless communication between mobile device are also mentioned. The graphical user interface of application, solution of an artificial inteligence and the network communication are described in the implementation part of this work.

Klíčová slova

Android, logické hry, síťové hry, Java, mobilní telefony, šachy, piškvorky, reversi.

Keywords

Android, logical games, network games, Java, mobile phones, chess, five in a row, reversi.

Citace

Michal Kubík: Logické síťové hry pro Android, bakalářská práce, Brno, FIT VUT v Brně, 2012

Logické síťové hry pro Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Kouřila. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Michal Kubík
16. května 2012

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce, Ing. Janu Kouřilovi, za odbornou pomoc, ochotu a čas, který mi při tvorbě této práce věnoval.

Obsah

1 Úvod

Hraní her bylo vždy velmi oblíbenou činností pro krácení volného času nebo jen tak pro zábavu. Dříve však lidé pro hraní her potřebovali soupeře a potřebné vybavení, jako je například šachovnice. Dnes je díky moderním technologiím v podobě přenosných zařízení umožněno lidem hrát hry téměř kdekoliv a kdykoliv, přičemž jim stačí pouze ono přenosné zařízení. Dokonce se hráči mohou obejít i bez lidského soupeře. Ten dnes bývá zastoupen umělou inteligencí – programem, který imituje lidské chování, v tomto případě provádí tahy v dané hře. Uživatel má však i nadále možnost hrát proti lidskému soupeři a to buď na jednom zařízení, nebo stále častěji přes počítačovou síť.

Z důvodu rychle rostoucí oblíbenosti přenosných zařízení se mnoho vývojářů obrací právě k těmto přístrojům. Díky tomu dochází ke vzniku mnoha užitečných aplikací, které pak umožňují uživatelům využívat všechny přednosti přenosných zařízení. Jednou z těchto aplikací by měla být i aplikace Logické hry, která bude poskytovat vášnivým, ale i občasným hráčům možnost zabavit se třeba při čekání na autobus a zároveň přitom potrápit svůj mozek.

V teoretické části této práce se zabývám postupy, které jsou často používány pro tvorbu umělé inteligence. Zmiňuji také možnosti síťové komunikace, která je potřebná pro hraní her po síti. Z technologií, které slouží k bezdrátové komunikaci, popíši Near Field Communication, Bluetooth a Wi-Fi. Dále uvádím důležité informace o systému Android, o jeho historii a o částech, které tvoří tento systém. Popsán je i proces vývoje aplikace pro systém Android, včetně tvorby grafického uživatelského rozhraní a následné publikování hotové aplikace.

V části návrhu a implementace se zabývám návrhem a tvorbou grafického uživatelského rozhraní s možností zobrazení na různě velikých obrazovkách. Uveden je také popis implementace umělých inteligencí pro hry piškvorky a reversi a krátce je zmíněn i způsob předávání síťových zpráv. V závěru popisuji způsoby testování, a možnosti dalšího směřování vývoje aplikace.

2 Analýza problému

V této kapitole budou rozebrány zejména způsoby tvorby umělé inteligence pro hraní her a možnosti síťové komunikace.

2.1 Tvorba umělé inteligence

Při tvorbě umělé inteligence je důležité nejdříve specifikovat problém, který chceme řešit a poté správně určit metodu, která se bude pro řešení daného problému hodit nejvíce. Mezi metody řešení úloh patří podle [1]:

- Metody založené na prohledávání stavového prostoru.
- Metody založené na omezujících podmínkách.
- Metody založené na rozkladu úloh na podproblémy.
- Metody hraní her.

Ve své aplikaci jsem využil posledních jmenovaných metod. Dále se tedy budu věnovat právě těmto metodám.

2.1.1 Metody hraní her

Tyto metody se využívají ve hrách pro dva hráče, kteří se pravidelně střídají a oba chtějí zvítězit. Pro složité hry, u kterých můžeme prozkoumat jen několik následujících tahů, často používáme algoritmus MiniMax, popřípadě jeho upravené verze, které zvyšují efektivitu prohledávání stavů hry.

Algoritmus MiniMax

Algoritmus MiniMax spočívá v prohledávání stavů hry, které mohou nastat poté, co hráč zahraje jeden z povolených tahů a v následném vyhodnocení těchto stavů z hlediska přínosu pro aktuálního hráče. Možné stavy hry jsou uspořádány do stromové struktury, kde každá větev představuje jeden tah hráče. Procedura MiniMax je volána rekurzivně až do předem určené hloubky, přičemž dochází ke střídání hráčů na tahu. V každé úrovni zanoření je z možných tahů vždy vybrán ten, který je pro daného hráče nejvýhodnější. Cílem je vybrat takový tah, který povede k nejmenší možné ztrátě v případě, že by se hra dále vyvíjela z pohledu aktuálního hráče nejhorším možným způsobem. K vyhodnocení jednotlivých stavů slouží vyhodnocovací funkce, která podle zadaných kritérií určí bodové ohodnocení daného stavu [1, 2].

Postup algoritmu MiniMax:

1. Pokud je aktuální stav hry konečným stavem nebo pokud byla dosažena maximální povolená hloubka zanoření, zavolá se vyhodnocovací funkce a výsledná hodnota je vrácena rodičovskému uzlu.
2. Postupné provádění metody MiniMax pro každý z povolených tahů v aktuálním stavu hry.
3. Jestliže je na tahu hráč A, vybere se nejvyšší hodnota vrácena jedním z potomků a tato hodnota je předána rodičovskému uzlu.
4. Jestliže je na tahu hráč B, vybere se nejnižší hodnota vrácena jedním z potomků a tato hodnota je předána rodičovskému uzlu.

5. Pokud se jedná o kořenový uzel, vrátí se nejvyšší nalezená hodnota a také tah, který k této hodnotě vede.

Algoritmus AlfaBeta

AlfaBeta vylepšuje algoritmus MiniMax tím, že úmyslně přehlíží některé větve ve stromové struktuře stavů hry a umožňuje tím rychlejší vyhodnocení, popřípadě zanoření do větší hloubky při hledání nejlepšího tahu. Předpokladem je co nejlepší určení pořadí, v jakém budeme prohledávat větve stromu [1, 3].

Postup algoritmu AlfaBeta:

1. Zavedeme si pomocné proměnné α a β a inicializujeme je tak, že α bude představovat nejnižší možnou hodnotu a β nejvyšší možnou hodnotu. Poté zavoláme proceduru AlfaBeta.
2. Pokud je aktuální stav hry konečným stavem nebo pokud byla dosažena maximální povolená hloubka zanoření, zavolá se vyhodnocovací funkce a výsledná hodnota je vrácena rodičovskému uzlu.
3. Jestliže je na tahu hráč A, bude se provádět procedura AlfaBeta pro každý možný tah, dokud platí $\alpha < \beta$. Pokud je hodnota vrácená potomkem větší než α, nastavíme proměnnou α na tuto novou hodnotu. Nakonec vrátíme hodnotu proměnné α a pokud se nacházíme v kořenovém uzlu stromu, předáme také nejlepší zjištěný tah.
4. Jestliže je na tahu hráč B, bude se provádět procedura AlfaBeta pro každý možný tah, dokud platí $\alpha < \beta$. Pokud je hodnota vrácená potomkem menší než β, nastavíme proměnnou β na tuto novou hodnotu. Nakonec vrátíme hodnotu proměnné β.

Algoritmus Negascout

Tento algoritmus dále rozvíjí myšlenku přehlížení těch větví stromu, které už nepřinesou do výběru nejlepšího tahu žádnou změnu. Nejdůležitějším předpokladem pro správnou funkčnost tohoto algoritmu je výborné uspořádání tahů podle jejich předpokládané kvality. Pokud se uspořádání podaří a nejlepší tah je nalezen hned v první prohledávané větvi, mělo by dojít k nárůstu výkonu. Při dalším prohledávání je totiž nastavena hodnota β na $\alpha + 1$, čímž vznikne nulové vyhledávací okno, které zvýší rychlost procházení ostatními větvemi. Další změnou je nerozlišování, zda je na tahu hráč A nebo hráč B přímo v proceduře. Musí však platit, že hodnocení stavu hráčem A je negací hodnocení stavu hráčem B. Jinými slovy, nejlepší tah hráče A znamená nejhorší možný stav pro hráče B vzhledem k aktuálnímu stavu hry [4].

Postup algoritmu Negascout:

1. Pokud je aktuální stav hry konečným stavem nebo pokud byla dosažena maximální povolená hloubka zanoření, zavolá se vyhodnocovací funkce a výsledná hodnota je vrácena rodičovskému uzlu.
2. Nastavíme pomocnou proměnnou b na hodnotu proměnné β.
3. Pro každý možný tah provedeme proceduru Negascout. Místo toho, abychom potomkům předali současné hodnoty proměnných α a β za hodnotu α dosadíme -b a za hodnotu β pak -α.
4. Hodnotu vrácenou potomkem znegujeme a zjistíme, zda leží mezi hodnotami proměnných α a β. Pokud ano, opět provedeme proceduru Negascout, tentokrát však za hodnotu α dosadíme -β.
5. Jestliže je hodnota vrácená potomkem po znegování větší než α, nastavíme proměnnou α na tuto novou hodnotu.
6. Pokud je α větší nebo rovno β, vrátíme hodnotu proměnné α a ukončíme proceduru.

7. Nastavíme proměnnou b na α + 1 začneme prohledávat další uzel stromu.
8. Po prohledání všech možných stavů vrátíme hodnotu proměnné α a pokud se nacházíme v kořenovém uzlu, předáme také tah, který vede k nejlepšímu stavu hry z pohledu hráče, který proceduru zavolal.

2.2 Síťová komunikace

Přenosná zařízení, pro něž je aplikace určena, poskytují několik možností vzájemné komunikace. Každá z možností s sebou nese určité výhody i nevýhody, na něž musíme přihlížet s ohledem na účel naší aplikace. Při vývoji své aplikace jsem vzal v úvahu pouze bezdrátové způsoby komunikace, neboť se domnívám, že pro přenosná zařízení je bezdrátová komunikace mnohem vhodnější, než komunikace přeš datové kabely.
Následuje popis dostupných bezdrátových technologií pro přenos dat:

Near field Communication (NFC)

NFC je technologie, která slouží k zasílání menšího objemu dat na velmi krátkou vzdálenost (v řádu jednotek centimetrů), při nízké spotřebě energie. Tento způsob spojení je vhodný například pro bezkontaktní platby nebo pro zasílání vizitek a jiných informací mezi dvěma telefony [5]. Ve své aplikaci jsem se rozhodl nevyužít této technologie i přesto, že z pohledu množství a rychlosti (uvádí se až 424 kbit/s) přenášených dat by bylo NFC více než dostačující. Důvodem je malá vzdálenost mezi komunikujícími přístroji, kterou by bylo nutno udržovat po celou dobu spojení. Navíc v dnešní době stále většina zařízení se systémem Android nemá zabudovánu hardwarovou podporu NFC [6].

Wi-Fi

Tento standard bezdrátové komunikace je dnes velmi rozšířen a jeho účelem je vytváření lokálních bezdrátových sítí. Wi-Fi umožňuje komunikaci na vzdálenost jednotek až desítek metrů a rychlost přenosu je velmi vysoká. Důležitým prvkem je přístupový bod, ke kterému se zařízení musejí připojit, aby následně mohla komunikovat mezi sebou, případně aby získala přístup k internetu [7]. Nevýhodou této metody je tedy nutnost komunikace přes přístupový bod. Tento problém řeší systém Android technologií Wi-Fi Direct [8], která poskytuje možnost přímého propojení zařízení, bez využití přístupového bodu. Wi-Fi Direct však musejí podporovat obě zařízení, která si přejeme propojit. Technologie je podporována jen v zařízeních se systémem Android 4.0, kterých je k dnešnímu dni (7. 5. 2012) pouhých 5% z celkového počtu Android zařízení [9]. Z tohoto důvodu jsem se rozhodl nevyužít tuto technologii ve své aplikaci.

Bluetooth

Tato technologie umožňuje vytváření osobních počítačových sítí. Jeden z přístrojů musí být ve viditelném režimu, čímž umožní ostatním získat informace o svém názvu, MAC adrese a jiné důležité údaje. Přístroj, který si přeje se s tímto zařízením propojit, odešle požadavek na spárování obou přístrojů. Proces spárování spočívá ve vytvoření číselného kódu, který je potřeba uvést v obou zařízeních. Po propojení obou přístrojů již může probíhat samotná komunikace a to buď po zabezpečeném, nebo nezabezpečeném kanálu. Spárování přístrojů je zaznamenáno v historii každého zařízení, a pokud není záznam o spárování explicitně odstraněn uživatelem, není potřeba příště tento proces před zahájením komunikace provádět [10]. Technologie Bluetooth umožňuje komunikaci přenosných zařízení na vzdálenost 5 až 10 metrů. Při použití nejrozšířenější verze

Bluetooth 2.0 můžeme data přenášet rychlostí 2,1 Mbit/s. Maximální vzdálenost i přenosová rychlost jsou pro účely aplikace dostačující [11]. Výhodou je také veliká rozšířenost této technologie na přístrojích se systémem Android [6] a možnost propojení přístrojů bez potřeby přístupového bodu, jako je tomu např. u technologie Wi-Fi. V aplikaci Logické hry jsem pro komunikaci přístrojů při hraní síťové hry použil právě tuto technologii.

3 Logické hry

Jedná se o takové hry, které po člověku vyžadují logické myšlení. Často jsou tyto hry hrány dvěma proti sobě soupeřícími hráči, z nichž každý se snaží zvítězit. Typicky se hraje na herní ploše, která je složena z herních polí. Počet herních polí je u každé hry buď pevně dán, anebo si jej určí sami hráči. Hráčům jsou přiřazeny herní symboly, které se odlišují např. tvarem nebo barvou. V následující části budou popsána pravidla pro hry, které jsou v aplikaci implementovány. Ve všech hrách, které jsou v aplikaci obsaženy, dochází k pravidelnému střídání tahu dvou hráčů, kteří proti sobě soupeří. Není přitom povoleno, aby se hráč dobrovolně vzdal svého tahu.

3.1 Piškvorky

Ve hře je přiřazen jednomu z hráčů symbol křížku, druhému pak symbol kolečka. Každý z hráčů při svém tahu zakreslí na herní plochu svůj symbol, přičemž cílem hry je, aby hráč vytvořil řadu pěti svých symbolů. Řada může být vytvořena horizontálně, vertikálně i diagonálně. Úkolem hráče je zabránit svému soupeři ve vytvoření takové řady, a zároveň se pokusit sám tuto řadu sestavit ze svých symbolů. Při zaplnění celé herní plochy bez možnosti určení vítěze nastává remíza. Velikost herní plochy je určena domluvou hráčů. V aplikaci je na výběr počet řádků a sloupců, přičemž dostupné jsou hodnoty 15, 30 a 60. V případě hry pro více hráčů je velikost herní plochy určena nastavením hráče, který hru vytvářel. Hru začíná hráč s křížky a v případě dalších zápasů začíná ten hráč, který v poslední hře skončil jako poražený.

3.2 Reversi

Hra se hraje na herní ploše o rozměrech 8×8 herních polí. Herním symbolem jsou zde kameny, které jsou z jedné strany bílé a z druhé strany černé. Kámen je považován za bílý, pokud je otočen bílou stranou nahoru, jinak říkáme, že je kámen černý. Hráči se před začátkem hry dohodnou, za kterou barvu bude každý z nich hrát. V aplikaci Logické hry je zakládajícímu hráči přiřazena černá barva a připojujícímu se hráči barva bílá. Každý hráč může při svém tahu položit právě jeden kámen. Při pokládání kamene však musí být dodrženo pravidlo, že alespoň v jednom směru od právě pokládaného kamene se nachází řada soupeřových kamenů o minimální délce jednoho kamene, která je zakončena kamenem právě táhnoucího hráče. Tato řada může být vedena horizontálně, vertikálně nebo diagonálně. Při položení kamene dojde k otočení všech soupeřových kamenů, které se nacházejí právě ve výše zmíněných řadách. Pokud hráč, který je právě na řadě, nemá k dispozici žádný povolený tah, dostane se na řadu opět soupeř. Hra končí po zaplnění herní plochy, nebo pokud ani jeden z hráčů nemůže provést platný tah. Vítězem je ten hráč, který má na konci hry na herní ploše nejvíce kamenů své barvy. Pokud mají oba hráči kamenů stejně, hra je vyhodnocena jako remíza.

3.3 Šachy

Tato část je volně převzata z [12]. Šachy se hrají na ploše o rozměrech 8×8 herních polí, které mají střídavě tmavou a světlou barvu. Herní symboly označujeme jako figury. Jeden z hráčů hraje za černé figury a druhý hráč za bílé figury. Podle toho pak mluvíme o bílém hráči nebo černém hráči. Každý hráč má na začátku stejnou sadu figur a při počátečním rozmístění jsou figury obou hráčů umístěny

na opačných stranách herní plochy se zrcadlově obráceným rozložením. V jednom tahu posunuje hráč vždy jednu herní figuru na jiné místo v herní ploše. Tah je povolen pouze na pole, která jsou prázdná nebo na kterých se vyskytuje soupeřova figura. Pokud na cílovém poli stojí soupeřova figura, je tato figura odstraněna z herní plochy. Tento proces se označuje jako braní soupeřovy figury. Figur se ve hře vyskytuje několik druhů a každá z nich má určen jiný způsob pohybu.
Figury, které se ve hře vyskytují, jsou následující:
Král – Může se pohybovat o jedno pole v jakémkoliv směru, nesmí však přitom dojít k jeho ohrožení soupeřovou figurou. Pokud s králem a jednou z věží nebylo dosud taženo, může král provést speciální tah zvaný rošáda. Při tomto tahu postoupí král o dvě pole směrem k věži a věž pak krále překročí o jedno pole. Všechna mezilehlá pole přitom musejí být prázdná a zároveň žádné z těchto polí nesmí být ohrožováno soupeřem. Toto je jediný tah ve hře, který umožňuje současný pohyb dvěma figurami.

Dáma – Pohybuje se o libovolný počet polí v jakémkoliv směru.

Věž – Po herní ploše se pohybuje horizontálně nebo vertikálně, a to o libovolný počet polí. Věž se také může účastnit rošády, speciálního tahu krále.

Střelec – Pohybuje se o libovolný počet polí v diagonálním směru.

Jezdec – Jako jediná figura ve hře se může pohybovat po skocích ve tvaru písmene L a nezáleží přitom, zda v cestě stojí jiné figury. Jedná se o pohyb o dvě pole horizontálně a jedno pole vertikálně, nebo naopak dvě pole vertikálně a jedno horizontálně.

Pěšec – Jeho pohyb může směřovat pouze vpřed a to vždy o jedno pole, s výjimkou svého prvního tahu, kdy je dovoleno pěšci táhnout o dvě pole vpřed. Pohyb je uskutečněn ve směru vertikálním s tím, že není povoleno brát při tomto pohybu soupeřovy figury. Pokud by však mohl pěšec brát soupeřovu figuru při diagonálním pohybu vpřed, je mu umožněn i tento pohyb. Speciální pohybem pěšce je braní mimochodem, známé také jako en passant. Tento pohyb může být proveden, pokud soupeřův pěšec táhne o dvě pole vpřed, přičemž při pohybu pouze o jedno pole vertikálně by došlo k jeho ohrožení naším pěšcem. Tento tah se pak provádí na pole, kde by pěšec stál, pokud by postoupil pouze o jedno pole vpřed. Soupeřův pěšec je však odstraněn z herní plochy a počítá se, jako by byl brán naším pěšcem. Jedná se o jediný tah ve hře, kterým je možno brát soupeřovu figuru, aniž by došlo k přímému kontaktu mezi figurami. Pokud pěšec dojde až na konec herního pole, provede hráč proměnu pěšce na jinou figuru. Možností je proměna na dámu, věž, střelce nebo jezdce.
Cílem hry je ohrozit soupeřova krále a přitom ubránit krále vlastního. Pokud je hráčův král ohrožen, říkáme, že se nachází v šachu, a hráč musí provést takový tah, aby tuto hrozbu odvrátil. Pokud žádný takový tah neexistuje, hra končí porážkou hráče, jehož král je ohrožen, tato situace se nazývá mat. Hráč může prohrát hru i tak, že se sám soupeři vzdá. V jiných případech hra končí remízou, to může nastat například v případě, že hráč nemá kam táhnout s žádnou svou figurou a současně se jeho král nenachází v šachu, nebo pokud se ve hře nachází již jen takové figury, kterými není možno dosáhnout matu. Další možnosti remízy jsou trojí opakování stejné pozice nebo padesát po sobě jdoucích tahů, ve kterých nebyla sebrána žádná figura. Na remíze se mohou také oba hráči domluvit kdykoli v průběhu hry. V aplikaci Logické hry je možné dosáhnout remízy provedením padesáti tahů bez braní, nemožností tahu jednoho z hráčů v případě, že se jeho král nenachází v šachu a nemožností dosáhnout matu.

4 Technická specifikace

V této kapitole je popsán systém Android, jeho historie a části, ze kterých je tento systém složen. Je zde také uveden výčet verzí systému, které byly dosud vydány. Dále jsou popsány možnosti vytváření aplikací pro systém Android, tvorba grafického uživatelského rozhraní a také publikace aplikací. Zmíněn je také postup při komunikaci s využitím Bluetooth.

4.1 Android

Android je softwarový balíček, určený především pro přenosná zařízení, který se skládá z operačního systému, middleware (software, který poskytuje služby ostatním aplikacím) a dalších klíčových aplikací [13].

4.1.1 Historie

Společnost Android Inc. byla založena v roce 2003 za účelem vývoje blíže nespecifikovaného softwaru pro mobilní přístroje. V srpnu roku 2005 byla firma odkoupena společností Google Inc. a později se začalo spekulovat o snaze Googlu proniknout na trh s novým mobilním zařízením (tzv. gPhone), které mělo být konkurencí pro iPhone od firmy Apple Inc. i pro další mobilní telefony. Místo toho byl v říjnu 2007 oznámen vznik Open Handset Alliance (OHA), seskupení společností, zabývajících se telekomunikací, vývojem hardware a software. Mezi zakládajícími členy OHA byly např. společnosti Google, HTC, Intel, T-Mobile a Texas Instruments. V den svého založení, představilo OHA svůj první produkt, platformu pro mobilní telefony – Android. Místo jediného mobilního zařízení tak postupně vzniklo mnoho přístrojů, využívajících platformy Android. V září roku 2008 byl na trh společností HTC uveden první mobilní telefon s operačním systémem Android ve verzi 1.0. Zároveň došlo k uvolnění zdrojových kódů Androidu jako open-source pod licencí Apache [14, 15]. Do dnešního dne bylo vydáno přes 200 různých zařízení [6], převážně mobilních telefonů, tabletů a elektronických čteček, které oficiálně běží na operačním systému Android.

4.1.2 Architektura operačního systému Android

Operační systém Android se skládá z pěti sekcí, které jsou znázorněny na obrázku 4.1. Podle [13] jsou to:

Aplikace

Android je dodáván se sadou základních aplikací, jako je e-mailový klient, internetový prohlížeč, správce kontaktů nebo SMS program.

Aplikační rozhraní

Poskytuje vývojářům přístup ke všem důležitým funkcím systému. Každý vývojář tak má přístup ke všem prostředkům, které jsou používány základními aplikacemi Androidu. Aplikace může mít zveřejněny své vlastnosti a jiná aplikace pak může této aplikace využít pro dosažení svých cílů. Takto mohou být nahrazeny i ty nejzákladnější programy v systému novými aplikacemi.

Knihovny

Android obsahuje sadu C/C++ knihoven, které jsou využívány různými částmi systému. Přístup k nim je zajištěn přes aplikační rozhraní. Mezi nejdůležitější knihovny patří:

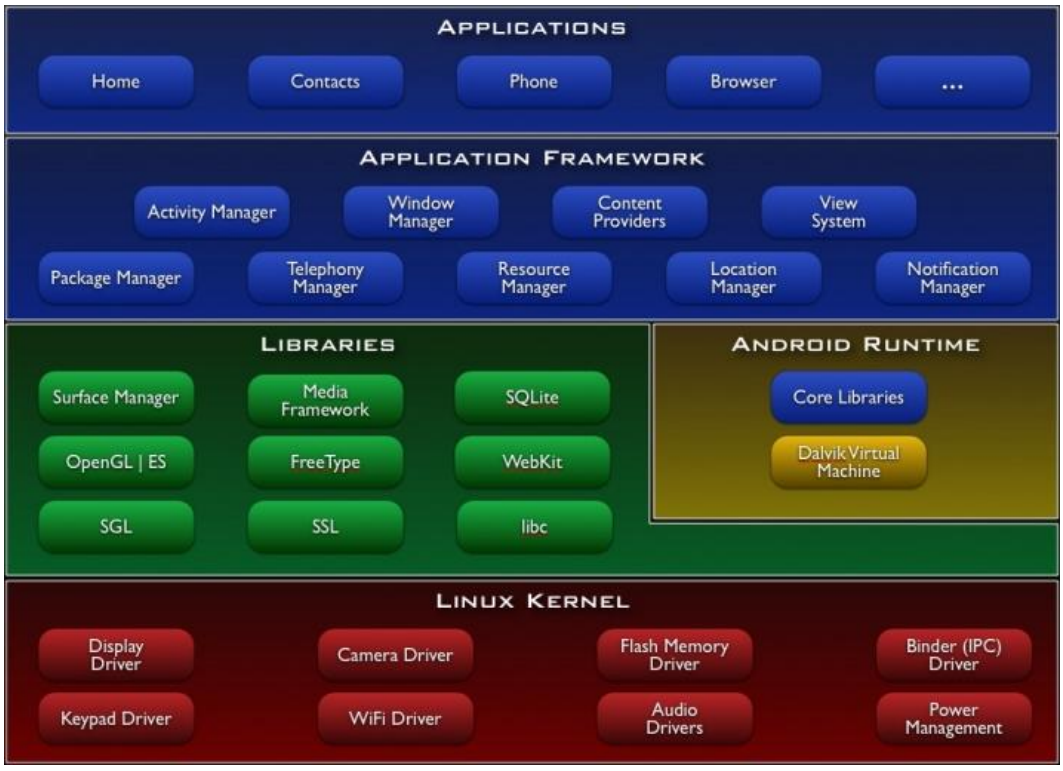
- Systémová knihovna C – odvozená z BSD implementace standardní knihovny C (libc), upravená pro vestavěná zařízení, která jsou založená na Linuxu.
- Knihovna médií – umožňuje přehrávání nebo zachytávání audia, videa a obrazu ve formátech jako jsou MPEG4, H.264, MP3, AAC, AMR, JPG nebo PNG.
- Správce povrchu – zajišťuje správné poskládání 2D a 3D grafických vrstev z různých aplikací.
- LibWebCore – jádro webového prohlížeče.
- SGL – slouží pro vykreslování 2D grafiky.
- 3D knihovny – implementace založená na OpenGL ES. Pokud je to možné, je využita hardwarová 3D akcelerace, jinak se použije optimalizované 3D softwarové vykreslení.
- SQLite – poskytuje odlehčenou verzi relační databáze.

Běhové prostředí

Každý program napsaný v jazyce Java je nejdříve převeden do formátu Dalvik Executable (.dex) a poté je spuštěn ve vlastním procesu s novou instanci virtuálního stroje Dalvik. Je tak umožněna větší nezávislost aplikací a jednodušší správa paměti. Dalvik byl navržen tak, aby bylo možné spustit současně více než jednu jeho instanci.

Linuxové jádro

Android je postaven na Linuxovém jádře verze 2.6, které obstarává abstrakční vrstvu mezi hardwarem a softwarem, správu paměti a procesů nebo také síťové a bezpečnostní služby.



Obrázek 4.1: Architektura systému Android [13]

4.1.3 Přehled nejdůležitějších verzí Androidu

Tato sekce je převzata inspirována z [15, 16].

Android 1.0

Jedná se o první verzí Androidu, která byla předinstalována na komerčně prodávaném mobilním zařízení. Tato verze také představila vzhled uživatelského rozhraní, který je dodnes používán bez větších změn i v novějších verzích Androidu, určených pro mobilní telefony.

Android 1.5 Cupcake

Tato verze přináší podporu vestavěného fotoaparátu a softwarové klávesnice. Zároveň bylo uživatelům umožněno vyměnit základní aplikaci pro klávesový vstup za jakoukoli alternativní aplikaci. Zajímavostí je, že počínaje touto verzí je každá nová verze systému Android pojmenována po anglickém názvu dezertu. Jednotlivé názvy jsou pak přidělovány podle abecedního pořadí.

Android 1.6 Donut

V této verzi byla implementována podpora telefonů s vyšším rozlišením displeje a došlo k vylepšení panelu pro rychlé vyhledávání.

Android 2.0/2.1 Eclair

Eclair přinesl vylepšení vestavěného fotoaparátu podporou blesku, digitálního přiblížení a dalších klíčových funkcí. Zadávání textu bylo vylepšeno lepší predikcí a korekcí slov. Počínaje touto verzí podporuje Android HTML5 a Bluetooth verze 2.1.

Android 2.2 Froyo

Největší novinkou této verze byla bezpochyby možnost instalovat aplikace na paměťovou kartu. Došlo také k řadě optimalizací, které vedly ke zvýšení výkonu. V neposlední řadě byla přidána podpora Adobe Flash 10.1 a podpora vícejazyčných klávesnic. V oblasti bezdrátových přenosů přibyla možnost vytvořit ze zařízení wi-fi přístupový bod, který umožňuje připojení jiných zařízení.

Android 2.3 Gingerbread

Jedná se o v současnosti nejrozšířenější verzi systému Android. Přináší změny v uživatelském rozhraní, které mají zefektivnit práci s přístrojem. V této verzi dochází k oficiální podpoře zařízení s více než jedním fotoaparátem, případně kamerou a zařízení s větším rozlišením displeje. Přibyla také podpora technologie Near Field Communication (NFC), umožňující bezdrátovou komunikaci zařízení na krátkou vzdálenost.

Android 3.0-3.2 Honeycomb

Honeycomb se stal první verzí operačního systému Android navrženou speciálně pro tablety. Vestavěné aplikace byly přizpůsobeny větší obrazovce a nedostupnosti hardwarových kláves. Zdaleka nejdůležitější je však podpora plnohodnotného multi-taskingu, kdy v předchozích verzích docházelo pouze k rychlému přepínání aplikací.

Android 4.0 Ice Cream Sandwich

Nejnovější verze operačního systému Android umožňuje použití jak na mobilních telefonech, tak i na tabletech a tím sjednocuje vývoj aplikací pouze na jednu verzi systému. V této verzi došlo k různým úpravám uživatelského rozhraní a nasazení nového písma s názvem Roboto. Mnoho vlastností bylo také přeneseno z verze Honeycomb a tyto vlastnosti jsou tak nyní k dispozici i pro mobilní telefony. K masovému rozšíření Ice Cream Sandwich by mělo dojít v průběhu roku 2012.

4.2 Vývoj aplikací pro Android

Vývojový balíček Android SDK obsahuje sadu nástrojů, které umožňují vývoj aplikací pro Android. Tyto nástroje jsou dostupné přes příkazovou řádku nebo použitím vývojového prostředí Eclipse s nainstalovaným pluginem ADT (Android Development Tools).
Nástroje balíčku Android SDK:

- Android – slouží pro správu virtuálních zařízení.
- Android emulátor – napodobuje hardwarové zařízení se systémem Android a umožňuje testování aplikací, aniž bychom potřebovali skutečné zařízení.
- Android Debug Bridge – umožňuje komunikaci s emulátorem nebo se zařízením připojeným přes USB.

4.2.1 Struktura Android aplikací

Aplikace pro Android jsou psány v programovacím jazyce Java. Každá aplikace se může skládat až ze čtyř základních druhů komponent. Jednotlivé komponenty mají zcela rozdílné vlastnosti a jejich kombinací je umožněn vznik plnohodnotné aplikace. Zvláštností aplikací pro systém Android je, že často neposkytují pouze jeden vstupní bod do celé aplikace, ale každý z použitých stavebních prvků může být zpřístupněn pro vzdálené použití z jiných aplikací. To umožňuje využití funkcí implementovaných v jiných aplikacích a vývojář se tak nemusí zabývat programováním těchto komponent pro svou aplikaci.
Přehled základních prvků aplikací [17]:

Aktivity

Každá *aktivita* představuje v aplikaci jednu obrazovku s uživatelským rozhraním. Aplikace se obvykle skládá z několika navzájem provázaných aktivit. Typicky je jedna *aktivita* označena za hlavní a slouží pak jako vstupní bod do aplikace v případě, že je aplikace poprvé spuštěna. Každá z *aktivit* může spustit jinou aktivitu a využít tak jejich služeb. V případě spuštění nové *aktivity* je stávající *aktivita* pozastavena a uložena na zásobník pro pozdější obnovení činnosti. Zásobník funguje na principu „last in, first out“ a při ukončení jedné *aktivity* je tak vždy obnovena *aktivita*, která se právě nachází na vrcholu zásobníku.

Služby

Služba umožňuje běh dlouhotrvajících operací v pozadí a uživatel tedy při práci s aplikací nebude blokován probíhající operací. *Služba* však na rozdíl od *aktivity* neposkytuje žádné uživatelské rozhraní pro komunikaci s uživatelem. Vhodné použití *služby* může být například poslouchání hudby nebo správa síťových přenosů. Pokud bychom však chtěli provádět operace náročné na procesor nebo jakékoli blokující operace, bude potřeba použít jiný přístup. *Služba* totiž běží ve stejném procesu

a dokonce ve stejném vlákně, jako celá aplikace, která *službu* spouští. Řešením je vytvoření nového vlákna, čímž se zajistí bezproblémový běh blokujících operací na pozadí, aniž by toto byl uživatel schopen zaregistrovat.

Poskytovatelé obsahu

Poskytovatelé obsahu umožňují správu dat aplikace a případně i sdílení dat s jinými aplikacemi. Data poskytována těmito *poskytovateli obsahu* mohou být uložena například v souborovém systému, na vzdáleném webovém úložišti nebo v SQLite databázi.

Přijímače broadcastových zpráv

Přijímače broadcastových zpráv slouží k zachycení systémových zpráv vyvolaných samotným systémem nebo jednou z uživatelských aplikací. Pokud si přejeme v naší aplikaci zachytávat určitý druh zpráv, je potřeba zaregistrovat náš *přijímač* dynamicky pomocí metody *registerReceiver()* třídy *Context* nebo staticky v souboru *Manifest*. Díky *přijímači* můžeme reagovat na určité události provedením vhodné operace nebo spuštěním služby, která bude vykonávat operaci na pozadí.

Soubor Manifest

Vývojář musí deklarovat všechny komponenty aplikace v souboru *AndroidManifest.xml*, jinak nebude možné tyto komponenty použít. V souboru se také definují různá oprávnění, hardwarové a softwarové požadavky nebo nejnižší úroveň API, která je vyžadována pro běh aplikace. Oprávnění, která jsou aplikací vyžadována, musí uživatel při instalaci odsouhlasit, jinak nebude aplikace nainstalována.

4.2.2 Grafické uživatelské rozhraní

Při spuštění *aktivity* je zobrazeno definované uživatelské rozhraní. Uživatelské rozhraní v Androidu je tvořeno hierarchií objektů *View*. Tyto objekty mohou reprezentovat různé grafické elementy, jako jsou například obrázky nebo tlačítka. Úpravu vlastností objektů *View* můžeme provádět přímo v kódu nebo v XML souborech definováním layoutů. Layouty vznikají kombinací několika objektů *View* a zanořováním dalších layoutů. Mezi základní layouty patří například *LinearLayout, RelativeLayout* nebo *TableLayout*. Definování grafického uživatelského rozhraní v XML souborech za použití layoutů je doporučeno kvůli větší přehlednosti a snadnějšímu ladění chyb v návrhu uživatelského rozhraní.

4.3 Používání Bluetooth v systému Android

Kapitola je volně převzata z [10]. Android poskytuje aplikační rozhraní pro práci s Bluetooth, díky čemuž jsme schopni zapínat a vypínat zařízení Bluetooth, vyhledávat přístroje s aktivním Bluetooth v okolí, připojovat se k okolním přístrojům a následně také komunikovat s těmito přístroji na úrovni přenášení aplikačních dat. Ovládací prvky zařízení Bluetooth jsou implementovány v balíčku *android.bluetooth*. Abychom k tomuto balíčku mohli přistupovat, musíme v souboru Manifest uvést oprávnění *BLUETOOTH* a pokud si navíc přejeme, aby byla aplikace schopna vyhledat okolní zařízení, je nutné také uvést oprávnění *BLUETOOTH_ADMIN*.

Poté, co již máme přístup k balíčku *android.bluetooth*, je důležité, abychom si vytvořili instanci objektu BluetoothAdapter, který reprezentuje Bluetooth zařízení umístěné v přístroji a umožňuje toto zařízení přímo ovládat. Dále je potřeba ověřit, že je zařízení Bluetooth momentálně

spuštěné a pokud není, musíme uživateli zobrazit požadavek o povolení zařízení aktivovat. Zobrazení požadavku a aktivaci zařízení nemusíme v naší aplikaci sami implementovat, stačí zaslat požadavek systému a ten tuto práci udělá za nás. Stejně tak za nás systém vyřídí požadavek na aktivaci viditelnosti zařízení, musíme pouze specifikovat dobu, po kterou si přejeme zařízení zviditelnit. Aktivace viditelnosti je nutná, pokud potřebujeme zajistit, aby byl náš přístroj viditelný a tedy dostupný pro okolní zařízení.

Abychom mohli zahájit komunikaci mezi dvěma přístroji, potřebujeme na obou zařízeních Bluetooth soket připojený na stejném *RFCOMM* kanálu. Pro získání tohoto soketu musí jeden z přístrojů převzít chování serveru a vyčkávat na připojení. Druhý přístroj pak vyhledá server a následně jako klient požádá o spojení. Při vyhledávání okolních zařízení dostáváme instance objektu BluetoothDevice, které obsahují název přístroje, jeho stav, *MAC* adresu a další důležité informace. Právě díky znalosti *MAC* adresy pak můžeme zažádat o připojení k serveru. Klient i server musí definovat *UUID*, 128 bitové číslo, jednoznačně identifikující poskytovanou službu. Pokud klient požádá o připojení k serveru a jejich *UUID* se shoduje, získají oba instanci třídy *bluetoothSocket*.

Po získání Bluetooth soketu získáme přístup k objektům *OutputStream* a *InputStream*, které umožňují zápis zpráv do soketu a čtení přijatých zpráv. Je velmi důležité, aby čekání na příchozí zprávy a zapisování zpráv neprobíhalo v hlavním vlákně, které se stará o vykreslování uživatelského rozhraní, jelikož se jedná o blokující operace a celá aplikace by tak byla zpomalena. Stejně tak jsou blokující i procesy čekání na připojení klienta nebo inicializace připojení k serveru, proto by i tyto operace měly probíhat v odděleném vlákně.

4.4 Publikace aplikací pro Android

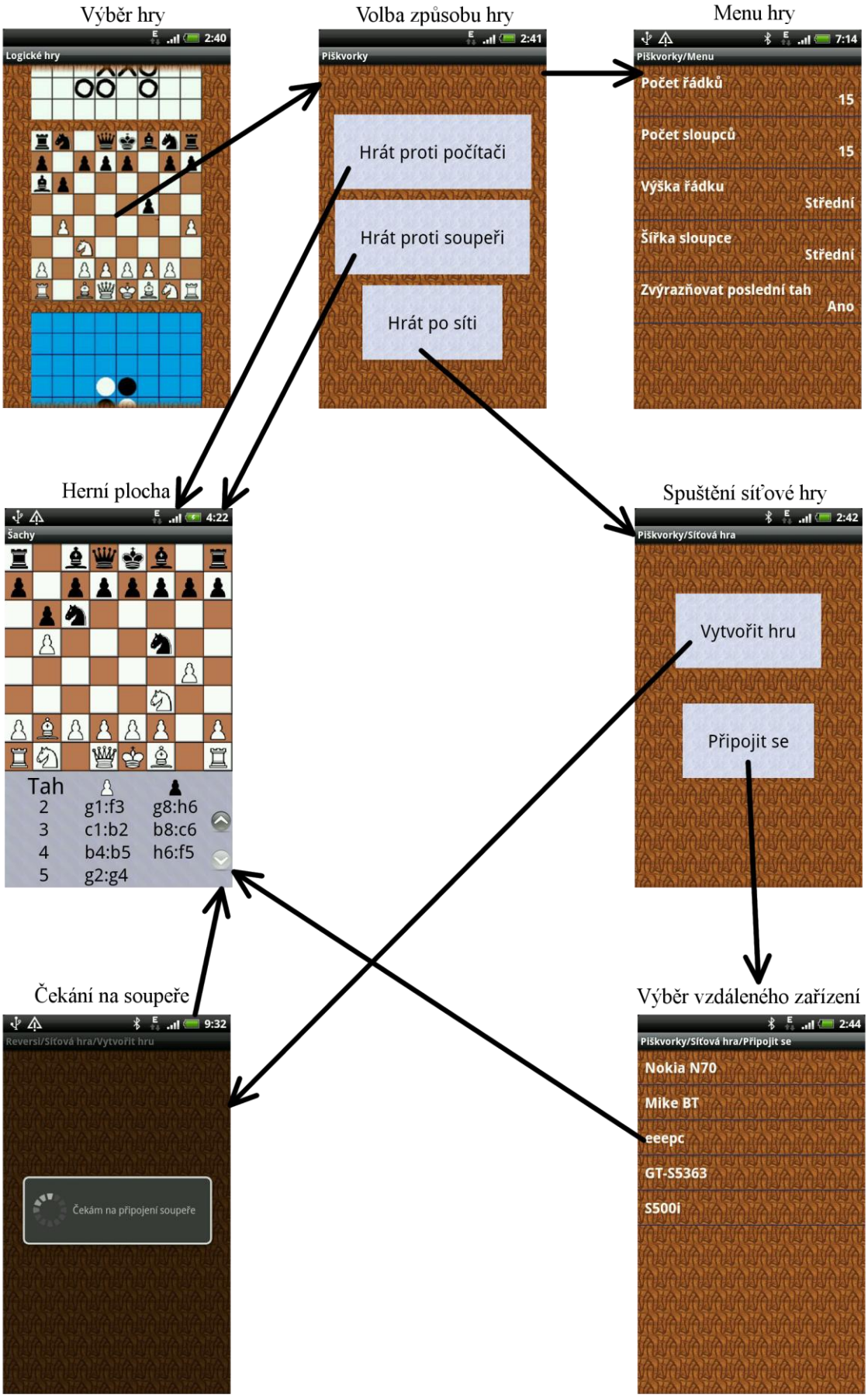
Abychom mohli aplikaci nainstalovat na Android zařízení, je nutné, aby tato aplikace byla podepsána osobním certifikátem autora. K podepsání aplikace lze použít nástroj Jarsigner. Samotný certifikát pak musí být podepsán osobním klíčem autora, který je možno vygenerovat nástrojem Keytool. Není povinně vyžadováno, aby byl certifikát podepsán certifikační autoritou. Při vývoji aplikace je možná instalace v *ladícím režimu*, přičemž Android SDK sám vytvoří vlastní ladící klíč, kterým aplikaci automaticky podepisuje při každé instalaci. Aplikace určená ke zveřejnění již však musí být podepsána osobním klíčem autora a podpis musí uživatel sám provést při každém vytvoření instalačního balíčku aplikace [18]. Pokud si vývojář přeje uveřejnit svou aplikaci na službě Google Play (dříve Android Market), musí si nejdříve u této služby vytvořit vývojářský účet a uhradit registrační poplatek ve výši 25 dolarů. Služba Google Play umožňuje publikování a poskytování aplikací zdarma nebo za určitý poplatek. Čeští vývojáři mohou v této službě uveřejňovat placené aplikace až od 12. 4. 2012.

5 Návrh Aplikace

V této kapitole bude popsán návrh uživatelského rozhraní aplikace. Dále zde bude popsán síťový a aplikační protokol, použitý pro komunikaci zařízení při hře pro více hráčů.

5.1 Organizace grafického rozhraní aplikace

Aplikace je rozdělena na několik logických celků, z nichž většina poskytuje grafické uživatelské rozhraní, přes které je možno orientovat se v různých částech aplikace. Toto grafické rozhraní bylo navrženo s ohledem na skutečnost, že zařízení se systémem Android mohou mít velmi rozdílné velikosti displejů. Aplikaci je tedy možno bez problému zobrazit na větších i menších obrazovkách. Velikost jednotlivých grafických objektů se mění s velikostí obrazovky, ovládání aplikace by tak nemělo dělat problémy ani na zařízeních s menším displejem. Grafické rozhraní je také přizpůsobeno možnosti otočení displeje. Některá grafická okna jsou nastavena tak, aby zobrazovala stejný obsah při orientaci na výšku i na šířku. Jiná okna pak při otočení displeje zobrazují grafické prvky v jiném uspořádání, které lépe vyhovuje dané orientaci obrazovky. Příkladem je například herní plocha, která při orientaci na výšku vykresluje ve spodní části obrazovky informační panel. Při orientaci na šířku by však tento panel ve spodní části displeje spíše překážel z důvodu nedostatku místa, proto je zajištěno, aby se panel vykreslil na pravém okraji obrazovky. Schéma propojení všech grafických částí aplikace je znázorněno na obrázku 5.1.



Obrázek 5.1: Schéma propojení grafického rozhraní aplikace

5.2 Navržený aplikační protokol

Navržený protokol slouží ke komunikaci přístrojů při síťové hře a zároveň je možné jej využít pro komunikaci s umělou inteligencí. V aplikaci je rozlišováno několik typů zpráv, přičemž každý typ zprávy má definovánu určitou podobu obsahu zprávy. Pokud zpráva obsahuje položky, pak je jejich syntaxe následující:

n*azev_polozky=hodnota_polozky;\n*

Rozlišované typy zpráv:

MESSAGE_READ

Tento typ označuje zprávu přijatou přes Bluetooth při síťové hře. Povolené položky obsahu zprávy jsou:

row – Číslo řádku, na který provedl soupeř svůj tah. Tato položka je použita ve hrách piškvorky a reversi.

col – Číslo sloupce, na který provedl soupeř svůj tah. Tato položka je použita ve hrách piškvorky a reversi.

sourceRow – Číslo řádku, ze kterého soupeř provádí svůj tah. Položka je použita ve hře šachy.

sourceCol – Číslo sloupce, ze kterého soupeř provádí svůj tah. Položka je použita ve hře šachy.

destinationRow – Číslo řádku, kam soupeř provedl svůj tah. Položka je použita ve hře šachy.

destinationCol – Číslo sloupce, kam soupeř provedl svůj tah. Položka je použita ve hře šachy.

wantNewGame – informace, zda si soupeř přeje začít novou hru, poté, co předchozí hra byla ukončena remízou nebo výhrou jednoho z hráčů. Povolené hodnoty jsou *true*, pokud si soupeř přeje hrát novou hru, nebo *false*, pokud soupeř novou hru odmítl. Tuto položku používají všechny implementované hry.

promotingTo – Položka používaná ve hře šachy, která určuje výběr hráče provedený pří proměně pěšce. Hodnoty, kterých může tato položka nabývat, jsou *queen* (dáma), *bishop* (střelec), *knight* (jezdec) a *rook* (věž).

AI_TURN

Tento typ zprávy označuje přijatý tah od umělé inteligence. Položky této zprávy jsou:

row – Číslo řádku, kam provedl soupeř svůj tah. Tato položka je použita ve hrách piškvorky a reversi.

col – Číslo sloupce, kam provedl soupeř svůj tah. Tato položka je použita ve hrách piškvorky a reversi.

MOVE

Zpráva, oznamující umělé inteligenci tah, který provedl uživatel. Položky této zprávy jsou stejné jako u typu *AI_TURN*.

INITIALIZE

Tato zpráva oznamuje umělé inteligenci začátek nové hry. Po přijetí této zprávy umělá inteligence inicializuje všechna data potřebná pro hraní hry. Tato zpráva neobsahuje žádné informační položky.

BEGIN

Touto zprávou oznamujeme umělé inteligenci, že má zahájit hru a provést první tah. V této zprávě nejsou obsaženy žádné doplňující informační položky.

CONNECTION_LOST

Zpráva zaslaná vláknem, starajícím se o síťové připojení, pokud dojde k přerušení spojení.

6 Implementace

V této kapitole je popsána tvorba grafického uživatelského rozhraní, řešení umělé inteligence pro hraní her piškvorky a reversi, a také způsob komunikace s vláknem, starajícím se o příjem a zasílání síťových zpráv.

6.1 Uživatelské rozhraní

K tvorbě uživatelského rozhraní jsem v aplikaci využil převážně deklarativní způsob skrze soubory XML. Díky vhodné volbě jednotek je aplikace zobrazitelná na přístrojích s různou velikostí obrazovky. Jedná se o jednotky dp (Density-independent Pixels) a sp (Scale-independent Pixels), z nichž dp vyjadřuje hodnotu, která se před použitím přepočítá podle hustoty pixelů na daném zařízení, a sp je hodnota odvozená od velikosti písma, kterou má uživatel ve svém systému nastavenou jako preferovanou. Dále jsou uvedeny a podrobně popsány grafické celky aplikace, které můžeme vidět na obrázku 5.1.

6.1.1 Výběr hry

Jedná se o vstupní obrazovku aplikace, kde je uživateli prezentována sada her, ze kterých si může jednu zvolit. Každá hra je zde prezentována obrázkem, pořízeným v průběhu dané hry. Uživatel by tak neměl mít problém na první pohled rozeznat, o kterou hru se jedná, a to i přes to, že obrázky nejsou doprovázeny žádným textovým popiskem. Obrázky jsou implementovány jako grafický objekt *Button*, představující tlačítko. Na každé tlačítko je pak jako pozadí nahrán obrázek z dané hry. Tato tlačítka jsou při orientaci displeje na výšku zabalena v grafickém elementu *ScrollView*, při orientaci na šířku pak v elementu *HorizontalScrollView*. Tyto elementy umožňují rolování v seznamu her, pokud tento seznam přesahuje velikost obrazovky. Zmáčknutí jednoho z tlačítek uživatele přesměruje na obrazovku s volbou způsobu hry. Stisk tlačítka *zpět* ukončí aplikaci.

6.1.2 Volba způsobu hry

Tato část aplikace dává uživateli na výběr ze tří způsobů hry. První z možností je hra proti umělé inteligenci, která byla implementována pro hry piškvorky a reversi. Druhá možnost nabízí hru pro dva hráče na jednom zařízení. Při výběru jedné z těchto dvou možností se ihned spustí daná hra. V případě výběru třetí možnosti, která představuje hru po síti, bude uživatel přesměrován na obrazovku nastavení síťové hry. Pokud však není na zařízení spuštěn Bluetooth, bude uživatel nejdříve požádán o povolení k jeho aktivaci. Žádost o povolení k aktivaci Bluetooth je nejdříve předána systému, ten zobrazí dialog, ve kterém si uživatel může vybrat, zda zařízení aktivuje či neaktivuje. O výsledku tohoto rozhodnutí nás systém informuje, musíme však implementovat metodu *onActivityResult*, kde nám bude zaslán kód identifikující uživatelovu volbu. Při stisku tlačítka *zpět* se obrazovka vrátí na výběr hry. Stisknutím tlačítka *menu* bude uživateli zobrazen seznam možností pro danou hru. Grafické prvky využité v tomto okně aplikace jsou stejné jako na obrazovce výběru hry v části 6.1.1.

6.1.3 Menu hry

V této části má uživatel k dispozici několik možností, které mění grafický vzhled dané hry nebo například velikost herní plochy. Každá z možností má nastavenu nějakou hodnotu, ať už je to základní hodnota nastavena při instalaci aplikace nebo hodnota vybraná uživatelem. Aktuální hodnota všech možností je vypsána vedle názvu dané možnosti. Uživatelem nastavené hodnoty jsou uloženy v objektu třídy *SharedPreferences*, kterou Android poskytuje pro ukládání menšího množství uživatelských dat. Data uložena v tomto objektu zůstanou zachována i při vypnutí a opětovném zapnutí aplikace, proto se ideální hodí i pro ukládání uživatelských nastavení v této aplikaci. Pro zobrazení seznamu možností je využit prvek *ListView* a adaptér, který poskytuje přístup k datům seznamu a uvádí, jak mají být položky zobrazovány. Pro dosažení správného zobrazovaní položek a pro případnou úpravu těchto položek byla implementována vlastní třída *menuAdapter*, rozšiřující třídu *ArrayAdapter<String>*. Označením jedné z položek se uživateli zobrazí dialog *AlertDialog, ve kterém* má na výběr z několika předem definovaných možností a jeho výběr, potvrzený příslušným tlačítkem v dialogu, bude uložen do paměti nastavení. Při každé změně nastavení je uživatel o této změně informován krátkou zprávou *Toast Notification* a zároveň je v seznamu upravena aktuálně vypsaná hodnota u dané možnosti. Při stisku tlačítka *zpět* je uživatel vrácen na výběr způsobu hry.

6.1.4 Spuštění síťové hry

Zde má uživatel na výběr pouze ze dvou možností. Při založení síťové hry bude vyžadováno povolení k zapnutí viditelnosti zařízení Bluetooth na 120 sekund. Žádost o aktivaci viditelnosti zařízení probíhá téměř totožně jako žádost o zapnutí Bluetooth popsaná v části 6.1.2. Poté, co uživatel povolí aktivaci viditelnosti, bude přesměrován na obrazovku čekání na soupeře. Druhou možností je připojení se k již založené hře. Zvolením této možnosti bude aktivována obrazovka s výběrem vzdáleného zařízení. Stiskem tlačítka *zpět* se aplikace vrátí k výběru způsobu hry. Pro zobrazení této obrazovky je využito stejných grafických objektů, jako u volby způsobu hry v části 6.1.2.

6.1.5 Čekání na soupeře

Po spuštění této aktivity bude aplikace čekat na připojení vzdáleného zařízení. Pokud se v časovém limitu nepřipojí žádný klient, bude uživatel vyzván k opětovnému aktivování viditelnosti Bluetooth. Pokud uživatel odmítne prodloužit viditelnost zařízení nebo pokud v průběhu čekání na klienta zmáčkne tlačítko zpět, bude přesměrován zpět na spuštění síťové hry. Grafické rozhraní tohoto okna je představováno pouze prvkem *LinearLayout*, který zajišťuje vyplnění celé obrazovky stejným pozadím, jaké je použité v ostatních částech aplikace. Pro zobrazení informace o čekání na soupeře je využita instance třídy ProgressDialog.

6.1.6 Výběr vzdáleného zařízení

Při spuštění této části aplikace se nejdříve vypíší názvy zařízení, se kterými se již v minulosti tento přístroj přes Bluetooth spároval. Dále pak budou postupně přibývat nová zařízení, ze kterých bude zachyceno broadcastové Bluetooth vysílání. Výběrem jedné z položek se aplikace pokusí s daným zařízením navázat spojení, a pokud vše proběhne v pořádku, oběma hráčům se spustí obrazovka se samotnou hrou. Při stisku tlačítka *zpět* bude uživatel vrácen zpět na obrazovku výběru síťových možností. Grafické rozhraní je zde řešeno velmi podobně, jako v menu hry v části 6.1.3.

6.1.7 Herní plocha

Nejdůležitější částí uživatelského rozhraní je herní plocha, na které každá z her probíhá. Skládá se ze čtverečků, které se označují jako herní pole. Další částí herní plochy jsou herní symboly, které jsou odlišné pro každou z her. Jedná se například o křížky a kolečka ve hře piškvorky nebo o figurky ve hře šachy. Herní plocha je celá řešena programaticky, jelikož je potřeba, aby se zobrazení na ploše měnilo v průběhu času podle akcí uživatele.

Původní návrh byl vytvořit pole tlačítek, která by představovala herní políčka. Tato myšlenka však byla jen stěží proveditelná, neboť Android nepodporuje zapouzdřující layout, který by umožnil listovat polem těchto tlačítek horizontálně i vertikálně zároveň. Proto jsem nakonec použil objekt *Canvas*, do kterého zakresluji veškeré objekty, které se mají zobrazit, jako bitmapy transformované na požadovanou velikost. Pokud herní plocha přesahuje velikost displeje na zařízení, je možné se pohybem prstu po této ploše plynule posunovat. Při položení prstu na displej je zaznamenána pozice a při každém posunu jsou vypočítány nové souřadnice, kam se má zobrazení herní plochy posunout. Aby nedocházelo k situacím, kdy se uživatel pokouší označit herní pole a přitom nepatrně pohne prstem, čímž by se místo označení pole posunulo zobrazení herní plochy, je nutné, aby uživatel pohnul prstem na obrazovce alespoň o minimální definovanou vzdálenost, aby se tento pohyb vyhodnotil jako úmyslný pohyb po herní ploše.

Stisknutím herního pole může dojít k aktivaci určité události, typicky provedení tahu na této pozici. Pokud hra skončí vítězstvím jednoho z hráčů nebo remízou, je hráčům nabídnuta možnost zahájit novou hru. Pakliže hráči odmítnou, mohou si prohlédnout herní plochu ve stavu, v jakém se nacházela na konci dané hry. Při síťové hře je potřeba, aby novou hru potvrdili oba hráči. V případě, že jeden z hráčů hru odmítne, je o tomto rozhodnutí informován i druhý hráč krátkou textovou zprávou, která se objeví na obrazovce. Stisknutí tlačítka *zpět* provede návrat k výběru způsobu hry.

6.2 Umělá inteligence

V aplikaci Logické hry má uživatel možnost zahrát si proti umělé inteligenci ve hrách piškvorky a reversi. Tvorba umělé inteligence pro hru šachy, která by dokázala hrát hru alespoň na průměrné úrovni, by byla časově příliš náročná. Z tohoto důvodu nebyla možnost hrát šachy proti umělé inteligenci zahrnuta do finální verze aplikace. Rozhodně se však jedná o výzvu do budoucna, kdy se o implementaci této umělé inteligence určitě pokusím.

Umělá inteligence je na začátku hry spuštěna ve vlastním vlákně, čímž se zajistí, že výpočet tahu nebude blokovat překreslování herní plochy, ke kterému dochází ve hlavním vlákně aplikace. Komunikace s herním jádrem, které se stará o zpracování tahů probíhá pomocí třídy *Handler*, která umožňuje zasílání zpráv mezi vlákny.

6.2.1 Reversi

Při návrhu umělé inteligence pro hru reversi jsem se rozhodl zaměřit na rychlost nalezení výsledného tahu. Proto jsem určil limit pro hledání tohoto tahu na dobu jedné sekundy. Tím je způsobeno, že nalezený tah bude pravděpodobně méně kvalitní než by byl tah nalezený za delší dobu. Na druhou stranu však uživatel získá velmi rychlou odezvu od umělé inteligence a nebude tak muset dlouze čekat na vyhodnocení tahu.

Vyhledávání tahu

V původní verzi aplikace jsem využil algoritmu Negascout, jelikož nabízí velikou redukci prohledávaných tahů oproti algoritmu MiniMax a při použití vhodně navržené vyhodnocovací funkce i oproti algoritmu AlfaBeta. Ukázalo se však, že ani tento algoritmus nebyl při výběru tahu dostatečně rychlý a i při relativně nízkých úrovních zanoření trval výběr tahu několik sekund. Důvodem je pravděpodobně menší rychlost procesoru na přenosných zařízeních. Řešením by mohla být větší optimalizace některých pomocných výpočtů nebo zvolení, popřípadě vytvoření vhodnějšího datového typu pro reprezentaci stavu herní plochy.

Ve výsledné verzi umělé inteligence je použit algoritmus MiniMax se zanořením do druhé úrovně pro každý aktuálně proveditelný tah. Podle ohodnocení těchto tahů se vyberou tři nejlepší tahy, které jsou dále rozvinuty a algoritmem MiniMax opět se zanořením do druhé úrovně, jsou pro každý z těchto tahů vybrány dva nejlepší následující tahy. Tento proces se opakuje i pro oba vybrané tahy, čímž získáme jeden nejlepší tah nalezený v šesté úrovni zanoření. Hodnota tohoto tahu je navrácena až do nejvyšší úrovně a je přiřazena tahu, který vedl do tohoto stavu. Tím získají původně vybrané nejlepší tři tahy nové ohodnocení a jako výsledný tah se vybere ten s nejvyšší hodnotou.

Ohodnocení pozice

Ve vyhodnocovací funkci jsem pro určení hodnoty daného stavu použil rozdíl mezi počtem vlastních a soupeřových kamenů. K výslednému rozdílu je připočtena určitá hodnota za každý tah, který je možno z tohoto stavu provést. Po přidělení vypočítané hodnoty danému tahu se k tomuto ohodnocení připočte také statická hodnota pozice, na kterou je tah proveden. V průběhu hry může dojít i ke změně hodnoty některých pozic. K tomu dochází, pokud dojde k zabrání některého z rohů herní plochy jedním z hráčů. Po uvážení jsem se rozhodl k výsledné hodnotě přičíst i ohodnocení vhodnosti tahu vzhledem ke svému okolí, pokud má být tah proveden na jeden z okrajů herní plochy. Tímto bylo docíleno lepšího zvládání soubojů, odehrávajících se na okrajích herní plochy, které jsou pro další vývoj hry obzvláště důležité.

6.2.2 Piškvorky

Při tvorbě umělé inteligence pro hru piškvorky jsem se inspiroval prací pana Ing. Jana Kouřila. Původním úmyslem bylo použití algoritmu Negascout, ten se ale neosvědčil již ve hře reversi. Ve hře piškvorky je však situace ještě obtížnější, neboť celkový počet prohledávaných tahů (dokonce i při uvážení jen těch vhodných) je mnohem vyšší než u hry Reversi. Při zanořování je pro každý provedený tah potřeba vypočítat útočnou matici a obrannou matici, podle kterých je pak určen další tah. Tento způsob vyhledávání nejlepšího tahu trval neúnosně dlouhou dobu, proto jsem nakonec od této možnosti upustil a na místo toho implementoval řadu optimalizací pro umělou inteligenci bez možnosti prohledávání do hloubky.

Výběr tahu tak probíhá vytvořením útočné a obranné matice, které přiřazují číselné ohodnocení každému hernímu poli podle zadané funkce. Mezi vylepšení patří například výběr tahu v případě, že více tahů dostane v jedné matici nejlepší ohodnocení. V takovém případě se přihlédne k tomu, jaké hodnocení dostal daný tah v druhé matici. Pokud ani hodnocení z druhé matice nerozhodne o nejlepším tahu, pak je ze skupiny tahů s nejlepším hodnocením vybrán jeden za pomocí generování náhodného čísla. Tím se mimo jiné zajistí i to, že úvodní tah umělé inteligence nebude vždy stejný.

6.3 Řešení síťové komunikace

Služba, starající se o síťovou komunikaci, je spuštěna stejně jako umělá inteligence ve vlastním vlákně. Čtení dat ze sítě probíhá ze vstupního datového proudu v neustálém cyklu, kdy při každém načtení nové zprávy jsou tato data zaslána instanci třídy *Handler* v hlavním vlákně aplikace. Zde dojde ke zpracování zprávy a provedení příslušné reakce na zprávu. Poté, co uživatel provede svůj tah, jsou data zaslána vláknu se síťovou komunikací a dále předána do výstupního datového proudu. Pokud během čtení dojde k chybě (např. když jeden z uživatelů vypne Bluetooth), zašle se hlavnímu vláknu zpráva *CONNECTION_LOST* a hra se přeruší.

7 Testování aplikace

S použitím ADB (Android Debug Bridge) bylo možno aplikaci průběžně testovat na telefonu HTC Desire HD, připojeným k počítači přes USB kabel. V telefonu byl v době testování nainstalován systém Android 2.3. Spouštění aplikace vždy probíhalo velmi rychle a trvalo nejdéle několik sekund, což velmi urychlovalo a tím i usnadňovalo testování. Důležité však bylo otestovat tuto aplikaci i na jiných přístrojích, než pouze na jednom konkrétním zařízení. Proto jsem využil možnosti pracovat s Android emulátorem. Tento emulátor umožňuje vytvoření virtuálního zařízení, kterému můžeme nastavit parametry jako je například velikost obrazovky nebo hustota pixelů. Díky tomu lze testovat aplikaci na zařízeních s různou hardwarovou konfigurací, aniž bychom tato zařízení skutečně vlastnili. Problémem je však vysoká náročnost tohoto emulátoru na procesor počítače a paměť RAM, obzvláště pak, pokud se rozhodneme emulovat zařízení s větším displejem.

Aplikaci jsem úspěšně otestoval na virtuálních zařízeních se systém Android od verze 2.1 až po nejnovější verzi 4.0.3. Dále jsem aplikaci zprovoznil na přístrojích s rozlišením 320 na 240 s hustotou pixelů 120 dpi, 480 na 320 s hustotou pixelů 160 dpi, 800 na 480 s hustotou pixelů 240 dpi a 1280 na 800 s hustotou pixelů 240 dpi. Není tedy problém aplikaci používat na telefonech s malými displeji ani na tabletech s velkými displeji. Toho je docíleno především používáním jednotek dpi, které jsou relativní k hustotě pixelů na obrazovce.

V pozdější fázi vývoje se projevil jeden veliký nedostatek Android emulátoru a tím je nepodporování Bluetooth. Vzhledem k důležitosti síťové komunikace přes Bluetooth pro mou aplikaci byl tento nedostatek kritický a to z důvodu nemožnosti testovat správnou funkčnost implementované síťové komunikace. Situaci jsem vyřešil stažením speciálně upravené verze Android-x86, kterou jsem nainstaloval na virtuálním stroji a následně propojil s ADB. Tento způsob již umožňoval využití Bluetooth a navíc se ukázalo, že systém ve virtuálním stroji funguje mnohem plynuleji než v emulátoru. Proto jsem nadále aplikaci testoval pouze na telefonu HTC a ve virtuálním stroji.

8 Závěr

Cílem práce bylo vytvořit mobilní aplikaci pro platformu Android, která umožňuje hraní několika logických her, ať už proti živému soupeři, nebo proti umělé inteligenci. Implementovány jsou hry piškvorky, šachy a reversi. Ke hrám reversi a piškvorky je pak k dispozici také umělá inteligence, proti které může uživatel hrát. U všech her je také podporována možnost hraní proti lidskému soupeři na jednom zařízení, popřípadě hraní přes bezdrátovou síť za pomocí Bluetooth. Aplikace je přizpůsobena možnosti hraní na zařízeních s různě velikými obrazovkami a nedělá jí tedy problémy zobrazování na malém telefonu ani na větším tabletu. Ke každé hře je uživateli k dispozici sada několika možnosti nastavení, například je to počet řádků a sloupců nebo přizpůsobení herní plochy velikosti obrazovky.

Díky této práci jsem zjistil, jak probíhá vývoj aplikací pro mobilní zařízení a naučil jsem se efektivněji používat programovací jazyk Java. Při psaní tak rozsáhlého projektu se také projevilo, jak důležité je komentovat napsaný zdrojový kód, neboť i člověk, který tento kód psal, po čase zapomene, co přesně některé složitější konstrukce provádějí. Při studování a řešení umělé inteligence jsem se pak naučil mnoho o způsobech, jakými se implementace umělých inteligencí provádí. Přínosem také bylo řešení zobrazení na různých velikostech displejů. Díky tomu jsem se naučil, jak se Android zachová na různých hardwarových konfiguracích a jak lze tohoto využít při optimalizování zobrazení. V neposlední řadě jsem také zjistil, jak Android zachází s aplikacemi a jak zajišťuje jejich zabezpečení.

Jako možná rozšíření do budoucna se z mého pohledu nejdůležitějším jeví dokončení umělé inteligence pro hru šachy. Uživatelé aplikace jistě ocení, pokud budou mít na výběr možnost hraní proti umělé inteligenci ve všech třech dosud implementovaných hrách. Dalším rozšířením by mělo být zvětšení počtu herních nastavení v menu každé hry. Například by bylo vhodné, aby si uživatel mohl zvolit, který z hráčů začne hrát jako první. Velmi vhodné by také jistě bylo vylepšení již implementovaných umělých inteligencí pro hry reversi a piškvorky tak, aby mohly lépe konkurovat lidskému hráči. Případně by uživateli mohl být umožněn výběr obtížnosti umělé inteligence. V budoucnu by dokonce mohly nabídku současných tří her rozšířit i některé další hry, jako například dáma. Jedním z dalších rozšíření by také mohlo být umožnění importování již implementovaných her přímo do aplikace. Uživatel by tak mohl mít všechny logické hry v jedné aplikaci a tím by mu bylo usnadněno jejich vyhledávání. Na možnost přidání nových her je aplikace připravena, ovšem importování cizích aplikací by bylo potřeba nejdříve podrobněji nastudovat a zjistit, jaké možnosti poskytuje Android při řešení takového problému. Po umístění aplikace na Google Play se jistě objeví i některé uživatelské postřehy a návrhy, které by taktéž bylo možné implementovat.

Literatura

[1] ZBOŘIL, František. *Základy umělé inteligence*: *studijní opora* [online]. Vysoké učení technické v Brně, Fakulta informačních technologií, 2006. [cit. 2012-5-15]

[2] Minimax. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-15]. Dostupné z: http://en.wikipedia.org/wiki/Minimax

[3] Alpha-beta pruning. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-15]. Dostupné z: http://en.wikipedia.org/wiki/Alpha-beta_pruning

[4] Negascout. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-15]. Dostupné z: http://en.wikipedia.org/wiki/Negascout

[5] Near field communication. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-15].
Dostupné z: http://en.wikipedia.org/wiki/Near_field_communication

[6] Comparison of Android devices. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-15].
Dostupné z: http://en.wikipedia.org/wiki/List_of_Android_devices

[7] Wi-Fi. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-15]. Dostupné z: http://en.wikipedia.org/wiki/Wi-fi

[8] Wi-Fi Direct. *Android Developers* [online]. [cit. 2012-05-15].
Dostupné z: http://developer.android.com/guide/topics/wireless/wifip2p.html

[9] Platform Versions. *Android Developers* [online]. [cit. 2012-05-15].
Dostupné z: http://developer.android.com/resources/dashboard/platform-versions.html

[10] Bluetooth. *Android Developers* [online]. [cit. 2012-05-15].
Dostupné z: http://developer.android.com/guide/topics/wireless/bluetooth.html

[11] Bluetooth. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-15]. Dostupné z: http://en.wikipedia.org/wiki/Bluetooth

[12] Šachy. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-15]. Dostupné z: http://cs.wikipedia.org/wiki/%C5%A0achy

[13] What is Android?. *Android Developers* [online]. [cit. 2012-05-15].
Dostupné z: http://developer.android.com/guide/basics/what-is-android.html

[14] Android (operating system). In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-05-15].
Dostupné z: http://en.wikipedia.org/wiki/Android_(operating_system)

[15] History of Android. YADAV, Manish. *Tech2crack* [online]. 30.12.2011 [cit. 2012-05-15]. Dostupné z: http://www.tech2crack.com/history-android/

[16] The History of Android. RUBIO, Justin. *IGN* [online]. 22.12.2011 [cit. 2012-05-15]. Dostupné z: http://www.ign.com/articles/2011/12/23/the-history-of-android

[17] Application Fundamentals: Application Components. *Android Developers* [online]. [cit. 2012-05-15]. Dostupné z: http://developer.android.com/guide/topics/fundamentals.html

[18] Signing Your Applications. *Android Developers* [online]. [cit. 2012-05-15]. Dostupné z: http://developer.android.com/guide/publishing/app-signing.html

Příloha A

Obsah CD

K bakalářské práci je přiloženo CD, které obsahuje tyto části:

aplikace

- **apk** – spustitelný soubor s přeloženou aplikací
- **src** – zdrojové soubory aplikace

dokumentace

- **doc** – technická zpráva vytvořená v programu MS Word 2007
- **pdf** – technická zpráva ve formátu pdf

readme.txt – příručka k překladu a instalaci aplikace