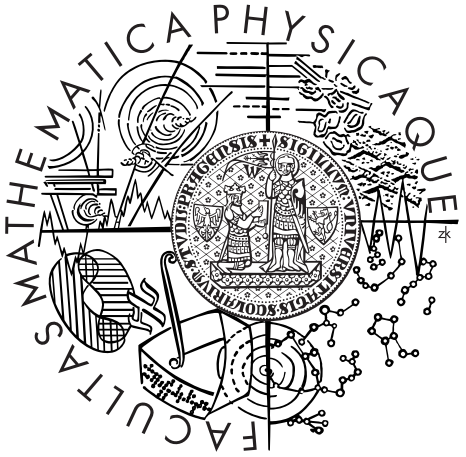
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Miroslav Macík

Implementace hry Shannon switching game pro iOS

Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce:	Mgr. Barbora Vidová Hladká, Ph.D.
Studijní program:	Informatika
Studijní obor:	IP

Praha 2013

Děkuji mé vedoucí Mgr. Barboře Vidové Hladké, Ph.D., Tereze Zábojníkové, rodině a přátelům.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Implementace hry Shannon switching game pro iOS

Autor: Miroslav Macík

Ústav: Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: Mgr. Barbora Vidová Hladká, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: iOS je operační systém určený pro mobilní telefon iPhone, hudební přehrávač iPod a tablet iPad. Shannon switching game je logická grafová hra pro dva hráče. Hru vytvořil americký matematik Claude Shannon. Práce popisuje existující implementace této hry a implementaci pro systém iOS vytvořenou v rámci této práce. Tato implementace umožňuje hru proti virtuálnímu soupeři, hru dvou hráčů na jednom zařízení nebo na dvou zařízeních komunikujících přes internet. Součástí práce je i popis algoritmů pro volbu tahu virtuálního soupeře či způsob generování herních plánů.

Klíčová slova: iOS, on-line hra, grafy

Title: An iOS implementation of the Shannon switching game

Author: Miroslav Macík

Department: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Barbora Vidová Hladká, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Shannon switching game is a logical graph game for two players. The game was created by American mathematician Claude Shannon. iOS is an operating system designed for iPhone cellular phone, iPod music player and iPad tablet. The thesis describes existing implementations of the game and also specific implementation for iOS operating system created as a part of this work. This implementation allows you to play against virtual opponent and also supports multiplayer game consisting of two players playing on the same device or through the Internet against each other. Another component of the thesis is deep insight of algorithms describing next move decisions of virtual opponent or the techniques of game plan generation.

Keywords: iOS, on-line game, graphs

Obsah

Úvod

Shannon switching game je logická grafová hra pro dva hráče. Cílem této práce je seznámit se se systémem iOS a hru pro tento systém implementovat.

První kapitola se věnuje hře Shannon switching game. Nejprve jsou popsány pravidla hry, hrací plán a role hráčů. Druhou část kapitoly doplňují informace o již existujících implementacích této hry.

Druhá kapitola obsahuje informace o hře jako aplikaci pro systém iOS. Jsou zde popsány vlastnosti aplikace, její výhody i nevýhody.

Třetí kapitola je věnována uživatelské dokumentaci. Dokumentace popisuje hráči, jak tuto aplikaci spustit a jak ji používat. Součástí této kapitoly jsou také obrázky z této aplikace.

Čtvrtá kapitola popisuje implementaci aplikace. Kapitola obsahuje popis návrhu aplikace, následuje popis generování grafu, který je poté využit pro vygenerování grafických podkladů pro hru. Další částí je popis samotného generování grafických pokladů a na závěr jsou zmíněny algoritmy pro určení vítěze hry.

Pátá kapitola obsahuje úvod do problematiky programování pro systém iOS. Začít programovat v jazyce Objective-C znamená zvyknout si na několik konvencí, se kterými se člověk programující v jazycích C# nebo C++ nesetká. Kapitola také přináší základní informace o frameworku Cocoa a Cocoa Touch, který je vstupní branou pro tvorbu aplikací pro systém iOS. V několika sekcích, které se snaží čtenáře seznámit se základními vlastnostmi jazyka Objective-C a frameworku Cocoa, jsou obsaženy také krátké ukázky zdrojového kódu implementujícího danou sekci v praxi.

Šestá kapitola obsahuje programátorskou dokumentaci. Jsou zde zmíněny všechny důležité třídy a jejich metody. Kapitola je rozdělena na dvě části, část týkající se generování grafu a část týkající se generování grafických podkladů a průběhu samotné hry.

1. Shannon switching game

1.1 Popis hry

Shannon switching game je strategická hra pro dva hráče. Hru vytvořil americký matematik Claude Shannon, nezávisle na něm také David Gale, jehož hra je známa pod názvy Bridg-It nebo Gale.

Hrací plán
Hrací plán tvoří konečný souvislý graf, který obsahuje dva označené vrcholy A a B.

Role hráčů
Hru hrají dva hráči. Jeden z hráčů, zvaný Shorter, musí ve svém tahu označit (obarvit) jednu z hran grafu. Druhý hráč, zvaný Cutter, musí ve svém tahu smazat jednu hranu grafu.

Průběh hry
Hru zahajuje hráč Shorter, poté se tahy hráčů střídají po jednom tahu.

Konec hry
Hra končí ve chvíli, existuje-li cesta složená pouze z označených hran mezi vrcholy A a B, nebo tuto cestu nelze vytvořit - tedy pokud jsou vrcholy A a B ve dvou různých komponentách souvislosti herního grafu. V prvním případě je vítězem hry hráč Shorter, v případě druhém vítězí hráč Cutter.

1.2 Existující implentace

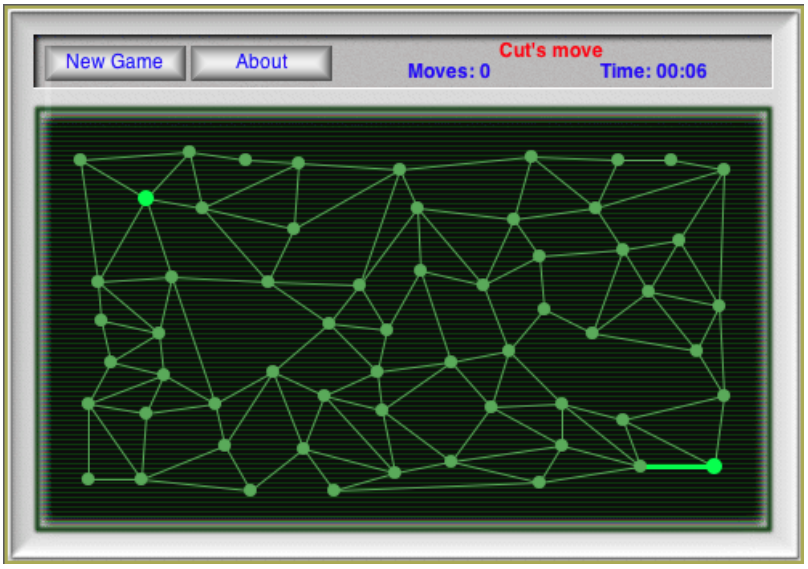
Jako jedinou hru implementující přímo Shannon switching game jsem nalezl hru pod názvem Graph Game. Dále jsem nalezl hru Bridj-It, která implementuje herní pojetí, jež vymyslel David Gale.

1.2.1 Graph Game

Graph game je hra založená na pravidlech Shannon switching game, kterou implementoval Mikhail Kryshen. Tato hra je vytvořena v jazyce Java. Její distribuce probíhá ve formě binárních souborů, hru lze hrát také ve formě Java Appletu na webové stránce [9]. Zdrojový kód hry je k dispozici pod licencí GNU General Public License. Hra má zajímavé grafické zpracování, viz obrázek 1.1.

Vzhledem k systému iOS je hlavní nevýhodou této hry použití programovacího jazyka Java, protože programy vytvořené v tomto jazyce, tedy ani Java Applet na webové stránce, nelze v zařízeních se systémem iOS spustit.

Hra má zajímavé pojetí, pro každou hru je vygenerován nový graf.



Obrázek 1.1: Zpracování hry Graph game

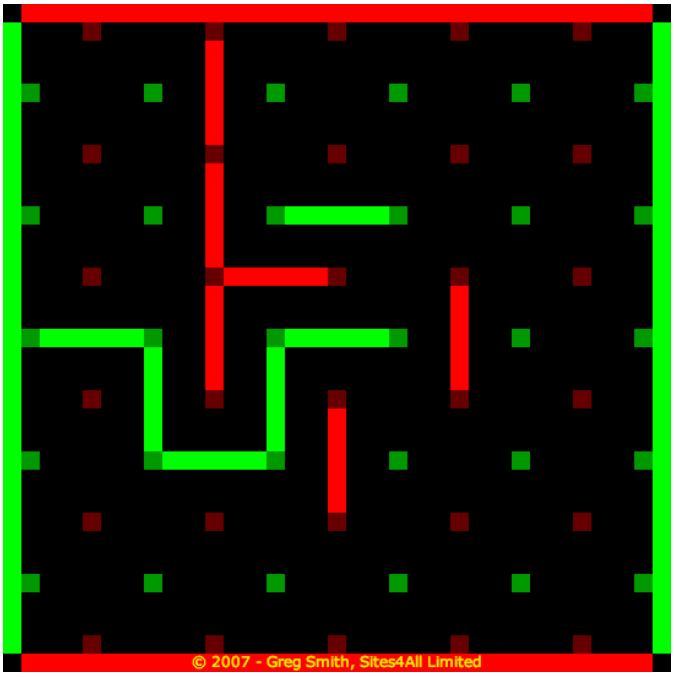
1.2.2 Bridj-It

Bridj-It je implementací hry Gale. Hra nabízí mírně odlišné pojetí než Shannon switching game. Hru hrají také dva hráči, z nichž každý má jednu barvu. Základem hry je čtvercová síť teček dvou barev, označující barvu hráče. Obvod čtverce je také obarven těmito barvami, a to tak, že barva protilehlé strany je stejná. Hráč může ve svém tahu spojit dvě tečky hranou v horizontálním nebo vertikálním směru. Hrany se nemohou překrývat s hranami soupeře. Vítězí hráč, kterému se podaří spojit protilehlé strany vlastní barvy.

Hru implementoval Greg Smith v jazyce PHP. Hra je volně dostupná na webu [3].

Výhodou hry je její dostupnost. Jedná se o webovou stránku, tedy lze hru hrát na jakémkoliv zařízení, které obsahuje internetový prohlížeč.

Hra má jednoduché zpracování, viz obrázek 1.2. Jedná se o HTML stránku, herní prvky jsou vytvořeny pomocí standardních HTML bloků. Hru lze hrát pouze proti počítači a hru začíná hráč, který v předchozím kole prohrál.



Obrázek 1.2: Zpracování hry Gale

Hra Gale má však jednu zásadní nevýhodu. Vzhledem ke hracímu poli, které je vždy pevně dané, lze dokázat, že bude-li hrát začínající hráč chytře, tak pro něj vždy existuje výherní strategie.
Důkaz lze nalézt v [6] a jeho hlavní ideou je nalezení dvou hranově disjunktních koster grafu. Pokud takové dvě kostry v herním grafu existují, existuje i výherní strategie pro začínajícího hráče. Pokud totiž druhý hráč zamezí použití určité hrany prvnímu hráči, existuje vždy ještě jedna hrana (z druhé kostry grafu), která má ekvivalentní spojovací funkci právě jako zablokovaná hrana.

2. Shannon switching game pro iOS

Cílem této práce je implementovat hru Shannon switching game pro systém iOS. Hra pro tento systém doposud nebyla implementována.

2.1 Co aplikace nabízí

Aplikace nabízí možnost hrát hru Shannon switching game na zařízeních se systémem iOS. V současné době je dostupná pro mobilní telefon iPhone, přehrávač iPod Touch a tablet iPad. Hru hrají vždy dva hráči, a to buď na jednom zařízení, nebo na dvou různých zařízeních připojených k síti Internet.

2.2 Lokální hra

Lokální hru hrají dva hráči na jednom zařízení. Hráči se ve svých tazích střídají. Další možností je hra proti virtuálnímu soupeři. Výhodou lokální hry je, že nevyžaduje internetové připojení.

2.3 Síťová hra

Síťovou hru hrají dva hráči na dvou různých zařízeních. Zařízení mezi sebou komunikují skrz internetovou síť. Pro tuto hru je nutné internetové připojení. Soupeře pro síťovou hru si může hráč přímo zvolit, nebo nechat systém vybrat soupeře náhodného.

2.4 Pojetí hry

Hra se odehrává v mikrosvětě, kde vrcholy grafu symbolizují buňky a hrany spojnice mezi nimi. Jeden ze dvou vrcholů grafu, které má za úkol Shorter spojit, je reprezentován jako poslední živá buňka. Obarvení hrany Shorterem je symbolizováno jako uzdravení spojnice mezi buňkami. Vrcholy, které jsou spojeny zdravými hranami a zároveň jsou i těmito hranami připojeny k základnímu zdravému vrcholu, jsou také symbolizovány jako zdravé. Cílem hry pro Shortera je vlastně uzdravit druhý z označených vrcholů grafu, který se nalézá na opačném konci grafu. Naopak tah Cuttera je symbolizován jako přeseknutí nemocné spojnice mezi buňkami, což vede k jejímu úplnému odstranění. Cílem Cuttera je tedy přesekat nemocné spojnice mezi buňkami tak, aby zabránil Shorterovi spojit oba označené vrcholy uzdravenými spoji.

2.5 Požadavky aplikace

Hra vyžaduje systém iOS 5.0 a vyšší. Požadavkem pro síťovou hru je také internetové připojení.

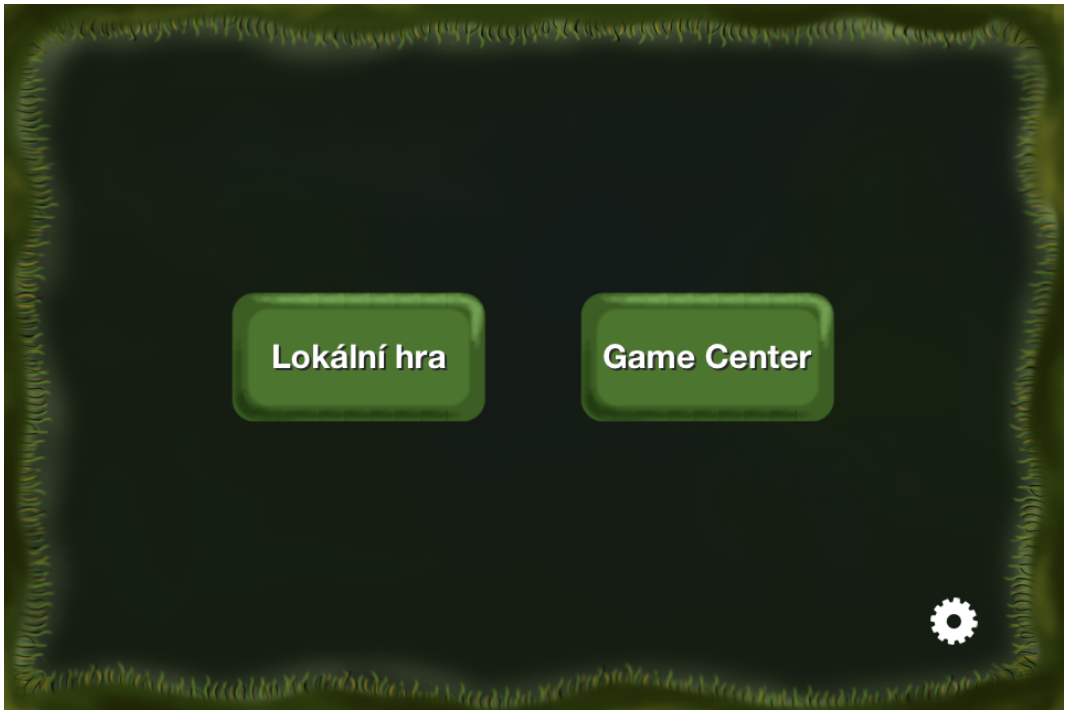
3. Uživatelská dokumentace

3.1 Dostupnost hry

Aplikace je dostupná standardním způsoben skrze Apple App Store pod názvem Life For Cells [10].

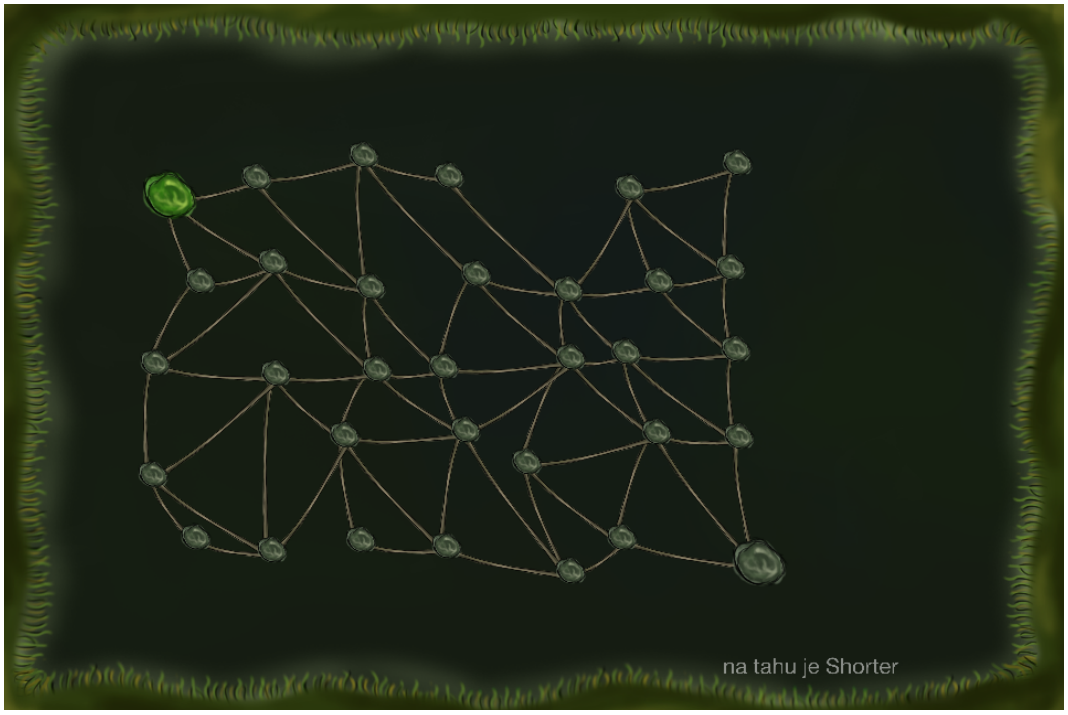
3.2 Ovládání hry

Ovládání hry je velice jednoduché. Po spuštění hry lze vybral lokální nebo síťovou hru zprostředkovanou pomocí služby Game Center. Úvodní obrazovku můžeme vidět na obrázku 3.1.

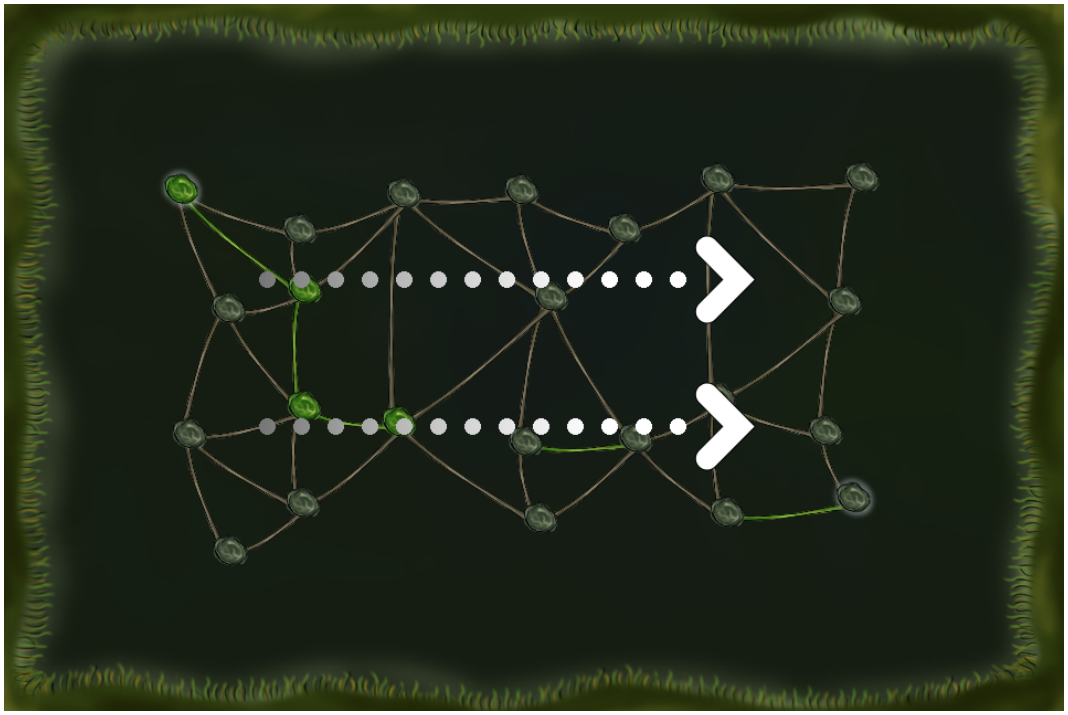


Obrázek 3.1: Uživatel může zvolit lokální nebo síťovou hru prostřednictvím služby Game Center

V samotné hře poté hráč označí, respektive smaže hranu dotekem hrany na obrazovce. Herní obrazovka může vypadat například jako na obrázku 3.2.
Hru je také možné kdykoliv ukončit přetažením dvou prstů přes obrazovku, náznak gesta je na obrázku 3.3.

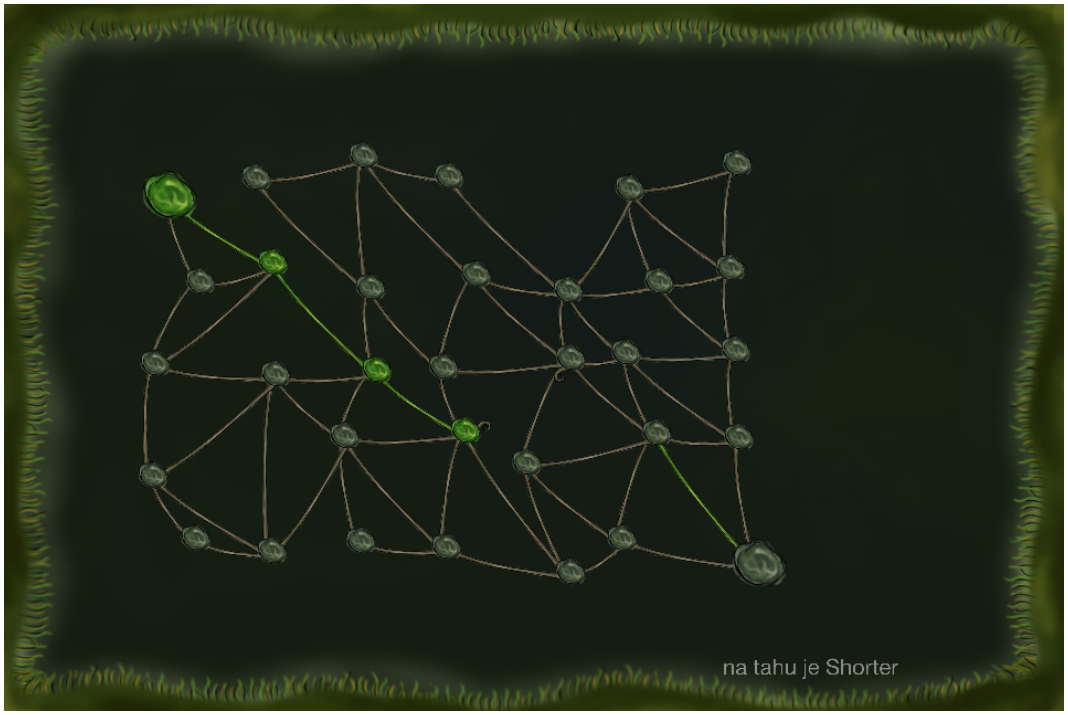


Obrázek 3.2: Začátek hry



Obrázek 3.3: Hru je možné kdykoliv ukončit potažením zleva doprava (takzvané swipe gesto) dvěma prsty

Průběh hry je ukázán na obrázku 3.4. Hrany označené hráčem Shorter jsou označeny zeleně. Pro lepší přehlednost je vždy poslední použitá hrana zvýrazněna. Pokud byla poslední hrana vybrána hráčem Shorter, je kolem hrany naznačena barevná záře. Byla-li poslední hrana přeseknuta hráčem Cutter, jsou u vrcholů naznačeny zbytky přeseknuté hrany. Obě tyto označení mizí z důvodů přehlednosti ihned po provedení dalšího tahu.



Obrázek 3.4: Průběh hry

3.3 Nastavení hry

Z hlavní obrazovky hry lze pomocí symbolu ozubeného kola zobrazit obrazovku nastavení. Na této obrazovce může uživatel nastavit

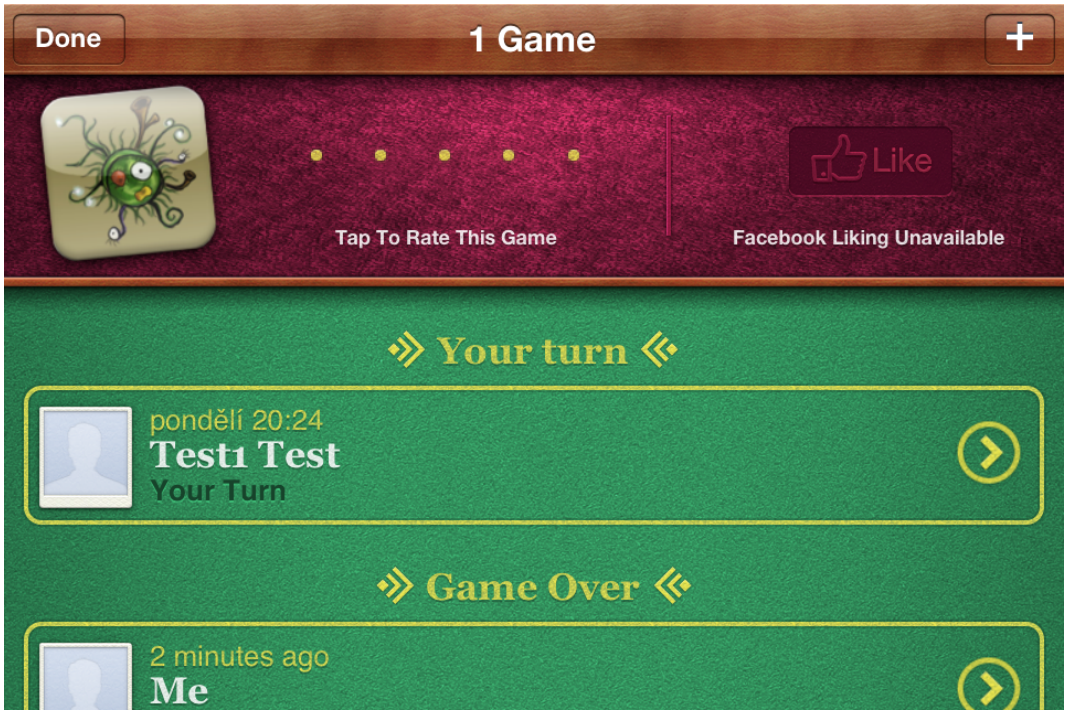
- velikost hracího plánu, kde uživatel může volit mezi třemi druhy velikostí, na základě zvolené velikosti je generována patřičně velká síť vrcholů (viz kapitola popisující implementaci),
- zda-li bude hrát virtuální hráč a případně jakou bude mít roli,
- jaká bude hustota vrcholů v grafu, tedy jaká bude pravděpodobnost vygenerování vrcholu (viz kapitola popisující implementaci).

Obrázek 3.5: Nastavení hry

3.4 Game Center

Pro síťovou hru je nutné využít služby Game Center. Při spuštění aplikace je uživatel vyzván, aby se do služby přihlásil, případně zaregistroval. Po přihlášení je zobrazena obrazovka služby Game Center, které vidíme na obrázku 3.6. Nyní lze vybrat některou z rozehraných her nebo založit hru novou.

Při zakládání nové hry můžeme ke hře přizvat kamaráda, nebo pouze spustit hru a systém nám už automaticky oponenta přidělí. Výběr můžeme vidět na obrázku 3.7.



Obrázek 3.6: Uživatel může zvolit již probíhající hru nebo založit hru novou (viz symbol + v pravém horním rohu)

Obrázek 3.7: Uživatel může vyzvat kamaráda, nebo si nechat soupeře přidělit systémem

4. Popis implementace

4.1 Návrh

Návrh programu lze rozdělit na dvě části. Jedna je určena na generování grafu, druhá pak na uživatelské prostředí a komunikaci s uživatelem. Cílem návrhu je co největší oddělení těchto částí. Část určená pro generování grafu se zaměřuje pouze na graf a už se nestará, jak bude daný graf zobrazen uživateli. Naopak část starající se o uživatelské prostředí nezajímá, jaký graf je jí předán, a pouze se stará o to, aby byl správně vykreslen. Součástí této části jsou také metody, které mají na starost interakci s uživatelem. Ty totiž souvisí právě s vykresleným obsahem.

4.2 Generování grafu

Jedním z požadavků hry je, aby byl vygenerovaný graf rovinný, tedy že existuje nějaké jeho rovinné nakreslení takové, že se žádné dvě hrany nepřekrývají. Tato podmínka není nutná, ale pro snadné ovládání jsou nekřížící se hrany vítané. Dalším ne nutným, ale vítaným požadavkem je, aby neexistovaly příliš dlouhé hrany, které by mohly usnadnit hru Shorterovi. Spojením těchto dvou požadavků vzniká jednoduchý algoritmus, jak docílit žádaného výsledku. Stačí vytvořit graf ve tvaru mřížky obsahující také úhlopříčné hrany, poté s určitou pravděpodobností vynechat některé vrcholy a získáme rovinný graf přesně dle těchto požadavků.

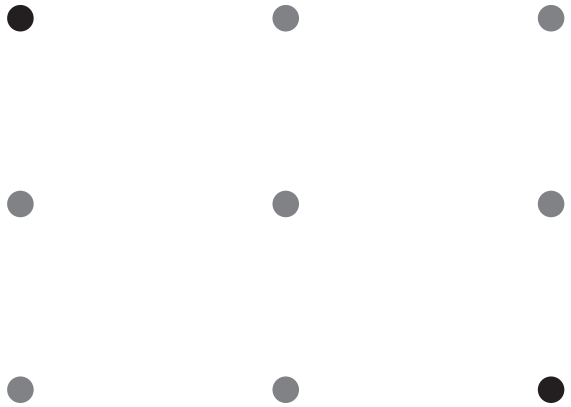
Generování grafu lze rozdělit do dvou částí – na generování vrcholů a generování hran.

4.2.1 Generování vrcholů

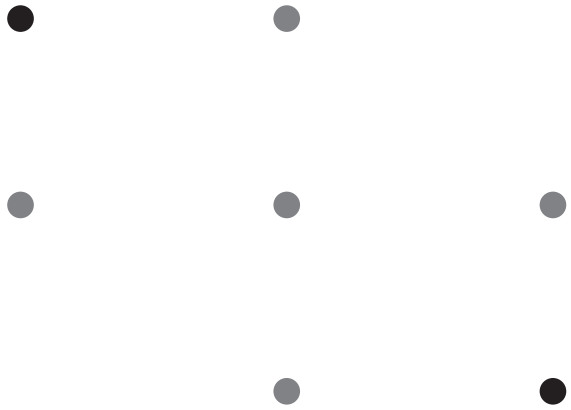
Celá tvorba grafu začíná tvorbou vrcholů. Nejprve je vytvořena síť všech vrcholů, jakou můžeme vidět na obrázku 4.1. Pro příklad je velikosti mřížky 3 x 3 vrcholy.

Následně jsou s určitou pravděpodobností některé vrcholy odstraněny. Pravděpodobnost pro lokální hru lze změnit v nastavení (uživatel může nastavit hodnotu mezi 0,8 a 1), pro síťovou hru je pravděpodobnost pevně nastavena na hodnotu 0,9. Výsledný graf může vypadat například jako na obrázku 4.2. Ve skutečnosti je toto realizováno již při generování vrcholů, tedy na vygenerování vrcholů nepotřebujeme dva průchody, ale pouze jeden. Předtím, než je vrchol vytvořen, je vygenerováno náhodné číslo. Je-li toto číslo menší nebo rovno zvolené pravděpodobnosti, je vrchol vygenerován. V opačném případě se přechází ke generování následujícího vrcholu a daný vrchol není vygenerován.

Nezávisle na zvolené pravděpodobnosti jsou vždy vygenerovány vrcholy A a B, které má za úkol spojit Shorter. Jako vrchol A je označen vrchol na pozici $(1, 1)$, jako B je označen vrchol na pozici (m, n), kde m je šířka a n je výška mříže.



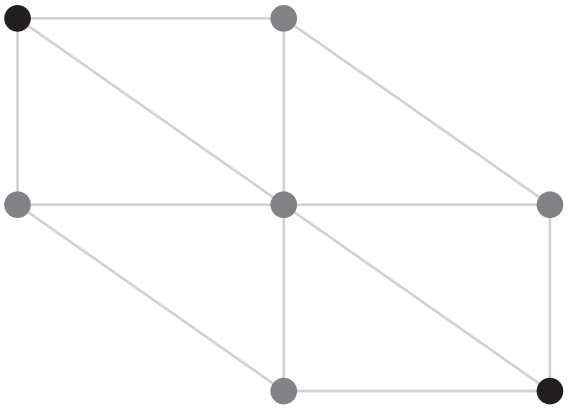
Obrázek 4.1: První krok generování grafu – jsou vygenerovány všechny vrcholy tvořící síť, dosud zatím nespojené



Obrázek 4.2: Druhý krok generování grafu – některé z vygenerovaných vrcholů jsou odstraněny

Důležité také je, aby byl výsledný graf spojitý. Pokud by spojitý nebyl, mohla by nastat situace, kdy by vrcholy A a B ležely ve dvou různých komponentách souvislosti a tedy by byl ihned po zahájení hry vítězem hráč Cutter.

Dalším nastavitelným parametrem je velikost grafu. Protože je generován graf ve tvaru mříže, lze nastavit jeho šířku a výšku. Velikost generovaného grafu je závislá na nastavení aplikace. Reálné velikosti odpovídající možným hodnotám v nastavení jsou 5 x 4, 7 x 5 a 10 x 8.



Obrázek 4.3: Příklad vygenerovaného herního grafu

4.2.2 Generování hran

Generování hran pokračuje tam, kde skončilo generování vrcholů. Nyní jsou hrany vytvořeny mezi zbylými vrcholy. Pravděpodobnost vytvoření nové hrany je pevně nastavena na hodnotu 0,95. Nová hrana musí splňovat následující pravidla:

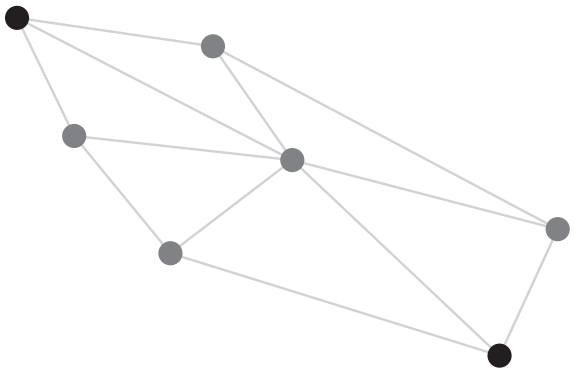
- nekříží žádnou z již existujících hran,
- pokud se nejedná o svislou hranu, tak musí vést pouze do sousedního vrcholu, včetně úhlopříček, a to jenom v případě, že na sousedním místě vrchol existuje, tedy nebyl vymazán při generování vrcholů,
- pokud se jedná o svislou hranu, tak musí vést k nejbližšímu vrcholu pod daným vrcholem, avšak může ignorovat smazané vrcholy.

Aplikováním těchto pravidel na jednotlivé vrcholy grafu získáme herní graf neboli herní plán. Ten může vypadat například jako na obrázku 4.3.

4.3 Generování grafiky pro hru

Metody pro generování grafické části hry jsou dalším celkem, který tvoří tento projekt. Základem grafiky pro hru jsou obrázky pro jednotlivé části grafu — tedy vrcholy a hrany — a doplňující grafika, například pozadí hry.

Před samotným začátkem vykreslování grafu jsou souřadnice jednotlivých vrcholů posunuty o náhodný počet pixelů ve vertikálním i horizontálním směru, což zaručuje, že graf nebude vypadat jako prostá mřížka. Počet pixelů, o které může být vrchol posunut, je zvolen tak, aby nedošlo ke křížení hran nebo překrytí vrcholů. Jako výsledek můžeme získat například graf na obrázku 4.4.



Obrázek 4.4: Příklad náhodné úpravy pozic vrcholů grafu aplikovaný na obrázek 4.3

Vykreslování grafu je realizováno průchodem grafu a provedením těchto kroků nad každým vrcholem:

1. vytvoříme jeho grafickou reprezentaci a tuto reprezentaci vykreslíme na herní plán,
2. vytvoříme grafickou reprezentaci pro hrany vedoucí z tohoto vrcholu a tuto reprezentaci také vykreslíme na herní plán.

Aplikací tohoto postupu na všechny vrcholy získáme výslednou grafickou reprezentaci herního grafu.

4.4 Určení vítěze

Vždy na konci tahu každého hráče probíhá ověření, zda-li se tento hráč nestal vítězem dané hry. Základem ověření je vždy prohledávání do hloubky, ale pro každého z hráčů se mírně liší. Je-li na tahu

- Cutter, je použito obyčejné prohledávání do hloubky, které je spuštěno z vrcholu A a ověřuje, zda-li existuje nějaká cesta do vrcholu B,
- Shorter, je použito upravené prohledávání do hloubky, které používá pro přesun mezi vrcholy pouze hrany označené právě tímto hráčem.

Není-li nalezena žádná cesta mezi vrcholy A a B po tahu Cuttera, vítězí tento hráč. Naopak pokud je po tahu Shortera cesta mezi těmito vrcholy nalezena a je složena pouze z označených hran, vítězí Shorter. Pokud není splněna ani jedna z těchto podmínek, hra pokračuje tahem soupeře.

4.5 Virtuální protihráč

Aplikace v rámci lokální hry nabízí také možnost hrát proti hráči, který je ovládán počítačem. Logika tohoto virtuálního protivníka je stejná v případě, že zastává roli Shortera, i v případě, že ovládá Cuttera.

4.5.1 Algoritmus hry virtuálního protihráče

Algoritmus je založen na použití dvou grafových algoritmů. Prvním je Fordův-Fulkersonův [8] pro výpočet maximálního toku v síti a druhým je Dijkstrův algoritmus [7] pro nalezení nejkratší cesty v ohodnoceném grafu.

Herní graf označíme jako $G = \{V, E\}$, kapacitu hrany budeme značit jako $c : E \to \mathbb{R}$, tok jako $t : E \to \mathbb{R}$ a délku hrany jako $d : E \to \mathbb{R}$.

Postup výběru nejvhodnější hrany
Nejprve je v síti pomocí Fordova-Fulkersonova algoritmu nalezen maximální tok. Jako zdroj je použit vrchol A a jako stok vrchol B. Kapacita vrcholů je určena podle vzorce (4.1).

$$c(e) = \begin{cases} |E| & \text{je-li hrana označená} \\ 1 & \text{jinak} \end{cases} \tag{4.1}$$

Po nalezení maximálního toku následuje hledání minimálního hranového řezu grafu. Díky znalosti maximálního toku na jednotlivých hranách je nalezení řezu jednoduché – stačí nám jeden chytrý průchod grafem do šířky (viz algoritmus 1).

Algoritmus 1 Nalezení minimálního hranového řezu v síti s maximálním tokem

```
for all v ∈ V do
  v ← nenavštíven
end for
fronta ← vrchol A
while fronta není prázdná do
  v ← vrchol z fronty
  v ← navštíven
  for all vrchol s spojený hranou e s vrcholem v do
    if t(e) < c(e) then
      fronta ← s
    end if
  end for
end while
minimální řez ← {}
for all navštívený vrchol v do
  for all vrchol s spojený hranou e s vrcholem v do
    if s je nenavštívený then
      minimální řez ← e
    end if
  end for
end for
return minimální řez
```

Vzhledem k nastaveným kapacitám hran a předpokladu, že hráč Shorter dosud nezvítězil, máme jistotu, že se v minimálním hranovém řezu neobjeví žádná z označených hran.

Důkaz tohoto tvrzení je triviální, stačí nám vyjít z předpokladu, že z vrcholu A do vrcholu B neexistuje cesta tvořená z pouze označených hran. Poté budeme postupně hledat všechny cesty z vrcholu A do vrcholu B a mazat hrany, které mají kapacitu jedna. Na každé takové cestě smažeme minimálně jednu takovou hranu. Tyto smazané hrany díky předpokladu jistě tvoří nějaký hranový řez grafu a jejich podmnožina bude tvořit řez minimální. Tedy minimální řez bude obsahovat pouze hrany s kapacitou jedna.

Tímto je první část hledání nejlepší hrany hotová. Následuje část druhá, ve které využijeme Dijkstrův algoritmus. Budeme hledat nejkratší cestu z vrcholu A do vrcholu B. Délky hran si ohodnotíme podle vzorce (4.2).

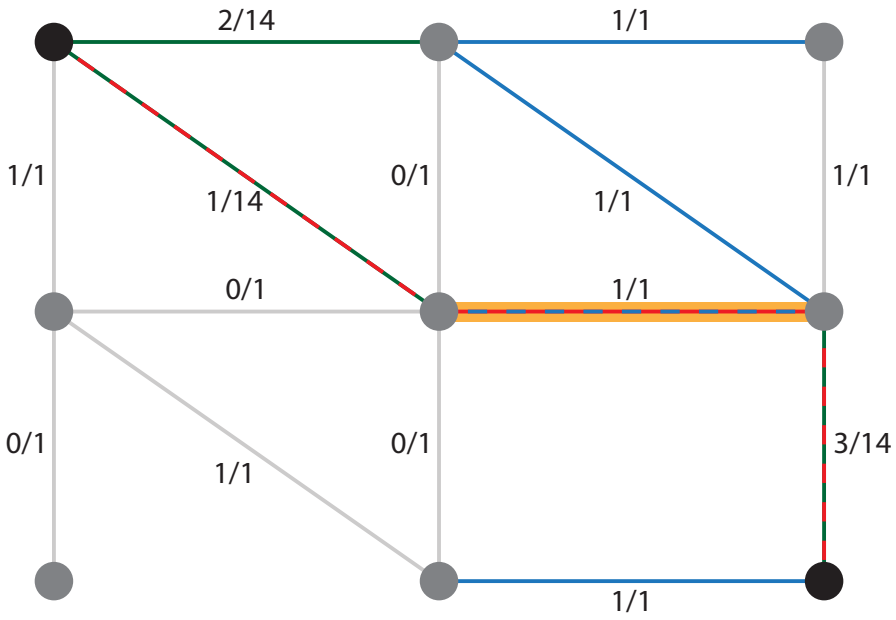
$$d(e) = \begin{cases} 0 & \text{je-li hrana označená} \\ 1 & \text{jinak} \end{cases} \tag{4.2}$$

Tento vzorec nám vlastně říká, že průchod po označené hraně nás nic nestojí. A to je přesně to, co potřebujeme – cesta z označených hran může být co do počtu hran dlouhá, ale oba hráče zajímá pouze to, kolik doposud neoznačených hran dělí vrchol A od vrcholu B.

Nyní máme dvě množiny hran. Jedna množina obsahuje hrany tvořící minimální hranový řez a druhá množina obsahuje hrany ležící na nejkratší cestě z vrcholu A do vrcholu B.

Z povahy minimálního řezu a nejkratší cesty plyne, že tyto množiny mají právě jednu společnou hranu. A přesně tuto hranu vybere náš virtuální soupeř.

Příklad výběru nejvhodnější hrany můžeme vidět na obrázku 4.5. Zeleně označené hrany byly obarveny hráčem Shorter, červeně označené hrany tvoří nejkratší cestu spočtenou Dijkstrovým algoritmem s ohodnocením hran dle (4.2). Modře označené hrany tvoří minimální hranový řez spočtený algoritmem 1. Hrana s oranžovým podkladem je vybrána jako nejvhodnější, protože leží na nejkratší cestě z vrcholu A do vrcholu B a zároveň je obsažena v minimálním hranovém řezu.



Obrázek 4.5: Příklad výběru nejvhodnější hrany virtuálním soupeřem

4.6 Síťová hra

Aplikace nabízí kromě hry dvou hráčů na jednom zařízení a hry proti virtuálnímu soupeři také hru dvou hráčů na dvou zařízeních komunikujících přes internetovou síť.

4.6.1 Game Center

Game Center je služba pro hraní her více hráčů na více zařízeních. Tato služba se poprvé objevila v systému iOS 4.1 pro zařízení iPhone a iPod, od verze iOS 4.2 pak i pro zařízení iPad. Game Center je vlastně jakousi herní sociální sítí. Hráčům nabízí možnost hrát s přáteli nebo s náhodně vybranými soupeři. Jednotlivé hry také mohou pomocí této služby odměňovat hráče body nebo oceněními za určité výkony.
Vývojářům nabízí Game Center jednoduchou možnost, jak do svých her přidat hru více hráčů. V podstatě stačí do her přidat pouze komunikaci mezi hrami jednotlivých hráčů. Další věci, jako je výběr soupeře, už implementuje sama služba Game Center.

4.6.2 Realizace síťové hry

Síťová hra je implementována pomocí služby Game Center jako tahová hra. Výhodou implementace tahové hry je, že uživatel může mít rozehraných několik her současně a vždy, když je na tahu, je upozorněn systémem. Čas na jeden tah není omezen, proto není nutné být připraven na tah soupeře a ihned na něj odpovídat.

Aktuální stav hry, tedy seznam všech provedených tahů jednotlivých uživatelů, je uložen na serveru, který Game Center provozuje. Proto není potřeba stav

jednotlivých her ukládat v zařízení. Nevýhodou tohoto systému ale je, že hráč po provedení svého tahu musí na server odeslat kompletní seznam provedených tahů, nikoliv pouze jím provedený tah. Naopak výhodou je jednoduchost použití toho řešení. Pokud uživatel ukončí aplikaci a za určitou dobu chce ve hře pokračovat, je automaticky ze serveru stažen aktuální stav hry.

Komunikace mezi hráči

Komunikace mezi hráči je řešena také pomocí služby Game Center. Tato služba poskytuje potřebné metody pro zasílání zpráv mezi zařízeními. Po vytvoření hry je nejprve provedena synchronizace a poté jsou zasílány zprávy při provedení tahu.

Synchronizace je provedena ihned po spuštění hry dvou hráčů. V rámci synchronizace jsou nastaveny stejné klíče (seed) náhodného generátoru čísel. To zaručí, že oběma hráčům bude vygenerován stejný graf. Synchronizace se skládá z několika jednoduchých kroků:

- na každém zařízení je vygenerováno náhodné číslo x typu int,
- číslo x je odesláno soupeři,
- čeká se na přijetí čísla y, které vygeneroval soupeř,
- jakmile je číslo y přijato, je klíč generátoru zvolen jako $max(x, y)$.

Synchronizační zprávy nejsou nijak speciálně formátovány.

Zprávy o provedení tahu jsou odesílány v průběhu hry, zprávu odesílá vždy hráč, který provedl tah. Tato zpráva je odeslána soupeři a zároveň je uložena na serveru.

Pokud má uživatel v době přijetí zprávy spuštěnou aplikaci a zároveň zobrazenou aktuální hru, je tah uskutečněn, tedy je provedeno oživení nebo odstranění hrany. Má-li uživatel spuštěnou aplikaci, ale hraje jinou hru, je upozorněn zprávou, že je v dané hře na tahu. Pokud nemá vůbec aplikaci spuštěnou, je mu zobrazena notifikace, která jej vyzve k provedení tahu.

Součástí zprávy je vždy také klíč náhodného generátoru, kterým bylo dané hrací pole vygenerováno. Klíč musí být součástí zprávy, protože pouze z dat uložených na serveru musí být možné vygenerovat aktuální stav hry, tedy hrací pole, oživené a také odstraněné hrany.

Zprávy o provedení tahu jsou odesílány ve formátu JSON[1].

[1] JavaScript Object Notation, `http://www.json.org`

```
{
    seed = 0;
    turns = (
        20,
        26,
        21,
        30,
        22
    );
}
```

Příklad 4.1: Příklad zprávy odeslané na konci tahu.

Příklad 4.1 ukazuje zprávu, kterou na konci svého tahu odeslal hráč Shorter. Zpráva obsahuje slovník dvou hodnot. Hodnota `seed` určuje hodnotu klíče náhodného generátoru, který byl použit pro vytvoření daného herního plánu. Hodnota `turns` obsahuje pole, které má jako své prvky označení hran, které již byly použity. Tyto hrany jsou seřazeny tak, jak byly postupně označeny, tedy první hranu tohoto pole (hrana s číslem 20) označil Shorter ve svém prvním tahu, další hranu (číslo 26), ve svém prvním tahu smazal Cutter.

5. Programování pro iOS

Začít programovat pro systém iOS vyžaduje naučit se několik nových věcí. Nejprve je potřebné seznámit se s jazykem Objective-C, poté s frameworkem Cocoa Touch a s některými řešeními různých typických problémů, například používání delegátů. Tato kapitola si klade za cíl ukázat alespoň několik hlavních problémů, případně věcí, které jsou odlišné od známějších programovacích jazyků, jako je C# nebo C++.

5.1 iOS

iOS je operační systém vyvinutý společností Apple Inc. Prvně byl představen v roce 2007 při příležitosti uvedení telefonu iPhone. Nejprve byl použit v zařízeních iPhone a iPod. Systém je přizpůsoben pro ovládání dotykem či více dotyky.

Tvorbě aplikací pro systém iOS se věnuje [1] a [2]. Tyto knihy jsou zaměřeny na systém iOS 5 a vývoj aplikací pro tento systém popisují formou tutoriálů.

5.2 Objective-C

Objective-C je objektově orientovaný programovací jazyk vytvořený jako rozšíření jazyka C. Základy programování v jazyce Objective-C lze najít v [4].

5.2.1 Syntaxe

Syntaxe jazyka pro práci s objekty je odvozena z jazyka Smalltalk, ostatní syntaxe je identická s jazykem C. Hlavní rozdíl oproti běžným jazykům je spouštění metod. V jazyce Objective-C nejsou volány metody, ale odesílány zprávy.

```
object.method(parameter);
object->method(parameter);
[object method:parameter];
```

Příklad 5.1: Srovnání volání metod v jazyce C# a C++ s odesíláním zpráv v jazyce Objective-C.

Zprávy jsou vyhodnocovány až za běhu programu. Zpráva se skládá z objektu, který má tuto zprávu zpracovat, ze selektoru, který určuje název metody, kterou má být zpráva zpracována, a z nepovinných parametrů zprávy. Během odesílání zprávy neprobíhá žádná kontrola, zda-li umí příjemce tuto zprávu zpracovat. Pokud ji zpracovat neumí, tak nastává vyjímka.

5.3 Cocoa a Cocoa Touch

Cocoa je rozhraní určené programátorům na vývoj aplikací pro systém Mac OS X. Toto rozhraní je vlastně rozšířením jazyka Objective-C. Cocoa Touch je upravenou verzí tohoto rozhraní. Hlavním rozdílem je úprava ovládání, které je přizpů-

sobeno ovládání dotykem. Cocoa Touch je určeno pro vývoj aplikací pro systém iOS. Rozhraní Cocoa se věnuje [4].

5.3.1 Rozšíření Objective-C

Cocoa přidává kompletní řadu nových tříd. Hlavní třídou je bezesporu `NSObject`. Tato třída je předkem všech dalších tříd obsažených v rozhraní Cocoa, případně Cocoa Touch. `NSObject` poskytuje všechny základní metody pro práci s objekty.

V rozhraní jsou také definovány nové datové typy, například `NSInteger`, který je naprosto schodný s datovým typem `int` v jazyce Objective-C.

5.4 Třídy

Jako objektově orientovaný jazyk nabízí Objective-C použití tříd. Použití třídy vyžaduje rozhraní a implementaci. Rozhraní deklaruje třídu a definuje její proměnné a metody, implementace pak samotný kód jednotlivých metod.

5.4.1 Dědičnost

Třídy v jazyce Objective-C mohou mít maximálně jednoho předka, ten se zapisuje v deklaraci metody za název deklarované metody a dvojtečku.

5.4.2 Deklarace metod

Název metody a její parametry se navzájem prolínají, což podporuje čitelnost kódu. Deklarace metody nejprve obsahuje příznak, zda-li se jedná o statickou metody (označeno znakem +) či nikoliv (znak -). Následuje návratový typ v závorkách a poté název metody s parametry. V názvu jsou parametry odděleny dvojtečkou.

```
@interface myClass : parentClass
{
        int value;
}

- (float)divideNumber:(float)dividend byNumber:(float)divisor;
+ (void)staticMethod;
@end
```

Příklad 5.2: Rozhraní třídy `myClass`. Tato třída je potomkem třídy `parentClass`, obsahuje proměnnou `value`, implementuje metodu `divideNumber:byNumber:` a statickou metodu `staticMethod`.

5.5 Protokoly

Protokoly umožňují deklarování metod, které musí (případně může) třída, používající tento protokol, implementovat. Protokoly vlastně částečně nahrazují absen-

ci abstraktních metod. Na druhou stranu má toto řešení i několik výhod. Jednou z nich je, že protokol mohou implementovat třídy, které nemají nic společného a nemusí mít žádnou třídu jako společného předka. Další výhodou je, že můžeme při deklaraci proměnné specifikovat, že objekt dané třídy musí implementovat daný protokol a už se nemusíme starat, jaký objekt bude v proměnné uložen.

```
@protocol Serialization <NSObject>
@optional
- (NSDictionary *)serializeToDictionary;

@required
- (NSString *)serializeToString;
@end

@interface MyClass : NSObject <Serialization>
```

Příklad 5.3: Protokol Serialization obsahující nepovinnou metodu `serializeToDictionary` a povinnou metodu `serializeToString`. Nakonec pak ukázka hlavičky deklarace metody, která daný protokol implementuje.

5.6 Kategorie

Kategorie nabízejí způsob, jak rozšířit již existující třídy o nové metody. Hlavní výhodou je, že nemusíme znát implementaci dané třídy a nemusíme vytvářet vlastní třídu pomocí dědičnosti, abychom mohli novou metodu implementovat. Pomocí kategorie tedy můžeme rozšířit jakoukoliv již existující třídu.

```
@interface NSMutableArray (Additions)
- (void)removeFirstObject;
@end

@implementation NSMutableArray (Additions)
- (void)removeFirstObject
{
    if (self.count == 0)
    {
        return;
    }

    [self removeObjectAtIndex:0];
}
@end
```

Příklad 5.4: Rozhraní a implementace kategorie `Additions`, která rozšiřuje třídu `NSMutableArray` o metodu `removeFirstObject`.

Kategorie tedy nabízejí jednoduchý způsob, jak rozšířit již existující třídy o nové metody. Pokud bychom však chtěli provést rozšíření i o nové proměnné, budeme muset použít dědičnost a vytvořit si vlastní třídu.

5.7 Správa paměti

Správa paměti je založena na systému počítání referencí. Třída `NSObject` poskytuje metody `retain` respektive `release`, pomocí kterých se zvýší, respektive sníží, hodnota `retainCount` o číslo jedna. Jakmile je tato hodnota rovna nule, může být objekt dealokován. Při vytváření nového objektu pomocí metod `alloc` nebo `new` je `retainCount` automaticky nastaven na hodnotu jedna.
Zvláštním případem je metoda `autorelease`, která garantuje, že `retainCount` bude snížen o číslo jedna. Zavoláním této metody je objekt přesunut do autorelease poolu. Snížení proběhne při zániku autorelease poolu, což je zpravidla na konci bloku.

Automatic Reference Counting
Jednou z velkých novinek systému iOS 5 je Automatic Reference Counting, často uváděné jako ARC. ARC je součástí kompilátoru LLVM 3.0 a je samo schopno provádět práci s pamětí. Tedy při použití ARC není potřeba používat metody `retain`, `release` nebo `autorelease`, kompilátor tyto metody sám vkládá na správná místa.

5.8 Distribuce aplikací

Distribuce aplikací zahrnuje několik kroků, které je potřeba vykonat, než se aplikace může dostat do zařízení uživatelů. Nejprve je potřeba zaregistrovat ID aplikace, poté založit samotnou aplikaci, vygenerovat distribuční certifikát a na závěr vytvořit build aplikace s požadovaným certifikátem.

Systém certifikátů
Každá iOS aplikace musí být podepsána certifikátem. Certifikátů existuje několik druhů a liší se podle způsobu, jakým chceme aplikaci použít:

- certifikáty vývojáře, které jsou určeny pro ladění aplikace na zařízení,
- Ad Hoc certifikáty, které slouží pro distribuci testovací verze aplikace (například ukázka aplikace klientům),
- App Store certifikáty, určené pro distribuci aplikace pomocí služby App Store.

iOS Provisioning Portal a systém certifikátů
iOS Provisioning Portal je webová stránka, která umožňuje vytváření ID pro nové aplikace a správu již existujících ID. Dále poskytuje služby spojené s certifikáty aplikací a registraci zařízení pro Ad Hoc distribuci.

Založení aplikace a iTunes Connect
iTunes Connect je stejně jako provisioning portal webová stránka. Tato stránka však slouží pro zakládání aplikace. K založení aplikace je potřeba vytvoření ID aplikace a vyplnění údajů o aplikaci.

Ad Hoc distribuce a distribuce pomocí App Store
Obě tyto distribuce jsou jedinou možností, jak nainstalovat aplikaci do zařízení, které nevlastní vývojář. Je mezi nimi však velký rozdíl.

Ad Hoc distribuce je možná pouze na zařízení, od nichž známe identifikátor UDID. Tento identifikátor je 32 znaků dlouhý řetězec složený z písmen a čísel, který jednoznačně identifikuje zařízení se systémem iOS. Toto UDID musíme nejprve zaregistrovat v systému iOS Provisioning Portal a poté vytvořit Ad Hoc certifikát, který bude daný identifikátor obsahovat. Aplikaci podepsanou tímto certifikátem je pak možné nainstalovat na zařízení s daným identifikátorem jako běžnou aplikaci pomocí programu iTunes. V současné době je možné na jeden vývojářský účet zaregistrovat maximálně 100 UDID identifikátorů ročně.
Typickým příklad použití distribuce pomocí Ad Hoc certifikátu je například možnost zaslat aplikaci klientovi, pro kterého aplikaci vyvíjíme již během vývoje.

Distribuce pomocí App Store pak slouží pro samotnou distribuci aplikace veřejnosti. Aplikace podepsané App Store certifikátem lze distribuovat pouze pomocí obchodu App Store. Před distribucí musí být aplikace zaevidována v systému iTunes Connect.

6. Programátorská dokumentace

Aplikace je naprogramována v jazyce Objective-C za použití frameworku Cocoa Touch. Zde jsou uvedeny popisy důležitých tříd, které jsou v aplikaci použity. Nejprve je uvedena část zaměřující se na konstrukci grafu. Následuje část zaměřená na generování grafické části aplikace a samostatná část věnující se spojení těchto obou částí ve finální aplikaci. Kompletní programátorskou dokumentaci lze nalézt v Příloze 1.

6.1 Tvorba grafu

Tvorbu grafu zajišťují tři základní třídy.

6.1.1 GraphNode

Reprezentuje vrchol grafu. Každý vrchol grafu je určen číslem `tag`, které musí být v rámci grafu unikátní. Také je důležité, aby byly vrcholy číslovány vzestupně. Dále obsahuje seznam hran `edges` a svou pozici `position` pro zobrazení v grafické reprezentaci grafu.

6.1.2 GraphEdge

Reprezentace hrany grafu. Každá hrana grafu je určena číslem `tag`, které musí být v rámci grafu unikátní a jednotlivé hrany musí být číslovány vzestupně, obsahuje dva vrcholy grafu `fromNode` a `toNode`, které určují odkud a kam hrana vede. Hrana také obsahuje označení `marked`, uchovávající informaci o tom, zda-li byla označena malířem.

6.1.3 Graph

Reprezentace grafu pro hrací plán hry. Reprezentuje graf pomocí vrcholů `GraphNode`, které jsou spojeny hranami `GraphEdge`. Umožňuje inicializaci a vygenerování rovinného grafu dané šířky `width`, výšky `height` a zadaných pravděpodobností `nodeProbability` a `edgeProbability` pomocí metody `initWithWidth:height:nodeProbability:edgeProbability:`.

Graf také eviduje dva vrcholy `startNode` a `endNode`, které má za úkol spojit malíř. Pro potřeby hry jsou důležité metody `markEdgeWithTag:` a `removeEdgeWithTag:`, dále metody pro ověření vítězství `existsMarkedPathFromStartToEndNode` a `existsPathFromStartToEndNode`.

Ve hře proti virtuálnímu soupeři jsou navíc využívány metody `minimalCutEdges` a `shortestPathIgnoringMarkedEdges`, na základě kterých je vybrána nejvhodnější hrana pro tah virtálního protivníka.

6.2 Grafické uživatelské rozhraní

Třídy, které slouží pro generování grafického uživatelského rozhraní, například vrcholů nebo hran.

6.2.1 NodeUI

Statická třída pro generování grafické reprezentace vrcholů grafu. Nejdůležitější metodou je `nodeImageViewForNode:`, která vytvoří grafickou reprezentaci konkrétního vrcholu grafu.

6.2.2 NodeImageView

Rozšíření třídy `UIImageView` pro potřeby zobrazování grafické reprezentace vrcholu grafu. Třída umožňuje označit, zda-li byl daný vrchol oživen, pomocí příznaku `recovered`.

6.2.3 EdgeUI

Statická třída pro generování grafické reprezentace hran grafu. Nejdůležitější metodou je `edgeImageViewFromNode:toNode:`, která vytvoří grafickou reprezentaci konkrétní hrany grafu spojující dva vrcholy.

6.3 Hra

Hlavní třídy, které reprezentují obrazovky hry.

6.3.1 MainVC

Třída reprezentující úvodní obrazovku. Úvodní obrazovka umožňuje výběr typu hry a na základě uživatelova výběru spouští daný typ hry. Jsou k dispozici tyto typy her:

- lokální hra, hra pro dva hráče na jednom zařízení,
- síťová hra, hra pro dva hráče na dvou různých zařízeních, které mezi sebou komunikují pomocí sítě internet.

6.3.2 GameVC

Základní třída zprostředkovávající průběh hry. Jejím úkolem je:

- vygenerovat hrací plán (graf),
- vygenerovat grafické znázornění grafu,
- zpracovávat akce provedené hráči,
- určit výherce hry,
- v případě hry proti virtuálnímu soupeři zajistit výběr nejvhodnějšího tahu.

6.3.3 MultiplayerGMVC

Třída pro hru dvou hráčů na dvou zařízeních pomocí sítě Internet. Tato třída je rozšířením třídy `GameVC`, kterou rozšiřuje hlavně o síťovou komunikaci.

6.3.4 Settings

Třída typu singleton, která poskytuje přístup ke globálnímu nastavení.

6.3.5 SettingsVC

Třída reprezentující obrazovku nastavení hry. Uživatel může nastavit:

- velikost herního plánu, skutečné velikosti jsou nadefinovány v metodách třídy `Settings` `graphWidthForGraphSize:` a `graphHeightForGraphSize:`,
- zda-li bude v rámci lokální hry hrát proti virtuálnímu soupeři a jakou bude zastávat případnou roli,
- hustotu vrcholů herního grafu.

6.3.6 GCTurnBasedMatchHelper

Upravená singleton třída z knihy tutoriálů [1]. Třída usnadňuje implementaci tahové hry pomocí služby Game Center.

Závěr

Práce představuje vytvoření aplikace implementující hru Shannon switching game pro systém iOS. Hru je možné hrát lokálně nebo prostřednictvím Internetu.

V rámci této práce proběhlo:

1. seznámení s programovacím jazykem Objective-C
2. seznámení s frameworky Cocoa a Cocoa Touch
3. seznámení s vývojovým prostředím Xcode
4. seznámení s distribucí aplikací pro systém iOS – App Store, iTunes Connect a systém certfikátů
5. návrh a implementace lokální hry
6. seznámení se s možnostmi síťové hry pomocí služby Game Center a rozšíření hry o možnost hrát prostřednictvím Internetu
7. návrh a implementace rozšíření lokální hry o možnost hrát proti virtuálnímu soupeři
8. testování hry

Možné rozšíření

V aplikaci se nabízí několik možných rozšíření.

Jedním z nich může být vylepšení taktiky virtuálního hráče. V tahových hrách (například v šachu) se pro výpočet tahů počítačem řízeného protihráče využívá algoritmus minimax. Tento algoritmus prochází rekurzivně všechny možné tahy a hledá dle zadané hodnotící funkce, který z tahů bude nejlepší. Jako funkce pro ohodnocení tahu by mohla být využita délka nejkratší cesty z vrcholu A do vrcholu B, tak jak je již nyní použita pro výpočet tahu virtuálního soupeře. Pak by se virtuální Shorter snažil tuto délku minimalizovat a Cutter naopak maximalizovat. Problémem algoritmu je jeho exponenciální časová složitost v^h, kde v je větvící faktor stromu možných tahů a h je hloubka zanoření rekurze.

Další možným rozšířením by mohlo být rozšíření algoritmu pro generováni herního grafu tak, aby generoval skutečné rovinné grafy a nevytvářel je ze sítě, jak je uvedeno v kapitole popisující implementaci. Také by tento generátor mohl kontrolovat *spravedlnost* vygenerovaného grafu, tedy má-li v grafu hráč Shorter stejnou šanci na vítězství jako hráč Cutter.

Seznam obrázků

Seznam použité literatury

[1] Steve Baranski, Jacob Gundersen, Matthijs Hollemans, Felipe Laso Marsetti, Cesare Rocchi, Marin Todorov, Ray Wenderlich. *iOS 5 By Tutorials: Volume 1*. 30. 4. 2012. ISBN 978-1475224269.

[2] Steve Baranski, Jacob Gundersen, Matthijs Hollemans, Felipe Laso Marsetti, Cesare Rocchi, Marin Todorov, Ray Wenderlich. *iOS 5 By Tutorials: Volume 2*. 30. 4. 2012. ISBN 978-1475245813.

[3] *Bridj-It*. `http://www.uksites4all.co.uk/development/bridjit/`

[4] Čada, Ondřej. *Cocoa, úvod do programování počítačů Apple*. ISBN 978-80-247-2778-3.

[5] Nancy Crisler, Patience Fisher, Gary Froelich. *Discrete Mathematics Through Applications*. 2. vydání. W. H. Freeman, 1999. ISBN 9780716736523.

[6] Dean, Sandy. *The Game of Bridg-It*. `http://scimath.unl.edu/MIM/files/MATExamFiles/Dean_MATpaper_FINAL.pdf`

[7] Dijkstra, Edsger Wybe. *A note on two problems in connexion with graphs*. Numerische Mathematik 1: 269–271, 1959. `http://www-m3.ma.tum.de/twiki/pub/MN0506/WebHome/dijkstra.pdf`

[8] Ford, Lester Randolph; Fulkerson, Delbert Ray *Maximal flow through a network*. Canadian Journal of Mathematics 8: 399–404, 1956.

[9] *Graph Game*. `http://www.coolmath-games.com/graphgame/`

[10] *Life For Cells*. `https://itunes.apple.com/us/app/life-for-cells/id641673105`

Přílohy

Práce obsahuje také přiložený CD-ROM obsahující zdrojové kódy aplikace a kompletní programátorskou dokumentaci.

	Adresář	Popis
1	**/documentation**	Programátorská dokumentace
2	**/application**	Zdrojový adresář aplikace
3	/application /Cells.xcodeproj	Soubor projektu pro vývojové prostředí Xcode
4	/application /Cells	Adresář se zdrojovými soubory aplikace