

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

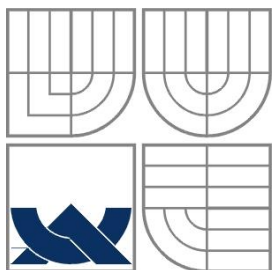
GUI PRO DESKOVOU HRU GHOST STORIES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

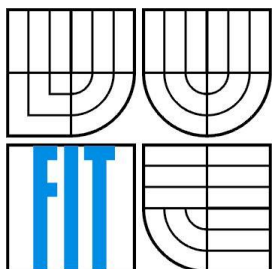
AUTOR PRÁCE
AUTHOR

TOMÁŠ SLÁDEK

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

GUI PRO DESKOVOU HRU GHOST STORIES

GHOST STORIES BOARD GAME GUI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ SLÁDEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. DAVID KUBÁT

BRNO 2013

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací počítačové verze kooperativní deskové hry Ghost Stories, která umožňuje lokální i síťovou hru. Text práce se postupně zabývá vybranou hrou, jejími pravidly, návrhem aplikace, detaily k implementaci a testování a zhodnocením dosažených výsledků. Výstupem práce je aplikace rozdělená na server a klienta, napsaná v jazyce C++ s využitím knihovny Qt verze 4.8 a její součásti, jazyka QML, pro tvorbu GUI.

Abstract

This bachelor's thesis describes the design and implementation of a PC version of a cooperative board game called Ghost Stories. The resulting application allows for both local and online playing. The text of this thesis details the chosen game and its rules, design and implementation of the application itself, testing and the evaluation of results. The output of this thesis is an application divided into server and client parts, written in C++ using the Qt 4.8 framework and its featured QML language for GUI creation.

Klíčová slova

GUI, desková hra, Qt framework, QML, síťová aplikace

Keywords

GUI, board game, Qt framework, QML, network application

Citace

SLÁDEK, T.: *GUI pro deskovou hru Ghost Stories*. Bakalářská práce, Brno, FIT VUT v Brně, 2013.

GUI pro deskovou hru Ghost Stories

Prohlášení

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

.....
Tomáš Sládek
15. května 2013

Poděkování

Chtěl bych poděkovat svému vedoucímu, Ing. Davidovi Kubátovi za odborné konzultace. Dále bych chtěl poděkovat všem testerům za ochotu a trpělivost zkoušet aplikaci během vývoje.

© Tomáš Sládek, 2013

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Obsah.....	1
1 Úvod.....	2
2 O hře Ghost Stories.....	3
2.1 Herní plán a komponenty.....	3
2.2 Pravidla hry.....	6
2.2.1 Fáze duchů (Yin).....	6
2.2.2 Fáze hráčů (Yang).....	7
3 Návrh aplikace.....	9
3.1 Grafické uživatelské rozhraní.....	9
3.2 Datová reprezentace hry.....	12
3.3 Síťová komunikace.....	14
4 Implementace.....	17
4.1 Qt framework.....	17
4.2 Klient – logická část.....	17
4.3 Klient - GUI.....	19
4.4 Síťová část - Server.....	21
5 Testování a zhodnocení výsledků.....	23
5.1 Další rozvoj.....	24
6 Závěr.....	25
A Finální návrh GUI.....	1
B Struktura uložené hry.....	1

1 Úvod

Deskové neboli stolní hry jsou součástí kultury velké části lidstva již dlouhou dobu, až do 20. století ale patřila mezi všeobecně známé a rozšířené jen hrstka her. Na konci 19. století přišel spolu s rozvojem tisku i rozvoj stolních her[3] a vzniklo množství úspěšných her – např. Snakes and ladders, Monopoly nebo Risk – které jsou dnes už považovány za „klasické“ a jejichž mechanismy byly nenáročné a často založené jen na náhodě. Velká vlna popularity deskových her, která trvá dodnes, vznikla v 90. letech 20. Století, odstartována tituly jako Osadníci z Katanu nebo Carcassonne. Tyto „moderní“ deskové hry se vyznačují obvykle složitějšími mechanismy, tematickým zasazením a kvalitním grafickým zpracováním.

Tato práce se zabývá návrhem a vytvořením aplikace, která umožňuje hrát jednu z těchto „moderních“ deskových her na PC. Vybraná hra se jmenuje Ghost Stories, vyšla v roce 2008 a řadí se do skupiny čistě kooperativních deskových her – tedy takových, kde všichni hráči spolupracují na dosažení určeného cíle, přičemž překonávají překážky určené herními mechanismy (mezi nejznámější hry v tomto žánru patří např. hra Pandemic). Elektronická verze této hry je v současnosti dostupná pouze pro mobilní přístroje s operačním systémem iOS.

V rámci práce vznikly dvě aplikace: klient, který zajišťuje lokální hru, zobrazuje stav hry a kontroluje dodržování pravidel, a server, který obstarává komunikaci mezi klienty při hře více hráčů na různých zařízeních. Obě aplikace jsou napsány v jazyce C++ s využitím knihoven Qt a Qt Quick pro tvorbu grafického uživatelského rozhraní.

Druhá kapitola tohoto dokumentu přibližuje vybranou hru po všech stránkách. Obsahuje představení hry, jejího tématu a základních principů, stručný výklad nejaktuálnější verze oficiálních pravidel, přehled herních komponent, karet i desek a několik poznámek k historii hry.

Třetí kapitola se zabývá návrhem aplikací a je rozdělena tematicky na tři části. První část představuje grafické uživatelské rozhraní – herní plochu, znázornění herních komponent na obrazovce a způsob ovládání hry i menu aplikace a jeho členění. Druhá část popisuje návrh datových struktur a logického uspořádání pro uchovávání dat o herních prvcích i stavu hry. Třetí část obsahuje návrh síťové komunikace – výběr komunikačního protokolu pro komunikaci v síti a návrh vlastního protokolu pro komunikaci klient-server.

Ve čtvrté části jsou v jednotlivých podkapitolách rozebrány aspekty implementace aplikací. První část přibližuje knihovnu Qt a jazyk QML a druhá část navazuje na první popisem implementace GUI v jazyce QML. Třetí část se věnuje klientské aplikaci a způsobu, jakým dohlíží na dodržování pravidel hry a manipulaci s informacemi o stavu hry. Čtvrtá část pokrývá serverovou aplikaci.

Poslední, pátá, kapitola představuje průběh testování aplikace několika dobrovolníky a zhodnocení zpětné vazby z testů, včetně výčtu opravených chyb a seznamu oblastí, ve kterých by šlo aplikaci rozšířit či vylepšit.

2 O hře Ghost Stories

Ghost Stories je kooperativní hra určená pro 1-4 hráče, kterou navrhl Antoine Bauza a v roce 2008 ji vydalo francouzské nakladatelství deskových her Repos Production. K dispozici je i v České Republice, a ačkoliv český překlad pravidel existuje pouze v neoficiální verzi, na samotných herních komponentech se text nenachází, takže hru lze hrát bez znalosti cizích jazyků.

Hra je tematicky zasazena do neurčené vesnice v Číně. V této vesnici je uložena urna s ostatky zlého vládce jménem Wu-Feng, který byl před mnoha lety poražen a odeslán do pekla. Během svého dlouhého pobytu v pekle se mu podařilo vypátrat umístění svých ostatků na zemi, a jelikož je potřebuje k tomu, aby se vrátil do světa živých, posílá do vesnice jednu vlnu svých služebníků za druhou. Na obranu vesnice se postaví čtveřice mnichů a roli těchto mnichů se ujímají hráči. Jejich úkolem je ubránit vesnici před nájezdy duchů a nakonec porazit jednu či více inkarnací Wu-Fenga. Pokud přitom všichni ztratí životní energii, vesnice utrpí příliš velké škody nebo se jim nepodaří porazit poslední inkarnaci Wu-Fenga včas, končí hra neúspěchem.

Ghost Stories se řadí mezi nejobtížnější deskové hry a přestože pravidla nabízí čtyři úrovně obtížnosti, není snadné vyhrát ani na tu nejjednodušší. Prohra je zde častým jevem, ale poměřit úspěšnost pokusu lze i v takovém případě alespoň spočítáním výsledného skóre podle tabulky uvedené v pravidlech.

2.1 Herní plán a komponenty

Herní plán není velký pevný kus, jak je u takových her obvykle zvykem, ale je modulární – skládá se z jednotlivých menších částí, jejichž uspořádání se náhodně mění s každou novou hrou. Plán se skládá ze dvou druhů desek - vesnice samotné (vnitřní čtverec) a přístupových cest k ní (obdélníkové desky po stranách vesnice).



Obrázek 2-1: Sestavený herní plán

Vesnice se skládá z devíti čtvercových polí. Každé jednotlivé pole je unikátní a představuje jednu z devíti lokací ve vesnici, mezi nimiž se pohybují figurky mnichů a jejichž služby mohou hráči v průběhu hry využívat. Na každém z těchto polí je tematický obrázek a v pravém dolním rohu je symboly naznačena funkce dané budovy. Na začátku hry se tyto budovy náhodně poskládají do čtverce 3x3, čímž vytvoří ústřední vesnici, po které se hráči pohybují. V průběhu hry se na poli vesnici mohou vyskytovat figurky hráčů a v případě dvou budov i jiné žetony nebo zvláštní figurky. Pole jsou oboustranná, přičemž jedna strana zobrazuje aktivní lokaci a druhá neaktivní, tedy lokaci, která byla zničena duchy a hráči už nemohou využít jejích služeb.



Obrázek 2-2: Obě strany jednoho pole vesnice

Přístupové cesty k vesnici slouží zároveň jako hlavní desky jednotlivých hráčů, jejich barvy odpovídají barvám figurek mnichů a hráči by se měli ke hře posadit tak, aby měli desku své barvy před sebou. Kromě barvy se od sebe jednotlivé desky liší ikonami v levém dolním rohu, které vyznačují speciální schopnost mnicha, jemuž deska odpovídá. Každá deska má dvě strany, na nichž jsou rozdílné schopnosti (tedy každý mnich má výběr ze dvou zvláštních schopností). Strana desky se určuje náhodně před začátkem hry, stejně jako přidělení barev hráčům. Společným znakem všech čtyř desek jsou pak tři obdélníková místa, která zabírají většinu desky. Na tato místa se pokládají karty příchozích duchů, s kterými hráči následně bojují. Kromě karty ducha se na desce může nacházet i figurka ducha, která se posouvá mezi umístěním na kartě a dvěma kruhovými políčky nad každým místem pro kartu.



Obrázek 2-3: Přístupová cesta k vesnici

Duchové a inkarnace Wu-Fenga jsou reprezentováni kartami, kterých je celkem 65. Karty duchů (55) mají modrou rubovou stranu, kdežto karty inkarnací Wu-Fenga jsou z rubové strany červené. Všechny karty sdílejí stejnou strukturu: většinu plochy zabírá obrázek ducha, v levém horním rohu se nachází jeho odolnost vyznačená barevnými kruhy (v případě karet s červeným rubem navíc ještě ikonka označující, že se jedná o inkarnaci Wu-Fenga), v pravém horním rohu je název (nepodstatný a jediný text ve hře) a dolní část karty obsahuje symboly, které určují schopnosti daného ducha. Tyto schopnosti jsou rozděleny do tří částí (a ne vždy jsou přítomny všechny): vlevo akce při příchodu ducha, uprostřed akce prováděná každé kolo a vpravo akce při odstranění ducha.



Obrázek 2-4: dva duchové a jedna inkarnace Wu-Fenga

Mezi další herní komponenty patří:

- 4 figurky mnichů.
- 8 figurek duchů.
- 2 figurky Buddhy.
- 4 šestihranné kostky s barvami místo čísel, jedna v šedém provedení a tři bílé.
- 1 šestihranná kostka se speciálními symboly místo čísel, tzv. „Curse die“ (kostka prokletí).
- 20 kruhových žetonů v pěti různých barvách (tzv. tao žetony).
- 20 osmiúhelníkových žetonů s nápisem Qi.
- 4 žetony se symbolem Yin-Yang v barvách hráčů.
- Jeden velký půlkruhový žeton označující, že nelze používat tao žetony.
- Jeden malý oválný žeton (váže se ke zvláštní schopnosti jednoho z hráčů).
- 4 velké oválné žetony v barvách hráčů k označení neaktivní schopnosti.
- 3 menší oválné žetony.



Obrázek 2-5: Herní komponenty

2.2 Pravidla hry

Účelem této sekce je přiblížit čtenáři nejdůležitější pravidla a strukturu hry, nikoliv podat kompletní výklad pravidel, mnohé méně podstatné věci jsou zde vynechány – před hraním hry je nutné přečíst si originální pravidla[1].

Hra se skládá z jednotlivých kol hráčů, které po sobě následují po směru hodinových ručiček podle rozsazení hráčů. Hráči se takto střídají v odehrávání kol, dokud není dosaženo jedné z podmínek konce hry. Jedno kolo se dělí na dvě fáze, v terminologii hry „Yin“ a „Yang“, které odpovídají akcím duchů (Yin) a mnicha, jehož hráč ovládá (Yang). Před hrou se hráčům náhodně přidělí mniši a jejich schopnosti, vytvoří se vesnice, hráči dostanou příslušné komponenty podle zvolené obtížnosti a vytvoří se balíček duchů. Tento balíček sestává z náhodně zamíchaných karet duchů a jedné až čtyř karet inkarnací Wu-Fenga, vložených ke konci balíčku. Umístění inkarnací, jejich počet a celkový počet karet v balíčku duchů se řídí zvolenou obtížností a počtem hráčů.

Vítězství dosáhnou hráči tehdy, pokud odstraní ze hry poslední z inkarnací Wu-Fenga dřív, než dojdou ke konci balíčku duchů, a pokud do té doby všichni mniši nezemřou nebo nedopustí zničení tří lokací ve vesnici. Při ztrátě všech žetonů Qi mnich umírá, jeho figurka se přesouvá na pole vesnice „hřbitov“, ztrácí všechny žetony, které měl ve svém vlastnictví a dokud ho spoluhráči neoživí, provádí hráč ve svém kole pouze fázi duchů.

2.2.1 Fáze duchů (Yin)

V této fázi hráč obsluhuje duchy na desce své barvy. Pokud se na desce nachází jedna či více karet duchů a tyto duchové mají neprázdné prostřední pole schopností, vykoná hráč akci odpovídající symbolu na kartě ducha. Aktivní schopnosti duchů jsou ve hře dvě, jedná se o ikonky figurky ducha a černé kostky. V terminologii hry se tyto duchové nazývají „*haunter*“ a „*tormentor*“.

Duchům typu „*haunter*“ přísluší černé figurky duchů, které se pohybují po desce a ohrožují pole vesnice před sebou. Figurky rotují po třech umístěních – na kartě a dvou políčkách na desce přímo nad kartou ducha. Pokud přijde na řadu takovýto duch a jeho figurka dosud není ve hře, hráč ji umístí na kartu ducha. Pakliže už se figurka ve hře nachází, posune ji hráč o jedno políčko dál na desce. Při dosažení třetího pole duch ničí nejbližší aktivní lokaci vesnice před sebou – hráč tuto lokaci otočí tak, aby byla položená zničenou stranou nahoru. Pokud figurka ducha na posledním poli již stojí, hráč ji vrátí zpět na příslušnou kartu.

Za každého ducha typu „*tormentor*“ musí hráč hodit černou kostkou prokletí a poté vykonat, co určí hod. Následující popis kostky odkazuje na strany zobrazené na obrázku 2-6. Dvě strany kostky jsou prázdné (a) a při jejich hodu se nic neděje. Strana s figurkou a vesnicí (b) znamená zničení nejbližší aktivní lokace ve vesnici před duchem, jehož akce je vykonávána. Symbol karty (c) značí příchod nového ducha – hráč vezme vrchní kartu z balíčku duchů a umístí ji do hry podle standardních pravidel umístování duchů. Padne-li strana s přeškrtnutými žetony (d), ztrácí hráč všechno tao žetony, které v danou chvíli vlastní. Poslední strana se symbolem Qi způsobí, že hráč ztrácí jeden bod životní energie (pokud tímto způsobem přijde hráč o poslední žeton Qi, stává se jeho mnich mrtvým a jeho kolo končí po vykonání poslední akce ducha).



Obrázek 2-6: Černá kostka

Po vykonání všech akcí duchů hráč zkontroluje, zda má na desce alespoň jedno volné místo. Pokud jsou všechny tři místa zaplněné kartami, hráč ztrácí jeden žeton Qi a fáze duchů tímto končí. V případě že alespoň jedno místo volné je, vytáhne hráč vrchní kartu z balíčku duchů a umístí ho do hry. Karta ducha se umístí na tu desku, která odpovídá barvě ducha (černý duch přísluší desce hráče, který ho vytáhl). Pokud je tato deska plná, lze ducha umístit na jakékoliv jiné volné místo na kterékoliv desce. Po umístění ducha do hry je nutné vykonat akci, které je uvedena v levém dolním rohu na kartě ducha. Těchto akcí je více druhů a v některých případech mohou být i dvě najednou.



Obrázek 2-7: Akce při příchodu ducha do hry

Symbyly podle obrázku 2-7 zleva:

- Příchod nového ducha do hry podle běžných pravidel.
- Dokud je tento duch ve hře, hráč, na jehož desce leží, ztrácí svou schopnost.
- Figurka ducha je okamžitě umístěna na kartu.
- Dokud je tento duch ve hře, mají hráči k boji s duchy o jednu kostku méně.
- Hráč, k němuž duch přišel na desku, ztrácí jeden žeton Qi.
- Dokud je tento duch ve hře, nelze k boji s duchy používat tao žetony.
- Zničí první aktivní lokaci před místem, kam byl duch umístěn.

2.2.2 Fáze hráčů (Yang)

V této fázi hráč ovládá svého mnicha. Fáze sestává ze tří základních kroků, které následují po sobě. Prvním je pohyb, kdy hráč může posunout svého mnicha do sousední lokace ve vesnici. Tento pohyb může být i diagonální, avšak vždy pouze o jedno pole. Ve druhém kroku se hráč rozhodne, zda chce využít lokace, na které jeho figurka stojí, nebo bojovat s duchem. Bojovat lze vždy jen s duchem, před jehož kartou figurka mnicha stojí (v rohu vesnice je tedy možné svést boj se dvěma duchy najednou). Využití lokace se řídí svými pravidly pro každou lokaci zvlášť a může jít např. o získání jednoho či dvou žetonů tao, posunutí karty ducha z jednoho místa na jiné nebo obnovení zničené lokace.

Boj s duchy probíhá tak, že hráč hodí kostkami, které má v danou chvíli k dispozici (obvykle všechny tři bílé kostky) a porovná padlé barvy s odolností ducha vyznačenou na jeho kartě. Bílá barva na kostce se počítá podle potřeby jako kterákoliv jiná barva. Pokud hráč hodí stejně nebo více, než je odolnost ducha, odstraní hráč ducha ze hry. Pokud hráči k poražení ducha scházejí barvy, může ke svému hodu přidat tao žetony ze své zásoby nebo ze zásoby spoluhráče, který stojí na stejné lokaci. Tyto přidané žetony hráči po úspěšném odstranění ducha ztrácí. Odstraněný duch odchází ze hry a spolu s ním končí i případné efekty, které měl na hru (odebrání kostky, znemožnění schopnost hráče apod.). Než je ale duch zcela odebrán ze hry, musí hráč ještě provést akci, která přísluší odchodu ducha ze hry – značí ji symbol v pravém dolním rohu karty ducha. Může se jednat o pozitivní efekt – zisk tao žetonů, Qi žetonů nebo žetonu Yin-Yang – nebo negativní efekt v podobě hodu kostkou prokletí (účinky se aplikují na hráče, který ducha odstranil).

Posledním krokem ve fázi hráče je umístění figurky Buddha nebo zisk žetonů neutrálních schopností. Pokud má hráč u sebe figurku Buddha, kterou získal v jednom z předchozích kol, a jeho

figurka stojí vedle neobsazeného místa pro kartu ducha, může se rozhodnout položit figurku Buddha na toto místo. Duch, který by byl později umístěn na figurku Buddha, bude bez provedení jeho akcí odstraněn ze hry a figurka se vrací zpět do zásoby. Pokud hráč skončí své kolo uprostřed vesnice, získává veškeré žetony neutrálních schopností, které na dané lokaci leží (tato část se týká jen hry v méně než čtyřech hráčích).

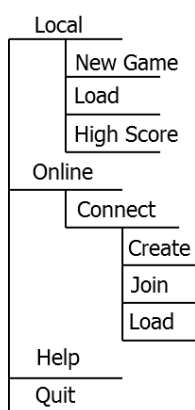
Mezi jednotlivými kroky ve svém tahu může hráč zahodit žeton Yin-Yang a obnovit libovolnou zničenou lokaci nebo využít kterékoliv lokace navíc ke své běžné akci.

3 Návrh aplikace

Návrh aplikace je rozčleněn do tří vzájemně nezávislých částí – síťové komunikace, GUI a datové reprezentace hry a jejího průběhu. Rozdělení je vhodné z praktických důvodů, jelikož takový návrh umožňuje měnit jednu z částí, aniž by to výrazně zasáhlo ostatní oblasti aplikace.

3.1 Grafické uživatelské rozhraní

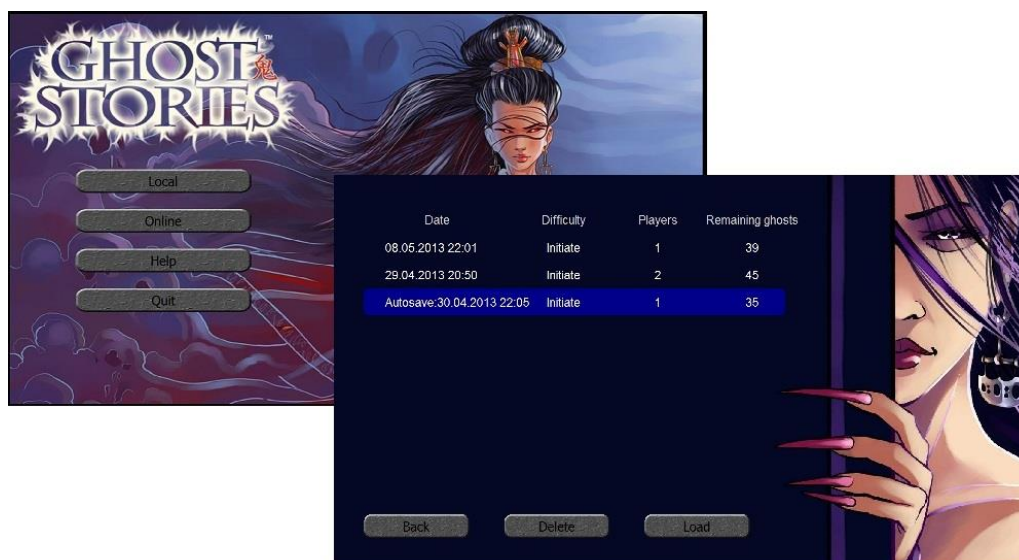
Následující text se týká pouze klientské aplikace, jelikož server je pouze konzolová aplikace a grafické uživatelské rozhraní postrádá. Po stránce grafického návrhu lze návrh uživatelského rozhraní rozdělit na dvě části – menu hry a samotná herní obrazovka.



Obrázek 3-2: Struktura menu

Menu sestává z několika různých sekcí uspořádaných do stromové struktury, kde každá sekce je na samostatné obrazovce, jak je u menu ve hrách typické. Hlavní menu obsahuje odkazy na možnosti lokální hry, síťové hry, nápovědy a ukončení aplikace. V možnosti lokální hry najde uživatel odkazy na založení nové hry (obrazovka s možnostmi nastavení obtížnosti a počtu hráčů), načtení uložené hry (obrazovka s nabídkou uložených her a možností jejich odstranění) a tabulku nejlepších výsledků. Odkaz na síťovou hru vede na okno se seznamem serverů, kde má uživatel možnost přidat nový server, odstranit stávající ze seznamu, nebo se na vybraný sever přihlásit. Po přihlášení se zobrazí seznam běžících her na daném serveru a uživatel se může k jedné z nich připojit, založit vlastní novou hru (obdobná obrazovka jako u vytvoření lokální hry, navíc však možnost pojmenovat hru), nebo načíst dříve rozehranou

hru, která je na serveru uložená (v tomto případě uživatel uložené hry mazat nemůže). V menu nápovědy se nachází odkazy na PDF soubor s pravidly hry a další PDF soubor vysvětlující ovládání grafického uživatelského rozhraní aplikace.



Obrázek 3-1: Hlavní menu a obrazovka s výběrem uložených her

Název případu užití	Vytvoření nové síťové hry
Identifikátor	UC1
Stručný popis	Uživatel vytvoří novou hru
Primární aktér	Uživatel
Sekundární aktér	Klientská aplikace
Vstupní podmínky	Klientská aplikace je správně spuštěna a vybraný server je aktivní
Základní scénář	<ol style="list-style-type: none"> 1. Vybrat v hlavním menu položku „Online“. 2. Vybrat server, vyplnit přihlašovací údaje a stisknout „Connect“. <ol style="list-style-type: none"> a. Server zkontroluje přihlašovací údaje, pokud jsou špatně, nutno zadat znovu a správně. b. Údaje jsou správné, server uživatele přihlásí 3. Stisknout tlačítko „Create“ 4. Vyplnit jméno a nastavit obtížnost a počet hráčů. 5. Server založí novou hru a klient ji živiteli zobrazí
Následné podmínky	Server založil novou hru a čeká na připojení dalších hráčů
Alternativní toky	Uživatel si tvorbu nové hry rozmyslel Uživateli se nepodařilo se přihlásit k serveru

Tabulka 3-1: Příklad užití pro vytvoření nové síťové hry

Z hlediska návrhu uživatelského rozhraní je obrazovka se samotnou hrou zajímavější a složitější než menu. Aplikace byla od první myšlenky navrhována v širokoúhlém formátu, a to hned ze dvou důvodů - širokoúhlý formát je dnes mezi zobrazovacími zařízeními u počítačů nejběžnější a jelikož zpracovávaná desková hra má půdorys čtverce, zbude po stranách herní desky dostatek místa na ovládací prvky a zobrazení potřebných informací o hře. Návrh tedy od počátku počítal i s umístěním herní desky uprostřed obrazovky s potřebnými informacemi po obou stranách, jelikož pozornost uživatele se bude většinu času obracet právě na samotnou herní desku. Hlavní myšlenka návrhu byla vytvořit takové rozhraní, které bude obsahovat všechny potřebné informace o hře, aby uživatel nemusel přepínat obrazovky či otvírat a zavírat různá okna.



Obrázek 3-3: První hrubá varianta návrhu

První varianta návrhu (viz Obrázek 3-3) byla nakonec zamítnuta úplně pro příliš mnoho nedostatků. Nejzávažnější z nich je skutečnost, že z obrazovky není dostatečně patrné, který hráč je právě na tahu (v lokální hře) či kterého hráče představuje uživatel (v síťové hře). Mezi další nedostatky patří

nedomyšlený systém zobrazení boje v pravém horním rohu (je možné bojovat i se dvěma duchy najednou, a to by při tomto systému nešlo dobře znázornit) či prázdné místo v levém dolním rohu, které působí v jinak informacemi zahlcené obrazovce nepatříčně. Rozhodnutí uvádět životní energii hráčů nad herní deskou bylo rovněž špatné, a sice proto, že vyústilo ve zmenšení samotné herní desky – přičemž herní deska obsahuje velké množství detailů a na menších zobrazovacích zařízeních by uživatel mohl mít potíže tyto detaily rozeznávat.

Finální návrh tyto problémy řeší a navíc obsahuje i další prvky. Samotný návrh hlavní obrazovky je v této práci kvůli své velikosti a množství detailů umístěn mezi přílohami, konkrétně se jedná o přílohu A. Základní koncept zůstává stejný – prostřední část obrazovky vyplňuje samotná herní deska, která ale zabírá celou výšku obrazovky a je tak největší, jaká může při daném rozvržení být. Zobrazení herní desky a všech herních komponent, které se na ni nacházejí / mohou nacházet je zcela věrné deskové předloze. Herní plocha je postavena tak, aby deska aktivního hráče byla vždy ve spodní části obrazovky, hráč tedy vidí hru tak, jak by ji viděl ze své pozice u stolu. Vlevo od desky aktivního hráče je znázorněn balíček duchů a číslo ukazující počet karet zbývajících v balíčku. Vpravo se nacházejí tlačítka pro jednotlivé akce, které může hráč ve svém tahu vykonat. V pravém horním rohu herní desky se zobrazuje žeton, který signalizuje neaktivní tao žetony. Levý horní roh je vyhrazen pro zobrazení různých událostí v průběhu hry, např. hodů kostkou prokletí, zisku tao žetonů při využití lokace ve vesnici nebo výběru odměny po poražení ducha.

Levý i pravý okraj obrazovky mimo herní desku jsou vertikálně rozděleny a vytváří tak čtyři oblasti, z nichž každá obsahuje jiný informační či ovládací prvek. V pravém horním rohu je umístěn prohlížeč herních prvků, který zobrazuje vybrané pole vesnice nebo kartu ducha ve zvětšené podobě oproti herní desce. U polí vesnice se navíc zobrazí jejich popis převzatý z pravidel hry. Pravý dolní roh obsahuje přehled žetonů, které má k dispozici aktivní hráč. Maximální počet zobrazitelných žetonů Qi je sice jen 10 oproti celkovému počtu 20 žetonů dostupných ve hře, ale pravděpodobnost, že by jeden hráč měl v průběhu hry více než 10 žetonů Qi je s ohledem na herní mechaniky extrémně nízká. Řada žetonů pod žetony Qi zároveň slouží jako ovládací prvky – Buddhy a žeton Mantry lze ve



Obrázek 3-4 Obrazovka boje s duchy

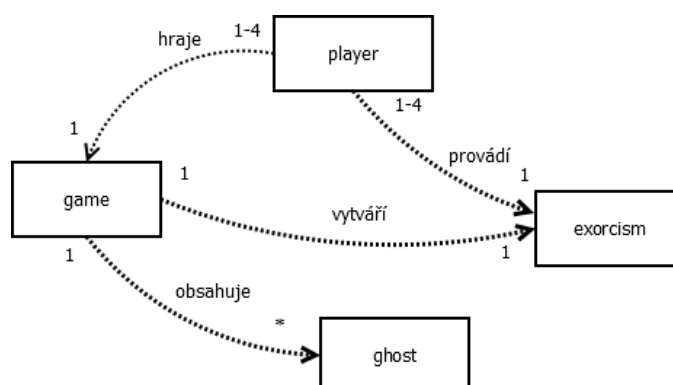
správnou chvílí myší přetáhnout na příslušná místa a kliknutí na žeton Yin-Yang nebo žeton neutrálních schopností zobrazí příslušnou nabídku. Spodní segment s tao žetony je čistě informační.

Levý dolní roh obsahuje okno pro textovou komunikaci, které je určeno primárně pro síťovou hru, funguje ale i při lokální hře (v takovém případě se vložený text samozřejmě nikam neodešle, jen se zobrazí v okně). V levém horním rohu vidí hráč všechny zdroje ostatních tří hráčů. Původně úmysl byl vyhnout se v návrhu číslům, aby okno na první pohled připomínalo stolní hru, ale tento přístup se v případě zobrazení zdrojů ostatních hráčů ukázal být značně chaotickým a nepřehledným, ba dokonce nerealizovatelným – dostupných žetonů je ve hře příliš mnoho, než aby se daly zobrazit skutečně jako žetony. Pro zvýšení přehlednosti jsou nulová čísla označující počet tao žetonů zvýrazněna barvou odpovídajícího žetonu. Nulový počet žetonů je při hře u hráčů častý, nula je tedy vždy zobrazena v černé barvě, aby splývala s pozadím a hráč neměl neustále na očích tabulku plnou nul.

Jediným prvkem hry, který se mi nepovedlo vložit do návrhu v rámci jedné obrazovky, jsou souboje s duchy. Jelikož je tato akce ve hře prováděna vždy kompletně od začátku do konce bez přerušení, bylo rozhodnuto o zakrytí herní desky po čas boje plochou navrženou speciálně pro boj s duchy (Obrázek 3-4). V prostřední části obrazovky jsou hráči k dispozici všechny zdroje, které může v tomto boji využít: hozené kostky, vlastní tao žetony i žetony spoluhráčů stojících ve stejné lokaci. Nahoře i dole jsou duchové, se kterými hráč právě bojuje (pokud je duch jen jeden, je vždy v horní třetině). Duchové mají vedle sebe vyznačenou odolnost – tuto odolnost se hráč snaží vyrovnat s pomocí dostupných surovin. Kliknutím na kartu ducha jej hráč vybere (kolem vybraného ducha se zobrazí bílý okraj) a následným kliknutím na platný zdroj tento zdroj přiřadí vybranému duchovi (červený žeton na červeného ducha, žlutá na jedné z kostek na žlutého ducha, apod. – bílou kostku lze přiřadit kterémukoliv duchovi). Pokud je hráč hotov, ať už proto, že přiřadil dostatečný počet zdrojů, nebo už nemá co přiřazovat, klikne na ukončovací tlačítko, hra vyhodnotí výsledek boje a zobrazí herní desku s případnými důsledky boje.

3.2 Datová reprezentace hry

Vzhledem k povaze deskové hry jako uzavřeného systému o konečném počtu komponent s jasně definovanými vlastnostmi a vztahy mezi nimi není složité převést jednotlivé prvky hry do objektové podoby a navrhnout jejich reprezentaci v programu. Důležité je zvolit měřítko tak, aby převod nebyl příliš obecný ani podrobný. Návrh systému hry byl rozdělen na objekty hry, hráče, ducha a objektu pro událost boje s duchem. Vzhledem k jednoduché povaze vztahů mezi herními komponenty a jejich nízkému množství nebylo nutné navrhovat uspořádání než jednoduché objekty bez dědičnosti.



Obrázek 3-5: Diagram tříd ve hře

Třída `player` reprezentuje hráče a obsahuje veškeré informace, které se hráče týkají – jméno, barvu, počet vlastních žetonů, umístění figurky, druh zvláštní schopnosti a stav desky včetně duchů, kteří se na ní nacházejí. Třída `ghost` pak představuje ducha a obsahuje vlastnosti, které ducha definují – jméno, pořadové číslo, odolnost, sadu schopností a příznak, zda se jedná o obyčejného ducha, nebo inkarnaci Wu-Fenga. Instance třídy `exorcism` je jen jedna a její data se resetují před každým bojem. Z instancí tříd `ghost`, `player` a `game` získává všechna potřebná data, která mohou ovlivnit výsledek boje, a po provedení hráčových akcí umožní instanci třídy `game` zjistit, jak boj hráčů s duchy dopadl.

Třída `game` je aplikací návrhového vzor singleton neboli jedináček, zastřešuje v rámci aplikace herní prostředí a je tedy žádoucí, aby po celou dobu běhu aplikace existovala vždy pouze jedna instance této třídy. Kromě instancí tříd hráčů a duchů tato třída přímo uchovává většinu informací, které se netýkají duchů či hráčů – např. uspořádání lokací ve vesnici, nastavení obtížnosti, stav hozených kostek... kromě toho také obsahuje stavové proměnné, pomocí kterých lze přesně určit, v jakém momentě se hra právě nachází (jedná se o informace, které při reálném hraní deskové hry nejsou reprezentovány žádnými komponenty, ale hráči je mají v paměti).

Některé informace je nutné skladovat i po ukončení aplikace, a jelikož jsou proměnlivého charakteru, nemohou být vloženy přímo do kódu. Je tedy nutné vybrat způsob jejich uchování mezi jednotlivými spuštěními aplikace. Pro tuto práci bylo vybráno prosté textové ukládání do formátu Extensible Markup Language, zkr. XML. XML je jazyk určený pro serializaci dat za účelem jejich publikování či výměny mezi aplikacemi a zabývá se pouze strukturou obsahu, nikoliv jeho stylizací – což je ideální pro potřeby uchovávání dat generovaných počítačem a znovu i čtených počítačem. Jeho textová povaha a prostá struktura ale zároveň umožňuje snadný lidský zásah. Knihovny Qt, jichž tato práce využívá, navíc poskytují dobrou podporu tohoto formátu a obsahují DOM parser pro XML (Document Object Model) – tedy takový parser, který načte dokument a v paměti z něj vytvoří strom dat, což se pro data hry hodí.

Aplikace během svého životního cyklu pracuje hned s několika XML reprezentacemi dat: seznamem duchů, nejlepším skóre a uložené hry. Seznam duchů je převzat přímo z pravidel hry, a jelikož aplikace není explicitně navržena s podporou vlastních duchů a v návodu ke hře není v tomto směru ani zmínka, nepředpokládám, že by uživatel chtěl jejich seznam modifikovat. Přesto je teoreticky možné vytvořit si vlastního ducha kombinací vlastností a schopností duchů stávajících či modifikovat seznam předdefinovaných duchů. Tato možnost je zde spíše z hlediska budoucího rozvoje hry, kdy by mohlo být podporované vlastní duchy vytvářet. Seznam nejlepšího skóre se načítá při startu aplikace a aktualizuje se na konci každé hry, kdy se výsledné skóre dostalo mezi 10 nejlepších.

Posledním typem souboru je uložená hra, která je z těchto tří zdaleka nejobemnější, jelikož obsahuje všechny informace o hře, nutné k jejímu obnovení (ukázka struktury uložené hry viz příloha B). Soubory s uloženými hrami se nacházejí v samostatné složce „saves“, kterou si aplikace v případě potřeby sama vytváří. Tento způsob ukládání informací o hře znamená, že uživatelský zásah do uložených dat je nejen nedetekovatelný, ale i velmi jednoduchý – což ale považuji spíše za výhodu. Umožňuje to hráči experimentovat s různým nastavením uložených her a vytvářet si situace, které by při normálním hraní byly jen obtížně dosažitelné, zvláště bereme-li v potaz množství náhody obsažené v generování nové hry. Ruku v ruce s modifikací uložených her se samozřejmě objevuje otázka možného podvádění. Tento problém je u kooperativní hry tohoto stylu, kdy hráči stojí jen proti hře samotné, prakticky neexistující, zanedbatelný, neboť jakékoliv podvádění zde postrádá smysl.

```

<ghost>
  <id>22</id>
  <name>Fury of Depth</name>
  <res>
    <b>4</b>
  </res>
  <turnAction>A_TORMENT</turnAction>
  <endAction>A_GAINQIYI</endAction>
</ghost>
<ghost>
  <id>23</id>
  <name>Creeping Ooze</name>
  <res>
    <g>1</g>
  </res>
  <turnAction>A_HAUNTER</turnAction>
</ghost>

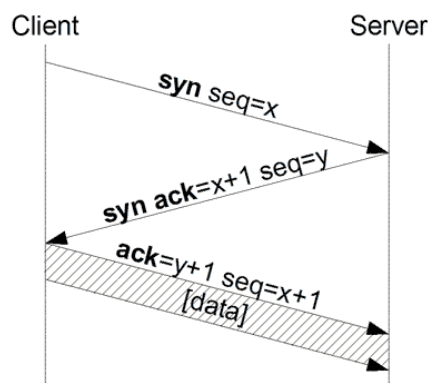
```

Kód 3-1: Ukázka struktury XML souboru s duchy

3.3 Síťová komunikace

Možnost síťové hry je bezpochyby jeden z hlavních důvodů, proč se pokoušet o převod deskové hry na PC. Způsobů jak zpracovat spojení aplikací po síti je mnoho a každý přístup má své pro a proti. Při návrhu takové aplikace je nutno rozhodnout o několika aspektech síťové komunikace: transportním protokolu, typu sítě z hlediska komunikace aplikací a samotném aplikačním protokolu.

V oblasti transportního protokolu je možností více, nejtypičtějším zástupci jsou však protokoly Transmission Control Protocol (TCP) a User Datagram Protocol (UDP). Protokol UDP je jednoduchý protokol, který nezaručuje doručení paketů ani jejich správné pořadí. UDP nepoužívá potvrzování o doručení a aplikace si tak nemůže být jistá, že příjemce data skutečně obdržel. Díky tomu ale má protokol UDP malou režii, což ho činí ideálním pro situace, kdy jde více než o kvalitu spojení o rychlost, a na občasnou ztrátu paketu či jejich zamíchání se nehledí (streamování, VoIP). Protokol TCP je opakem UDP – upřednostňuje spolehlivost spojení před rychlostí. TCP je spojově orientovaný protokol a každé spojení začíná ustanovením spojení s pomocí tzv. „three-way handshake“ mechanismu, kdy si komunikující strany navzájem vymění úvodní pakety a potvrzení o jejich doručení, poté následuje skutečná datová komunikace. TCP zaručuje doručení paketů beze ztráty a ve správném pořadí za cenu nižší rychlosti přenosu.



Obrázek 3-6: Ustavení spojení v protokolu TCP, tzv. Three-way handshake [2]

Z výše popsaného je zřejmé, že vhodnou volbou pro deskovou hru je transportní protokol TCP – u tahové hry záleží mnohem víc na spolehlivosti spojení, než jeho rychlost. Ostatně pro účely tahových her je protokol TCP využíván takřka bezvýhradně a dnešní rychlý internet umožňuje využívat protokol TCP i v náročnějších hrách, např. World of Warcraft, Guild Wars nebo StarCraft II. Po transportním protokolu je nutné vybrat architekturu komunikace mezi aplikacemi. Možnosti jsou dvě: architektura klient-server nebo klient-klient. Uspořádání typu klient-klient sice nabízí robustnější způsob síťové hry (v případě výpadku se aplikace připojí k jinému klientovi) a méně zatěžuje provoz sítě, ale decentralizace, kterou tento systém přináší, je pro tuto aplikaci nežádoucí. Zvolená architektura je tedy klasický typ server-klient, kdy klientovi stačí znát adresu serveru a sever prostředkovává komunikaci s ostatními klienty.

Poslední v řadě rozhodnutí před navržením vlastního aplikačního protokolu je rozhodnutí o způsobu funkce server – může být buď iterativní, nebo rekurzivní. Iterativní typ zpracovává požadavky postupně podle času příchodu, a zatímco zpracovává jeden, ostatní čekají. Naproti tomu rekurzivní server se za běhu typicky větví do více vláken, aby mohl obsluhovat všechny klienty zároveň, a každému klientu je přiřazen deskriptor (ukazatel na schránku pro přijímání a odesílání dat), sloužící pro komunikaci s daným klientem. Režim serveru souvisí do jisté míry se zvoleným transportním protokolem a pro zvolený protokol TCP se lépe hodí server rekurzivního typu.

Aplikační protokol, navržený pro tento projekt, je textový. S textovým formátem se snadno pracuje a je člověku rychle srozumitelný. Skládá se vždy z jednoslovného příkazu psaného velkými písmeny a případných argumentů, oddělených podtržítkem. Díky efektivnímu návrhu předávání informací o herním stavu je počet příkazů aplikačního protokolu minimální a server je zároveň téměř kompletně odstíněn od samotné herní logiky, což potenciálně umožňuje jeho využití pro jiné hry. Navržený aplikační protokol obsahuje následující příkazy:

- | | |
|-------------------------------------|-----------------------------|
| • LOGIN_[username]_[pass] | • LOGINOK |
| • GETGAMELIST | • LOGGEDELSEWHERE |
| • GETSAVELIST | • BADPASS |
| • STARTGAME_[name]_[players]_[diff] | • SAVELIST_[data] |
| • LOADGAME_[gameID] | • GAMELIST_[data] |
| • JOINGAME_[gameID] | • GAMESTART_[gameID] |
| • UPDATEALL_[gameID]_[data] | • LOADED_[data] |
| • CHAT_[gameID]_[data] | • GAMEFULL |
| • UPDATE_[gameID]_[data] | • CHAT_[data] |
| • PING | • DISCONNECT_[pos] |
| | • JOINED_[pos]_[data] |
| | • PING |
| | • UPDATE_[data] |
| | • PLAYERJOINED_[pos]_[name] |

V levém sloupci jsou příkazy, které posílá klient, v pravém pak příkazy, které posílá server. Komunikace vždy začíná ze strany klienta příkazem LOGIN. Server ověří totožnost přihlašovaného uživatele. Nyní nastanou tři případy: přihlašovací údaje nesouhlasí (server odpoví BADPASS), uživatel je již přihlášen z jiného místa (server odpoví LOGGEDELSEWHERE) nebo přihlašování v pořádku projde (server odpoví LOGINOK) a komunikace může pokračovat dále. Bez této počáteční autorizace nebude server reagovat na příkazy z daného zdroje. Po přihlášení klient vyžaduje seznam probíhajících her zprávou GETGAMELIST, na což mu server odpoví GAMELIST s připojeným

seznamem probíhajících her. Pokud chce hráč načíst uloženou hru, proběhne totéž s příkazy GETSAVELIST a SAVELIST. O vytvoření nové hry žádá klient příkazem STARTGAME s argumenty v podobě jména hry, počtu hráčů a obtížnosti. Server odpovídá příkazem GAMESTART, ke kterému připojí vygenerovaný identifikační řetězec pro nově vytvořenou hru. Pokud se uživatel chce připojit k probíhající hře, pošle klient žádost o připojení ke hře JOINGAME spolu s identifikačním řetězcem žádané hry. Pokud je hra již plná, server odpoví GAMEFULL, jinak uživatele do hry přidá a odpoví JOINED s argumenty pozice, kterou hráč ve hře zaujímá, a samotnými herními daty. Příkazy UPDATE a UPDATEALL slouží k vyměňování dat herního stavu, přičemž UPDATE je určen k aktualizaci pouze právě změněných informací a UPDATEALL slouží k přenosu kompletního stavu hry. Příkaz CHAT se na obou stranách používá k přenosu zpráv, které si uživatelé během hry spolu vyměňují. Server navíc odesílá nepárové zprávy DISCONNECT a PLAYERJOINED. První zmíněná oznamuje klientovi, že jeden z uživatelů připojených k probíhající hře se odpojil a je tedy nutné hru pozastavit. Druhý příkaz naopak oznamuje připojení dalšího uživatele ke hře.

Zvláštní zmínku zaslouží příkazy PING. S těmito příkazy původní návrh nepočítal, ale během implementace se ukázaly jako nutné. Rozhraní pro práci s TCP sockety je v knihovně Qt sice dobré, mezi nedostatky ale patří nemožnost nastavit pro komunikaci posílání keep-alive zpráv¹. Pro účely aplikace je ale žádoucí vědět, kdy se uživatel nebo server odpojí, a keep-alive mechanismus je tedy implementován pomocí zpráv PING na úrovni aplikačního protokolu. Pokud v předdefinovaném intervalu nedostane komunikující strana příkaz PING nebo jiný libovolný příkaz, považuje protistranu za odpojenou a provede patřičné kroky k ukončení komunikace.

¹ Keep-alive zprávy jsou mezi komunikujícími subjekty posílány v definovaných intervalech, aby se ověřilo, že spojení je stále aktivní.

4 Implementace

Po implementační stránce lze práci rozdělit na tři samostatné celky: GUI část klienta, logickou část klienta a server. GUI část byla implementována v jazyce QML s využitím součástí Qt Quick z knihovny Qt. Různé jiné části knihovny Qt využívá i logická část aplikace i server – nicméně je pouze konzolová aplikace a grafické uživatelské rozhraní postrádá. Logická část klientské aplikace uchovává data o hře (jejichž vizualizaci zprostředkovává GUI), dohlíží na dodržování pravidel hry a v případě síťového hraní obstarává komunikaci se serverem.

Práce sestává ze dvou projektů vytvořených ve vývojovém prostředí Qt Creator: `Ghost_Stories` a `Ghost_Stories_server`.

4.1 Qt framework

Qt je multiplatformní knihovna zaměřená na tvorbu GUI a založená na C++, jejíž vývoj začal už v roce 1991 pod rukou Haavarda Norda a Erika Chambe-Enga, kteří v této souvislosti v roce 1994 založili firmu Trolltech. Do té doby vyšlo 5 hlavních verzí knihovny a firma Trolltech přešla v roce 2008 do vlastnictví finské Nokie, která vývoj knihovny v roce 2012 posunula společnosti Digia[4]. Qt je populární knihovna a využívá ji mnoho rozšířených softwarových produktů, z nejznámějších např. KDE Display Manager, VLC Media player, Maya či Skype a používá se i při tvorbě vizuálních efektů ve filmech, např. v Lucas Film[5].

Knihovna Qt je od začátku dostupná ve dvou variantách – pro komerční vývoj a pro open-source projekty, pod licencí GPL a LGPL. Do dnešního dne je Qt dostupné pro velké množství platform, např. Windows, OS X, Android, X11 a knihovna přerostla původní určení (tvorbu GUI) a nabízí mnoho dalších funkcí, např. práci se síťovými zařízeními, práci se soubory, s databázemi nebo zpracování XML souborů. Tvorbu aplikací v jazyce C++ s pomocí knihovny Qt vhodně popisuje kniha C++ GUI Programming with Qt 4[8].

V době psaní této práce je k nejnovější stabilní verze 5.0, vydaná 19. prosince 2012, která představuje výraznou změnu oproti předchozím 4.x verzím v podobě hardwarové akcelerace grafiky založené z velké části na jazycích QML a JavaScript. Ve verzi 5.1, která by měla vyjít v červnu 2013, se očekává podpora operačních systémů Android a iOS. K vytvoření této práce byla použita knihovna Qt ve verzi 4.8.1 – v počátečních fázích vývoje aplikace nebyla verze 5.0 ještě ani vydaná a dnes stále ještě zůstává hlavním učebním zdrojem pro verzi 5.0 její dokumentace, jelikož literatury týkající se této verze není mnoho k dispozici. S Qt 4.x je situace opačná, jelikož je Qt oblíbená knihovna a první verze řady 4 byla vydaná už v roce 2005, literatury je k dispozici dostatek.

Pro tvorbu projektů v jazycích C++ a QML nabízí Qt vlastní vývojové prostředí, které se nazývá Qt Creator, je součástí balíku Qt SDK (Software Development Kit) a disponuje obvyklými funkcemi vývojových prostředí jako zvýraznění syntaxe (včetně podpory QML), navigací v kódu, kontextovou nápovědou a tvorbou projektů (soubory shrnující informace o projektu jsou uloženy s koncovkou `.pro`). Tato aplikace byla kompletně vytvořena v programu Qt Creator verze 2.4.1.

4.2 Klient – logická část

Hlavní částí programu jsou třídy `client` a `game`, první jmenovaná obstarává komunikaci se serverem a druhá samotný běh hry. Obě třídy jsou aplikací návrhového vzoru Singleton neboli jedináček, a celý v rámci celého programu se tedy nachází jen jedna jejich instance, dostupná globálně skrze statické metody `client::getClient()` a `game::getGame()`. Atributy třídy

game lze zhruba rozdělit na tři druhy: přímá reprezentace herních prvků, stavové informace o hře a data potřebná pro běh aplikace. Mezi reprezentaci herních prvků patří informace, které se v původní deskové hře skutečně fyzicky nacházejí – např. pole čtyř instancí třídy `player`, reprezentující jednotlivé hráče, zásoby žetonů `stashTao` a `stashQi` nebo hozené kostky v poli `taoRoll`. Stavové informace o hře jsou takové atributy, které přímo nereprezentují komponenty ve hře, ale vypovídají o tom, v jakém stavu se hra právě nachází. Například počítadlo `bossesToBeat`, označující počet inkarnací Wu-Fenga, které zbývá porazit, boolean `TakingAction`, jehož hodnota vypovídá o tom, zda hráč právě provádí akci (a tedy nemůže udělat jiné věci, než tuto akci dokončit), nebo atribut `currentPlayer`, jež označuje, který hráč je právě na tahu. Ostatní data potřebná pro běh aplikace jsou různorodá, od tabulky nejlepších výsledků `hiscore`, přes obsah komunikačního okna `chattext` až po seznam serverů `serverList`.

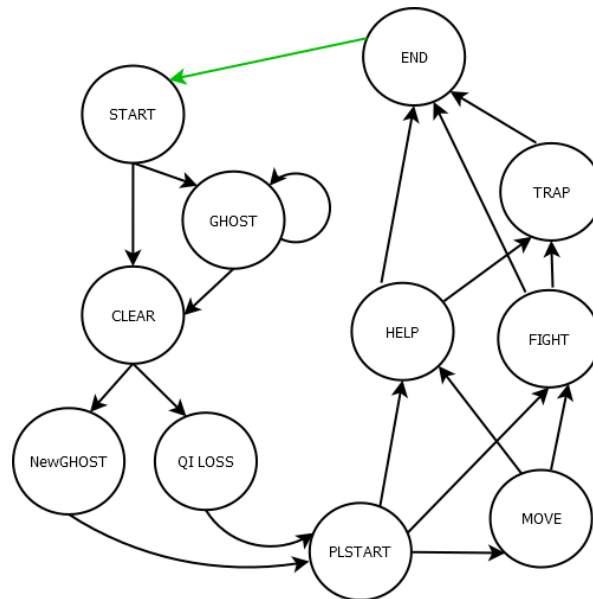
Po spuštění aplikace konstruktor třídy `game` inicializuje všechny potřebné atributy na příslušné hodnoty, načte duchy a nejlepší výsledky z příslušných XML souborů na disku a náhodně přiřadí barvy a schopnosti hráčům a zamíchá lokace ve vesnici. K získávání náhodných čísel v celé aplikaci slouží metoda `randInt(int low, int high)`, která náhodně generuje čísla od `low` po `high` transformací z generátoru `qrand()`, který je součástí Qt. Jako seed pro generování náhodných čísel slouží aktuální čas při spuštění aplikace v milisekundách. Při vytváření nové hry se volá metoda `SetupLocalGame`, resp. `SetupNetworkGame` pro síťovou hru. Tyto metody pracují podobně, obě připraví stavové a herní atributy na hodnoty příslušící zvolenému počtu hráčů a obtížnosti – počet Qi, Tao, Yin-Yang a neutrálních žetonů, počet lokací, které mohou hráči ztratit a příznak `isAbsent`, že daný hráč není ve hře. Rovněž proběhne sestavení balíčku duchů, při čemž je stejně jako při míchání lokací i hráčů využit modifikovaný algoritmus Fisher-Yates shuffle (známý i jako Knuth shuffle). Tento algoritmus generuje náhodné permutace konečné množiny prvků a to tak, že všechny permutace mají stejnou pravděpodobnost výskytu. O popularizaci tohoto algoritmu se zasloužil Donald E. Knuth[6]. Další výhodou Fisher-Yates shuffle je, že jeho složitost je pouze $O(n)$ ve srovnání s naivním algoritmem míchání, jehož složitost je $O(n^2)$.

```
for i from n - 1 downto 1 do
  j ← random integer with 0 ≤ j ≤ i
  exchange a[j] and a[i]
```

Algoritmus 4-1: Fisher-Yates shuffle

Po inicializaci hry do počátečního stavu je zavolána metoda `Run()`, která provádí potřebné akce v závislosti na aktuální nastavené fázi hry. Fáze hry jsou definovány spolu s ostatními výčty v souboru `misc.h` pod názvem `turnPhase` a odpovídají fázím hry, tak jak jsou uvedeny v pravidlech. Vzhledem k povaze deskové hry není složité navrhnout na základě pravidel stavový automat; desková hra sama o sobě v podstatě stavový automat je. Automat navržený pro tuto hru je na obrázku 4-1. Nepokrývá samozřejmě všechny možné kombinace stavů, které mohou ve hře nastat, jedná se o znázornění průběhu kola jednoho hráče.

Stavy `START` a `GHOST` odpovídají fázi `GHOSTACTION`, kdy hra kontroluje karty duchů na desce aktivního hráče a provádí činnosti odpovídající schopnostem duchů (posun figurky – `MoveHaunter()`, hod kostkou – `CurseRoll()`). Pokud jsou všechny duchové obslouženi, fáze se přepíná na `OVERRUNCHECK` (odpovídá stavu `CLEAR`). V této fázi hra zkontroluje, zda má hráč na desce před sebou volné místo. Pokud ano, uvolní vrchního ducha z balíčku duchů k umístění (přepnutí fáze na `NEWGHOST`, nastavení atributu `ghostReadyToDeploy` na hodnotu `true` a čekání, kam hráč ducha umístí), v opačném případě odebere aktivnímu hráči jeden žeton Qi. Na



Obrázek 4-1: Stavový automat jednoho kola hry

konci těchto akcí se fáze přepne na MOVE (stavy PLSTART + MOVE v diagramu automatu). V této chvíli umožňuje hra hráči posunout figurku svého mnicha a jiné věci, které pravidla dovolují dělat před pohybem. Pokud se hráč pohne nebo rovnou využije akci, hra přepne fázi na ACTION (stavy HELP + FIGHT v diagramu automatu). V této fázi je hráči umožněno bojovat s duchy nebo využít pomoci lokace, na které právě stojí. Využití pomoci v lokaci nastaví stavové proměnné podle vybrané lokace a hra následně vyčkává na uživatelský vstup (výběr žetonu, kliknutí na plochu, posunutí karty apod.). Pokud hráč spustí boj s duchem, hra naplní všechny atributy instance třídy `exorcism`, nastaví příznak, který obrazovku boje s duchy zobrazí místo herní desky, a čeká na uživatelský vstup. Během tohoto vyčkávání je nastavena proměnná `takingAction` na hodnotu `true` a dokud akce neskončí, nemůže hráč v rámci herního UI dělat nic jiného, ani hru ukončit. Když už nemá hráč žádné akce k dispozici, hra se přepne do fáze PLACEBUDDHA (stavy TRAP + END v diagramu automatu). V této fázi hráč může umístit na desku figurku Buddy ze své zásoby a fáze trvá tak dlouho, dokud neklikne na tlačítko, kterým ukončí svůj tah. Zelená šipka v diagramu automatu označuje přepnutí aktivního hráče na dalšího v pořadí. To vykonává metoda `NextPlayer()`, která resetuje a inicializuje atributy spojené se změnou aktivního hráče.

Ve výše uvedeném popisu průběhu herního kola chybí mnoho detailů, které jsou ale povahou spíše drobnosti a týkají se výjimek a drobných pravidel, které se v deskových hrách běžně nacházejí – příkladem jsou například zvláštní schopnosti hráčů – jeden si smí před pohybem vzít jeden tao žeton, druhý se smí pohybovat o dvě políčka místo jednoho, třetí nemusí házet kostkou prokletí atd. Všechny tyto výjimky jsou na příslušných místech v kódu kontrolovány a ošetřeny, stejně jako možnost, že aktivní hráč využívá schopnost jiné desky při hře méně než čtyř hráčů.

4.3 Klient - GUI

Grafické uživatelské rozhraní je zpracováno v jazyce QML. QML spadá do sady nástrojů pro tvorbu uživatelských rozhraní Qt Quick[10], která patří do knihovny Qt. Je to deklarativní jazyk založený na jazyce JavaScript, jehož hlavním využitím je návrh grafických uživatelských rozhraní. S jeho pomocí lze poměrně jednoduše vytvářet graficky různorodá uživatelská rozhraní. Jazyk QML je poměrně

nový, vznikl v roce 2009 a využívá se zejména v oblasti mobilních aplikací, kde je důležité ovládání hmatem a plynulé animace, nicméně není na mobilní aplikace omezen.

Zdrojové soubory v jazyce QML popisují objektový strom prvků neboli elements. Mezi základní prvky, které jsou součástí Qt, patří např. grafický prvek obdélník (`rectangle`) nebo obrázek (`image`). Kromě grafických prvků obsahuje QML ještě prvky behaviorální, jako např. odchyťování událostí myši (`mouseArea`) nebo animace prvků (`animation`). Ze základních prvků lze skládat prvky vlastní a ty pak dále používat jako standardní prvky QML. Takto lze vytvářet jednoduché prvky jako např. tlačítka, ale i celé složité aplikace.

Vlastní prvek v jazyce QML je tvořen souborem s názvem „názevPrvku.qml“, v němž se nachází pouze jeden kořenový prvek. Takto vytvořený prvek lze pak v dalších zdrojových souborech používat pod názvem „názevPrvku“. Struktura zdrojového QML souboru vypadá následovně:

```
Image {
    id: stoneb
    source: "pics/stoneB.png"
    anchors.top: parent.top
    anchors.left: parent.left
    anchors.topMargin: 0.05*parent.height
    anchors.leftMargin: 0.05*parent.width
    width: 0.4*parent.width
    fillMode: Image.PreserveAspectFit
    visible: Game.getPlayerQiColor(1) === 0 && type === 0
    MouseArea {
        anchors.fill: parent
        enabled: stoneb.visible
        onClicked: {
            if (Game.isTurn() && Game.getUsingCemetery()) {
                Game.Resurrect(1, tileId)
            }
        }
    }
}
```

Kód 4-1: Úryvek kódu v jazyce QML

V ukázce je prvek `Image` a v něm další prvek `MouseArea`. Objekt v jazyce QML je specifikován názvem prvku, jehož je instancí, následovaným složenými závorkami. Uvnitř složených závorek lze definovat další atributy objektu, v příkladu 4-1 např. zdrojový soubor pro obrázek (`source`) či jeho ukotvení k jiným objektům (`anchors`) – tyto atributy jsou definovány způsobem `název:hodnota`. Speciální místo zaujímá atribut `id`, který objektu přiřazuje identifikátor, s jehož pomocí se lze na objekt odkazovat i mimo objekt samotný. Například `MouseArea` využívá k nastavení svého atributu `enabled` hodnotu atributu `visible` z obrázku `stoneb`.¹ Jelikož QML vychází z JavaScriptu, lze jako hodnotu atributu použít hodnotu výrazu (viz např. atribut `visible` v kódu 4-1). Atribut `onClicked` uchovává akci, která se provede, pokud objekt typu `MouseArea` zaregistruje kliknutí v rámci své plochy. Výsledkem ukázkového kusu kódu tedy bude obrázek, na který lze kliknout, čímž se spustí nějaká akce.

¹ V tomto případě by místo `id` objektu `stoneb` šlo použít i termín `parent`, jelikož objekt typu `MouseArea` je zde přímým potomkem objektu `stoneb`.

GUI aplikace je kompletně vytvořené v QML a skládá se z jednoho dokumentu pro každou obrazovku v menu, hlavního souboru pro herní obrazovku a různých dalších prvků, které se využívají zejména v herní obrazovce (např. prvek `Board` uložený v `Board.qml`). S částí aplikace napsané v C++ je uživatelské rozhraní spojeno odkazy na instanci tříd `client` (v QML pod označením `Client`) a `game` (v QML pod označením `Game`). Odkazy jsou vytvořeny při startu programu v `main.cpp` pomocí metody `QDeclarativeContext::setContextProperty`. Skrz tyto odkazy lze volat ty metody odkazovaných objektů, které jsou při deklaraci opatřeny makrem `Q_INVOKABLE`. Toho se v této aplikaci využívá zejména pro změnu stavových proměnných hry, což signalizuje instanci třídy `game`, že hráč vykonal akci, kterou je potřeba zpracovat.

Herní obrazovka je nejobsažnější QML dokument z celého projektu a je poskládána z množství různých prvků. Vesnice uprostřed herní desky je reprezentována typem `Village`, což je mřížka o devíti objektech typu `VillageTile`. `VillageTile` se stará o zobrazení těch herních prvků, které se mají na konkrétním poli právě nacházet, k čemuž slouží atribut `tileId`, určující typ pole. Po stranách vesnice jsou objekty typu `Board`, znázorňující desky hráčů a vše, co na nich leží. Obdobně jako typ `VillageTile` má typ `Board` k dispozici atribut `boardId`. Pravá dolní sekce se zdroji aktivního hráče je implementována typem `PlayerStats`. Zobrazení polí vesnice a karet duchů je rozděleno na dva prvky, `TileViewer` a `GhostViewer`, které se střídají podle potřeby. `TileViewer` obsahuje popis jednotlivých polí vložený do prvku `Flickable`, který umožňuje posouvat textem, pokud je příliš dlouhý. Zdroje ostatních hráčů v levém horním rohu zobrazuje prvek `TeamGrid`. Karty s duchy ležící na herní desce nejsou součástí jednotlivých desek, ale tvoří prvek sám o sobě, aby je bylo možno přetažením přesouvat mezi deskami. Pro zobrazení boje s duchy slouží prvek `Exorcism`, který v případě potřeby překrývá herní desku a znemožňuje hráči interakci s ní, dokud není boj dokončený.

Zobrazení herních prvků je na prvky uživatelského rozhraní navázáno prvkem `Connections`, který je napojen na signál `uiChanged()`, vysílaný z C++ kódu, který signalizuje změnu dat, kterou je potřeba uživateli zobrazit. Po obdržení tohoto signálu je provedeno obnovení uživatelského rozhraní.

4.4 Síťová část - Server

Server je implementačně poměrně prostý – využívá třídy z Qt, která představuje rekurzivní TCP server a zapouzdřuje práci se systémovými knihovnami. Tato třída se nazývá `QTcpServer`[7]. Při startu serveru je spuštěna instance třídy `QTcpServer`, která začne naslouchat pro příchozí spojení na všech adresách lokálního stroje (`QHostAddress::Any`, což je ekvivalentem adresy 0.0.0.0) a defaultně na portu 1337. Port, na kterém aplikace naslouchá, lze změnit uvedením čísla požadovaného portu jako prvního a jediného argumentu při spuštění aplikace. Signál příchozího připojení je v konstruktoru serveru spojen se slotem `handleNewConnection()`.

Současně se startem načte aplikace seznam zaregistrovaných uživatelů a jejich hesel. Hesla se na server posílají a jsou i uložena ve formě MD5 hashe, který je vytvořen s pomocí metody `QCryptographicHash::hash()`. Kombinace hesel a uživatelských jmen se ukládá na disk vždy při registraci nového hráče. Registrace probíhá automaticky, když se o připojení k serveru pokusí uživatel, jehož jméno v seznamu uživatelů uvedeno není.

Při příchodu nového připojení získá server socket, na kterém probíhá komunikace, metodou `QTcpServer::nextPendingConnection`. K tomuto socketu jsou pak přivázány dvojice signálů a slotů pro čtení/odpověď a signálu odpojení. Následně požádá server komunikující stranu o přihlášení. Veškerá další komunikace probíhá obdržáním signálu `readyRead()` a zavoláním slotu

`reply()`, v jehož rámci server zjistí, jaký požadavek na server přišel, a podle toho vybere příslušnou akci.

Nově vytvořená hra dostává od serveru přidělený unikátní identifikační string, kterým se na hru server i klient odkazuje. Identifikační řetězec generuje metoda `QUuid::createUuid()`. Na příkaz připojení ke hře reaguje server přidáním žadatele do zvolené hry a notifikací ostatních hráčů ve stejné hře o připojení nového hráče. Při žádosti o seznam běžících či uložených her je klientovi odeslán seznam žádaných her ve formátu XML, obsahující název, počet hráčů, obtížnost a identifikační řetězec dané hry. Aktualizace herních dat mezi klienty je řešena prostým příkazem `UPDATE`, jehož argumentem je kus XML obsahující právě změněné herní data. Aniž by server tato data zpracovával, pošle je zbývajícím klientům ve stejné hře. Klienti si tato data pak sami zpracují. Stejně probíhá i přeposílání zpráv pro komunikaci hráčů ve hře.

Druhá ze dvou tříd v projektu serveru je třída `heartbeat`. Tato jednoduchá třída obsahuje jen dva atributy, jednu metodu a jeden slot. Instance této třídy slouží k implementaci mechanismu `keep-alive` na straně serveru. Server musí hlídat stav spojení se všemi připojenými klienty a každý klient tedy musí mít vlastní přidružený `keep-alive` mechanismus. Při navázání spojení se vytváří instance třídy `heartbeat` navázaná na socket příchozího spojení a začíná běžet časovač `t` (třída `QTimer`). Časovač je nastaven na 20 sekund, a pokud doběhne do konce, spustí se slot `dcTimeout()`, který odpojí přidružený socket. Požadavky přicházející na server od klienta resetují časovač příslušného socketu – pro případ delší neaktivity vysílá klient periodicky příkaz `PING`, který slouží k resetování časovače. Tyto příkazy v době nečinnosti mají podstatně kratší periodu než časovač odpojení, aby se zamezilo odpojení z důvodu pouhého zpomalení na lince.

5 Testování a zhodnocení výsledků

Důležitou součástí vývojového cyklu aplikace je testování. V průběhu vývoje byly testovány všechny nové prvky hned, jak byly v aplikaci implementovány – tyto testy se zaměřovali spíše na jejich funkčnost z programového hlediska než na správnost dodržování pravidel. V raných fázích vývoje probíhalo obvykle testování specifických částí aplikace za specifických podmínek. Před takovým testem se počáteční podmínky při spuštění aplikace upravily tak, aby bylo možno testovat přímo daný scénář bez nutnosti nejprve procházet hrou až do vybraného bodu. Testování prvků GUI napsaných v QML probíhalo jejich přímým zobrazením bez závislosti na herních datech, podmínky zobrazení byly přidány teprve tehdy, když se daný prvek zobrazoval korektně.

Další fází testování bylo testování konkrétních oblastí hry ve chvíli, kdy byly všechny části hry zhruba hotové. Zde přišly na řadu specifické oblasti, např. zobrazování polí vesnic ve správném pořadí, i když pole rotuje podle aktivního hráče, nebo správnost chování hry v konkrétní situaci, kdy se na jedné desce hráče nachází tři duchové se symbolem kostky a ve fázi duchů je nutné zajistit 3x po sobě hod černou kostkou. Další z cílených testů se týkal síťové komunikace a její spolehlivosti a korektnosti – bylo ověřeno, že server i klient reagují na vzájemné zprávy podle očekávání, ale přesto se chyby objevily, například v pořadí argumentů, kdy klient předpokládal pořadí argumentů u příkazu opačné, než jak jej server odesílal, a hra se kvůli tomu chovala neočekávaně.

Když se aplikace dostala do fáze vývoje, kdy bylo možno hru hrát od začátku do konce bez výrazných chyb, záseků či padání programu, vstoupili do procesu vývoje dobrovolníci, aby hru testovali při reálném hraní. Celkový počet dobrovolných testerů hry byl 8. Jejich zkušenosti s hrou se lišily od úplné neznalosti až po více než deset 10 odehraných her v deskové předloze. Tyto testy probíhaly zejména při reálné hře ve všech možných kombinacích počtu hráčů a obtížností se primárním úkolem testerů zde bylo ověřit, zda hra opravdu důsledně hlídá dodržování pravidel. S pochopením pravidel problém nebyl, s pochopením ovládní hry už místy ano – ovládní není dostatečně intuitivní samo o sobě a na základě těchto poznatků vznikly instrukce, jak hru při hraní ovládat. Tyto instrukce jsou dostupné ze pod položkou „Help“, která se nachází v hlavním menu hry. Zpětná vazba testerů byla vesměs pozitivní, ale jelikož testování probíhalo iterativně už během vývoje, bylo objeveno a posléze i opraveno velké množství chyb. Žádná z nich nebyla natolik závažné povahy, že by vedla k pádu celé aplikace, ale některé znemožňovaly další průchod hrou či naopak umožňovaly obcházení pravidel, nebo hráče mátlly špatným zobrazováním prvků GUI. Všechny takto nalezené chyby byly následně nasimulovány, přezkoumány a nakonec opraveny. Příklad za všechny: Duchové, kteří berou hráčům kostku, po odstranění z herní plochy sice zmizeli i se symbolem sebrané kostky, kostka se hráčům již ale nevrátila, protože, jak bylo při krokování programu zjištěno, bylo místo počítadla kostek při odstranění ducha inkrementováno jiné počítadlo. Kromě odhalených chyb měli testeři i několik připomínek k samotnému designu a způsobu ovládní – vybrané připomínky, které byly uznány za dostatečně přínosné, jsou uvedeny v sekci 5.1 mezi dalšími směry rozvoje aplikace.

Nakonec proběhly testy nestandardního chování programu – účelem bylo zjistit, jak se program zachová, pokud dojde k narušení prvků, na které při svém běhu spoléhá. Mezi tyto patří zejména externí soubory, formát textu uživatelského jména, hesla nebo chatu a vypadávající připojení k serveru. Na většině externích souborů aplikace závislá není a jejich absence tedy nijak nevádí. Jediný problém představoval soubor `ghosts.xml`, bez něhož si hra nemá odkud načíst duchy a nelze tedy vytvořit hru. V takovém případě se ale hra chovala neočekávaně a bylo nutné dodělat kontrolu a upozornit uživatele na chybějící soubor už při startu hry. Ošetření poškození dat bylo v původním návrhu opomenuto a proto se poslední úpravy a opravy týkaly zejména této oblasti.

5.1 Další rozvoj

Na základě návrhů testerů i po úvahách autora vznikl seznam věcí, které by se na aplikaci daly vylepšit, případně čím by se dala ještě rozšířit. Následující seznam je řazen pouze podle příslušnosti ke konkrétní části aplikace a nebere v potaz případnou složitost návrhu. Vývoj hry bude pokračovat i nadále, nicméně nelze s jistotou říci, které z těchto prvků se zpracování dočkají, a které ne.

Herní obsah

- Bonusové karty duchů vydané jako mini-rozšíření.
- Rozšíření White Moon.
- Rozšíření Black Secret.

Možnosti GUI

- Vrátit zpět právě provedený pohyb.
- Zrušit výběr akce, pokud není dokončená (hráč si provedení rozmyslí před hodem kostkou apod.).
- Kromě duchů a vesnice zobrazovat ve zvětšené verzi a s nápovědou i schopnosti hráčů.
- Částečně automatizovat boje s duchy (okamžitě přidělit to, co je jednoznačné) a jasněji ukázat, že duch je poražený.
- Zakladatel hry může před začátkem hry vyhodit připojené hráče.
- Hra komentuje události (např. v okně pro chat).

Klient

- Editor nových karet duchů.
- Možnost detailně vytvořit vlastní hru (přidělení mnichů a schopností, rozložení vesnice, uspořádání a skladba duchů...).
- Vhodná hudba do pozadí.
- Statistiky her.
- Ovládání menu klávesnicí.

Server

- Hry dostupné jen přes heslo.
- Tabulka nejlepších výsledků napříč celým serverem.
- Posílení bezpečnosti proti brute-force útokům přihlašování.
- Statistiky her.

6 Závěr

Cílem této práce byl nastudovat vhodnou problematiku, navrhnout řešení a implementovat aplikaci pro hraní deskové hry Ghost Stories na PC, včetně síťové hry několika hráčů, a následně ji podrobit testování. Nastudování potřebné problematiky nebylo obtížné v případě obecné práce s Qt, síťové komunikace a pravidel zvolené hry. Nastudování jazyka QML, který byl zvolen pro implementaci GUI, se ukázalo být o něco obtížnější – jelikož je QML jazyk starý teprve necelé 4 roky, trpí nedostatkem kvalitní literatury a studium tohoto jazyka bylo tedy nutné uskutečnit zejména s pomocí on-line návodů[9] a samotné dokumentace[10].

Při návrhu řešení bylo nutno vyřešit několik problémů. Zaprvé, jak poskládat všechny informace o hře na jednu obrazovku, aby uživatel během hry nemusel přepínat okna? První návrh GUI a jeho variace tento problém neřešili uspokojivě, ale druhý návrh už byl uznán za vhodný. Navržené řešení je poměrně intuitivní (využívá např. přetahování prvků tam, kde se v deskové předloze s komponenty skutečně hýbe), nestačí to ale k tomu, aby uživatel znalý pravidel deskové hry mohl aplikaci plně používat bez dalších instrukcí – vzniklo proto PDF s návodem, jak hru ovládat. Další problém spočíval ve využití komunikace na transportní vrstvě TCP skrz prostředky knihovny Qt. Ty sice značně zjednodušují práci po síti, ale neumožňují nastavit keep-alive mechanismus pro TCP spojení. Aplikace tedy implementuje vlastní keep-alive mechanismus v rámci aplikačního protokolu. Logika dohlížející na dodržování pravidel hry využívá jednoduchého stavového automatu, který se restartuje vždy se změnou aktivního hráče.

Aplikace je implementována v jazycích C++ a QML s využitím knihovny Qt pro různé dílčí činnosti (práce se soubory, síťová komunikace, parsování XML) a využívá i mechanismu signálů a slotů, který je pro knihovnu Qt typický. Práci lze rozdělit na dva projekty dle zvolené architektury klient – server. Klient zprostředkovává zobrazení hry, dohlíží na dodržování pravidel a provádí činnosti, které hráč oproti deskové variantě sám nutně dělat nemusí. Server je jednoduchá aplikace umožňující komunikaci mezi více klienty a udržující databázi uživatelů a uložených her, oboje v podobě XML souborů. Výsledná implementace serveru je dost obecná na to, aby se dal v případě potřeby s minimálními úpravami použít pro jakoukoliv jinou hru podobného typu, která svá data rovněž ukládá ve formátu XML. Vzhledem k implementaci nad TCP protokolem by ale nebylo vhodné používat jej v situacích, kdy je žádoucí rychlá odezva.

Testování prokázalo, že aplikace je připravena k běžnému používání. Výhrady testerů byly minimální a veškeré odhalené chyby se dočkaly opravení. Některé návrhy na vylepšení použitelnosti aplikace, které z testování vzešly, jsou uvedeny v kapitole 5.1. Hra sice korektně zprostředkovává hru podle pravidel deskové předlohy, nicméně na uživatelském dojmu z aplikace (UX, User Experience), by se ještě dalo ledacos zlepšit. Výsledná aplikace věrně zprostředkovává deskovou předlohu Ghost Stories, umožňuje hrát hru na jednom počítači i po síti a počítačové zpracování pravidel hru urychluje (odpadá chystání/sklízení hry a manipulace s herními komponenty).

Literatura

- [1] BAUZA, Antoine. Ghost Stories Rules: v. 1.3. In: *Repos Production* [online]. 2009 [cit. 2013-05-10]. Dostupné z: http://rprod.com/uploads/GS_RULES_US_ver1_3-web.pdf
- [2] *300px-Tcp-handshake.png* [diagram] [online]. 2005 [cit. 2013-05-10]. Dostupné z: <http://upload.wikimedia.org/wikipedia/commons/c/c7/300px-Tcp-handshake.png>
- [3] APPELCLINE, Shannon. Board Game History: The Birth of the Modern Board Game. In: *Mechanics & Meeples* [online]. 2013 [cit. 2013-05-10]. Dostupné z: <http://www.mechanics-and-meeples.com/2013/01/14/board-game-history-the-birth-of-the-modern-board-game/>
- [4] Qt: The Power of a Complete Development Framework. In: *Qt* [online]. 2013 [cit. 2013-05-14]. Dostupné z: <http://qt.digia.com/Product/>
- [5] Lucasfilm Ltd. Uses Trolltech's Qt to Create User Interface for Cutting-Edge Content Creation. *Business Wire* [online]. 2007 [cit. 2013-05-10]. Dostupné z: <http://www.businesswire.com/news/home/20071015006427/en/Lucasfilm-Ltd.-Trolltechs-Qt-Create-User-Interface>
- [6] KNUTH, Donald Ervin. The art of computer programming. 3rd ed. Upper Saddle River: Addison-Wesley, c1998, xiii, 762 s. ISBN 02-018-9684-2.
- [7] WATZKE, David. Konzolové programy v Qt 4 : TCP server. Abclinuxu [online]. 16. 9. 2009, [cit. 2013-05-10]. Dostupný z WWW: <http://www.abclinuxu.cz/clanky/programovani/konzolove-programy-v-qt-4-3-tcp-server>
- [8] BLANCHETTE, Jasmin a Mark SUMMERFIELD. *C++ GUI Programming with Qt 4: Prentice Hall Open Source Software Development Series*. 2. ed. Trolltech ASA: Prentice hall, c2008, xxi, 718 s. ISBN 978-0-13-235416-5.
- [9] NOMOVOK a QUIT CODING. *Qt Quick Game Programming*. 2010. Dostupné z: http://quitcoding.com/download/Qt_Quick_Game_Programming_1_0.pdf
- [10] Qt Project: Qt Quick. DIGIA. *Qt Project* [online]. 2012 [cit. 2013-05-11]. Dostupné z: <http://qt-project.org/doc/qt-4.8/qtquick.html>

Seznam příloh

Příloha A	Finální návrh GUI
Příloha B	Struktura uložené hry
Příloha C	CD s elektronickou verzí práce, zdrojovými kódy a návodem k aplikaci

A Finální návrh GUI



B Struktura uložené hry

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<savegame>
<date>02.05.2013 22:07</date>
<playerNumber>1</playerNumber>
<difficulty>1</difficulty>
...1
```

mnoho dalších stejně uložených proměnných

```
...
<playerPerspective>
  <val>0</val> ... <val>1</val>
</playerPerspective>
```

```
<villagePerspective>
  <val>0</val> ... <val>2</val>
</villagePerspective>
```

```
<reverseVillage>
  <val>0</val> ... <val>8</val>
</reverseVillage>
```

```
<village>
  <val>4</val> ... <val>8</val>
</village>
```

```
<villageLife>
  <val>0</val> ... <val>1</val>
</villageLife>
```

```
<NPC>
  <val>0</val> ... <val>2</val>
</NPC>
```

```
<rerolls>
  <val>1</val> ... <val>1</val>
</rerolls>
```

```
<stashTao>
  <val>4</val> ... <val>4</val>
</stashTao>
```

```
<taoRoll>
  <val>0</val> ... <val>0</val>
</taoRoll>
```

```
<ghostsIncoming>
  <val>41</val>
  ...
  <val>6</val>
</ghostsIncoming>
```

```
<players>
```

```
<val>
<isAbsent>0</isAbsent>
<playerName></playerName>
<location>4</location>
```

```
...
```

Množství dat o hráči

```
...
```

```
<taoPool>
  <val>1</val> ... <val>0</val>
</taoPool>
```

```
<board>
<val>34</val> ... <val>0</val>
</board>
```

```
<buddha>
  <val>0</val> ... <val>0</val>
</buddha>
```

```
<mantra>
  <val>0</val> ... <val>0</val>
</mantra>
```

```
<dieCaptured>
  <val>0</val> ... <val>0</val>
</dieCaptured>
```

```
<haunter>
<val>
  <val>1</val> ... <val>0</val>
</val>
```

```
...
<val>
  <val>0</val> ... <val>0</val>
```

```
</val>
</haunter>
```

```
</val>
...
```

```
</players>
</savegame>
```

¹ Tři tečky označují vynechaná místa, kde se opakuje velké množství dat s podobnou strukturou