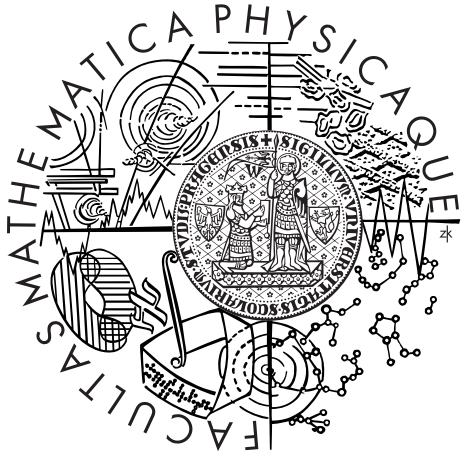
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jan Roztočil

Nástroj pro tvorbu a ladění herních strategií

Středisko informatické sítě a laboratoří

Vedoucí bakalářské práce: RNDr. Libor Forst
Studijní program: Informatika
Studijní obor: Programování

Praha 2013

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) 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: Nástroj pro tvorbu a ladění herních strategií

Autor: Jan Roztočil

Katedra: Středisko informatické sítě a laboratoří

Vedoucí bakalářské práce: RNDr. Libor Forst

Abstrakt: Cílem práce je vytvořit implementaci strategické deskové hry Risk, která bude podporovat jak hru proti počítačem ovládaným soupeřům, tak hru více lidských hráčů prostřednictvím sítě Internet. Součástí úkolu je navrhnout znovupoužitelný síťový protokol. V rámci implementace bude poskytnuto rozhraní pro definování nových počítačových strategií. Program bude koncipován jako nástroj pro vývoj těchto strategií, s jeho pomocí bude možné strategie ladit i za běhu hry. Program bude podporovat operační systémy MS Windows a Linux.

Klíčová slova: hra Risk, multiplayer, herní strategie, vývojářský nástroj

Title: Game strategy creation and debugging tool

Author: Jan Roztočil

Department: Středisko informatické sítě a laboratoří

Supervisor: RNDr. Libor Forst

Abstract: The aim of this thesis is to create an implementation of a strategic board game of Risk. A play against computer-controlled opponents will be supported, as well as multiplayer game of human players over the Internet. A design of a reusable communication protocol is also a part of the task. An interface for defining a new computer strategies will be provided within the implementation. The program will be oriented as a tool for develepment of those strategies, it will help to debug strategies within an actual game. The program will support MS Windows and Linux operating systems.

Keywords: Risk game, multiplayer, game strategy, development tool

Obsah

Úvod

Implementace deskových her jako počítačových programů je velmi populárním tématem po celou dobu existence moderních počítačů. Jedním příkladem za všechny je šach. Už od 50. let minulého století je cílem řady šachových nadšenců i vědců vyvíjet silné počítačem řízené hráče, a demonstrovat tak schopnosti počítačového myšlení.

Obecně není vývoj umělých hráčů a strategií snadným úkolem. Šach má tu výhodu, že — přes svou značnou složitost a obrovské množství možných stavů — má poměrně kompaktní herní prostředí. To ovšem neplatí pro každou deskovou hru.

Strategická desková hra Risk, která je předmětem této práce, je mnohem mladší než šach a zdaleka o ní nebylo sepsáno tolik teorie. Nicméně podobně jako šach vyžaduje dobré strategické myšlení a taktické rozhodování.

Cílem této práce je ukázat, jak je možné vytvářet a ladit počítačem řízené strategie této deskové hry a poskytnout nástroje, které tento proces zjednoduší.

K tomu je však nutné nejprve vytvořit kompletní implementaci vlastní hry. Navíc je vyžadováno, aby tato implementace podporovala nejen hru proti počítačovým hráčům, ale i lidským soupeřům prostřednictvím Internetu. Přes síť by měly být schopny hrát i samotné umělé strategie.

První kapitola tohoto textu se věnuje seznámení s vlastní hrou Risk, krátce popisuje její pravidla a některé bežné strategické postupy. Druhá kapitola představuje vlastní aplikaci a popisuje způsob jejího používání. Třetí kapitola se zabývá aplikací z programátorského hlediska, rozebírá její návrh a diskutuje problémy vziklé při vývoji.

1. O hře Risk

Risk je strategická desková hra, kterou navrhl Alebert Lamorisse a v roce 1957 ji pod názvem *La Conquête du Monde* uvedla na trh firma Parker Brothers (dnes vlastněná společností Hasbro). Hra je standardně určena pro 3 – 6 hráčů. Hrací plán je rozdělen na jednotlivá území v několika kontinentech. Cílem je dobýt všechna území a ovládnout svět.

Hra se za dobu své existience objevila v řadě různých variant, tato práce si jako předlohu bere původní verzi hry, resp. její remake, který vyšel v České republice v roce 2007.

1.1 Pravidla hry

Podrobný návod ke hře [1] lze stáhnout na webu společnosti Hasbro a je také obsažen na přiloženém CD, nicméně pro úplnost jsou pravidla v následujícím textu krátce shrnuta.

Hraje se na mapě světa rozdělené do 42 území na 6 kontinentech. Hráči vlastní tato území, posilují je, přesouvají armády a útočí na soupeře. Cílem je dobýt všechna uzemí na hracím plánu.

1.1.1 Rozebrání území

Na začátku hry získají hráči určité množství jednotek, přesný počet záleží na počtu hráčů a je uveden v [1]. Tyto jednotky pak postupně umisťují na herní plán, tím pro sebe území zabírají.

Když jsou všechna uzemí rozebrána, pokračují hráči v jejich posilováni. Postupně umisťují zbylé jednotky na svá uzemí. Na konci této fáze hry jsou všechna území rozebrána a na každém území je přítomna alespoň jedna jednotka, což je ostatně invariant, který platí po celou dobu hry.

1.1.2 Vlastní hra

Hráči se postupně střídají, každý ve svém kole provede následující kroky: povinně získá a umístí nové jednotky, provede libovolný počet útoků a volitelně přesune jednotky z jednoho území na druhé.

Získání posil

Každý hráč získá na začátku svého kola určité množství nových jednotek. Přesný počet závisí na počtu ovládaných území a kontinentů. Počet jednotek za území lze získat ze vztahu 1.1. Bonus za kontinetny je uvedený opět v [1], obecně však platí, že čím větší kontinent, tím větší bonus. Hráč získá bonus za kontinent, pouze pokud ovládá na začátku svého tahu všechna území na tomto kontinentu.

$$\min\left(\left\lfloor\frac{\#\text{území}}{3}\right\rfloor, 3\right) \tag{1.1}$$

V tomto kroku může hráč také směnit karty za další bonusové jednotky. Ve hře jsou tři typy karet: pěchota, kavalerie a dělostřelectvo. *Sada karet* je definována jako trojice karet stejného typu nebo jedna karta od každého typu. Pokud má hráč na ruce více než 4 karty, musí měnit. S každou provedenou směnou se zvyšuje počet posil, které hráč za sadu karet získá.

Všechny posily získané v této fázi hry musí hráč okamžitě rozmístit na svá uzemí.

Boj

Hráč si zvolí své uzemí, ze kterého chce na soupeřovo území zaútočit. K útoku může použít 1, 2 nebo 3 jednotky, přičemž na útočícím území vždy musí alespoň 1 jednotka zůstat. Obránce může bránit 1 nebo 2 jednotkami. Oba hráči hodí jednou kostkou za každou svou jednotku, hodnoty na kostkách se seřadí sestupně. Souboje se vyhodnocují pořadě, útočník vyhraje, pokud je hodnota na jeho kostce ostře větší než obráncova. Poražená jednotka je vyřazena ze hry. Hráč může útok opakovat, nebo může zaútočit na jiné území.

Pokud útočník území v tomto souboji dobyl, všechny jednotky použité k útoku se přesunou na toto území. Navíc sem může útočník přesunout libovolné množství dalších jednotek z původního území.

Pokud hráč získá všechna území soupeře, vyřadí ho tím ze hry a získává všechny jeho karty.

Posílení pozice

Když hráč skončí se všemi útoky, může přesunout libovolný počet jednotek z jednoho svého území na druhé. Mezi těmito územími musí existovat cesta opět po vlastních územích.

Pokud hráč v tomto kole dobyl alespoň jedno území, získáva jednu kartu. Tím jeho kolo končí.

Konec hry

Hra končí, pokud některý hráč získal všechna území na hracím plánu.

1.2 Pravděpodobnost

Pravděpodobnost hraje v Risku důležitou roli a její pochopení je součástí úspěchu. Tabulka 1.1 (vygenerovaná pomocí nástroje *Battle Simulator*, viz 3.9) ukazuje pravděpodobnost výhry útočníka v případě, že útočí A jednotek a brání D. Je vidět, že pokud jsou armády vyrovnané, tak při malém počtu jednotek má navrch obránce, přeci jen mu stačí pouze dorovnat hod kostkou útočníka. Ovšem při vetším počtu jednotek, začíná hrát ve prospěch útočníka fakt, že hází 3 kostkami. Jeho pravděpodobnost na vítězství dosáhne 50% již při 5 jednotkách na každé straně, v průměrném případě však útočníkovi zbyde jen 1.64 jednotky. Pokud by se spolu utkalo 100 jednotek, pravěpodobnost, že vyhraje útočník se blíží 85%.

Tabulka 1.1: tabluka pravěpodobností výhry útočníka

$A\backslash D$	1	2	3	4	5	6	7	8	9	10
1	0.419	0.105	0.027	0.007	0.002	0.000	0.000	0.000	0.000	0.000
2	0.751	0.363	0.209	0.091	0.048	0.021	0.012	0.005	0.002	0.001
3	0.917	0.656	0.468	0.315	0.204	0.136	0.083	0.052	0.033	0.021
4	0.972	0.785	0.644	0.476	0.360	0.251	0.181	0.122	0.086	0.057
5	0.990	0.890	0.771	0.638	0.510	0.399	0.297	0.226	0.162	0.118
6	0.997	0.935	0.855	0.745	0.638	0.518	0.425	0.329	0.258	0.192
7	0.999	0.967	0.911	0.834	0.737	0.642	0.535	0.444	0.355	0.287
9	1.000	0.980	0.946	0.887	0.819	0.732	0.643	0.545	0.463	0.378
9	1.000	0.990	0.967	0.930	0.875	0.810	0.727	0.647	0.558	0.479
10	1.000	0.995	0.982	0.953	0.915	0.862	0.799	0.725	0.650	0.565

1.3 Strategie a taktika

Risk je strategická hra, a jako taková vyžaduje od hráčů, pokud chtějí uspět, kvalitní strategická a taktická a rozhodnutní. Mezi strategií a taktikou je ovšem určitý rozdíl.

Taktika

Taktika popisuje jednorázovou jasně definovanou akci, která přináší hráči okamžitý zisk, nebo alespoň posílení současné pozice. Účinek taktiky lze dopředu dobře spočítat. Příkladem z šachu je třeba vidlička[1] nebo navázání jedné figury k obraně jiné, vyšší hodnoty. V Risku se taktická rozhodnutí nejčastěji týkají toho, které území dobýt a kolik k tomu použít jednotek, kolik jich zanechat na dobytém uzemí, nebo jak rozdělit posily pro připravení co nejlepší pozice.

Strategie

Proti tomu *strategie* je dlouhodobý plán, jehož okamžitý přínost lze jen obtížně vyčíslit. Nicméně pokud je správný, často rozhoduje hru. Analogií, opět z šachu, může být snaha získat převahu ve středu šachovnice nebo třeba pokus vyměnit „špatného“ střelce za soupeřova „dobrého“ (jejich nominální hodnota je stejná, ale „špatný“ střelec má výhled zakrytý linií pešců).

V Risku se za strategii považuje například rozhodnutí, zda-li preferovat zabírání kontinentů nebo většího počtu území, přičemž oba postupy mohou dávat stejný zisk v počtu jednotek. V případě kontinentů, je strategickým rozhodnutím i pořadí, v jakém je hráč zabírá.

Celkovou strategii určuje i povaha hráče. Agresivní hráč se bude snažit zabírat velký počet území a riskovat jejich případnou ztrátu, kdežto defenzivně laděný hráč bude svá již dobytá území opevňovat a vyčkávat na akci soupeřů.

Strategie však nesmí být pouze rigidní posloupností kroků, hráč by měl být schopen svou strategii — v závislosti na vzniklé situaci — vhodně měnit.

[1] situace, kdy figura (nejčastěji jezdec, dáma nebo pěšec ohrožuje dvě soupeřovy figury naráz, přičemž hrozbu nelze nijak odvrátit)

Diplomacie

Nedílnou součástní hry Risk je bezesporu diplomacie. Schopnost uzavírat aliance a dohody — a ve vhodnou chvíli je rušit — je pro úspěch ve hře možná stejně důležitá jako strategická a taktická rozhodnutí. Nicméně toto je relevantní pouze pro lidské hráče, a ideálně pokud hrají v osobním kontaktu, proto se diplomacií dále zabvat nebudeme, v knize Ehsana Honary [2] jsou jí věnovány celé dvě kapitoly.

V programu je pro hráče k dispozici chat, takže se mohou libovolně dohadovat. Počítačem řízení hráči žádné dohody neuzavírají.

1.4 Základní strategie a taktiky hry Risk

Zde jsou uvedeny některé základní postupy při hraní Risku.

1.4.1 Obecné rady

Ze začátku hry je vhodné soustředit se na získání některého malého kontinentu, který se dobře brání. Stabilní bonus za jeho obsazení je velkou výhodou do začátku hry. Vhodnými kandidáty jsou *Austrálie* resp. *Jižní Amerika*, které jsou ke zbytku světa připojeny přes 1 resp. 2 území.

Další radou je snažit se, aby jednotky byly vždy umístěné na hraničních území a přispívaly tak obraně nebo vytvářely tlak na soupeře.

1.4.2 Zamezení bonusů z kontinetnů

Účinnou taktikou je zabrání jednoho soupeřova území na kontinentu, který jinak celý vlastní. Účelem je zamezit, aby soupeř za tento kontinent získal bonus. Bonus se získává vždy na začátku tahu a hráč musí kontrolovat všechna území na daném kontinentu.

1.4.3 Zablokování slabého hráče

Když se hra dostane do fáze, kdy jednomu hráči zbývá jen několik území a je tedy těsně před vyřazením, tak pokud lze všecha území tohoto hráče bezpečně dobýt a získat tak jeho karty, pak je je to samozřejmě správný postup. Pokud jsou však jeho území roztroušená, pak je dobrou strategií izolovat jedno jeho území mezi svými a zajistit, že nikdo jiný jej nebude moct snadno dobýt. Když ostatní hráči získají zbylá uzemí tohoto hráče, je už snadné porazit jeho poslední území a získat odměnu.

Pochopitlně nevýhodou této strategie je, že pokud by se izolovanému hráči podařila získat karta do sady a směnou by obdržel větší množství posil, může kolem sebe nadělat značné škody.

1.4.4 Snadné získaní karty

V pozdější fázi hry může být obtížné v rámci jednoho tahu dobýt nějaké území a získat tím nárok na kartu. Karty v této fázi hry už dávájí nezanedbatelné množství jednotek a hráč, kterému se daří karty získávat pravidelně, má bezesporu výhodu.

Tato taktika funguje tak, že skupina několika hráčů využívá jedno území, které každý hráč ve svém tahu dobyde, získá tak kartu za toto kolo, ale na území nechá jen povinnou 1 jednotku.

Hráči tak mohou činit zcela samovolně, protože se jim to prostě hodí. Nebo se na této taktice mohou spolu domluvit, pak už jen záleží na tom, kdo první domluvu poruší a zabere územe pro sebe natrvalo.

1.5 Varianty hry

Hra Risk existuje v řadě variantách. Obvykle se liší v rozložení nebo témetickém zaměření hracího plánu. Dále existují verze hry pro 2 hráče, hry s různými úkoly nebo neutrálními hráči.

1.5.1 Hodnota odměn

Velmi populární modifikací je, že odměny získané směnou karet progresivně nerostou, nýbrž každá sada má fixně danou hodontu. Hra v této variantě je řádově pomalejší a ztrácí určitou dynamiku, na druhou stranu získá charakter opravdu strategické hry, kde se dobrá pozice nedá snadno prorazit obrovským množstvím právě získaných vojáků. Výběr této varianty není v aplikaci podporován, odměny mají prograsivní charakter, tak jak je to uvedeno v Pravidlech.

1.5.2 Rozdělení počátečních území

Tato modifikace pravidel určuje, jestli si počáteční území rozdělí hráči (jak je popsáno v 1.1.1), nebo zda-li jim je přidělí server. Výběr toho pravidla je v aplikaci implementován (viz 2.1.1)

2. Uživatelská dokumentace

2.1 Instalace a spuštění

MS Windows

Pro spuštění programu je třeba operační systém Windows 7 (64-bit). Program není nutné instalovat, v kořenovém adresáři projektu jsou připravené spustitelné soubory `client.exe` a `server.exe`, spolu se všemi vyžadovanými dynamickými knihovnami. Stačí pouze překopírovat adresář `risk/` z přiloženého CD na lokální disk a zajistit, aby zde měl program právo pro zápis.

Linux

Pro spuštění je potřeba program nejprve přeložit, stačí použít program `make`, podrobný popis je v sekci 3.2.

V nálsedujícím textu jsou jména spustitelných souborů uváděna bez přípony .exe, všechny příklady nicméně platí pro Windows i Linux.

2.1.1 Server

Server musí běžet ještě před tím, než je spuštěn klient. Server může využít následujích několik voleb (argumentů příkazového řádku):

`--port`
: číslo TCP portu, na kterém bude server dostupný, výchozí hodnota je 50123

`--num-players`
: počet hráčů potřebných ke hře, výchozní hodnota je 3

`--random-areas`
: server rozdělí počáteční území hráčům náhodně

`--debug`
: spustí server s speciální debug konzolí, více v sekci 2.4

Příkladem spuštění serveru může být příkaz:

```
./server --port 12345 --num-players 6 --random-areas --debug
```

2.1.2 Klient

Klientská aplikace má také k dispozici řadu voleb:

`--host`
: adresa nebo doménové jméno serveru, výchozí hodnota je *localhost*

`--port`
: číslo TCP portu, na kterém server poslouchá, výchozí hodnota je 50123

`--username`
: uživatelské jméno hráče, který hraje s tímto klientem

`--gui`
: přepínač určující, zda-li bude spuštěno grafické rozhraní

`--record`
: přepínač určující, jestli se má hra nahrávat pro pozdější možnost přehrání

`--replay`
: přehraje dříve nahranou hru, argumentem je cesta k souboru se záznamem

Klienta lze sputit třeba takto:

```
client --host u-pl5.ms.mff.cuni.cz --username hrac1 --gui
```

V klient i server mají volbu `--help`, která zobrazí přehled všech voleb dostupných pro daný program. Většina voleb má krátkou i dlouho variantu. Dlouhým volbám je možné předávat argumenty v notaci s mezerou nebo znakem =, tedy například je přípustné jak `--username hrac1`, tak `--username=hrac1`.

Spuštění hry

Typický scénář, jak spustit hru na lokálním počítači, je nejprve spustit server s volbou `--random-areas` a nechat ho, aby rozdělil počáteční území náhodně, případně lze přidat volbu pro počet hráčů, výchozí počet je 3. A poté spustit několik klientů, každého s jiným uživatelským jménem. A nakonec spustit jednoho klienta s grafickým rozhraním. Jakmile je k serveru přihlášen dostatečný počet klientů, začne hra.

Chybová hlášení

Může se stát, že se klientovi nepodaří k serveru připojit, v takovém případě server zpravidla buď neběží vůbec, nebo není po síti dostupný.

Dále může nastat, že klientovi se sice podaří k serveru připojit, nicméně server spojení odmítne. Může pro to mít dva důvody. Uživatelské jméno, které se uživatel snaží použít, je již zabrané jiným hráčem. Nebo je počet přihlášených hráčů na serveru již maximální.

Ve všech případech klient uživatele o nastalé situaci informuje a definovaně se ukončí.

2.2 Ovládání klienta

2.2.1 Počítačový hráč

Pokud je klient spuštěn bez volby `--gui`, automaticky bude hrát jako počítačem řízený hráč podle strategie Standard (3.5.1). Klient vypisuje do konzole řadu informací o stavu hry, už však není možné ho dál ovládat. Parametry počítačové strategie lze ovšem nastavovat prostřednictvím konfiguračního souboru, podrobný popis souboru je sekci 3.5.2.

Tato strategie však nemá dobře implementovanou první fázi hry, území vybírá náhodně, proto je lepší, aby uživatel — pokud si chce opravdu zahrát — spouštěl server s volbou `--random-areas` (zkráceně jen `-r`). V dalších fázích už strategie hrát umí.

2.2.2 Grafické rozhraní

Pokud je při startu klienta volba `--gui` zapnuta, spustí se grafické uživatelské rozhraní.

Dominantím prvekem je herní plán – mapa světa, která zobrazuje aktuální rozdělení světa mezi hráče. Nad každým územím je také uveden příslušný počet jednotek. Do mapy lze levým tlačítkem myši klikat a získat tak bližší informace o jednotlivých územích.

Pod mapou světa jsou dvě záložky textových oken. Jedno zobrazuje logovací zprávy a druhé slouží jako chat mezi hráči.

Mezi mapou a logovacími okny je navíc ještě lišta, která ukazuje pořadí hráčů během kola a aktuální výši odměn za směnu karet.

Na pravé straně se nachází hlavní ovládací panel, ten je rozdělen do 3 záložek. Záložka *General* zobrazuje informace o lokálním hráči, vybraném území a znázorňuje hody kostkou. Dále obsahuje tlačítka na provádění akcí. V záložce *Players* jsou informace o ostatních hráčích, o počtu jejich území, jednotek a kontinentů. V poslední záložce *Debug* je jediné tlačítko *Evaluate*, které zobrazí ohodnocení stavu hry pomocí evaluátoru Standard (viz 3.5.3) pro všechny aktivní hráče.

Implicitně je zapnuto, že hru ovládá uživatel. Toto lze změnit zaškrtnuním pole *Autoplay* v záložce *General* na ovládacím panelu. Potom bude místo uživatele hrát počítač, a to podle stejné strategie, jakou by použil klient bez GUI.

V následujícím textu jsou popsány postupy, když hru ovládá uživatel. Hra probíhá ve třech fázích.

První fáze

V první fázi se rozebírají uzemí na herní plánu. V horní části ovládacního panelu je textové pole, které indikuje, kdo je právě na tahu. Když je na tahu hráč (uživatel), kliknutním myši v mapě vybere některé zatím neobsazené území, svou volbu potvrdí a tím jeho tah automaticky skončí.

Druhá fáze

Ve druhé fázi se posilují zabraná území. Pokud je hráč na tahu, vybere své území, které plánuje posílit a v objevivším se dialogu vybere počet jednotek, které chce na dané území poslat. Ze strategického hlediska je dobré posílat vždy jen jednu jednotku, aby měl hráč možnost regovat na akce soupeřů, v principu jich ale může poslat více. Na ovládacím panelu je zobrazen počet zbývajících posil. Odesláním jednotek tah hráče opět automaticky končí.

Třetí fáze - vlastní hra

Ve třetí a poslední fázi se odehrává vlastní hra. Na začátku kola dostane hráč tolik posil, na které má — na základě okupovaných území a kontinentů — nárok.

Následně má hráč k dispozici sadu akcí: posílení území, přesun jednotek z jednoho území na druhé, útok na nepřátelské území a výměnu karet. Uživatel může tyto akce provádět v jakémkoliv pořadí a může je libovolně opakovat. V pravidlech je sice přesně popsáno, v jakém pořadí mají probíhat kroky v poslední fázi, nicméně pro snadnější ladění počítačových strategií je lepší, pokud má uživatel větší flexibilitu.

Zde jsou sepsané jednotlivé akce, v závorce jsou anglické názvy, tak jak jsou uvedeny v GUI a podržené písmeno označuje klávesovou zkratku, kterou lze pro vybrání akce také použít:

Útok (Attack) Po označení této akce vybere hráč tahem myši dvojici území. Z území, nad kterým tah myši začal, se bude útočit na území, nad kterým tah myši skončil. Pokud je vybrána validní dvojice, tj. území spolu sousedí, jsou nepřátelská a útočník má dostatek jednotek pro útok, zobrazí se dialog útoku. Hráč má možnost vybrat, zda-li bude útočit 1, 2 nebo 3 jednotkami, případně zvolit, aby se útok opakoval, dokud není území dobyto. Informace o probíhajícím útoku se zobrazují v textovém poli uprostřed dialogu. Útok lze kdykoliv přerušit uzavřením tohoto dialogu.

Posílení území(Reinforce) Pokud má hráč nějaké volné posily, může zvolit tuto akci. Proces této akce je úplně stejný, jako při posilování území ve druhé fázi hry.

Přesun jednotek (Transfer) Podobně jako v případě útoku vybere hráč dvojici území. Obě uzemí musí hráčovi patřit a musí mezi nimi existovat cesta po vlastních územích. Pokud jsou tyto podmínky splněny, zobrazí se dialog, pomocí nehož hráč vybere počet jednotek k přesunu.

Výměna karet (Exchange Cards) Tato akce zobrazí dialog pro výměnu karet. Pokud má hráč nějakou kompletní sadu, může směnit za bonusové jednotky.

Konec kola (End round) Touto akcí ukončí hráč svůj tah.

2.3 Přehrávání záznamů her

Program poskytuje možnost nahrát si libovolnou partii a tu si později ze záznamu přehrát. Nahrávání není automatické a je nutné ho zapnout při spouštění klienta volbou `--record`. To je z toho důvodu, aby když na lokálním počítači hraje 6 klientů zároveň, nevznikalo veliké množství duplicitních záznamů.

Nahrané hry se ukládají do adresáře `replays/`. Jméno souboru se skládá z data a času hry a přípony `*.rep`.

Pokud chce uživatel záznam přehrát, spustí klient s volbou `--replay`, jako argument této volbě předá soubor se záznamem hry, např:

```
./client --gui --replay replays/2013-05-16_20-43-49.rep
```

Záznam lze přehrávat v režimu s grafickým rozhraním i bez něj. V případě, že je GUI povolené, je možné záznam pozastavovat a znovu pouštět pomocí ovládacích prvků na panelu v pravé části obrazovky.

Pro přehrání záznamu není nutné, aby bežel server.

2.4 Debug režim serveru

Server spuštěný v ladicím režimu (s volbou `--debug`), má aktivovanou tzv. *debug konzoli*. Jedná se o běžnou konzoli na standardním vstupu, která přijímá příkazy, jimiž lze měnit stav hry v jejím průběhu. Lze použít následující sadu příkazů:

`SET_OWNER` ⟨`new-owner-id`⟩ ⟨`area1`⟩ ... ⟨`areaN`⟩
: danému seznamu území nastaví nového vlastníka

`SET_UNITS` ⟨`area-id`⟩ ⟨`num-units`⟩
: na území s daným ID nastaví počet jenotek na hodnotu ⟨`num-units`⟩

`DESTROY_PLAYER` ⟨`player-id`⟩ ⟨`new-owner-id`⟩
: téměř zničí hráče ⟨`player-id`⟩, všem jeho územím, až na jedno, nastaví jako nového vlastníka hráče ⟨`new-owen-id`⟩

`XCHG_OWNER` ⟨`area1-id`⟩ ⟨`area2-id`⟩
: vymění vlastníky uvedených území

Pokud je do konzole zadán jiný příkaz, který není uveden v seznamu výše, je poslán klientům, jako by to byl příkaz samotného komunikačního protokolu.

3. Implementace

Tato kapitola popisuje aplikaci z pohledu programátora. Nejprve jsou zde rozebrány technické záležitosti, jako použité knihovny a způsob kompilace programu. Další části už se zabývají vlastním návrhem aplikace. Podrobněji jsou popsána některá důležitá rozhraní. Závěr se věnuje dvěma nástrojům, které byly pro potřeby projektu vytvořeny.

3.1 Volba technologií

3.1.1 Jazyk

Jako programovací jazyk byl vybrán C++, a to ve standardu C++11. Autorovi přišlo zajímavé prozkoumat, jak je na tom implementace této nové normy napříč platformami. Navíc C++11, díky nové sadě vlastností a rozšíření standardní knihovny, dělá z C++ příjemný moderní programovací jazyk při zachování jeho efektivity a typové bezpečnosti.

Ovšem při rozhodování, které vlastnosti z C++ použít, bylo nutné brát v potaz požadavek na multiplatformnost. Také z pochopitelných důvodů nebylo možné použít nejnovější verze různých překladačů. Proto byly v projektu použity pouze následující vlastnosti: smart pointers, typová inference, lambda funkce, rvalue reference a move sémantika. Variadic templates jsou použité pouze na Linuxu.

3.1.2 Knihovny

V programu je využívána knihovna Boost (`http://www.boost.org/`). Modul Boost Asio je použit pro implementaci síťové komunikace. Modul Boost ProgramOptions je použit pro zpracování argumentů programu. Dále je knihovna využita např. pro generování náhodných čísel a další drobné pomocné operace.

3.1.3 Grafický framework

Jako grafický framework byla zvolena knihovna Qt (`http://qt-project.org/`). Hlavními důvody jsou tyto: knihovna je napsaná v jazyce C++, je mezi uživateli poměrně rozšířená a má velmi dobrou dokumentaci. Navíc je multiplatfomní. Jako nástroj pro sestavní aplikace používá knihovna Qt program qmake, který umožňuje z jednoho projektového souboru vygenerovat jak Makefile na unixových systémech, tak i projekt pro Visual Studio v MS Windows. Součástí této knihovny je i modul QtXML, který slouží k parsování a generování souborů ve formátu XML, což se programu využívá pro práci s konfiguračními soubory.

3.2 Kompilace a sestavení programu

Linux

Pro kompilaci na Linuxu lze použít standardní program make. V kořenovém adresáři projektu je ručně psaný Makefile, který sestaví klienta resp. server pomocí

příkazů `make client` resp. `make server`. Pokud není target uveden, sestaví se oba programy. K této operaci jsou použity Makefile soubory vygenerované programem qmake.

Pro korektní překlad je potřeba použít kompilátor, který podporuje výše uvedené vlastnosti jazyka C++11. Otestovány jsou dva překladače, clang 3.2 (`http://clang.llvm.org/`) a gcc (`http://gcc.gnu.org/`) ve verzi minimálně 4.6.

Clang je součástní projektu LLVM. Jedná se o poměrně nový překladač, nicméně v kvalitě výsledného programu se známějšímu gcc vyrovná. Navíc oproti gcc poskytuje mnohem lepší diagnostiku chyb. Také je mezi běžně dostupnými překladači asi nejdál v implementaci nového standardu C++. Proto byl clang pro tento projekt zvolen jako výchozí překladač.

Pro změnu překladače na gcc stačí v souboru Makefile změnit hodnotu proměnné *cxx* na *gcc*.

Předpokládá se, že uživatel má ve svém systému naistalované knihovny, na nichž projekt závisí. Potřeba je knihovna boost ve verzi alespoň 1.52 a framework Qt verze 4.8.

MS Windows

Pro překlad na systému MS Windows jsou v adresáři `projects/msvc2012` připraveny projekty pro Visual Studio 2012 (tato verze je potřeba opět kvůli C++11). Knihovny jsou vyžadovány stejné jako v případě linuxu, je však možné, že uživatel bude muset v projektech upravit cesty k nim.

Pro Windows však jsou exe soubory již vytvořené (viz 2.1), takže není nutné celý projekt kompilovat.

3.3 Návrh aplikace

Obě aplikace, klient i server, společně sdílejí zdrojové kódy. Pro přehlednost jsou zdrojové kódy rozděleny do řady modulů, které přesněji vymezují jejich fukčnost a dávají projektu logickou strukturu. Každý modul má svůj vlastní jmenný prostor a příslušné zdrojové kódy jsou v samostatném adresáři: `src/<module-name>`.

Zde je uveden přehled všech modulů s krátkým popisem a v následujícm textu jsou rozebrány užší vazby mezi nimi.

ai
: obsahuje zdrojové kódy počítačových strategií a rozhraní skrze která lze další strategie definovat

client
: tento modul obsahuje síťový kód a logiku klienta, zde je main funkce pro klientskou aplikaci

game
: zde jsou datové struktury reprezentující aktuální stavy hry, tento modul sdílejí klient i server

gui
: zde je kompletní implementace grafického rozhraní

protocol
: v tomto modulu je abstrahován komunikační protokol, je zde uveden úplný seznam příkazů, jsou tu bázové typy pro implementaci protokolu na serveru i klientu a v neposlední řadě třída `ProtocolProcessor`

server
: obsahuje kompletní implementaci serveru, zde je main funce pro server

utils
: zde jsou pomocné typy a funce, různé konverze, časovač, generátor náhodných čísel, generická továrna nebo bázový typ pro stavové objekty

view
: tento modul abstrahuje způsoby zobrazení aplikace

3.3.1 Model klient-server

Implementace vlastní hry je založena na modelu klient-server. Tento přístup je zvolen především proto, aby bylo možné spustit kdekoliv v Internetu samostatně server, ke kterému by se mohli připojovat i jiní klienti, než ten implementovaný v rámci této práce.

Server

Server je v systému jedinou hlavní autoritou a všichni k němu připojení klienti jsou si navzájem rovni. Server je stavový, jeho počáteční stav - inicializace slouží k registraci klientů a potvrzení jejich připravenosti ke hře. V tomto stavu také server přiděluje hráčům ID, kterými se po zbytek hry budou identifikovat. Zbylé stavy odpovídají jednotlivým fázím hry.

Server řídí logiku celé hry, určuje pořadí hráčů a oznamuje jim, kdy jsou na tahu, dále zprostředkovává boje, generuje hody kostkou a přiděluje karty.

Server přijímá zprávy od klientů, aktualizuje svůj interní stav hry a přeposílá zprávy ostatním klientům. Server kontroluje přijaté zprávy, zda-li odpovídají platným operacím ve hře, takže klienti se mohou spolehnout na validitu dat, která přicházejí ze serveru a nemusejí je znovu kontrolovat.

Komunikační protokol

Komunikační protokol je navržen jako textový. Oproti binární variantě se lépe ladí, názvy příkazů poměrně přesně popisují akci, ktrou dané příkazy provádějí. Protokol je tak i snadno pochopitelný pro případné další implementátory. V aplikaci se nepředpokládá nějaký větší datový tok, proto ani nevadí větší objem dat ve srovnání s binární reprezentací.

Protokol je navržen pro běh nad transportním protokolem TCP. V principu je možné provozovat ho i nad UDP, ovšem protokol jako takový neposkytuje žádnou kontrolu integrity dat ani potvrzování přijatých zpráv. Kompletní text protokolu je uveden v Příloze B.

ProtocolProcessor

`ProtocolProcessor` je typ ze jmenného prostoru protocol, který slouží ke zpracování příkazů protokolu. Uživatelské třídy od něj buď podědí nebo jej přidají jako datovou položku, poté si pomocí metody `registerCallback` zaregistrují příkazy, které má procesor umět zpracovat. Předává se jméno příkazu a funktor, který se má zavolat po přijetí příkazu.

`ProtocolProcessor` je použit jednou v klientu ve třídě `ClientProtocol` a podruhé na server ve třídě `ServerProtocol`. Sada příkazů, které rozpoznávájí klient a server se liší, proto mají každy vlastní implementaci.

`ProtocolProcessor` je také použit k implementaci debug režimu serveru, když funguje jako bázový typ třídy `CommandProcessor`.

3.4 Implementace klienta

Vlastní klientská aplikace se sestává z několika součástí. Základem je kód v modulu client, zde se vytváří aplikace, je zde vyřešena komunikace se serverem a řízení klienta. Dále je to uchování stavu hry, které se provádí v modulu game. Zde jsou uloženy informace o hráčích, o stavu mapy a průběhu hry. Nezbytnou součástí je i zobrazení výstupů programu, jež je v modulu view a gui.

Aplikace je navržena tak, aby každá ze zmíněných součástí měla jasně definovaný účel, zároveň aby však nebyla v programu napevno nakódovaná a umožňovala snadnou výměnu za jinou svou implementaci. Pro tento požadavek se výborně hodí návrhový vzor Model-View-Controller.

3.4.1 Vzor Model-View-Controller

Obecně je Model-View-Controller (MVC) návrhový vzor, který odděluje uchování dat (model), operace nad nimi (controller) a způsob jejich zobrazní (view). Cílem je rozpoutat vazby mezi nimi a umožnit výměnu jednotlivých komponent, aniž by bylo nutné pokaždé procházet celý systém. Tento vzor není nijak striktní ohledně konkrétních tříd nebo vztahů mezi nimi, nicméně nabízí dobrý konceptuální pohled na problém.

V našem konkrétním případě je modelem modul game, základem controlleru je třída `GameController` a jako view funguje třída `AppView`.

Měnit model není pro potřeby projektu žádoucí, ovšem změnou kontroleru lze vybrat, zda-li bude hru ovládat hráč, nebo jestli bude hrát počítač. Podobně výběrem view lze určit, jestli bude hra zobrazena v grafickém rozhraní nebo bude mít jen konzolový výstup.

Volba ovládání

Volba způsobu řízení hry se provádí skrze výběr implementace rozhraní `GameController`. Toto rozhraní obsahuje metody reprezentující situace, ve kterých se vyžaduje, aby klient aktivně reagoval. Především jde o chování v jednotlivých fázích hry. Dále je to výzva k obraně napadeného uzemí, výzva k poslání jednotek na nově dobyté území a oznámení o nově získaných posilách.

V programu jsou následující implementace:

`client::DefaultController`
: slouží jako ukázková implementace kontroleru, je vybrána pokud žádná lepší není k dispozici, přesto jde o plnohodnotnou implementeci, která se sice rozhoduje náhodně, nicméně korektně

`ai::ComputerController`
: toto je abstraktní třída, která slouží jako základ pro ovládání hry počítačem (vice v sekci o počítačových strategiích3.5)

`client::GuiController`
: tato třída slouží pro ovládní hry skrze grafické uživatelské rozhraní

Volba pohledu

Způsob zobrazení lze vybrat pomocí potomků třídy `AppView`. V programu jsou dostupné dvě implementace – třída `NullView` má většinu metod definovaných jako prázdné, v několik případech vypisuje informace do konzole. třída `GuiView` předává aktualizace pohledu do grafického rozhraní.

Ne všechny kombinace pohledu a ovládání však dávají smysl. Klient spuštěný s grafickým rozhraním může mít připojen jak `GuiController`, tak některý `ComperController`. Klient bez grafického rozhraní pochopitelně `GuiController` použít nesmí.

3.5 Vlastní implementace počítačového hráče

Jak bylo uvedeno výše, počítačem ovládané strategie jsou potomky abstraktní třídy `ComputerController`. Tato třída pomocí nové sady virtuálních metod implementuje metody svého předka (`GameController`). Tím je zaručeno, že se všechny nové počítačové strategie budou chovat korektně, alespoň ve smyslu pořadí volání svých metod. Tyto nové metody jsou čistě virtuální a skrze jejich implementaci se ovlivňuje chování odvozených počítačových strategií.

V programu jsou implementovány dvě strategie. Stragegie Test (`TestStrategy`) slouží spíše jako ukázka použití rozhraní. Přestože se chová korektně, tak hraje naprosto náhodně a nemá tedy další význam. Lépe se chová strategie Standard.

3.5.1 Strategie Standard

Strategie Standard (`StandardStrategy`) ke svému rozhodování používá ohodnocení stavu hry a pravděpodobnostní tabulku úspěšnosti bitev. Navíc lze parametry této stratige nastavovovat skrze konfigurační soubor (`data/ai/standard.xml`).

Pro vývoj této strategie nebyl použit žádný algoritmus pro prohledávání stromu hry (Minimax, resp. Alpha-beta pruning). Přeci jen se může hrát až v 6 hráčích, navíc značný vliv má náhoda, ať už při hodu kostkou či při rozhodování hráčů. Takový herní strom by byl obrovský a vůbec není jasné, jak ho kvalifikovaně prořezávat. Místo toho se strategie snaží získat pro sebe v každém kole pokud možno co nejlepší pozici a tu udržet.

3.5.2 Konfigurace strategie

Konfigurační soubor je XML soubor, jehož prostřendictvím lze nastavovat některé parametry strategie.

Za prvé je možné určit statickou hodnotu konkrétních území, ta se pak použije jako základ při ohodnocování pozice. Skrze element `area` se určí ID konkréntního území a jeho hodnota, např:

```
<area id="0" value="1.3"/>       <!-- alaska -->
```

Podobně to funguje při nastavení statické hodnoty kontinentu, pro tu se využívá element `continent`:

```
<continent id="0" value="5"/>    <!-- north america -->
```

V obou případech platí, že pokud není hodnota nastavena, používá se výchozí 1.0.

Dále je možné nastavovat další parametry skrze element `param`. Jeho formát je následující:

```
<param key="<name-of-param>" value="<value>"/>
```

Atribut `key` obsahuje jméno parametru a `value` je vždy desetinné číslo typu *double*. Zde je uveden seznam přípustných parametrů:

`area_coeff`
: představuje modifikátor váhy jednoho území, čím větší hodnota, tím víc si bude strategie cenit území a tím bude agresivnější.

`continent_coeff`
: modifikátor váhy kontinentů, čím větší bude hodnota v poměru k `area_coeff`, tím víc si bude strategie cenit kontinentů a nebude se snažit tolik expandovat

`component_size`
: určuje, jak moc si strategie cení souvislých komponent

`army_value_modifier_size`
: koeficient, kterým se násobí počet jednotek na území, slouží spíše pro snížení relativního vlivu počtu jednotek na celkové ohodnocení stavu

`border_area_coeff`
: koeficient, který se použije při počítání vlivu jednotek, které stojí na hraničním území, funguje jako bonus

`inner_area_coeff`
: podobně jako v předchozím případe, pouze má vliv na vnitřních územích (takových, které nemají za souseda žádné nepřátelské území), funguje jako penalizace, aby to donutilo jednotky zbytečně se na takových územích nehromadit

`min_probability_attack`
: nejměnší pravděpodbnost výhry, jakou musí mít strategie při útoku, čím nižší hodnota bude, tím odvážněji bude strategie hrát

`max_loss_ratio_attack`
: maximální očekávaný poměr přeživších jednotek po útoku, nastavení této hodnoty např. na 0.75 zamezí strategii pouštět se do útoku, o kterém se předpokládá, že přežije méně než 75% původních jednotek

K parametrům se ze zdrojového kódu přistupuje skrze třídu `EvalConstants`. Pokud je potřeba přidat nový parametr, stačí přidat další členskou položky třídy a zaregistrovat její jméno v metodě `initConstMap`. Parser se pak sám postará, aby byla její hodnota načtena.

3.5.3 Ohodnocování pozice

Pro ohodnocování pozice je připravena základní bázová třída `PositionEvaluator`, která bere ukazatel na typ `WorldMap` a ID hráče, pro něhož se má ohodnocení provést. Vlastní ohodnocení pak probíhá v potomcích této třídy v implementaci abstraktní metody `evaluate`. Výstupem evaluátoru je nezáporné racionální číslo.

Pro strategii Standard je napsán evaluátor `StandardEvaluator`, který ohodnocuje dohromady jak území, tak armády. Ohodnocení vychází ze základních hodnot vlastněných území a kontinetnů, na ty jsou pak použity modifikátory z konfigurčního souboru. V potaz se tedy bere váha území a kontinentů, preference souvislosti, pozice armád na hraničních a vnitřních území.

3.5.4 Pravděpodobnostní tabulka

Pro dobré rozhodování potřebuje strategie vědět, jakou má pravděpodobnost, že vyhraje souboj s daným počtem jednotek. Důležitý je i odhad, kolik jednotek během souboje padne.

Pro tento výpočet byl vytvořen program *Battle Simulator* 3.9. Vygenerované tabulky jsou uloženy v adresáři `data/ai` v souborech s příponou `.prob`.

Ve zdrojovém kódu se pak k tabulkám přistupuje skrze instance třídy `ProbabilityTable`. Ta má na sobě metodu `get`, která pro daný počet útočníků a obránců vrátí typ `ProbabilityRecord`, který už přímo obsahuje data z tabulky.

3.6 Grafické rozhraní

Jak je uvedeno na začátku kapitoly, grafické rozhraní je napsané nad knihovnou Qt. Všechny dialogy a widgety[1] jsou ručně psané a jejich kompletní zdrojové kódy jsou v modulu gui.

Jediným problémem bylo, jak zajist správnou aktualizaci grafického rozhraní. Grafické rozhraní totiž beží ve svém vlastním vlákně (*gui thread*), a zpracování síťového protokolu beží v hlavním vlákně (*main thread*)[2]. Když by se kód z hlavního vlákna snažil aktualizovat GUI přímo voláním metod, je zřejmé, že to není správný postup, program nejspíše rovnou spadne kvůli race condition.

[1] v Qt je widget jakýkoliv ovládací prvek nebo skupina prvků, v kódu jde o potomky třídy `QWidget`

[2] podle dokumentace by dokonce gui thread měl být zároveň main thread, autor však nenašel důvod, přoč by tomu tak nutně být muselo

Naštěstí existuje v knihovně Qt systém signálů a slotů [10], což je v podstatě implementace návrhového vzoru Observer [7], která se v Qt používá především pro notifikaci událostí na ovládacích prvcích GUI. Tento systém má vlastnost, že sloty, v reakci na nějaké signál, jsou volány synchronně a to v *gui threadu*. Navíc samotné emitování signálů je *thread safe* operace.

Tato vlastnost se hodí při řešení problému s předáváním dat z hlavního vlákna. Signály a sloty mohou používat pouze potomci třídy `QObject`, proto je v modulu `gui` třída `GuiInfo`, která propojuje příslušné signály a sloty a emitování signálu zabaluje do bežných metod. Díky tomu lze signály vyvolávat i z částí kódu, které jinak na Qt nezávisí.

Kreslení mapy

Základem mapy je černobílý obrázek, který obsahuje pouze hranice území. Spolu s ním se používá ještě jeden odpovídající obrázek, ve kterém jsou všechna území vykreslena nějakou unikátní barvou. Tento obrázek slouží jako barvná maska k rozlišení jednotlivých území.

Ke každé mapě existuje konfigurační soubor ve formátu XML (ideálně vyrobený map editorem, viz 3.8). Tento soubor popisuje pro každé území jeho název, klíčovou barvu v masce, seznam sousedních území a nějaký bod (pivot), který je součástí tohoto území. V případě, že je nějaké území nesouvislé (např. Japonsko je skupina ostrovů), obsahuje seznam takových pivotů, pro každou komponentu jeden.

Když uživatel v grafickém rozhraní vybere nějaké uzemí kliknutím myši v mapě, tak bod pod myší se promítne do barevné masky a podle informací v konfiguračním souboru se vybere správné území.

Území, které patří nějakému hráči je třeba obarvit barvou tohoto hráče. K tomu se používá algoritmus záplavového vyplňování (*flood fill* [11]). Ten funguje tak, že z daného výchozího bodu vybarvuje souvislou oblast danou barvou, dokud je referenční barva stejné jako ta ve výchozím bodě. V našem případě je referenční barvou barva masky.

Flood fill je spuštěn z každého pivotního bodu, díky tomu je kompletně vybarveno i nesouvislé území.

3.7 Dokumentace zdrojových kódů

Ke zdrojovým kódům je dostupná dokumentace vygenerovaná pomocí nástroje Doxygen. Dokumentace se nachází v adresáři `doc/doxygen/html`. A lze ji případně znovu vygenerovat příkazem `make doc`.

3.8 Map Editor

Map Editor je samostatný nástroj, který byl vytvořen čistě pro usnadnění procesu výroby mapy. Jeho zdrojové kódy se nacházejí v adresáři `tools/map_editor`. Editor vychází z obrázku s barevnou maskou mapy světa a umožňuje snadno „naklikat“ seznam pivotů a sousedů pro všechna uzemí.

Editor byl použit pouze k vytvoření jediné mapy, té jež je použita ve hře. Nicméně nic nebrání tomu, aby byl využit i k vytvoření úplně nové mapy se zcela jinou topologií, než má klasická mapa světa.

3.9 Battle Simulator

Battle Simulator je jednoduchý program, který slouží k výpočtu pravděpodobnosti výhry útočníka při různých kombinacích počtu útočící a bránících jednotek. Dále také určí kolik jednotek průměrně útočníkovi zbyde, pokud souboj vyhraje.

Nástroj tyto hodnoty počítá metodou Monte Carlo, zkrátka odsimuluje souboj pro všechny dané dvojice počtů útočníků a obránců. Výsledky ukazují, že pokud se pro každou tuto dvojici provede simulace alespoň 10000x, tak se takto spočítaná pravděpodobnost liší — od té vypočítané matematickou metodou Markovových řetězců — pouze v řádu desetin procenta.

Jedná se o stochastickou simulaci, která musí dokončit každý souboj. Při skutečné hře však útočník může souboj zastavit, pokud má mimořádnou smůlu při hodu kostkou.

Vygenerované tabulky jsou *de facto* čtvercové matice, kde na pozici (a, d) je dvojice čísel $p|e$, kde a resp. d je počet útočících resp. bránících jednotek, p je pravděpodbnost, že vyhraje útočník a e průměrný počet jednotek, které útočníkovi zůstanou, pokud vyhraje.

Zdrojové kódy programu jsou v adresáři `tools/battle_simulator`, formát argumentů programu je následující:

```
simulator <max-units> <num-iterations> <output-file>
```

Například pro vygenerování tabulky `table.prob` pokrývající souboje až 20 jednotek, při 10000 opakování každého souboje, se použije příkaz:

```
./simulator 20 10000 table.prob
```

Závěr

V rámci této práce se podařilo vytvořit téměř kompletní implementaci deskové hry Risk, která může být využita jak pro hraní po síti s jinými hráči, tak proti počítačovým soupeřům.

Implementace jedné počítačové strategie je také součástí výsledného projektu. Tato strategie mj. demonstruje, jakým způsobem používat rozhraní pro tvorbu nových strategií.

Především má však uživatel k dispozici nástroj, který mu vývoj strategií usnadní. Může použít vlastnosti jako debugovací konzoli na serveru, možnost prohlížení již odehraných her nebo nastavování parametrů strategie prostřednicvím XML souboru.

Z programátorského hlediska se například podařilo skloubit framework Qt a knihovnu Boost, což zj. pro jejich odlišný způsob využívání vláken, je cenná zkušenost. Dále byla prozkoumána použitelnost nové normy jazyka C++ napříč platformami. Ukázalo se, že sice nelze použít úplně všechny nové vlastnosti jazyka, nicméně existuje spousta velmi užitečných vlastností, které se dají využít bez problému.

Další z výstupů práce je dokument s kompletním popisem použitého komunikačního protokolu. Podle tohoto textu je možné naprogramovat alternativní implementace klienta i serveru.

Možná rozšíření

Bezesporu nejzajímavějším rozšířením projektu by byla lepší implementace počítačové strategie založená na sofistikovanějším principu, než je statické ohodnocování. Některé nápady jsou rozebrány například v diplomové práci Michaela Wolfa [3].

Seznam použité literatury

[1] *RISK: Hra o dobytí světa - Návod ke hře* Hasbro, 2007. Dostupné z: `http://www.hasbrohracky.cz/download/risk_navod_cz.pdf`

[2] Honary, Eshan. *Total Diplomacy: The Art of Winning RISK*. BookSurge Publishing, 2007. ISBN 1-4196-6193-0.

[3] Wolf, Michael. *An Intelligent Artificial Player for the Game of Risk*. Diploma thesis. Darmstadt University of Technology, Department of Computer Science, 2005

[4] Osborne, Jason A. *Markov Chains for the RISK Board Game Revisited*. North Carolina State University, Raleigh

[5] Robinson, Garrett *The Strategy of Risk*. Massachusetts Institute of Technology, 2009. Dostupné z: `http://web.mit.edu/sp.268/www/risk.pdf`

[6] Alexandrescu, Andrei. *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley Professional, 2001. ISBN 0-201-70431-5.

[7] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. *Design Patters. elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. ISBN 0-201-30998-X.

[8] *C++ reference* [online]. Dostupné z: `http://en.cppreference.com/w/cpp`

[9] *Boost.Asio - reference manual* [online]. Dostupné z: `http://www.boost.org/doc/libs/1_53_0/doc/html/boost_asio.html`.

[10] *Qt - online documentation* [online].
Dostupné z: `http://qt-project.org/doc/qt-4.8/`

[11] *Flood fill* [online], poslední aktualizace 14-5-2013 [cit. 19-5-2013] Wikipedie. Dostupné z: `http://en.wikipedia.org/wiki/Flood_fill`

Příloha A: Obsah přiloženého CD

`risk/`
: adresář s kompletním projektem, daty a zdrojovými kódy

`doc/`
: zde je vysázený text práce v pdf i PostScriptu a soubor s pravidly ke hře

`risk/data/`
: zde jsou data potřebná pro běh programu

`risk/doc/`
: tento adresář obsahuje text komunikačního protokolu a dokumentaci vygenerovanou nástrojem Doxygen

`risk/libs/`
: zde je část zdrojových kódů knihovny Loki (dostupná ke knize [6])

`risk/projects/`
: projekty pro MS Visual Studio 2012 pro klienta i server

`risk/replays/`
: adresář pro ukládání záznamů her

`risk/src/`
: zdrojové kódy aplikace, jsou rozělené do modulů, každý modul bydlí ve vlastním podadresáři

`risk/tools/`
: zdrojové kódy pomocných nástrojů *Map Editor* a *Battle Simulator*

Příloha B: Komunikační protokol

RISK GAME COMMUNICATION PROTOCOL

(ver 0.2)

Jan Roztocil [honza.roztocil@gmail.com]

Robert Guth [Robert.Guth@seznam.cz]

Listopad 2012

1. UVOD

Protokol je zalozen na modelu klient server. Server je jedinou a hlavni autoritou, vsichni klienti jsou pred serverem rovnocenni (tedy neni zaveden zadny model pristupovych prav).

Server si udrzuje aktualni stav hry, informuje o nem vsechny hrace a ridi prubeh jednotilivych kol.

Klienti jsou povinni sledovat zpravy, ktere posila server a reagovat na ty, ktere se jich tykaji (nebudou k tomu totiz nijak jinak vyzvani).

Pozn. k nazvoslovi: v nasledujicim textu budeme pro oznaceni klienta pouzivat terminy "hrac" i "klient" v rovnocennem vyznamu. Stejne tak budem pouzivat termin "zprava" ci "prikaz" pro data predavana mezi klientem a serverem.

Kompletni seznam prikazu a jejich formatu je uveden na konci tohoto dokumentu, v prubehu textu se na nej budeme hojne odkazovat.

2. IDENTIFIKACE HERNICH ENTIT

Hraci se identifikuji pomoci ID, ktere obdrzi od serveru po prihlaseni (viz sekce 3.1).

Uzemi na hernim planu se oznacuji celociselnym identifikatorem (viz Appendix A).

Karty se oznacuji svym typem: pechota, kavalerie, delostrelectvo, divoka karta (viz. Appendix B).

3. INICIALIZACE

Server je po spusteni ve fazi inicializace, kdy se prihlasuji hraci a je mozna jeho konfigurace.

3.1. PRIHLASOVANI HRACU

Hraci se prihlasuji prikazem LOGIN a server odpovida potvrzovacim prikazem LOGIN-ACK.

Pokud prihlaseni probehne bez problemu, posle server hraci LOGIN-ACK OK a zaroven mu posle seznam doposud prihlasenych hracu skrze n prikazu REGISTERED-PLAYER (kde n je pocet prihlasenych hracu). Dale dosud prihlasenym hracum posle informaci o novem hraci opet prikazem REGISTERED-PLAYER.

Kdyz ma hrac potrvzen svuj login a je pripraven ke hre, posle serveru zpravu READY.

3.2. KONFIGURACE SERVERU

3.2.1 NASTAVENI PRAVIDEL

Podrobny popis pravidel se nachazi v sekci 8. Nasteveni urciteho pravidla se provadi prikazem APPLY_RULE.

Napriklad, pokud chce klient nastavit pravidlo 0 na variantu 1 posle na server:

```
APPLY_RULE 0 1
```

vyznam jednotlivych argumentu je v tabulce v seznamu prikazu v sekci 7.2.

4. PRUBEH HRY

4.1. ZAHAJENI HRY

Pokud se na serveru zaregistroval dostatecny pocet hracu a vsichni projevili vuli hrat odeslanim prikazu READY, zahaji server hru.

Server rozhodne (zpravidla nahodne) o poradi, v jakem budou hraci hrat a informuje je o tom pomoci prikazu ORDER.

Potom server informuje o poctu posil, ktere budou moci v nasledujicich dvou fazi vyuzit, prikazem REINFORCEMENTS.

Nasledne zahaji prvni fazi hry prikazem START-PHASE 0 a vyzve prvniho hrace, aby zacal svuj tah prikazem START-ROUND.

4.2. FAZE HRY

Hra ma tri faze: rozebrani pocatecnich uzemi, posilovani pocatecnich uzemi a vlastni hra.

4.2.1. ROZEBRANI POCATECNICH UZEMI (PHASE 0)

Hrac, ktery je na tahu, si vybere neobsazene uzemi a informuje server prikazem REINFORCE <player-id> <area-id> 1, posila tedy na dane uzemi 1 jednotku. Tato jednotka se mu tedy odecte z celkoveho mnozstvni volnych posil. Server nasledne informuje ostatni hrace prikazem REINFORCE <player-id> <area-id> 1, kde <player-id> je id hrace, ktery prave obsadil uzemi.

V pripade, ze je nastavene pravidlo o nahodnem rozdeleni uzemi, tak je prvni krok teto faze vynechan.

Hrac ukonci sve kolo prikazem END-ROUND a server vyzve dalsiho hrace prikazem START-ROUND.

Kdyz jsou vechna uzemi rozebrana (ne nutne museji mit vsichni hraci stejny pocet uzemi). Server posune hru do dalsi faze prikazem START-PHASE 1

4.2.2. POSILOVANI POCATECNICH UZEMI (PHASE 1)

V teto fazi rozdeluje zbyle jednotky na sva zabrana uzemi.

V jednom svem kole musi hrac umistit na plan aspon 1 jednotku, pokud jiz vsechny jednotky rozmistil, ukonci okamzite kolo prikazem END-ROUND (viz. nize).

Hrac, ktery je na tahu, si vybere jedno ze svych uzemi a prida na nej jednu jednotku stejne jako v predchozi fazi hry prikazem REINFORCE <player-id> <area-id> <num-units>. Server reaguje take stejne.

Timto zpusobem muze hrac posilit v ramci sveho kola posilit i vice uzemi najdnou.

Hrac ukoci sve kolo prikazem END-ROUND a server vyzve dalsiho hrace

prikazem START-ROUND.

Kdyz vsichni hraci dokonci posilovani svych uzemi, zacne server fazi vlastni hry pomoci prikazu START-PHASE 2

4.2.3. VLASTNI HRA (PHASE 2)

Hra se sklada z jednotlivych kol. Na zacatku kazdeho kola je hrac, ktery ma byt prave na tahu, vyzvan serverem pomoci prikazu START-ROUND, aby zacal hrat. Svuj tah ukonci hrac prikazem END-ROUND.

Prubeh jednoho kola je popsan v samostatnem bode 5.

5. PRUBEH KOLA

5.1. ZISKANI A UMISTENI POSIL

5.1.1 ZISKANI POSIL ZA UZEMI

Hrac ziskava posily za pocet uzemi, ktera ovlada (viz. Pravidla). Server informuje hrace prikazem REINFORCEMENTS o poctu posil, ktere ziskal pro toto kolo. Stejnym prikazem o tom informuje i vsechny ostatni hrace.

5.1.2 POSILENI UZEMI

Hrac umisti sve posily na uzemi pomoci prikazu REINFORCE, podobne jako v 4.2.(1-2), ale nyni muze umistit na libovolne sve uzemi libovolne mnozstvi svych posil, napr:

REINFORCE 4 26 10 (posle 10 jednotek na Ural, id hrace je 4)

Server o tom opet informuje vsechny ostatni hrace (necht ma aktualni hrac ID 4):

REINFORCE 4 26 10 (hrac s ID 4 poslal 10 jednotek na Ural)

5.1.3 ZISKANI POSIL SMENOU KARET

Pokud ma hrac korektni kombinaci karet, kterou chce smenit, udela tak prikazem CARDS-EXCHANGE, ve kterem uvede typy karet, ktere hodla smenit. Od server mu prijde formou prikazu REINFORCEMENTS kolik ziskal jednotek.

Takto ziskane posily umisti hrac na sva uzemi stejne jako v bode 5.1.2.

V pripade, ze ma hrac vic nez 4 karty, musi aspon jednu sadu smenit.

5.2. BOJ

Rozhodne-li se hrac utocit, provede to prikazem ATTACK. Jak je zvykem server tuto informace preda hracum verejnou variantou tohoto prikazu. Pocet utocicich jednotek je dle pravidel 1-3.

Hrac, ktery je napaden ma za povinnost oznamit pocet obrancu, to provede prikazem DEFEND. Pocet branicich jednotek dle pravidel 1-2.

Server za hrace hodi kostkou a vysledek hodu zverejni prikazem DICE-ROLL. Je na klientech, aby interpretovali hod kostkou a odstranili porazene jednotky, prip. se vzdali prohraneho uzemi.

Dobyl-li hrac nejake uzemi, muze na nej presunout libovolne mnoztstvi jednotek z uzemi, ze ktereho byl veden posledni utok, tak jak je popsano v 5.3.

5.3. PRESUN JEDNOTEK

Hrac muze presunout jednotky na nejake uzemi, pokud ho prave dobyl nebo vyuziva moznost posilit svou pozici (kterou ma jednou za kolo, viz

Pravidla).

Presun jednotek se provadi prikazem TRANSFER. Server opet informuje ostatni hrace verejnou variantou prikazu TRANSFER.

5.4. ZISKANI KARTY

Pokud hrac ve svem kole dobyl alespon jedno uzemi, ziskava automaticky jednu kartu. Server ho o tom informuje prikazem CARDS-ACQUIRED, kde uvede konkretni typ karty, kterou hrac ziskal. Ostatni hrace server informuje verejnou variantou tohoto prikazu, tedy sdeli, ze hrac s danym ID ziskal 1 kartu.

5.5. VYRAZENI HRACE

Pokud hrac ziskal posledni uzemi jineho hrace, vyradil ho tim ze hry a ziskava vsechny jeho karty. Server ho informuje prikazem CARDS-ACQUIRED s prislusnym poctem a typem karet. Ostatni hrace server informuje verejnou variantou tohoto prikzu, tedy sdeli jim, kolik karet vitez od porazeneho ziskal.

Navic server informuje vsechny hrace o vyrazeni jednoho z nich prikazem PLAYER-DEFEATED.

5.6. KONEC HRY

Hraci obdrzi zpravu PLAYER-DEFEATED a zaroven ve hre zustal posledni hrac. Tento hrac je vitez.

6. ZISKANI STAVU HRY

Klient ma moznost dotazat se kdykoliv serveru na aktualni stav hry. Hodi se zjm. v pripade, kdy klient ztratil na nejakou dobu spojeni.

<TODO>

7. SEZNAM PRIKAZU

7.1. FORMAT PRIKAZU

Kazdy prikaz je retezec ASCII znaku slozeny z tokenu oddelenych jednou nebo vice mezerami. Prikaz je ukoncen znakem konce radku '\n' (0x0a). Prvni token prikazu je zpravidla psany velkymi pismeny a jde o jmeno prikazu. Ostani tokeny jsou argumenty prikazu. Maximalni delka prikazu je 127 znaku.

```
COMMAND <arg0> ... <arg_n>
```

7.2. SEZNAM PRIKAZU (CLIENT -> SERVER)

Pokud se v prikazu vyskytuje argument <player-id>, je klient povinnen na tomto miste poslat identifikator, ktery ziskal po zalogovani od serveru.

LOGIN <real-id>
- hrac se prihlasi se svym skutecnym jmenem, server odpovi zpravou LOGIN-ACK

READY <player-id>
- posila hrac na server pote, co ziskal potvrzeni o prihlaseni a je pripraven ke hre

REINFORCE <player-id> <area-id> <num-units>
- hrac informuje server poctu jednotek, ktere chce umistit na lokaci <area-id>

CARDS-EXCHANGE <player-id> <card_0> <card_1> <card_2>
- hrac sdeluje, ktere karty hodla smenit

TRANSFER <player-id> <from> <to> <num-units>
- hrac presouva <num-units> jednotek z uzemi s area-id <from> na uzemi s area-id <to>

ATTACK <player-id> <from> <target> <num-units>
    - hrac utoci z uzemi <from> na uzemi <target> poctem jednotek
      <num-units>
    - hrac ktery vlastni uzemi <target> ho musi branit, pouzije prikaz
      DEFEND

DEFEND <player-id> <area-id> <num-units>
    - v pripade, ze je hrac napadan, oznamuje serveru kolika jednotkama
      bude uzemi branit

END-ROUND <player-id>
    - hrac oznamuje konec sveho kola

CHAT PUBLIC <player-id> <message>
    - hrac posila zpravu vsem ostatnim hracum

CHAT PRIVATE <player-id> <recipient-id> <message>
    - hrac posila soukromou zpravu hraci <recipient-id>

APPLY_RULE <rule-id> <variant-id>
    - posila hrac na server, aby vynutil pouziti urciteho pravidla
    - oznaceni pravidel a jejich variant odpovida sekci tohoto
      dokumentu, ktera je popisuje

<rule-id>	pravidlo	<variant-id>	varianta
0	8.2.1	0	8.2.1.1
		1	8.2.1.2
1	8.2.2	0	8.2.2.1
		1	8.2.2.2

7.3 SEZNAM PRIKAZU (SERVER -> CLIENT)

Zpravy, ktere posila server klientum. Server posila tyto zpravy vsem klientum zaroven, pokud neni uvedeno jinak.

LOGIN-ACK OK <assigned-id>
LOGIN-ACK FAILED <error-id>

<error-id>	popis
1	bad name format
2	id already in use
3	too many players

    - posila server klientovi jako reakci na LOGIN prikaz, pokud je vse
      v poradku posle variantus OK a propoji ciselny identifikator,
      kterym se bude klient nyni identifikovat
    - tuto zpravu posila server pouze klientovi, ktereho se tyka

LOGOUT <player-id> <reason>

<reason>	oduvodneni
0	kicked by admin
1	user left game

    - server informuje ostatni hrace, ze se jeden z nich odhlasil ze
      hry nebo byl odhlasen serverem

REGISTERED-PLAYER <player-id> <real-id>
    - v server informuje ostatni hrace o pritomnosti noveho hrace
    - server posle tuto zpravu jako reakci na LOGIN prikaz

ORDER <player-id_0> ... <player-id_n>
    - server sdeluje poradi, v jakem bude probihat kolo

START-PHASE <phase-id>

<phase-id>	popis
0	rozebrani pocatecnich uzemi
1	posilovani pocatecnich uzemi
2	vlastni hra

- server zahajuje danou fazi hry (viz tabulka)

START-ROUND <player-id>
- server vyzyva hrace <player-id>, aby zacal svuj tah

TRANSFER <player-id> <from> <to> <num-units>
- hrac <player-id> presouva <num-units> jednotek z uzemi s area-id <from> na uzemi s area-id <to>

REINFORCE <player-id> <area-id> <num-units>
- hrac <player-id> posiluje uzemi <area-id> pocetem jednotek <num-units>

REINFORCEMENTS <player-id> <num-units>
- hrac <player-id> ziskava <num-units> posil

ATTACK <who-id> <from> <target> <num-units>
- hrac <who-id> zautocil z uzemi <from> na uzemi <target> s poctem jednotek <num-units>

DEFEND <who-id> <area-id> <num-units>
- hrac <who-id> brani utok na uzemi <area-id> poctem jednotek <num-units>

PLAYER-DEFEATED <who-id> <by-id>
- server posila informaci o vyrazenem hraci

CARDS-EXCHANGE <player-id> <card_0> <card_1> <card_2>
- hrac <player-id> smenil dane kary

CARDS-ACQUIRED <player-id> <num-cards>
- hrac <player-id> ziskal nove karty v poctu <num-cards>

CARDS-ACQUIRED <player-id> <num-cards> <card_0> ... <card_n>
- tuto zpravu posila server pouze hraci <player-id>
- informuje ho o kartach, ktere ziskal

DICE-ROLL <ad1> <ad2> <ad3> <dd1> <dd2>
- server posila toto zpravu vsem hracum jako vysledek probihajiciho boje
- vyznam argumentu:
  - ad1 - hod 1. kostkou utocknika
  - ad2 - hod 2. kostkou utocknika
  - ad3 - hod 3. kostkou utocknika
  - dd1 - hod 1. kostkou obrance
  - dd2 - hod 2. kostkou obrance
- v pripade, ze nejaka kostka neni pro aktualne probihajici boj relevantni, je hodnota prislusneho hodu 0

8. PRAVIDLA A VARIANTY

8.1. OFICIALNI PRAVIDLA

Hra se, az na vyjimky uvedene v 8.2, ridi pravidly od spolecnosti Hasbro (ke stazeni: http://www.hasbrohracky.cz/download/risk_navod_cz.pdf, strany 5-12). Na tento dokument se v textu odkazujeme pomoci terminu "Pravidla".

8.2. VARIANTY PRAVIDEL

8.2.1. ROZDELENI POCATECNICH UZEMI

8.2.1.1. VYBIRAJI HRACI (oficialni varianta)

Varianta uvedena v Pravidlech, uzemi vybiraji postupne hraci.

8.2.1.2. ROZDELI SERVER

Server pocatecni uzemi rozdeli mezi hrace nahodne.

8.2.2. POCET POSIL ZA KARTY

8.2.2.1. ROSTOUCI (oficialni varianta)

Pocet posil zavisi na poctu sad, ktere byly v cele hre doposud vymeneny. Pocty jsou: 4, 6, 8, 10, 15, 20, 25,...,60.

8.2.2.2 KONSTANTNI S RUZNOU HODNOTOU SAD

V teto variante nezalezi na poctu smenenych sad behem hry, hraci vzdy ziskavaji konstantni mnozstvi posil. Sady maji vsak ruznou hodnotu: Pechota 3 jednotky, Kavalerie 5 jednotek, Delostrelectcvo: 7 jednotek, od kazdeho typu jedna karta: 9 jednotek.

APPENDIX A

Tabulka uzemi a jejich indexu

UZEMI	INDEX
Aljaska	0
Severozapadni uzemi	1
Gronsko	2
Alberta	3
Ontario	4
Quebec	5
Zapad USA	6
Vychod USA	7
Stredni Amerika	8
Venezuela	9
Peru	10
Brazilie	11
Argentina	12
Island	13
Skandinavie	14
Velka Britanie	15
Zapadni Evropa	16
Stredni Evropa	17
Jizni Evropa	18
Ukrajina	19
Severni Afrika	20
Egypt	21
Kongo	22
Vychodni Afrika	23
Jizni Afrika	24
Madagaskar	25
Ural	26
Sibir	27
Jakutsk	28
Kamcatka	29
Irkutsk	30
Afghanistan	31
Cina	32
Mongolsko	33
Japonsko	34
Stredni vychod	35
Indie	36
Siam	37
Indonesie	38
Nova Guinea	39
Zapadni Australie	40
Vychodni Australie	41

APPENDIX B

Tabulka typu karet

<card-id>	typ karty
0	pechota
1	kavalerie
2	delostrelectvo
3	divoka karta