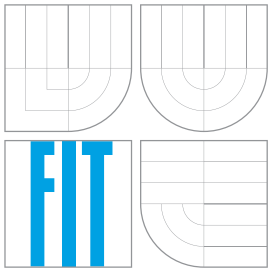
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATIZACE HRY "HEARTHSTONE"

AUTOMATION OF "HEARTHSTONE" GAME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FILIP ZELNÍČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Dr. Ing. PAVEL ZEMČÍK,

BRNO 2016

Zadání bakalářské práce/18995/2015/xzelni00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Zelníček Filip**
Obor: Informační technologie
Téma: **Automatizace hry "Hearthstone"**
Automation of "Hearthstone" Game
Kategorie: Uživatelská rozhraní

Pokyny:

1. Prostudujte dostupnou literaturu na téma pravidla a způsob hry "Hearthstone" a prostudujte též možnosti aplikace umělé inteligence na hraní této hry.
2. Vyberte vhodnou metodu automatizace a umělé inteligence a výběr odůvodněte.
3. Navrhněte způsob implementace vybrané metody a diskutujte možné vlastnosti takové implementace.
4. Implementujte vybranou metodu a demonstrujte výsledek na vhodném příkladě.
5. Diskutujte dosažené výsledky a možnosti pokračování práce.

Literatura:

- Dle pokynů vedoucího práce

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zemčík Pavel, prof. Dr. Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá počítačovou karetní hrou Hearthstone: Heroes of Warcraft a aplikací umělé inteligenci na tuto hru. Program umí zpracovat informace ze hry a následně je ohodnotit speciálním algoritmem, vyhodnotí je a podá hráči hrajícímu tuto hru informace, jak podniknout další kroky, které ho co nejvíce přiblíží k výhře v dané hře. V tomto směru bylo dosaženo velmi dobrého výsledku. Při testování proti umělé inteligenci ve hře bylo dosaženo více než 80% úspěšnosti. Tato práce může sloužit pro začínající hráče této hry, pro základní informace o hře a návod, či pomoc k jejímu hraní.

Abstract

This thesis is about computer card game called Hearthstone: Heroes fo Warcraft and about application of artificial intelligence in this game. Program knows how to deal with informations coming from game and how to evaluate them with a specific algorithm. After that it evaluate the data and gives to the player playing this game information what to do in the following step to win the game. In this scenario was reached really good results. It was tested against artificial intelligence in game and this program had winrate over 80%. This thesis can serve as little help for begginers in Hearthstone, or as a manual how to play this game.

Klíčová slova

Hearthstone, automatizace, umělá inteligence, karetní hra

Keywords

Hearthstone, automation, artificial intelligence, card game

Citace

ZELNÍČEK, Filip. *Automatizace hry "Hearthstone"*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Zemčík Pavel.

Automatizace hry "Hearthstone"

Prohlášení

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

.....
Filip Zelníček
18. května 2016

Obsah

Kapitola 1

Úvod

V 21. století se do povědomí a módy dostávají elektronické přístroje jako jsou mobilní telefony, počítače, notebooky a tablety. Tato zařízení primárně slouží k zjednodušení a zrychlení práce, ale mohou sloužit i jako zdroj informací a zábavy. Jako takový zdroj zábavy může sloužit například i počítačová hra a tato práce se zabývá právě jednou počítačovou karetní hrou, která se jmenuje Hearthstone: Heroes of Warcraft. Dále se tato práce zabývá její automatizací, neboli hraní této hry automatem.

Cílem této práce je vytvořit funkčního robota, který bude schopný sám analyzovat současný stav hry a navrhnout co nejlepší možný tah hráči, který jej následně provede. Poté co se tak stane, bude robot nabízet další možné tahy, které budou mít co nejvyšší pravděpodobnost úspěšného výsledku, tedy výhry v dané partii. Výsledky se budou testovat proti umělé inteligenci zabudované ve hře, která slouží pro nováčky k vyzkoušení si této hry. K tomuto testu budou využity herní balíčky, které budou obsahovat pouze základní karty, které má k dispozici každý hráč. Více informací o získávání dalších karet, než těch základních je popsáno v dalších kapitolách. Cílem této práce bude dosáhnout funkčním robotem co nejvyššího poměru výher a proher. V případě více výher než proher, bude robot považován za úspěšný. Navíc se pokusím porovnat nejen výsledky robota proti umělé inteligenci ve hře, ale i to, jak by hrál opravdový hráč, který s touto hrou má již nějaké zkušenosti.

V současné době je téměř vše ovládáno elektronikou a tím pádem řízeno nějakým inteligentním systémem. Tato práce se zabývá podobným problémem – využití umělé inteligence, nebo automatizace v reálném životě, aplikované na počítačové hře. Mou osobní motivací ke zpracování této práce je zájem o umělou inteligenci, zájem o samotnou hru Hearthstone a chuť si vyzkoušet využít dosažené znalosti z oboru na nějakém větším projektu, než s kterým jsem se mohl během bakalářského studijního programu setkat.

V kapitole 2 jsou popsány prvky Hearthstone, základní principy hry, dostupné rasy, herní plocha, různé typy schopností karet a nakonec v této kapitole nalezneme popsána pravidla hry. V další kapitole 3 se seznámíme s umělou inteligencí, nebo automatizací v Hearthstone a další známých hrách. Další kapitola 4 nám popíše mechanismus ohodnocení karet, způsob volby tahů a celkový chod robota. V kapitole 5 je popsána samotná implementace programu a jak vyhodnocuje situace ve hře a technické problémy, které mohou při chodu robota nastat.

Kapitola 2

Karetní hra Hearthstone

V celé této kapitole se práce zabývá shrnutím současného stavu a to konkrétně o samotné hře Hearthstone. O tom jak se tato hra hraje, jaké jsou její pravidla, co je k jejímu hraní potřeba. Dále zde nalezneme popis jednotlivých typů karet a různé typy tahů s nimi. Avšak toto téma je značně rozsáhlé a tato kapitola se bude pouze zabývat tou nejdůležitější částí, která je podstatná pro zbytek práce a její správné pochopení.

2.1 Cíle hry

Jako každá počítačová hra i Hearthstone slouží primárně jako zdroj zábavy. Lze ji hrát opakovaně stále dokola, pokaždé proti jiným soupeřům. Každý hráč na začátku začíná s 30 životy a určitým počtem karet, který je vysvětlen v dalších kapitolách. Během jednotlivých tahů si hráč líže karty podle určitých pravidel a během svého kola vykládá různé typy karet, nebo využívá již vyložených k odstranění karet soupeře, či sebrání životů soupeře. Jakmile hráč přijde o všech 30 životů, hra končí. Během jednotlivých kol, kdy se hráči střídají na tahu, může hráč do časového limitu vykládat své karty, či odstraňovat soupeřovy. Každá karta má nějaký svůj typ, schopnost a cenu. Více informací o typech karet je v následujících kapitolách. Pro některé hráče může být cílem ve hře posbírat všechny karty, nebo soutěžit proti jiným hráčům v bodovaných hrách a dostávat se na co nejvyšší příčku v žebříčku v celkovém měsíčním hodnocení. Na obrázku 2.1 je ukázka prostředí Hearthstone.

2.2 Historie

Hearthstone je on-line počítačová karetní hra vydaná společností Blizzard Entertainment v roce 2014. V současné době k 1. květnu 2016 má oficiálně přes 50 milionů zaregistrovaných uživatelů[25] a slaví rozšíření v podobě tří adventur a 3 rozšíření s novými sety karet, které zvětšují hráčské obzory a možnosti. Aktuálně hra obsahuje celkem 1053 karet[13], které lze využít ke hraní. Více informací o samotných kartách je v kapitole 2.3.

Jeden z důvodů úspěšnosti této hry je také její rozšíření na řadu platforem, jako jsou Windows, IOS, Mac, nebo Android. Díky zmíněnému rozšíření na mobilní telefony a tablety s operačním systémem IOS, nebo android drží Hearthstone cenu za nejlepší mobilní hru za rok 2014. Dále její úspěšnost vzniká úzkou spoluprací mezi vývojáři a světově známými streamery[1], kde před oficiálním vydáním rozšíření jsou streamerům poskytnuty nové karty,

[1]Streamer je člověk, který hraje denně vybranou pc hru a své počínání vysílá veřejně na internet, např. na *www.twitch.tv* a díky své sledovanosti vydělává na reklamách, nebo na smlouvě se serverem, kde vysílá.

Obrázek 2.1: Ukázka prostředí v Hearthstone

které pak ve svém vysílání zveřejňují. Další interakcí mezi společností Blizzard a samotnými hráči, je uvedení karty Fiery Bat v novém rozšíření, což je počítačová přezdívka vítěze prvního turnaje v Hearthstone na Blizzconu.

Hearthstone se již od svého oficiálního vydání z roku 2014 vyskytuje na velkých e-Sport[2] turnajích s obrovskými odměnami pro vítěze. Za zmínku stojí celoroční kvalifikace na Blizzcon, kde se z každé moderní hry[3] od společnosti Blizzard Entertainment kvalifikují 4 hráči z každého regionu (regiony jsou rozděleny na: Evropu, Ameriku, Asii a Čínu) a hrají o titul nejlepšího hráče/týmu na světě. K cestě na Blizzcon slouží několik oficiálních turnajů po celý rok. V roce 2015 se konal turnaj European Road to Blizzcon v Praze v O_2 aréně[26].

Nakonec se úspěšnost a oblíbenost této počítačové hry odráží i v podpoře nových hráčů ze strany Blizzardu. Když se do hry přihlásí nový hráč, tak dostane pouze základní karty a s troškou úsilí může už za hodinu hraní získat dalších několik desítek základních karet. Vzhledem k tomu, že zhruba třikrát do roka vychází rozšíření čítající od 30 do 150 karet, tak by nováček nebyl v reálném čase bez investice peněz schopen získat většinu lepších karet, bez kterých se vyvážené a kvalitní balíčky nedají na vyšší úrovni hrát. Samozřejmě záleží na vkusu hráče a jeho chuti hrát se základními kartami, nebo se všemi rozšířeními. Proto Blizzard zavedl model *Standard* a *Wild*[14]. Ve *Wild* se nedějí žádné změny a hráč může využít všechny dosud vydané karty při stavbě balíčku dle pravidel zmíněných v dalších kapitolách. Karty musí ale nejprve vlastnit, aby je mohl využívat.

Proti tomuto modelu přišel v roce 2016 model nový – *Standard*, kde je povoleno využívat kromě základních karet a karet ze setu *expert*, který vyšel současně s vydáním hry, pouze několik nejnovějších rozšíření. Tím pádem nový hráč potřebuje získat pouze zlomek karet, a nemusí posbírat okamžitě všechny kolekce. Toto umožňuje také manipulovat s příliš silnými

[2]E-Sport je turnaj, nebo kolekce turnajů v nějaké počítačové hře organizovaných úplně stejně, jako například sportovní turnaje jako mistrovství světa. Turnaje mohou probíhat pouze přes internet, avšak velké a populární turnaje probíhají v obrovských halách s publikem a doprovodnými akcemi

[3]Hearthstone, World of Warcraft, Heroes of the Storm, Starcraft II

kartami bez nutnosti je odstranit ze hry, nebo výrazně změnit, jak se tomu dělo před vydáním *Standardu*. Také tento model zajišťuje popularitu hry, kdy se co několik měsíců může změnit tzv meta hry, neboli styl jakým se většina her odehrává. Jestli jsou to rychlé, nebo pomalé hry, či která rasa je silnější, díky kombinaci karet, anebo naopak která se v daném období téměř nehraje. Další informace ohledně karet jsou v následujících kapitolách.

2.3 Karty

Hearthstone je počítačová karetní hra s možností sbírat karty a vytvářet z nich nové herní balíky. Podobných karetních her existují desítky a to jak počítačové, tak reálné. V reálných karetních hrách hráč sbírá opravdové karty, které fyzicky drží v ruce. V počítačových je vlastní pouze digitálně. Pravděpodobně nejpopulárnější reálnou sběratelskou karetní hrou je Magic the Gathering. V této kapitole jsou informace o typech karet v Hearthstone, jak získat další, kromě těch základních a jaký mají vliv na samotnou hru.

Každá karta má své jméno, typ, svou cenu v maně, kterou je třeba zaplatit pro zahrání karty z ruky, raritu a další schopnosti, které budou popsány v této kapitole. Karta může být neutrální, příslušná k nějaké rase, které jsou popsány v následujících kapitolách a nebo speciální. Speciální karty se dostávají do hry při aktivování efektu jiných karet. Avšak tyto speciální karty nelze získat do své vlastní kolekce karet.

Hráč při registraci a splnění herního tutoriálu dostane základní kolekci karet. V každém herním balíčku může hráč využít až dvě stejné neutrální karty, nebo karty pro jeho rasu. Jedinou výjimkou jsou karty, které jsou označené jako *Legendární*. Tyto karty může hráč využít pouze po jednom kusu v jeho herním balíku. *Legendární* karty hráč pozná podle oranžového diamantu ve středu karty. Pokud chce získat další karty má několik možností jak je získat a rozšiřovat svou sbírku.

- Koupit balíček 5 karet za herní měnu Zlato a nebo reálné peníze. Herní zlato hráč dostává za vyhrané hry a splněné úkoly, které dostává každý den
- Zakoupit a splnit adventuru za herní měnu zlato a nebo reálné peníze
- Vyhrát alespoň jednou týdně souboj v Tavern Brawlu
- Splnit speciální úkoly, nebo úspěchy ve hře (Achievementy)
- Vstoupit do arény a vyhrát co nejvíce her, než ji opustí. Odměnou je balíček 5 karet a jednotlivé karty
- Vyrobení si nových karet za odstranění nepotřebných, nebo nadbytečných karet. Hráč může využít funkci *disenchant*, která trvale zničí vybranou kartu a hráč z ní získá 1/4 její ceny v tzv arcane dust a následně pak za tyto arcane dusty může vyrobit libovolné karty, dokonce i legendární karty (cena za výrobu karty je úměrná její kvalitě)

Karty se dělí do 3 typů:

- Miniony
- Spelly
- Zbraně

Minion

Minion, neboli poskok, je karta, která po zahrání zůstává na hrací ploše. Má vlastní cenu, útok a životy. Po vyvolání z ruky musí minion čekat do dalšího kola, než bude moci zaútočit. Výjimkou jsou minioni se schopností *Charge* (2.3). V případě, že minion může útočit, může zaútočit na libovolný cíl (hráč, nebo soupeřův minion) pouze jednou za kolo. Výjimku tvoří minioni se schopností *Windufry*(2.3). Na obrázku 2.2 je zobrazen klasický neutrální minion[2]. Na obrázku 2.3 je zobrazen minion[9], který patří hrdinovi Hunter, což můžeme poznat podle zeleného zbarvení okrajů vrchní části karty.

Obrázek 2.2: Minion – Boulderfist Ogre

Obrázek 2.3: Minion – Tundra Rhino

A – útok, B – životy, C – vyvolávací cena, D – schopnosti, E – jméno , F – typ

Speciální schopnosti

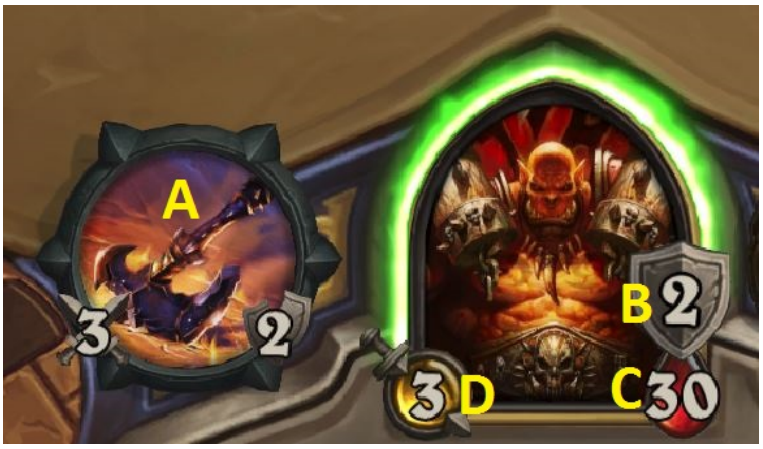
Speciální schopnosti jsou rozšiřující schopnosti minionů, které mají ve hře speciální efekt

- Battlecry – efekt, který minion vykoná poté, co přijde do hry z ruky
- Deathrattle – efekt, který minion vykoná poté, co je odstraněn ze hry
- Taunt – minion s *Taunt* musí být primárním cílem útoků soupeřova minionů, nebo hrdiny, v případě více minionů s *Taunt* jsou všichni prioritními cíli a soupeř vybírá na který cíl zaútočí první
- Charge – minion nemusí čekat do dalšího kola, aby mohl zaútočit
- Stealth – na miniona nemůže být zaútočeno, zacíleno kouzlo, nebo speciální schopnost
- Spell Damage – minion dočasně zvyšuje poškození kouzly o hodnotu *Spell Damage*

- Choose One – při zahrání karty s *Choose One* si hráč vybere jednu z možností a karta se chová dle popsaného textu
- Divine Shield – minion při prvním obdržení poškození nedostane žádné poškození, ale ztratí schopnost *Divine Shield*, ve hře je zobrazena žlutou bublinou kolem miniona
- Combo – tato schopnost je aktivní, pokud již hráč zahrál toto kolo z ruky jinou kartu
- Windfury – tento minion může zaútočit 2x za kolo
- Overload – pokud hráč zahraje kartu s *Overload*, tak v dalším kole bude mít k dispozici o hodnotu *Overload* méně mana krystalů
- Enrage – efekt se aktivuje pouze, když je minion poškozen
- Frozen – minion nemůže další kolo útočit, poté efekt *Frozen* zmizí
- Inspire – efekt tohoto miniona se provede vždy, když hráč použije svou speciální schopnost
- Discover – hráč dostane na výběr 3 náhodné karty dle kritérií a vybere si jednu z nich a tu si lízne, zbytek zmizí
- Další dodatečné schopnosti – tyto schopnosti jsou slovně popsány u konkrétních minionů

Weapon

Weapon, neboli zbraň je speciální karta, která je přiložena k hrdinovi. Současně může každý hráč aktivní pouze jednu zbraň. V případě, že zahraje druhou, tak první zbraň zaniká a aktivní zůstává pouze ta později zahraná. Každá zbraň má útok a durabilitu, která se zobrazuje stejně, jako životy u minionů. Útok určuje výši poškození, které může udělat hrdina cizímu hrdinovi, nebo minionovi. Durabilita určuje kolikrát lze danou zbraň použít. Na obrázku 2.4 je vidět detail hrdiny s přiloženou zbraní.



Obrázek 2.4: Detail hrdiny s přiloženou zbraní

A – Vyložená zbraň s 3 útok na 2 použití, B -– Armor na hrdinovi (2), C — Životy hrdiny (30), D -– Útok hrdiny (3)

Spell

Spell, neboli kouzlo, je karta, která má okamžitý vliv na hru, nemá tedy útok a životy, jako minion. Poté co vykoná svůj efekt, je karta kouzla přesunuta na hřbitov. Samotný efekt kouzla je popsaný na kartě. Jedinou výjimkou jsou speciální kouzla *Secret*. Tyto kouzla mohou vlastnit hrdinové Mage, Paladin či Hunter a zobrazují se okolo rámečku hrdiny. *Secrety* nevykonají svůj efekt okamžitě, ale čekají, až je splněna podmínka jejich aktivace. Až se tak stane, *Secret* se aktivuje a poté je odstraněn ze hry, jako každé jiné kouzlo. Hráč nemůže mít aktivních více stejných karet *Secret* naráz. Na obrázku 2.5 je příklad karty *secret* Mirror Entity[5], která patří k hrdinovi Mage (modré zbarvení). Soupeř vidí, ze hráč zahrál *Secret*, ale již neví, jaký efekt bude mít. Na obrázku 2.6 vidíme příklad klasického kouzla Shadow Bolt[6], které patří k hrdinovi Warlock (fialové zbarvení okrajů jeho karet).

Obrázek 2.5: Secret – Mirror Entity Obrázek 2.6: Spell – Shadow Bolt

2.4 Herní balíček

Jako v každé karetní hře je i v Hearthstone herní balík, ze kterého si hráč líže kary. Karty hráč drží na ruce a během chvíle, kdy je na tahu, může tyto karty zahrát. Herní balík si hráč může kdykoliv vytvořit, v hlavním menu hry, z karet, které aktuálně vlastní. V každém balíčku se mohou nacházet karty příslušné k dané rase a nebo neutrální minioni, kteří se mohou nacházet ve všech herních balících. Obvykle si hráč rozmyslí nějaký koncept a podle něj si pokusí balík poskládat, ale je možné využít i inteligentní návrhy herních balíků. V každém poskládaném balíku, (krom speciálního případu arény) se musí vyskytovat přesně 30 karet.

Každá stejná karta může být použita při konstrukci balíku maximálně 2x. Jedinou výjimkou jsou karty označené jako *Legendary* (legendární) a ty se mohou v balíku nacházet pouze v jednom kusu. Z dřívějších kapitol víme, že získat nové karty lze převážně při otvírání nových balíčků (pozor, neplést s herním balíkem). Jedná se o 5 náhodných karet, kde je

alespoň jedna karta *Rare* kvality a nebo vyšší. Jeden takový balík se dá pořídit v herním obchodě za 100 zlatých, což je herní měna. Tuto měnu hráč získává hlavně za výhry proti jiným hráčům (až 100 zlatých za den) a při splnění denního úkolu (40 až 100 zlatých za den). Další možnosti získat zlato jsou v aréně.

Pokud hráč nemá zlato a chce získat další karty, může disenchantnout (rozložit) některé své nepotřebné karty a za výsledek tohoto procesu (Arcane Dust) si vytvořit libovolné karty nové. Avšak z rozložení karty hráč nedostane její plnou cenu, která závisí na raritě té konkrétní karty. Tyto rarity máme následující:

- Common (běžná), tato kvalita se značí bílým krystalem uprostřed karty. Cena výroby *common* karty je 40 Arcane Dust, její rozložení 5 Arcane Dust.
- Rare (vzácná), tato kvalita se značí modrým krystalem. Cena její výroby je 100 Arcane Dust a její rozklad přinese 20 Arcane Dust.
- Epic (epická), epická kvalita se značí fialovým krystalem a je možné ji vyrobit za 400 Arcane Dustů a rozložit za 100 Arcane Dust. Jedná se o jedny z nejsilnějších karet.
- Legendary (legendární), tyto karty jsou velmi silné, vzácné a také drahé na výrobu. Jedna taková karta přijde hráče na 1600 Arcane Dust a z jejich rozkladu získá 400 Arcane dust.

Rarita (vzácnost) karet rozhoduje jaká je šance, že si hráč takovou kartu otevře v balíku a nebo cenu její výroby. Obecně platí, že vzácnost karet také souvisí s jejich efektem ve hře, avšak nalezneme jak špatné legendární karty, tak i velmi dobré *common*, nebo *rare* karty. Pokud hráč nemá dostatek zlata, nebo Arcane Dust, je zde možnost zakoupit karty za reálné peníze, závislé dle počtu zakoupených balíčků. Cena v reálných penězích se pohybuje okolo 1 Eura za balíček.

Během skládání nového balíku je doporučené držet se již zmíněného konceptu. Například můžeme herní balík stavět jako kontrolní, kde hráč hraje velmi pasivně a čeká, co udělá soupeř. Jakmile soupeř zahraje nějakou kartu, hráč s kontrolním balíkem na ni reaguje a takto se snaží odrazit všechny hrozby. Opakem kontrolního balíku je tzv *Aggro* balík, který se snaží zaplavit herní plochu co nejvíce levnými miniony a vyhrát hru dřív, než soupeř bude moci pořádně zareagovat, nebo se dostat do stavu, kdy až bude moci soupeř reagovat, tak bude hra v podstatě rozhodnutá. Třetím typem balíčků jsou tzv *Midrange* balíky, které jsou kombinací mezi kontrolními typy a *Aggrem*. Soustředí se na hru okolo čtvrtého až šestého kola, kdy se snaží hru rozhodnout. Čtvrtým typem se dají považovat tzv *Combo* balíčky, které kombinují více zajímavých herních mechanismů, nebo schopností karet, kdy může hráč vyhrát během jednoho kola. Nevýhodou těchto balíků je občas zdlouhavé čekání, než hráč dostane všechny potřebné karty do ruky. Herní balíky se můžou kdykoliv před, nebo po hře měnit, či upravovat. Hráč může mít celkem vytvořených až 18 herních balíků, ve kterých může opakovaně použít stejnou kartu, jako využívá v jiném, což je určitě výhoda proti opravdovým kartám.

2.5 Hrdina

Neboli hero, či povolání. Hrdinů ve hře najdeme 9 unikátních typů, kde každý hrdina má svou unikátní speciální (raciální) schopnost a specifickou zásobu karet, které patří pouze k danému hrdinovi. Tyto karty se značí barvou daného hrdiny. Za zmínku též stojí šest emocí, které jsou ukázané na obrázku 2.7 a kterými hráč může vyjádřit své pocity soupeři

(například ohodnotit dobře zahraný tah), což je jediná forma jak se může během hry komunikovat se soupeřem. Speciální schopnost lze použít jednou za kolo a efekty každé hrdinové jsou následující:

Obrázek 2.7: Ukázka emocí ve hře hrdiny Warlock

- Warlock – ztratí o dva životy a lízne si další kartu ze svého balíčku
- Priest – vyléčí cílovému charakteru až 2 životy (minion, nebo hrdina)
- Mage – udělí cílovému charakteru 1 poškození
- Rogue – přiloží hráči zbraň s 1 útok a 2 durability
- Druid –přidělí hráči 1 útok do konce kola a 1 armor
- Shaman – vyvolá jeden ze 4 náhodných totemů, pokud již není vyvolán na stole hráče
- Hunter – udělí 2 poškození nepřátelskému hrdinovi
- Paladin – vyvolá miniona s 1 útokem a 1 životem
- Warrior – dostane 2 armory

2.6 Pravidla

Celá práce se zabývá hrou Hearthstone, proto pro lepší pochopení hry a práce samotné v této kapitole nalezneme zkrácené informace o pravidlech, povolených a naopak zakázaných možnostech tahu v jednotlivých kolech

Před začátkem hry si hráč poskládá hrací balík s 30 kartami (více o kartách v kapitole 2.3), nebo vybere jeden z 9 předem navolených herních balíčků. Poté si pro danou hru zvolí jednu z 9 ras (více o rasách v kapitole 2.5) a mód hry. Módy ve hře jsou následující:

- Hráč proti hráči, kde si hráč vybere formát hry (*Standard* s pouze nejnovějšími kartami, nebo *Wild*, který obsahuje veškeré karty ve hře) a to jestli chce hrát bodovanou hru (*Ranked*) s odměnou na konci měsíce a nebo obyčejnou nebodovanou hru, za kterou nejsou odměny (*Unranked*).

- Tavern Brawl, kde se každý týden střídá nová mechanika, která je obvykle odlišná od klasických pravidel Hearthstone. Za první výhru v Tavern Brawlu je jeden balíček karet ze setu classic. Obvykle jsou zde nastavená velmi šílená pravidla a slouží k pobavení hráčů.

- Aréna, kde hráč staví balíček z karet které mu nabídne konkrétní aréna (výběr vždy jedné karty z 30) a slouží pouze pro danou arénu. V Aréně hraje hry, dokud nemá 3 prohry a nebo 12 výher, poté dostane odměnu a arénu opouští.

- Adventura, kde pokud hráč získal přístup může porazit počítač s umělou inteligencí a jednorázově získat nové karty. Adventura má obtížnost normal a heroic, kde normal slouží pouze pro zisk nových karet a heroic pro souboj nad velmi silnou a zvýhodněnou umělou inteligencí.

- Hráč proti umělé inteligenci, kde si hráč vybere obtížnost normal, nebo advanced a hraje proti počítači. Především na tento mód hry se zaměřuje tato práce.

Nyní začíná samotná hra[16], která trvá od 5 do zhruba 20 minut. Na začátku systém automaticky vylosuje, který hráč začne a který pojede druhý. Oba hráči dostanou do ruky 3 náhodné karty z jejich 30 kartového balíčku a mají možnost karty vyměnit za nové náhodné karty z jejich balíku (*Mulligan*). Hráč který je na tahu druhý má o jednu kartu z jeho balíčku v ruce více a navíc dostane speciální kartu *The Coin*[8] viz obrázek 2.8.

Obrázek 2.8: Speciální karta – The Coin

Po mulliganu obou hráčů začíná regulérní hra. Oba hráči se střídají v 75 sekund časově omezených kolech, kdy musí stihnout vymyslet a zahrát jejich tahy, pokud vše nestihnou do časového limitu, tak jim zbytek akcí propadá a na řadě je další hráč. Na začátku každého kola si hráč lízne jednu kartu z vrchu jeho herního balíčku, dostane prázdný mana krystal, kterých může mít maximálně 10 a všechny jeho prázdné mana krystaly se naplní manou. Hráč manu využívá jako měnu pro zaplacení ceny karet pro jejich vyložení/zahrání. Karty hráč hraje, neboli vykládá na hrací plochu, která se dá rozdělit na následující sekce, které jsou zaznačeny v obrázku 2.9.

A – Hrdina hráče, B – Hrdina soupeře, C – Balíček karet, D – Balíček karet soupeře, E – Tlačítko END TURN na ukončení současného kola, F – Speciální schopnost, G – Speciální schopnost soupeře, H – Ruka s kartami, I – Ruka s kartami soupeře, J – Historie zahraných tahů, K – Místo pro přiložení zbraně, L – Místo označující mana krystaly a manu (dle barev, nebo čísel), M – Minion s útokem 6 a aktuálně 5 životy, který nemůže útočit, N – Minion s útokem 2 a 4 životy, co může útočit, O – Soupeřův minion, P – Plocha pro vykládání minionů, Q – Soupeřova plocha pro vykládání minionů, R – Místo označující mana krystaly a manu soupeře

Během svého kola může hráč vykládat miniony, kterých může mít současně na hrací ploše až 7, hrát kouzla, které nejsou početně omezené na tah, útočit již vyloženými miniony z dřívějších kol a nebo jednou za tah využít speciální schopnost. Kolo končí při stisknutí tlačítka *END TURN*, nebo po uplynutí časového limitu. Jakmile jeden z hráčů dosáhne životy na nulu, nebo níže, okamžitě prohrává hru a hra končí. V případě, že oběma hráčům klesnou životy na 0 a méně, hra končí remízou, která se každému hráči zobrazí jako prohra.

Je na každém hráči, jestli zahraje libovolnou kartu, nebo (ne)zaútočí minionem, čí použije svou speciální schopnost. Neaktivní hráč pouze čeká, než začne jeho kolo.

Další pojmy

V této podkapitole jsou vysvětlené některé pojmy, se kterými se ještě můžeme setkat

- Fatigue – Je situace ve hře, kdy si hráč lízne veškeré karty z jeho hracího balíčku. Pří pokusu o líznutí si další karty (např na začátku tahu) hráč dostane 1 poškození, při dalším pokusu se výše tohoto zranění o jedna zvyšuje. Tento mechanismus je ve hře proti neomezeně dlouhým hrám
- Armor – je bonusový život, který může přesáhnout klasických 30 daného hrdiny. Zobrazuje se šedým štítem s hodnotou armoru.
- Board – herní plocha
- Lethal – situace, kdy jeden hráč může ukončit hru
- Trade – situace ve hře, kdy se dva, nebo více minionů zabije mezi sebou
- Character – libovolný minion, nebo hrdina
- Pack – balík s náhodnými pěti kartami, které si hráč přidá do své kolekce
- Deck – herní balíček

Obrázek 2.9: Herní plocha

Kapitola 3

Metody umělé inteligence pro hru

V této kapitole se stejně jako v kapitole předchozí práce zabývá shrnutím současného stavu. V kapitole 3 nalezne čtenář informace o tom, co je to umělá inteligence, co je to stavový prostor a jak ho prohledávat, dále se pak dozví něco o strojovém učení a něco o umělé inteligenci aplikované v některých hrách.

Umělá inteligence[21], anglicky artificial intelligence, neboli AI je druh umělého myšlení, který vykazuje nějaké známky inteligence, tedy schopnost řešit, nebo rozpoznat problémy na určité úrovni, či abstraktně myslet. S umělou inteligencí se v dnešní době můžeme setkat téměř kdekoliv – v mobilních telefonech, počítačích, vysavačích, pračce, ale třeba i na internetu, například ve vyhledávačích, nebo překladačích. Taková umělá inteligence je schopná se vyvíjet a učit.[22]

V této práci se primárně zabýváme umělou inteligencí aplikovanou na hry, konkrétně pak na hru Hearthstone. Kdo se už někdy setkal s počítačovou hrou, tak už ví, že je možné hrát nejen proti dalším hráčům, například přes internet, ale že je také možné hrát proti počítači, který zastupuje a ovládá právě taková umělá inteligence.

3.1 Prohledávání stavového prostoru

Při aplikaci umělé inteligence na různé hry, je třeba definovat stavy, do kterých se můžeme dostat a to, jak se mezi různými stavy pohybujeme. Tyto pohyby označujeme jako přechody. Dále je třeba si určit počáteční stav a stav, který hru ukončí, označený jako koncový stav. Poté hledáme nejvhodnější cestu za pomocí přechodů vedoucí od počátečního stavu, do stavu koncového[21]. Dost často se můžeme potkat s velmi rozsáhlými stavovými prostory (například ve hře šachy, nebo Go), kde bychom nemohli v reálném čase hledat mezi všemi možnostmi tahu a proto je velmi důležité tyto možné varianty co nejvíce zjednodušit, neboli odstranit nevhodně ohodnocené varianty přechodů. Prohledávání stavového prostoru lze hodnotit pomocí 3 vlastností:

- Časová složitost – popisuje časovou složitost k vyřešení dané úlohy určitou metodou. Tato složitost je nepřímo závislá na dostupném výpočetním výkonu.
- Prostorová složitost – popisuje množství paměti potřebné k řešení dané úlohy
- Kvalita získaných výsledků – popisuje úplnost a optimálnost dané metody. Tzn existuje řešení úlohy a pokud ano, tak jestli je nejoptimálnější.

Prohledávání do šířky

Prohledávání do šířky, neboli breath-first search[19] je jeden ze základních algoritmů pro procházení stavového prostoru. Jeho princip je prohledávání celého stavového prostoru přes všechny uzly. Při prohledávání nevyužívá žádnou speciální heuristickou analýzu, ale prochází jeden uzel za druhým.

Při procházení stavového prostoru pomocí algoritmu prohledávání do šířky je využití klasická fronta, do které vkládá výchozí uzel. Následně se vyjme uzel z fronty a všechny jeho následníky vložíme do fronty, takhle se prochází celý strom, dokud se nenalezne cesta, což odpovídá hledanému řešení. Algoritmus pro prohledávání do šířky:

1. Zařaď do fronty výchozí uzel.
2. Vyjmi z fronty uzel a prozkoumej ho.
   (a) Pokud je řešení nalezeno, ukonči prohledávání a vrať výsledek.
   (b) Pokud řešení není nalezeno, zařaď do fronty následující uzly (dostupné z tohoto uzlu), které ještě nebyly prozkoumány.
3. Je-li fronta prázdná, všechny uzly byly prozkoumány. Ukonči prohledávání a vrať výsledek, že řešení nebylo nalezeno.
4. Není-li fronta prázdná, opakuj od bodu 2.

Prohledávání do hloubky

Prohledávání do hloubky, neboli depth-first search[19] je také jeden ze základních algoritmů pro procházení stavového prostoru pomocí metody backtracking, neboli metody zpětného sledování. Tento algoritmus se využívá efektivně pouze pro nízký objem dat a nebo když není známa efektivnější metoda.

Algoritmus prohledávání pomocí backtrackingu je následující:

1. Zařaď do zásobníku výchozí uzel.
2. Vyjmi ze zásobníku uzel a prozkoumej ho.
   (a) Pokud je řešení nalezeno, ukonči prohledávání a vrať výsledek.
   (b) Pokud řešení není nalezeno, zařaď do zásobníku následující uzly (dostupné z tohoto uzlu), které ještě nebyly prozkoumány.
3. Je-li zásobník prázdný, všechny uzly byly prozkoumány. Ukonči prohledávání a vrať výsledek, že řešení nebylo nalezeno.
4. Není-li zásobník prázdný, opakuj od bodu 2.

Další algoritmy

Vzhledem k tomu, že existuje mnoho algoritmů pro prohledávání stavového prostoru[11], zde jsou uvedeny některé další:

- Prohledávání do hloubky s omezením
- Prohledávání oběma směry
- Iterativní prohledávání do hloubky
- Uspořádané prohledávání

3.2 Support Vector Machines

Support Vector Machines, neboli SVM[17] je metoda strojového učení, kdy počítač je schopen se naučit z dostatečného množství vzorků (v případě této práce odehraných her), jak by se táhlo, kdyby nastala určitá situace, byť by tato situace nemusela nikdy nastat během fáze učení. Nevýhodou této metody je potřeba co největšího vzorku dat, ze kterého se má počítač naučit. Řádově se jedná minimálně o desítky tisíc odehraných her. Poté co se počítač naučí z již odehraných her, je možné aplikovat tyto znalosti na další partie.

Důležitou součástí SVM je metoda jádrové transformace, neboli kernel transformation prostoru příznaků dat do prostoru transformovaných příznaků obvykle vyšší dimenze. Tato jádrová transformace umožňuje převést původně lineárně neseparovatelnou úlohu na úlohu lineárně separovatelnou. Využívají se pro to různé jádrové transformace. Výhodou této metody (a jiných metod založených na jádrových transformacích) je, to že se transformace dají definovat pro různé typy objektů, například pro stromy, grafy, posloupnosti DNA.

3.3 Hry

V této kapitole se zaměřím na umělou inteligenci v počítačových hrách, konkrétně srovnání her lidského jedince proti počítači. Těchto her by se dalo najít mnohem více, zde jsou uvedené zajímavé příklady.

Šachy

Šachy jsou desková hra pro dva hráče, která se hraje na desce o velikosti 8x8 polí. Hráči se střídají na tahu, kdy ve svém tahu hráč táhne jednou z 16 figurek, které má od začátku hry tak, aby odstranil figurky soupeře a mohl dosáhnout vítězného stavu tzv šach-mat. Každý druh figurky má vlastní druh pohybu, typický pro ten druh figurky. V této hře neexistuje prvek náhody, ale pouze znalost a zkušenost hráčů. Tato hra, stejně jako jakákoliv jiná hra slouží primárně pro zábavu, rozvíjení mysli člověka, ale také jako soutěžení s jinými hráči.

Ve druhé polovině 20. století se zároveň s vývojem počítačů objevují i takové, které obsahují umělou inteligenci. Jeden z nich se jmenuje Deep Thought od IBM. Tento počítač poměří síly s nejlepším hráčem šachů na světě – Garry Kasparovem v roce 1989, avšak Garry Kasparov tento počítač absolutně rozdrtí a tím vyvrátí domněnky, že by počítač mohl být chytřejší, jako lidská mysl. V roce 1996 přijde IBM s novým počítačem s umělou inteligencí na šachy – Deep Blue. Tento souboj měl speciální pravidla, kdy se hrálo podle klasických pravidel šachů, ale navíc tým mohl počítač mezi jednotlivými hrami přeprogramovávat,

takže Garry Kasparov nehrál jen proti počítači, ale i proti týmu programátorů. Už první hra vedla k výhře počítače, kdy si správně vyhodnotil možnosti tahu a nenechal se zaskočit tahy soupeře. Tento zápas byl první na světě, kdy počítač s umělou inteligencí dokázal porazit mistra v šachu. Během dalších zápasů však byl počítač poražen a celkové skóre v tomto roce bylo 4:2 ve prospěch lidské mysli.

Další rok se konala odveta s ještě výkonnějším počítačem, který byl naučený podle všech předchozích her Garriho Kasparova s počítači. Ten však neměl k dispozici ani jednu hru, kterou nový, předělaný Deep Blue hrál. Kasparov první hru s přehledem vyhrál. Ve druhé ho počítač porazil. Další tři hry skončily remízou. V poslední hře si opět počítač správně vyhodnotil situaci a zahrál nečekané tahy, čímž Kasparova porazil. Ačkoliv Kasparov obvinil IBM, že zasahovala do počítače i v průběhu her, nic nebylo prokázáno. Takže v této době uvažovat o výhře počítače nad člověk je diskutabilní, ale rozhodně tato událost vedla k posunu v chápání a ocenění umělé inteligence na světě.[20]

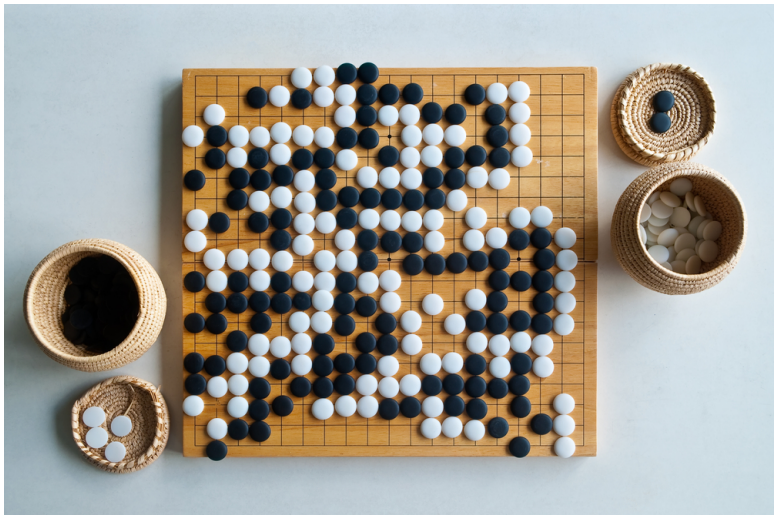
Obrázek 3.1: Ukázka ze hry šachy [12]

Go

Go[15] je desková hra pro dva hráče původem z Číny. Hra je stará již přes 4 tisíce let a stále je poměrně populární. Po celém světě ji hraje stovky milionů lidí. Hra se hraje na herní desce s 361 průsečíky, které se říká goban. Cílem této hry, je mít na konci hry více bodů za zajaté kameny soupeře. Hráči se střídají na tahu a pokládají jejich kameny tak, aby obklíčily kameny soupeře již položené na desce. S již vyloženými kameny se nesmí pohybovat.

Poté co byl poražen mistr světa v šachu Garri Kasparov počítačem, byl tento moment považován jako definitivní vítězství počítačové inteligence nad lidskou myslí. S tím však nesouhlasili příznivci hry go. Tvrdili, že v této hře se něco takového dohledné době nemůže stát. Překvapeni mohli být již v říjnu 2015, kdy počítač AlphaGo[18] měřil síly s evropským mistrem ve hře go a počítač zvítězil v poměru 5:0. Tento počin vzbudil pozornost po celém světě a již v březnu 2016 se tento počítač utkal s korejcem Leeem Sedolem, který byl mnoho let bez konkurence a proslavil se velmi netypickými herními postupy, které do té doby byly prakticky nemyslitelné. Lee tvrdil, že viděl hry počítače AlphaGo a byl si jistý, že tento počítač bez problému porazí. Už v první hře však počítač Leeho porazil, jelikož za dobu

mezi říjnem 2015 a březnem 2016 odehrál mnoho her, díky kterým se ještě lépe naučil tuto hru hrát. A během utkání 5 zápasů, se počítač učil doslova ze dne na den. Nakonec tato partie skončila v poměru 4:1 a touto výhrou se opět dokázalo, jak vývoj umělé inteligence postupuje vpřed.



Obrázek 3.2: Ukázka ze hry Go[23]

Hearthstone

Ve hře Hearthstone se také nachází několik druhů umělé inteligence. Avšak jejich úroveň nelze srovnávat s umělou inteligencí z předchozích kapitol a to protože tato umělá inteligence se přirozeně nerozvíjí, ani se neučí. Tím pádem se chová stále stejně předvídatelně. Ve hře Hearthstone můžeme nalézt následující typy umělé inteligence[10]:

- V hrách pro jednoho hráče, zde nalezneme každého z 9 hrdinů na úrovni začátečník s kartami, které má k dispozici i úplně nový hráč v této hře.
- V hrách pro jednoho hráče, zde dále nalezneme každého z 9 hrdinů na úrovni pokročilý s kartami, které má k dispozici středně pokročilý hráč.
- V adventurách, které vyšly jako rozšíření této hry nalezneme speciální umělou inteligenci se speciálními kartami, ve dvou obtížnostech – Normal a Heroic, kde se první obtížnost chová jako pokročilá obtížnost umělé inteligence ve hrách pro jednoho hráče a druhá obtížnost se chová stejně, jenom má k dispozici mnohem lepší karty, jejich počet, nebo je ještě jinak zvýhodněna. Touto mechanikou se vylepšují schopnosti počítače v různých hrách bez nutnosti vlastnit dostatečný výpočetní výkon.

V této práci se budeme při aplikaci a testování výsledků práce zabývat především základní umělou inteligencí, která má k dispozici stejné karty, jako úplně nový hráč. Tímto zajistíme jednoduché opakování tohoto testu.

Kapitola 4

Automatizace

V této kapitole je popsán algoritmus, který ohodnocuje a vybírá tahy pro karetní hru Hearthstone. Tento krok je důležitou součástí pro automatizaci uvedené hry, protože program by bez takového algoritmu nebyl sám schopen udělat patřičné tahy. Konkrétně je pak popsán jak funguje v programu, který je cílem této práce. Dále pak jaké jsou důsledky na danou hru, která využívá umělou inteligenci[24].

4.1 Typy

Ve hrách s umělou inteligencí se můžeme setkat s různým způsobem počítačového myšlení, kterého je více typů. Někdy může počítač využít výpočetního výkonu počítače a prohledat a vyzkoušet všechny možnosti tahu. Jindy zase může reagovat pouze na danou situaci, konkrétně na daném místě a vůbec neřeší zbytek situace ve hře. Případně je ovlivňován pouze malým okolím. Více informací je v následujících podkapitolách.

Programové chování

Programové chování je pouhá sekvence příkazů, které jsou poskládány za sebou a počítač je slepě vykonává. Jedná se o nejjednodušší variantu umělé inteligence. Počítač si nemůže vybrat z možností akcí, ale pouze strojově testuje jednu podmínku za druhou, až se dostane do konečného stavu. Je tedy logické, že tato umělá inteligence není příliš schopná řešit složitější problémy. Aby chování programu nebylo vždy totožné, vkládají autoři her do této umělé inteligence náhodný prvek. Ten poté toto rozhodování lehce ovlivňuje.

Také se zde objevují optimalizační metody, které hledají nejvhodnější strategie následujících tahů, nebo cíle vhodných útoků, nejrychlejší cestu k cíli atd. Počítač je schopen spočítat takových tahů tisíce za sekundu, proto se tato metoda může jevit jako vhodná, ale stále je velmi omezen ve výběru akcí.

Na lepší hratelnosti přidávají hrám tzv rámce, neboli frames, což jsou sice obyčejné programy, ale každý objekt ve hře je vykonává nezávisle na ostatních. Každý takový objekt má vlastní mechaniku a vykonává akce podle toho, co vidí. Vtrhne-li například nepřítel do vesnice, nastává v místě útoku zmatek, ale na druhé straně vesnice je obyvatelstvo v klidu. Nad rámci jednotlivých rámců jsou vyšší rámce, například taktický, nebo ekonomický. Tento hierarchický systém se lépe programuje a také vypadá mnohem lépe a přirozeněji.

Největší nevýhoda pevných programů je v tom, že se u složitých her prakticky není možné připravit na všechny varianty chování. A už vůbec ne bez chyb.

Pravidla

Tato varianta je velmi podobná programovému chování. Pravidla pro rozhodování jsou oddělená od samotného programu. Umělá inteligence si pak může vybrat, jaká pravidla aplikuje v závislosti na dané situaci. Nejdříve však tu situaci zhodnotí, může si prohlédnout celý herní prostor, herní mapu, atd. A potom si vybere nejvhodnější pravidlo, kterým bude na situaci ve hře reagovat. Také je možné, že se počítač z odehraných her bude učit nové mechaniky a taktiky pro danou hru, které si pak ukládá do databáze. Výhodou celé této varianty je možnost přepínat se mezi různými pravidly a přizpůsobovat své chování na požadavky ve hře. Například může mít vlastní pravidla pro útok a další pro obranu ve hře.

Vtištěná inteligence

V mnoha hrách je počítačová inteligence vytvářena nejen pravidly nebo programem, ale také vhodnými úpravami prostředí hry. Příkladem mohou být logické hry, ve kterých počítač nemá žádnou vlastní inteligenci, ale přesto není lehké ho porazit. Všechny rébusy a hádanky byly vytvořeny člověkem, v podstatě tedy hráč nesoupeří s umělou inteligencí počítače, ale inteligencí tvůrců hry. Častější je kombinace systému s doplňky, kde je počítačová inteligence hloupější, než by bylo vhodné, a tak je posilněna o příznivé náhodné jevy, jako jsou důležité suroviny, předem založené základny na nejvýhodnějších místech, terén vyhovuje jeho způsobu hry. Zatímco hráč má přesný opak. Tato úprava umělé inteligence není ovlivněna počítačem, ale je zakomponována ve hře samotné od autorů hry.

Adaptivní systémy

Tato umělá inteligence se umí učit znalosti během chodu programu. V současné době neexistuje mnoho her, které tuto variantu využívají. Umělá inteligence se poučí podle svých předchozích her a to podle toho jestli je poražena, nebo zvítězila. Získá z těchto her informace, které ji pomáhají být úspěšnější v dalších hrách. Tohle učení znemožní nepřátelskému hráči, obvykle člověku využívat stále stejné triky, kterými počítač dokáže pravidelně porazit. Naopak počítač se od hráče různým trikům může naučit.

Výhodou tohoto systému je, že počítač poté dokáže ovládat a manipulovat se všemi objekty hry zároveň, což hráč nedokáže. Pokud se takový počítač naučí dokonale hru hrát, je téměř nemožné ho poté porazit.

Je několik variant, jak se počítač může během her učit.

- Knowledge mining – Neboli těžba znalostí je technologie, která spolupracuje se systémem pravidel a tomu dodává nová pravidla chování. Využívá se převážně ve strategických hrách ze záznamů her. Počítač si záznam her vyhodnotí a přidá nová pravidla do své kolekce pravidel, naopak ty špatná, nebo málo funkční zase smaže. Takto se může učit velmi dlouhou dobu, až se z něj stane velmi kvalitní soupeř. Tato metoda je vhodná pro strategické hry.

- Casebased reasoning – Neboli usuzování na základě případů. Zde si počítač ukládá jednotlivé situace, to jak je někdo řešil a pak výsledek, který vyšel za použití daného řešení v dané situaci. Tato varianta je však náročná na počet her a také na výpočetní výkon počítače, než projde všechny položky v databázi a najde správnou, nebo alespoň nejvhodnější reakci. Tato metoda je vhodná pro logické hry.

- Neural networks – Neboli neuronové sítě. Tento způsob je nejrychlejší ze všech zde uvedených. Nejtěžší částí na neuronových sítích, je naučit počítač, jak se v jednotlivých situacích chovat. Tyto sítě se skládají z matematických neuronů, které simulují schopnosti lidského mozku. Ve fázi učení vznikají mezi umělými neurony nové spoje, ale nemění se počet vlastních prvků. Na rozdíl od předchozích metod, se jejich báze při učení zvětšují a snižují rychlost usuzování. V rychlosti vlastního hraní pak není žádný rozdíl mezi sítí naučenou a nenaučenou, a je proto ideální pro akční hry. Na druhou stranu mívají neuronové sítě tendenci ke špatnému učení a mohou si chvílemi počínat dost zvláštně a pro člověka nelogicky. Jejich síla roste s opakováním, a když se síť dobře naučí a poté nedá hráčům šanci.

Další varianty

Další variantou, jak vytvořit nějaké automatické chování ve hře je poskytnutí počítači veškeré informace o hře, tedy i takové, které by hráč hrající v pozici počítače normálně neměl. Jedná se tedy o podvádění, ale je také možné podávat počítači částečné informace při hře proti těžší variantě umělé inteligence, kde by implementace byla podstatně náročnější, než poskytnout skryté informace o soupeři. Další pomoci pro těžší varianty umělé inteligence je poskytnutí jim více zdrojů ve hře, mnohem lepší startovní pozici atd. V případě že jsou ve hře náhodné jevy, počítač je také může obrátit na svou stranu.

4.2 Ohodnocení karet

Abychom mohli úspěšně pracovat s kartami v této práci je třeba zjistit informace o kartách, což je popsané v dalších kapitolách. Poté co máme dostatek informací je třeba je nějak vyhodnotit. V prostředí Hearthstone se říká takovému hodnocení value, neboli hodnota.

Na začátku se zaměříme na samostatné miniony. Nejprve je třeba vybrat, jakého miniona vyvolat na stůl. Pro základní miniony s tzv vanilla atributy (bez jakýchkoliv speciálních schopností) je tento výpočet jednoduchý, protože není ovlivněn ničím jiným než číslem aktuálního kola, kartami, co jsou již vyložené na stole, a kartami, co má hráč na ruce, případně těmi, co si může líznout během nejbližších kol.

Nejprve by program měl vybrat správného miniona, kterého vyvolat na stůl. V ideálním případě to bude takový, který stojí přesně tolik many, kolik má hráč na dané kolo přiděleno. Tedy ve 4. kole se program snaží najít miniona za 4 many. V případě, že hráč nemá na ruce miniona za tolik many, jaké je aktuálně kolo, tak se program snaží najít kombinaci těch levnějších, aby využil co nejvíce zdrojů pro dané kolo. V případě, že má hráč na ruce více minionů bez speciálních schopností, co stojí stejně many, musí si program vybrat dle jejich útoku a životů tu nejvhodnější kombinaci.

Pokud má soupeř již nějaké miniony na stole a hráč se jich nemůže nijak efektivně zbavit, pak bude volit miniony s co nejvíce životy na stůl, nebo alespoň tak, aby útok soupeřových minionů byl nevýhodný pro soupeře. Pokud je stůl úplně prázdný, bude se program snažit vyvolat miniony s co největším útokem, jelikož další kolo budou moci udělat co největší zranění. Toto vyvolávání minionů je poměrně intuitivní a hráč s tím obvykle nemá problém, program tuto situaci musí vyhodnotit a na základě znalosti o hře se hráči pokusí poradit nejvhodnější tah.

Problém nastává při celkově velkém počtu minionů na stole a hlavně s minony, co mají speciální schopnosti. Mezi takové schopnosti může patřit například zvýhodnění *Spellpower*,

Taunt, ale i další schopnosti, které jsou popsané přímo na kartě. Takové schopnosti můžou být například vyléčení ztracených životů, zvýšení atributů vlastního miniona, líznutí karty navíc, aj. V těchto případech jsem hodnotil různé karty pro různé varianty hry. Tento počin je těžce spočítatelný, takže jeho výsledek je založen čistě na mých zkušenostech s touto hrou.

Dále je třeba si také uvědomit, že při vyvolání miniona nezáleží jen na tom, co má minion za atributy, ale také kolik stojí many, protože tato čísla dále figurují ve vzorci pro výpočet tzv value. A samozřejmě bonusové body za speciální schopnosti, které jsou individuální.

$$\text{value} = (\text{útok} + \text{životy} + \text{bonus})/\text{mana} + \text{mana}/4 \tag{4.1}$$

V rovnici 4.2 vidíme základní výpočet value v našem programu. Kde útok a životy jsou základní atributy miniona, kterého chceme vyvolat, nebo již máme vyvolaného na stole. Mana je cena dané karty a bonus je již zmíněná hodnota, která je pro každou kartu a situaci ve hře různá. Například karta *Consecration* od hrdiny Paladin, která udělí 2 zranění každému nepřátelskému charakteru celkem za 4 many. Tedy pokud by soupeř neměl žádného miniona na stole, toto kouzlo by udělilo 2 body zranění pouze nepřátelskému hrdinovi, což za 4 many není mnoho (za 4 many lze například zahrát miniona, co má 4 útok a 5 životů), Naopak v případě, že by soupeř měl na stole vyložených třeba 5 minionů, kde by 4 z nich měli 2 životy a méně, tak se toto kouzlo stává mnohem cennější, než bylo v předchozím příkladě. Také je nutno vzít v potaz, kombinaci s dalšími kartami vyloženými na stole, nebo s kartami na ruce. Právě tyto případy ovlivňuje proměnná bonus.

Obrázek 4.1: Chillwind Yeti[3]

Obrázek 4.2: Stormwind Knight[7]

Na obrázcích 4.1 a 4.2 je uveden příklad výpočet value a srovnání dvou minionů co se týče výhodnosti vyložení z ruky do hry.

První minion – Chillwind Yeti má 4 útok a 5 životů za 4 many bez dalších schopností. Takového miniona tedy podle vzorce ohodnotíme na:

$$(4+5+0)/4+4/4 = 3{,}25$$

Druhý minion – Stormwind Knight má 2 útok a 5 životů za 4 many. Navíc má tento minion speciální schopnost *charge*, kterou jsem ohodnotil bonusem, který se spočítá jako polovina z útoku, tedy celkový výpočet value:

$$(2+5+2/2)/4+4/4 = 3{,}00$$

V tomto případě tedy program vyhodnotí, že minion na obrázku 4.1 je výhodnější pro vyvolání do hry, než minion na obrázku 4.2.

Stejným způsobem jako hodnocení speciálních schopností minionů mohou být zbraně a nebo kouzla. Zbraň se může simulovat jako minion, ale každé kouzlo má svůj vlastní speciální efekt, který musí být nějak převeden na hodnotu value. Pokud by tomu tak nebylo, tak by program nebyl schopen využívat kouzla a tím pádem by to byla pro hráče karta, kterou nikdy z ruky nezahraje.

4.3 Robot

Robot v počítačových hrách, někdy též zkráceně bot je počítačem ovládaný hráč, který využívá umělou inteligenci pro volbu svých tahů. Také se ale o robotovi můžeme bavit ve smyslu programu třetí strany, který je určený pro nadstandardní automatizaci této hry.

Určitě je nutné uvést, že využívání programu, který nějak automaticky zvýhodňuje hráče před ostatními je z pravidla proti pravidlům dané hry. Takový robot může například hrát hru během doby, kdy hráč není vůbec u počítače, nebo mu umožňuje hrát a získávat herní měnu, herní zkušenosti, nebo další výhody v několika hrách zároveň, což by hráč nebyl schopen stíhat ovládat. Extrémním případem takového bota je pak program, co silně zasahuje do struktury počítačových her a například mění data hry, která jsou odesílána na server, tím pádem může například hráč mít přístup k surovinám navíc, nebo třeba 100% úspěšnost při střelbě ve hrách označovaných jako first person shooter, neboli FPS.

Boti nejsou povoleni ve hrách nejen kvůli zvýhodnění hráčů, kteří je využívají, nebo neoprávněným zásahům robota do hry, ale také protože kazí požitek z hry. Přeci jen to je umělá inteligence, co nemá žádné emoce a jen strojově vykonává úkony. Hráči mohou být poté deprimování, když se setkávají s jinými hráči, za které hraje pouze umělá inteligence, navíc když je tato umělá inteligence porazí. Přeci jen téměř každá hra má krom varianty pro více hráčů i variantu pro hráče jednoho. Takže kdyby se chtěl hráč utkávat proti počítači, zvolí tuto variantu.

Kapitola 5

Implementace

V této kapitole je popsána implementace samotného robota a dále jsou zde informace o logovacím souboru a práci s ním.

Výsledný program bude vypisovat hráči do konzole informace o stavu hry na levé straně a na pravé bude přidávat rady, jak zahrát tah. Cílem je dosáhnout co nejefektivnější výhry s minimálním úsilím uživatele tohoto programu. Proto může tento program sloužit jako návod, jak hrát hru Hearthstone.

Robot je implementovaný v jazyce C++, neboť je to jazyk, se kterým jsem se mohl potkat po většinu doby mého studia na FIT VUT a podporuje práci s řetězci, což bylo při implementaci důležité. Program primárně funguje na platformě Windows, který sám používám. Vzhledem k tomu že se jedná o program k počítačové hře, byla tato volba nejvhodnější. Další kroky by mohly vést k rozšíření tohoto programu i na jiné platformy, ale to není cílem této práce. Na obrázku 5.1 je vidět výstup hotového programu.

Obrázek 5.1: Výsledný program

5.1 Ovládání

Ovládání programu je jednoduché a poměrně intuitivní. Pro základní využití robota stačí program spustit, spustit hru a pak už jen stačí pouze číst informace, které vypisuje do konzole a v ideálním případě se řídit jeho pokyny. Pokud hráč udělá chybu, nebo se překlikne a zahraje jiný tah, než mu byl doporučen programem, tak pokud je to možné (hra není ukončena výhrou, nebo prohrou), tak program spočítá a navrhne nové možnosti. To se děje po každém odehraném tahu.

Program při stisknutí určitých kláves vypisuje stav na stole (*R pro hráče, E pro soupeře*), karty na ruce (T), nebo všechny možné varianty tahu (W), případně (V) pro všechny možné varianty tahu i se všemi variacemi cílů, pokud při hraní dané karty je třeba vybrat nějaký cíl. Program se ukončí stisknutím klávesy (Q). Všechny tyto ovládací klávesy jsou však nepovinné. Samotná hra se pak na PC ovládá myší, případně poklepáním, či přetažením karet na dotykové obrazovce.

Klávesy pro výpis nejsou nutné a slouží pouze k ověření správného zpracování dat, protože veškeré informace vidí hráč také na obrazovce hry. Klávesy pro výpis všech možných tahů pro dané kolo však z obrazovky na první pohled zjistit nejdou. Hráč sice vidí, kterou kartu lze zahrát a kterou ne (karty, které mají zeleně podsvícené pozadí je možné v aktuálním kole zahrát), ale nikde nevidí seznam všech možných cílů těchto karet. Tento příklad je zobrazen na obrázku 5.2.

Obrázek 5.2: Ukázka ze hry Hearthstone – karty, se kterými lze hrát

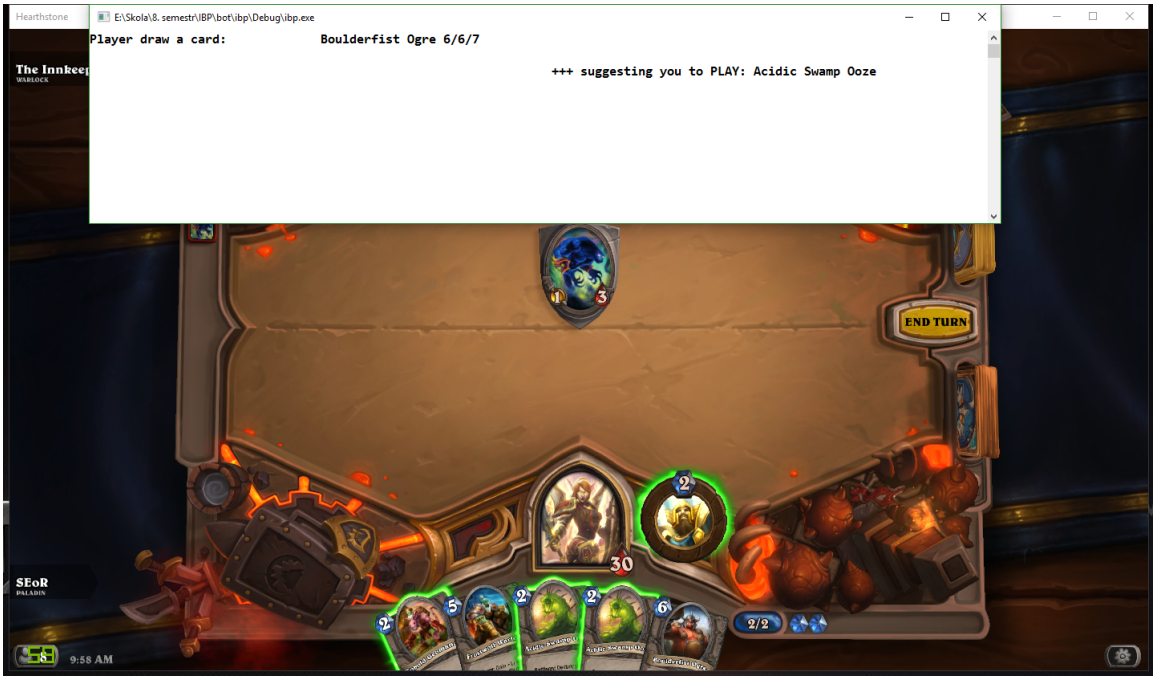
Pokud hráč chce hrát kartu nějakou kartu, nejdříve si vybere se kterou chce táhnout jednou na ní klikne myší a následně přetáhne myš na pozici, kam chce kartu zahrát. Také je možné držet levé tlačítko na myši pro stejný efekt. V případě že hráč chce zahrát miniona,

přetáhne kartu na pozici mezi jiné miniony, ti se mu graficky posunou a hráč tak může miniona zahrát na vybranou pozici. V případě že si to rozmyslí, může pravým tlačítkem, nebo kliknutím mimo hrací plochu tuto akci anulovat. Na obrázku 5.3 je ukázáno, jak se vybírá pozice na stole při vyložení miniona pro situaci z obrázku 5.2.

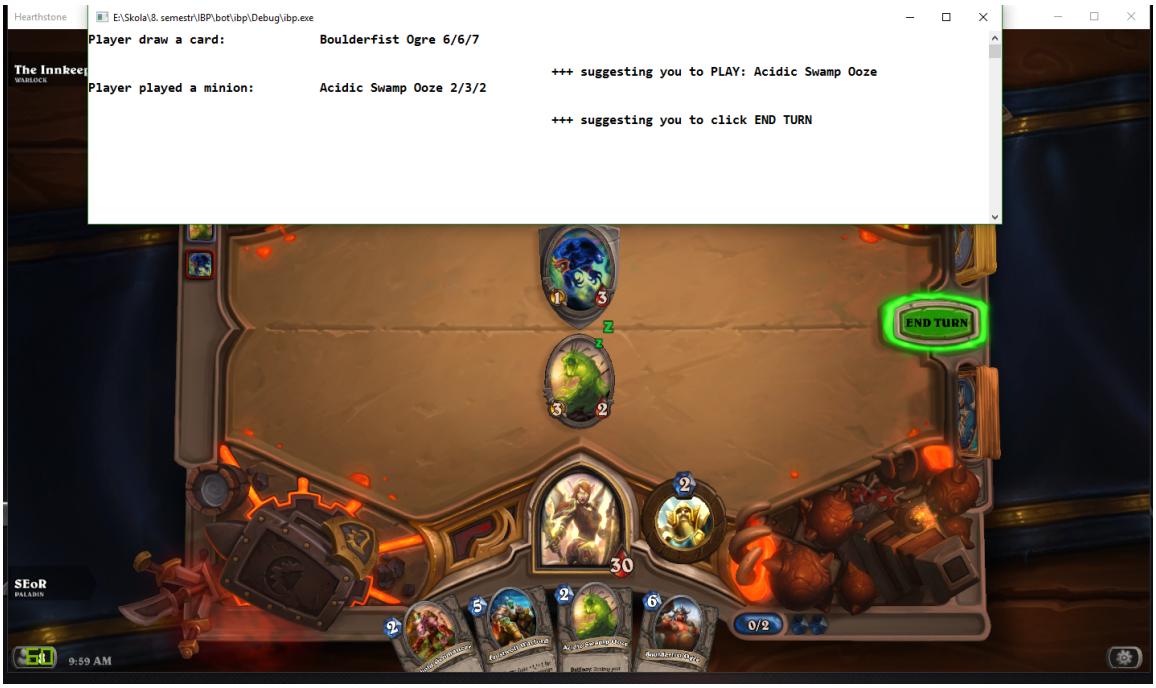
Na obrázku 5.4 je vidět příklad ze hry s výstupem programu, který hráči navrhuje tah. Dále hráč tento tah provedl stejným principem, jako na předchozích obrázcích a pak na obrázku 5.5 je vidět výsledek s dalšími informacemi o hře a o informaci, kterou program informuje hráče jak hrát dál.

Obrázek 5.3: Ukázka ze hry Hearthstone – jak vybrat pozici

V případě kouzel, nebo speciálních schopností, které mají konkrétní cíl, si hráč tento cíl vybere stejně, jako pozici na stole. Jediná věc, co je tu jinak, je že se karty neposouvají, ale drží pozici na stole. Pokud hráč zahraje kouzlo, které nemá určený cíl, stačí tuto kartu posunout ze své ruky, aby byla aktivovaná její vlastnost. Na obrázcích 5.6 a 5.7 je srovnání dvou takových kouzel. Útoky minionů probíhají stejně, jako hraní karet kouzel z ruky, jenom že minioni musí být vyložení na stole.



Obrázek 5.4: Ukázka před zahraným tahem



Obrázek 5.5: Ukázka po zahraném tahu

Obrázek 5.6: Ukázka kouzla s výběrem cíle[1]

Obrázek 5.7: Ukázka kouzla bez výběru cíle[4]

5.2 Práce s kartami na ruce

Pro práci s kartami na ruce, která je implementována jako vektor struktury musíme zvážit, kdy je třeba zapisovat informace (zisk karty do ruky) a kdy je z ní smazat, aby byl zachován správný počet a pořadí zbývajících karet na ruce. Zisk nové karty do ruky můžeme rozdělit na několik případů. Nejobvyklejší cesta je líznutí nové karty z balíčku do ruky. Tento jev se eviduje jednoduše přes konkrétní hledané podřetězce a karta se uloží na konec struktury, tak aby odpovídalo chronologické pořadí zisku karet v ruce.

Další možností jak získat kartu jinak, než obyčejné líznutí z herního balíčku je ze hry do ruky. V lepším případě některé karty mají speciální efekt, který vrací tyto karty do ruky, v tom horším tyto karty vrací karty jiné do ruky. Zde je ošetřeno několika podmínkami, aby se program nepokoušel zapsat data do neexistujícího indexu, nebo vůbec rozpoznal, že nějaká karta do ruky přišla, případně odešla.

Třetí variantou je zisk karty tzv odnikud. Tento jev se objevuje například při mechanice *Discover*, nebo *Gain*, která nebyla dříve vysvětlena, ale nejedná se o žádnou speciální mechaniku, pouze hráč dostane (například do ruky) nějakou konkrétní kartu, nebo kartu z nějaké kolekce jiných karet.

Po získání libovolné karty do ruky je třeba zapsat její atributy do struktury, aby se následně program mohl co nejpřesněji rozhodovat, jak je pro něj která karta výhodná zahrát v daném tahu. Do struktury ukládá pouze cenu karty, v případě miniona jeho útok a životy. Kouzla a někteří minioni jsou navíc popsáni hodnotami, které upravují jejich hodnotu ve hře. Tento krok je speciální pro každou kartu zvlášť a vychází z mých zkušeností s touto hrou. Jelikož se u většiny těchto karet nedá získat jiný přesnější popis jejich efektu, pokládám tento krok za nejpřesnější,

Při jakékoliv ztrátě karty z ruky snižuji počítadlo karet `cards_in_hand` o počet karet, které z ruky odešly, naopak při zisku karet toto počítadlo zvyšuji. V případě, že by logovací

soubor byl dokonalý a nenesl v sobě chyby popsané v další kapitole, tak by tento krok nebyl třeba a pro počet karet bychom na ně přistupovali přes volání `hand.size()`, ale jelikož někdy logovací soubor kartu nezaznamená správně, tak zde je náhradní mechanika, která většinou dokáže objevit problém a chybu co nejlépe opravit. Tento jev se pak ovládá při posouvání karet v ruce, což lze vyčíst z logovacího souboru.

Speciální práce s kartami též nastává při začátku hry, kdy si hráč líže karty do pozic na ruce v náhodném pořadí (jak je vyhodnotí hra), proto nelze postupovat s klasickou prací s kartami na ruce, ale nejdříve pomocí `hand.resize()` zvětšíme velikost hráčovi ruky na požadovanou velikost (3 pro začínajícího hráče, 5 pro hráče druhého na řadě) a karty srovnáme a zapíšeme, dle jejich správného pořadí. Poté již postupujeme klasicky, jak bylo vysvětleno v předchozích odstavcích. S další problémovou složkou se můžeme setkat při mulliganu, který nám před začátkem hry některé karty zahazuje a výměnou za ně jsou karty nové, které jsou opět v náhodném pořadí. Proto, než hra projde fází mulligan, je přepínač `bool mulligan_done` nastaven na `false`. Jakmile se tak stane, je do konce hry nastaven na `true` a karty na ruce se chovají opět normálně a program s nimi tak pracuje. Tento přepínač dále využívají jiné části programů pro správnou funkci. Tento přepínač se resetuje po skončení hry, nebo opětovném zapnutí hry a nebo programu, pokud je hra vypnutá.

5.3 Práce s miniony na stole

Ačkoliv se může zdát, že práce s miniony na stolech je identická, nebo velmi podobná práci s kartami na ruce, je tomu právě naopak. Jediné co mají společné, je uložení jako vektor struktur, který je zde využit pro snadné posouvání odstraněných minionů ze stolu, nebo karet na ruce. Pozice minionů na rozdíl od karet na ruce se mění též vyvoláním jiného miniona před jiného a nebo mezi dva další. Minioni jsou očíslovaní indexy jedna až sedm, při pohledu hráče z levé strany na stranu pravou. Maximální index je tedy takový, jaký je počet minionů u hráče na stole. Pro hráče i soupeře je vlastní struktura, která eviduje miniony na dané straně stole.

U hráče můžeme rozdělit změnu počtů minionů na stole také na několik případů. První z nich je případ, který vyvolává miniona buď špatně zapsaným řádkem v logovacím souboru, nebo je minion vyvolán nějakým speciálním efektem (existují zde jiní minioni, kteří poté co se zahrají, nebo odstraní ze stolu, tak vyvolají miniony jiné). Typickým příkladem může být například raciální schopnost hrdiny Thrall, který vyvolává jeden ze 4 náhodných totemů. Jakmile tento minion přijde na stůl, nejdřív z dalších informací v logovacím souboru program zjistí na jakou pozici minion přišel a podle toho do struktury zařadí další prvek. Jakmile se tak stane, tak zjisti aktuální útok a životy a zapíše je ke příslušné kartě na příslušné pozici. Pro práci z miniony máme zase kontrolní proměnnou `player_minions` a `enemy_minions`, které slouží ke stejnému případu, jako `cards_in_hand`.

Dalším typem práce s miniony hráče je unikátní proměna miniona v jiného. Nejedná se tedy o přímé odstranění karty, či vrácení na ruku, ale minion se po efektu jiného miniona, nebo kouzla změní na úplně novou kartu. Zde musíme nejdříve odstranit původního miniona z dané pozice, tu ovšem musíme nejprve zjistit. Poté se na pozici přidá typickým způsobem nový minion. Tato situace vzniká například při kouzlu *Sheep*, které vlastní mág Jaina Proudmoore, nebo *Hex*, které vlastní šaman Thrall. Pro tyto specifické karty jsou vytvořeny vlastní části programu, které jsou ještě zvlášť pro hráče a jeho soupeře.

Další variantou pro práci s miniony je klasické vyvolání z ruky. Hráč si opět vybere cílovou pozici, která se zaznamená do logovacího souboru a zapíší se veškeré informace

jako v předchozích odstavcích. Jediná výjimka je nutnost odstranit kartu z ruky, aby se ve vektoru, který schraňuje informace o kartách na ruce, simulovalo zahrání karty. Podobně to funguje i u soupeře, ale u něj logovací soubor neobsahuje pozici, odkud jde karta z ruky.

Při smrti minionů z libovolné části stolu se opět odstraní z vektoru a smažou se i další podpůrné data ke konkrétnímu minionu. Posledním typem práce s miniony je změna jejich atributů útok a počet životů, případně změna celkového počtu životů. Tyto změny jsou také přes speciální klíčové slova vyhledány a zpracovány tak, aby příslušnému minionovi zapsali změnu atributů tak, že výsledek bude stejný jako je pozice a atributy miniona na stole ve hře. Toto se děje například při útoku jednoho miniona na druhého. Mějme modelovou situaci, kdy máme 2/4 miniona A, tzn 2 útok a 4 životy a na druhé straně 1/3 miniona B. Hráč na tahu se rozhodne, že minion A (nebo minion B) zaútočí na druhého. Oba si najednou způsobí zranění rovna velikosti útoku nepřátelského miniona, tedy minion A ztratí jeden život a zbývají mu 3 a v logu má zaznamenaný 1 bod zranění. Minion B dostane 2 body zranění a tím pádem mu zbyde jeden život. Tyto udělené zranění jdou ve hře zvrátit efektem *Healing*, proto tyto změny musí být možné také vrátit do původní hodnoty. Tyto úpravy statistik ještě ovlivňují další efekty karet, které přidávají atributy minionům navíc. Například takový Shattered Sun Cleric přídá trvale cílovému přátelskému minionu +1 útok a +1 život. Na tento jev slouží speciální proměnná ve struktuře minionů `bonushp` a v ní se ukládá rozdíl daného atributu oproti normální hodnotě. Tedy pokud v našem modelovém příkladě dostane minion A +1 útok a +1 obranu, zvýší se jeho atributy na 3/5, ale jelikož má už nějaké body zranění, musí se jeho aktuální počet životů změnšit na 4 s možností vrátit jeden život zpátky.

Poslední kartou, který se vykládá z ruky je zbraň a nebo kouzlo. Zbraň se chová jako minion, jen je za něj označený samotný hrdina s číslem pozice 0. V případě kouzla, které má cílovou pozici nezobrazenou, i když ovlivní miniona na nějaké konkrétní pozici. Kouzlo se poté odloží na hřbitov a program zaeviduje výsledky na ploše, nebo ruce zase do příslušných proměnných.

5.4 Testování

Poté co byl program implementován bylo ho třeba otestovat. Nejprve jsem testoval, jestli je program schopen porazit umělou inteligenci ve hře s herním balíčkem obsahující také základní karty. Zde měl program přes 80% úspěšnost, což je v této hře považováno za úspěch.

Dalším krokem bylo otestovat tento program proti těžší umělé inteligenci ve hře Hearthstone. Zde už program narazil na nedostatek karet a ačkoliv dokázal zahrát některé hry, kdy těžšího soupeře porazil, Jednalo se však spíše o výjimky. Vítězství zazanamenal pouze v 17% případů. Zde by se programu dalo pomoci zlepšením umělé inteligence, nebo sestavením herního balíčku z pokročilých karet.

Třetím testovaným aspektem bylo srovnání hry programu vůči tomu, jak by se rozhodoval člověk. Zde byla drtivá většina případů totožná, ale ve specifických stavech hry se tento program choval strojově a nedokázal odhadnout situaci, jako by to udělal hráč. Tento problém by se mohl vyřešit také pokročilejší umělou inteligencí, založenou na strojovém učení, kdy by si pamatoval výsledky v této situaci. Ojediněle jsem se setkal s tím, že program přišel na možnost ukončit hru o jedno kolo dříve, než by takovou hru ukončil člověk.

Kapitola 6

Závěr

Cílem této práce bylo zaměřit se a pochopit umělou inteligenci aplikovanou v hrách a poté ji zkusit sám aplikovat na počítačovou hru. Tyto cíle byly splněny tak, že jsem se nejprve zaměřil na pochopení umělé inteligence aplikované v hrách a poté ji sám aplikoval na počítačovou karetní hru Hearthstone. Výsledkem toho je funkční program, který je schopen uživateli poskytnout textové informace o tom, jak hrát v daném kole tak, aby měl co největší šanci na výhru v rámci pravidel této hry.

V průběhu vývoje programu a i po jeho dokončení bylo třeba otestovat funkčnost programu a jeho kvalitu rozhodování. Program jsem porovnával proti umělé inteligenci obsažené ve hře a zjišťoval, jestli je program schopen tuto umělou inteligenci porazit. Výsledkem bylo více než 80% úspěšnost, což je v této hře velký úspěch. Dále jsem porovnával jestli rozhodování programu je identické s rozhodováním člověka v dané situaci. Ne vždy tomu tak bylo, avšak v této hře není jednoduché určit s absolutní jistotou, který tah je správný, a který naopak není.

Také byly splněny všechny body zadání. Nejprve bylo třeba nastudovat literaturu, pravidla a možnost implementace umělé inteligence na tuto hru. Poté bylo nutné zpracovat informace z hry, aby se mohl algoritmus automatizace implementovat. Program využívá kombinatorické zkoušení všech možných variant tahu a dle algoritmu ohodnocení jednotlivých tahů vybírá ten nejvhodnější pro hráče. Tento algoritmus ohodnocování tahů je založen nejen na výpočtu z číselných hodnot, ale i na osobní zkušenosti s danými situacemi.

Tuto implementaci podporuje hra samotná skrze logovací soubor, proto si myslím, že tato varianta byla nejvhodnější. Avšak mohou nastat problémy se zápisem do tohoto souboru, a pak může mít program problém rozpoznat situaci, ve které se hra nachází.

Práce mi ukázala, že počítačové myšlení, tedy umělá inteligence je poměrně složité téma samo o sobě a docílit toho, že se počítač se rozhoduje správně je ještě těžší. Na tuto práci by se v budoucnu dalo navázat zpracováním dalších karet, které v práci nebyly zpracovány, nebo tyto karty ještě nebyly vydány. Také by se mohla umělá inteligence programu vytrénovat pomocí algoritmů na strojové učení. Avšak pro tuto variantu je třeba sehnat dostatečně velký vzorek odehraných her, což v případě této práce nebylo časově možné.

Literatura

[1] *Obrázek karty Assasinate.* [Online; navštíveno 18.5.2016].
URL https://hydra-media.cursecdn.com/hearthstone.gamepedia.com/9/9d/Assassinate%28568%29.png

[2] *Obrázek karty Boulderfist ogre.* [Online; navštíveno 28.4.2016].
URL https://hydra-media.cursecdn.com/hearthstone.gamepedia.com/f/fd/Boulderfist_Ogre%2860%29.png

[3] *Obrázek karty Chillwind Yeti.* [Online; navštíveno 17.5.2016].
URL https://hydra-media.cursecdn.com/hearthstone.gamepedia.com/e/ea/Chillwind_Yeti%2831%29.png

[4] *Obrázek karty Frost Nova.* [Online; navštíveno 18.5.2016].
URL https://hydra-media.cursecdn.com/hearthstone.gamepedia.com/0/00/Frost_Nova%2849%29.png

[5] *Obrázek karty Mirror Entity.* [Online; navštíveno 30.4.2016].
URL http://hydra-media.cursecdn.com/hearthstone.gamepedia.com/0/0f/Mirror_Entity%28569%29.png

[6] *Obrázek karty Shadow Bolt.* [Online; navštíveno 30.4.2016].
URL https://hydra-media.cursecdn.com/hearthstone.gamepedia.com/3/34/Shadow_Bolt%28647%29.png

[7] *Obrázek karty Stormwind Knight.* [Online; navštíveno 17.5.2016].
URL https://hydra-media.cursecdn.com/hearthstone.gamepedia.com/f/f0/Stormwind_Knight%28603%29.png

[8] *Obrázek karty The Coin.* [Online; navštíveno 10.10.2015].
URL https://hydra-media.cursecdn.com/hearthstone.gamepedia.com/c/c9/The_Coin%28141%29.png

[9] *Obrázek karty Tundra Rhino.* [Online; navštíveno 28.4.2016].
URL https://hydra-media.cursecdn.com/hearthstone.gamepedia.com/1/15/Tundra_Rhino%28162%29.png

[10] *Practical mode.* [Online; navštíveno 18.5.2016].
URL http://hearthstone.gamepedia.com/Practice_mode

[11] *prohledavani stavoveho prostoru.* [Online; navštíveno 18.5.2016].
URL http://portal.matematickabiologie.cz/res/f/prohledavani-stavoveho-prostoru.pdf

[12] *Šachy albi*. [Online; navštíveno 16.5.2016].
URL https://im9.cz/iR/importprodukt-orig/f4f/f4fa3b4c6013a80e349445de90dd8886.jpg

[13] *Card*. 2016, [Online; navštíveno 1.5.2016].
URL http://hearthstone.gamepedia.com/Card

[14] *Standard format*. 2016, [Online; navštíveno 3.5.2016].
URL http://hearthstone.gamepedia.com/Standard_format

[15] association, B. G.: *How to play*. 2016-03-21, [Online; navštíveno 17.5.2016].
URL http://www.britgo.org/intro/intro2

[16] Blizzard: *How To Play*. 2016, [Online; navštíveno 13.9.2015].
URL http://us.battle.net/hearthstone/en/game-guide/

[17] Burges, C. J.: *A Tutorial On Support Vector Machines for Pattern Recognition*. [Online; navštíveno 16.5.2016].
URL http://cmp.felk.cvut.cz/cmp/courses/recognition/Lab_archive/RPZ_07-08w/svm2/SVM_tutorial-Burges98.pdf

[18] Hassabis, D.: *AlphaGo: using machine learning to master the ancient game of Go*. 2016-01-27, [Online; navštíveno 17.5.2016].
URL https://googleblog.blogspot.cz/2016/01/alphago-machine-learning-game-go.html

[19] Kolář, J.: *Teoretická informatika*. Česká technika - nakladatelství ČVUT, 2009, ISBN 978-80-01-04331-8.

[20] Kušová, T.: *Den, kdy počítač poprvé porazil člověka*. 2011-10-02, [Online; navštíveno 16.5.2016].
URL http://www.novinky.cz/veda-skoly/historie/224101-den-kdy-pocitac-poprve-porazil-cloveka.html

[21] Mařík, V.: *Umělá inteligence (1)*. Academia, 2000, ISBN 80-200-0496-3.

[22] Popelka, O.: *Umělá inteligence*. 2015-01-10, [Online; navštíveno 15.5.2016].
URL https://akela.mendelu.cz/~xpopelka/cs/ui/uvod/

[23] Poroong, S.: *goban*. [Online; navštíveno 14.5.2016].
URL http://blogs.discovermagazine.com/crux/files/2016/01/shutterstock_342026228.jpg

[24] Rybka, M.: *Umělá inteligence v počítačových hrách*. [Online; navštíveno 17.5.2016].
URL http://www.krokodyyl.wz.cz/inteligence.php

[25] Statista: *Number of Hearthstone: Heroes of Warcraft players worldwide as of April 2016*. 2016, [Online; navštíveno 1.5.2016].
URL http://www.statista.com/statistics/323239/number-gamers-hearthstone-heroes-warcraft-worldwide/

[26] Whirthun: *Join Us for the 2015 Hearthstone World Championship!* 2015-01-30, [Online; navštíveno 10.5.2016].
URL http://us.battle.net/hearthstone/en/blog/17776013/join-us-for-the-2015-hearthstone-world-championship-1-30-2015

Přílohy

Seznam příloh

Příloha A

Instalace a spuštění

Ke úspěšné instalaci a spuštění hry je třeba splnit minimální požadavky:

1. Vlastnit dostatečně silný hardware:
   - PC s Windows XP, nebo novější, procesor Intel Pentium D nebo AMD Athlon 64 X2, grafiku NVIDIA® GeForce 6800 (256 MB) nebo ATI Radeon X1600 Pro (256 MB), 2 GB RAM a 3 GB místa na disku
   - Mac OS X 10.9.5, Intel Core 2 Duo, NVIDIA GeForce 8600M GT nebo ATI Radeon HD 2600 Pro, 2 GB RAM a 3 GB místa na disku
   - iOS 6.0, iPad Air, iPad 2, iPad mini, iPhone 4s, iPod Touch 5
   - telefon, nebo tablet s Android 4.0, 1 GB RAM a 2 GB vnitřní paměti
2. Mít přístup na internet
3. Zaregistrovat se na *http://eu.battle.net/en/*
4. Nainstalovat Battle.net klienta
5. Nainstalovat hru Hearthstone

Poté co jsou všechny požadavky splněni, stačí spustit battle.net klient, přihlásit se a spustit hru v nabídce v levém sloupci.

Příloha B

Obsah CD

- Technická dokumentace
- Zdrojový kód
- Pomocné datové soubory
- Video funkčního programu
- Readme soubor

Příloha C

Programová implementace

Získávání dat ze hry

Veškeré nastavení se zapíše při zapnutí robota, takže uživatel pouze spustí aplikaci a ta za něj v případě dostatečných práv udělá vše potřebné. Nyní poté, co hráč spustí Hearthstone a vstoupí do nějaké hry, tak se každá akce, kterou udělá na hrací ploše, nebo se automaticky pustí z nějakého spouštěče, tak se zapíše do logovacího souboru (logu), která se nachází ve složce s nainstalovanou hrou. Pří instalaci do nabídnuté složky můžeme najít logovací soubor na adrese `C:\Program Files(x86)\Hearthstone\Hearthstone_Data\output_log.txt`. Tyto implicitní adresy se mohou měnit dle zadaných parametrů z příkazové řádky při spuštění programu.

Nyní začíná práce samotného robota. Na začátek je nutné podotknout, že logovací soubor má opravdu velmi hodně řádků a každou další akcí velmi rychle narůstá jejich počet. Jen po spuštění hry, bez jakékoliv další akce má logovací soubor přes 5 000 řádků. Jedna spíše kratší hra má pak řádků okolo 50 000. Naopak hry s pomalými balíčky mají řádků až 350 000.

Program načte řádek z logu a hledá některé shodné řetězce. Jejich princip je vysvětlen v následující kapitole. V případě, že není nalezen žádný vhodná řetězec k parsování, program si uloží do pomocné proměnné počet znaků daného řádku a přičte je do proměnné `char_num`, která v sobě nese počet všech již prozkoumaných řádků. Poté se program na 100 milisekund uspí a v následném vzbuzení v logovacím souboru přesune ukazatel o počet znaků, které již zpracoval. Tím se tedy přesune k novému řádku, pokud již existuje, jinak zase čeká desetinu sekundy, než se znovu spustí. V případě, že by nebylo využito ukládání počtu již načtených znaků a následný přesun ukazatele v souboru, stouplo by vytížení procesoru zhruba o dva řády a se zvyšujícím se počtem řádků by program přestal stíhat. Počet znaků z každého řádku program ukládá do již zmíněné proměnné `char_num`, která je 64 bitová, což zaručuje dostatečný rozsah počtu znaků. Poté co se hra vypne a program je znovu spuštěn, vymaže předchozí logovací soubor pro rychlejší a přesnější práci. V případě shody konkrétního řetězce, který je hledán přes cyklus `for`, který iteruje pro všechny hledané řetězce zapsané v poli `event_list[]`, se provedou příslušné akce, popsané v poli `action_list[]`. Tyto akce jsou přiřazené k jednotlivým prvkům `swich-case` ve funkci `look_for_action()`.

Pomocné datové soubory

Při práci s logovacím souborem získá program různé data, ale aby je mohl využít, potřebuje k tomu ještě další zdroj. Ze samotných dat hry se mi nepodařilo získat informace o kartách, proto jsem si tedy vytvořil vlastní pomocné soubory, které nesou data o jednotlivých kartách. Jedná se o soubory `name.txt, mana.txt, health.txt a attack.txt`.

V případě že program nalezne na právě zpracovaném řádku shodu s klíčovým řetězcem, který nějak popisuje část události ve hře, tak se v jíž zmíněném `swich-case` najde příslušná akce, která se provede. Obvykle je třeba získat nejdříve jméno karty, což se v programu děje pomocí funkcí `string::substr()` a `string::length`. Takto získané jméno karty se následně zašle do funkce `get_cardinfo(string name)`, která vrací číslo řádku pro danou kartu. Tato hodnota se využije jako vstup do funkce `get_card_stats(int i, string value)`. Výstupem z této funkce jsou již konkrétní hodnoty, které odrážejí opravdové atributy jednotlivých karet. Další funkcí, která slouží k parsování a práci s daným řádkem je `get_player_number(string line)`, která vrací pořadí daného hráče, který bych hledanou kartou ovlivněn.

Hlavní hledané řetězce

V této kapitole jsou popsány stěžejní řádky, které program hledá a jejich správné zpracování a výklad se následně odráží na chodu celého programu.

Vzhledem k tomu, že logovací soubor neudržuje žádné informace o aktuální pozici karet v ruce, nebo na stole, tak je třeba si tyto údaje vést a pravidelně aktualizovat. V programu nalezneme strukturu `CardInHands`, která obsahuje veškeré informace o kartách, které hráč drží na ruce. Další strukturou je `MinionsOnTable`, která je vytvořena jak pro záznam pozice karet na stole jak pro hráče, tak i pro soupeře. I zde si log nenechává žádné informace o aktuální poloze karet, takže jakmile je změna v pozici na stole, je třeba ji správně zapsat do struktury. Také se v této struktuře nachází aktuální hodnoty životů a útoků charakterů.

Využil jsem zde knihovnu pro oboustranný seznam `#include <deque>`, která v programu umožňuje například smazat prvek uprostřed herního stolu. Poté se pořadí prvků (karet zapsaných ve struktuře) automaticky posune a nemusí se hlídat meze pole s využitím této knihovny. Zde se vzácně objevila nečekaná chyba, která vedla k pádu programu. Ačkoliv jsem všechny objevené chyby odstranil a zajistil tak co nejplynulejší chod programu, přesto se můžou vyskytnout chyby, které vedou k fatálnímu důsledku.