VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MULTIPLATFORMNÍ KARETNÍ HRA S UMĚLOU INTELIGENCÍ

MULTIPLATFORM CARD GAME WITH ARTIFICIAL INTELLIGENCE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MARTIN TREJTNAR**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL MATÝŠEK**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Trejtnar Martin**
Program: Informační technologie
Název: **Multiplatformní karetní hra s umělou inteligencí**
Multiplatform Card Game with Artificial Intelligence
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s obecnými postupy tvorby her v enginu Unity. Zaměřte se na žánr karetních her a na multiplatformní vývoj.
2. Prostudujte a popište algoritmy a přístupy vhodné pro implementaci umělé inteligence v karetních hrách. Prostudujte dostupné nástroje a knihovny umožňující implementaci inteligentních agentů v enginu Unity.
3. Navrhněte karetní hru pro více hráčů s možností hry na různých zařízeních. Vyberte/navrhněte metodu umělé inteligence vhodnou pro tuto hru.
4. Implementujte navrženou aplikaci pro alespoň dvě platformy (například PC a mobilní zařízení). Implementujte inteligentní protihráče.
5. Zhodnoťte dosažené výsledky, vytvořte krátké prezentační video a diskutujte možnosti budoucího vývoje.

Literatura:

- Dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, rozpracovaný bod 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Matýšek Michal, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Tato práce se zabývá umělou inteligencí v karetních hrách. Cílem je implementovat multiplatformní hru tohoto žánru v herním enginu Unity, shrnout možné přístupy vytváření inteligentních protihráčů a pro zvolenou hru navrhnout a popsat metodu nejvhodnější, případně kombinaci několika. Provedený výzkum ukázal, že problémová doména je u karetních her většinou dosti specifická a to znesnadňuje užití univerzálních algoritmů. Zvolený problém je vyřešen formou rule-based umělé inteligence. Podařilo se vytvořit inteligentního hráče pro zástupce z kategorie *imperfect information games*, což je jeden z hlavních přínosů této práce. Ačkoli se dopouští drobných taktických prohřešků, jeho chování většinou blízce připomíná smýšlení středně zkušených hráčů.

Abstract

This thesis focuses on artificial intelligence in card games. The goal is to implement a multiplatform game of this genre in the Unity game engine, to summarize possible approaches that are being used in order to create intelligent agents and furthermore to design and describe the most suitable method or combination of methods for the chosen game. The research that was carried out has shown that the problem domain of card games is rather specific, making it more difficult to use the general-purpose algorithms. The problem given was solved using the rule-based artificial intelligence. The intelligent agent has been implemented for a game of imperfect information, which is considered to be the main contribution of this work to the community. Even though the artificial intelligence player is making minor tactical mistakes, his behavior closely resembles the way of thinking of semi-experienced players.

Klíčová slova

umělá inteligence, Unity, Umělá inteligence v karetních hrách, Multiplatformní vývoj, imperfect information games, Karetní hra Bang!, Nashova rovnováha, teorie her, C#

Keywords

artificial intelligence, Unity, artificial intelligence in card games, multi-platform development, imperfect information games, card game Bang!, Nash equilibrium, game theory, C#

Citace

TREJTNAR, Martin. *Multiplatformní karetní hra s umělou inteligencí*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Matýšek

Multiplatformní karetní hra s umělou inteligencí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Matýška. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Martin Trejtnar
10. května 2021

Poděkování

Tímto bych chtěl poděkovat panu Ing. Michalu Matýškovi za cenné rady, čas věnovaný konzultacím a celkové vedení mé bakalářské práce. Dále bych rád poděkoval Mgr. Vladimíře Černochové za jazykovou korekturu textu práce.

Obsah

1 Úvod **3**

2 Umělá inteligence v karetních hrách **4**
2.1 Dělení umělé inteligence ve srovnání s lidmi podle dosažené inteligence . . . 4
2.2 Perfect information games . . . 4
2.3 Imperfect information games . . . 5
2.4 Algoritmy a přístupy pro tvorbu umělé inteligence . . . 9

3 Herní Engine Unity **15**
3.1 Multiplatformní vývoj . . . 15
3.2 Unity Visual Editor . . . 15
3.3 Životní cyklus skriptu v Unity . . . 16
3.4 Unity Machine Learning Agents . . . 17

4 Karetní hra Bang! **19**
4.1 Role, postavy . . . 19
4.2 Průběh hry . . . 19
4.3 Hrací karty . . . 20

5 Návrh **21**
5.1 Síťový koncept . . . 21
5.2 Uživatelské rozhraní . . . 22
5.3 Inteligentní hráč . . . 25

6 Implementace klientské aplikace **28**
6.1 Finální podoba klientské aplikace . . . 28
6.2 Princip hraní karet . . . 29
6.3 Další prvky uživatelského rozhraní a vlastnosti aplikace . . . 30

7 Implementace serverové aplikace **32**
7.1 Finální podoba serverové aplikace . . . 32
7.2 Komunikace klient-server . . . 34
7.3 Střídání hráčů na tahu . . . 35
7.4 Zpracovávání hraných karet . . . 35
7.5 Výpočet vzdáleností mezi hráči . . . 36
7.6 Další prvky uživatelského rozhraní a vlastnosti aplikace . . . 37

8 Implementace inteligentního hráče **38**
8.1 Heuristika odhadu rolí . . . 38

8.2 Tah inteligentního hráče 40
8.3 Implementace dalších rozhodovacích mechanismů 41

9 Testování **44**

10 Závěr **47**

Literatura **48**

Kapitola 1

Úvod

Umělá inteligence je velmi důležitou součástí nejen videoherního průmyslu. Jedná se o poměrně mladý obor. Myšlenka vytvoření „něčeho“ co dovede napodobit chování a myšlení člověka, což je podstatou umělé inteligence, je však mnohem starší. Dalo by se říci, že se jedná o jednu z věcí, před níž ve třicátých letech minulého století varoval český spisovatel Karel Čapek ve svých antiutopických dílech. Dnes je však běžnou součástí světa okolo nás a důležitým vědním oborem.

V této práci se umělé inteligenci věnuji v kontextu her, konkrétně her žánru karetního. Cílem je naimplementovat karetní hru pro více hráčů s možností hry na různých zařízeních, vytvořit inteligentního protihráče a shrnout techniky a metody, jež se za tímto účelem v daném odvětví užívají. Pro demonstrační účely jsem zvolil oblíbenou karetní hru *Bang!* Jedním z důvodů je i fakt, že neexistuje její oficiální počítačové zpracování.

V kapitole 2 se nalézá klasifikace karetních her a popis vybraných algoritmů, v kapitole 3 je popsán herní engine Unity se zaměřením na multiplatformní vývoj a plugin *Machine Learning Agents*. Kapitola 4 je věnována samotné hře Bang!, její klasifikaci a vysvětlení nejzákladnějších pravidel v kontextu vývoje umělé inteligence. Následuje návrh 5 a implementace 6 hry, kde popořadě rozebírám síťový koncept, uživatelské rozhraní a inteligentního protihráče. Průběh testování inteligentního protihráče je popsán v kapitole 9.

Kapitola 2

Umělá inteligence v karetních hrách

Následující kapitola se zabývá klasifikací umělé inteligence vybraných her na základě dosažené inteligence. Bude objasněn rozdíl mezi *Perfect information game* a *Imperfect information game*, jež je klíčovou charakteristikou pro výběr přístupu tvorby umělé inteligence. Kapitola dále obsahuje stručný historický přehled s významnými milníky nejen v oblasti karetních her. Postupně se však na tento žánr podrobněji zaměří. Stručně budou popsány významné, úspěšně aplikované algoritmy.

2.1 Dělení umělé inteligence ve srovnání s lidmi podle dosažené inteligence

Při tvorbě umělé inteligence je důležité přesně změřit její sílu. V názvu kapitoly zmíněné rozdělení může být definováno následovně:

- **Sub-human** – hraje hůře než lidé.
- **Par-human** – hraje srovnatelně jako lidé.
- **High-human** – hraje lépe než většina reálných hráčů.
- **Super-human** – hraje lépe než všichni lidé.

V následujícím textu jsou rozebírané hry a aplikace umělé inteligence klasifikovány dle tohoto ohodnocení [37].

2.2 Perfect information games

U her *Perfect information games* je v každém tahu možné jasně definovat tah nejlepší, případně několik, nelze-li určit právě jeden takový. Zároveň je v každém momentu dané hry hráčům znám kompletní stav hry i předchozí akce. Nenacházejí se zde žádné skryté informace [30]. Nevyskytuje se v nich ani taková náhoda, že by se například nevědělo, jaká karta ze zamíchaného balíčku bude otočena.

Formálně může být sekvenční hra pro dva hráče definována následovně: Graf hry s *perfect information* je uspořádaná čtveřice $G = \langle L, \iota_I, \Sigma, \Delta \rangle$, kde L je konečná množina stavů, ι_I je stav počáteční, $\iota_I \in L$, Σ je konečná abeceda akcí a $\Delta \subseteq L \times \Sigma \times L$ je množina přechodů

mezi stavy. Požadujeme, aby graf hry G byl totální, tedy pro všechna $\ell \in L$ a $\sigma \in \Sigma$ existuje $\ell' \in L$ takové, že $(\ell, \sigma, \ell') \in \Delta$.

Tahově založená hra na grafu G je hrána dvěma hráči po nekonečně mnoho tahů. První kolo začíná v počátečním stavu ι_I, jež je jedním z vrcholů grafu G. V každém kole, je-li současný stav ℓ, hráč 1 volí tah $\sigma \in \Sigma$, poté hráč 2 volí lokaci ℓ' takovou, že $(\ell, \sigma, \ell') \in \Delta$. Další kolo poté začíná ve stavu ℓ' [23]. Hra končí, dojde-li ke splnění vítězných podmínek.

2.2.1 Šachy

Ačkoliv je tato kapitola zaměřena na hry karetní, jakožto typické zástupce skupiny *Perfect information games*, bude stručně zmíněna i tradiční hra – Šachy. V libovolném momentu hry je její stav jasně definovaný situací na herní desce 8×8. Je znám počet kamenů, jejich druh, pozice, dále informace, který ze dvou hráčů je momentálně na tahu.

Vytvoření šachového programu (počítače), který by byl na úrovni *Super-human*, patří historicky mezi největší výzvy informatiky. Tento cíl byl dosažen 11. května 1997, kdy šachový superpočítač *Deep Blue* firmy IBM porazil v šesti zápasové sérii šachového mistra světa Garryho Kasparova. Jedná se o významný milník i proto, že se tato událost dostala do povědomí široké veřejnosti a zapříčinila významné navýšení investic do vývoje umělé inteligence.

Deep Blue byl založen na základním schématu z roku 1949 složeného ze tří komponent amerického matematika Clauda Shannona. **Generátor tahů**, který umožňoval prohledávání v čase vpřed, **ohodnocovací funkce**, jež počítala ohodnocení budoucích stavů (herních konfigurací) a **řízení prohledávání**, které umožňovalo pohyb v čase zpět a ukládání ohodnocení potenciálně budoucích stavů do stavu současného. Toto schéma bylo u Deep Blue doplněno o **zásobník chytrých tahů** za účelem detekce opakování herních situací. V generátoru tahů bylo použito *Alfa-beta ořezávání* [33] a algoritmus *Quiescence search* (capture search) [13], jehož principem bylo i zvážení možnosti přijetí situace takové, jaká je, namísto sebrání soupeřova kamene, vyskytla-li se možnost. V případě sebrání kamene algoritmus pokračuje z pohledu druhého hráče a zvažuje opět stejné možnosti. *Alfa-beta ořezávání* spoléhá na to, že není nutné prohledávat všechny možné oponentovi protitahy na tah náš, o kterém se jednou prokáže, že je nevýhodný. Další strategií je *odložené vyhodnocování*, díky němuž se systém vyhne dopočítávání *ohodnocovací funkce* v situacích, kdy by některý z hráčů obětoval důležitý kámen bez pádného důvodu. Deep Blue používal *odložené vyhodnocování*, avšak jen do momentu, kdy nastala neobvyklá konfigurace [27].

2.3 Imperfect information games

Hry s *Imperfect information* nejsou plně pozorovatelné. Hráč ve svém tahu nemá k dispozici veškeré informace, je informován pouze částečně. Jinými slovy, hráči mají soukromé, skryté informace. To je typické zejména pro karetní hry, kde se často neví, jaké karty byly soupeřům rozdány. Proto algoritmy používané ve hrách informovaných nejsou schopné vyřešit *Imperfect information games* [25].

Formálně může být hra s *imperfect information* definována následovně:
$G = (N, \Theta, S, P, u)$, kde N je množina hráčů. Θ_i je konečná množina možných typů hráče i. S_i je množina možných strategií hráče i. $p_i \in P$ je pravděpodobnost (joint probability distribution) rozdělení typů hráče i. Pro konečný typový prostor předpokládáme, že $p(\theta_i) > 0$, pro všechny $\theta_i \in \Theta_i$. Symbol u_i reprezentuje užitkovou funkci (funkci odměny) $S \times \Theta \to \mathbb{R}$, která přiřazuje hodnotu z $\mathbb{R}$ pro $i \in N$ [29, 31].

2.3.1 Poker

Poker je jedním z nejtypičtější zástupců karetních *Imperfect information games*. Po dlouhá desetiletí byl považován za jednu z největších výzev v oboru umělé inteligence. Je také často užíván jako učebnicový příklad při objasňování základních konceptů teorie her. Hra má mnoho různých variant, z nichž však tou nejslavnější je varianta *No-limit Texas Hold'em* [17, 18].

No-limit Texas Hold'em je obyčejně hrán v počtu dvou, až devíti hráčů, což je nejběžnější varianta. Hra začíná rozdáním dvou karet všem hráčům. Ty jsou soukromou informací každého z nich. Jeden z hráčů je v dané hře v roli *big blind* (velká povinná sázka na slepo), po jeho levici sedí hráč v roli *small blind* (obvykle polovina hodnoty big blind). Tyto role se po každé hře posouvají po směru hodinových ručiček (před rozdáním dvojic karet). Následuje první kolo sázek. Při variantě *No-limit Texas Hold'em* není pravidly omezena maximální výše sázky. Pokračuje se spálením vrchní karty balíčku a otočením vrchních tří karet (veřejná informace). Následuje další kolo sázek. Poté je opět spálena vrchní karta balíčku a otočena čtvrtá karta. Poslední kolo sázek probíhá po otočení páté karty. Sázku je možné dorovnat či navýšit, třetí možností je složení karet. Hru vyhrává hráč, který z pěti veřejných a svých dvou soukromých karet sestaví nejlepší pětikaretní kombinaci. Hraje se s balíčkem padesáti dvou karet francouzského typu [20].

V roce 2017 byl představen algoritmus *DeepStack*, jehož autory jsou čeští a kanadští vědci. Program porážel profesionální hráče v pokerové variantě *Heads-up no-limit Texas Hold'em*, jež je hrána ve dvou hráčích. Výkonost umělé inteligence v Pokeru se měří v jednotkách mbb/g, kde 1 mbb vyjadřuje tisícinu povinné sázky big blind za hru. Za výraznou úspěšnost profesionálové považují hranici 50 mbb/g. *DeepStack* dle autorů po odehrání více než 44 tisíc her dosáhl výkonosti 492 mbb/g [35].

V témže roce byl vědci z Carnegie Mellon University v Pensylvánii [17] představen další program – Libratus – taktéž určen pro *Heads-up no-limit Texas Hold'em*. Libratus porazil tým čtyř hráčů z nejlepší světové desítky ve dvacetidenním turnaji, a to s 99.98% statistickou úspěšností a výkonností 147 mbb/g. Porazil též každého z hráčů individuálně. Algoritmus používal dva druhy abstrakce za účelem snížení výpočetní náročnosti. Prvním druhem je abstrakce akcí, kterých v libovolném momentu hry bývá až 20 tisíc. Ty jsou sdružovány a mapovány pouze na několik málo typových. Podstatnou částí těchto akcí je výše sázky. K tomuto dochází v každé jednotlivé hře, tudíž program nesází stále tytéž obnosy. Druhým druhem je abstrakce karet. Spočívá v sdružování podobných karetních kombinací na ruce. K druhé jmenované abstrakci nedochází v prvních dvou kolech sázení. Ve třetím kole sázek je takřka 55 milionů možných karetních kombinací zredukováno na 2,5 milionu. Libratus je založen na algoritmu *Monte Carlo Counterfactual Regret Minimization*, který je popsán v podkapitole 2.4.4. Systém, který porazil výše zmíněný tým čtyř profesionálů, používal sto CPU (centrálních procesorových jednotek) [17, 18].

Zásadním průlomem bylo představení programu *Pluribus*. Tato umělá inteligence je určena pro *No-limit Texas Hold' em*. Na rozdíl od svých předchůdců však pro variantu hranou šesti hráči – nejhranější forma pokeru. Jádrem algoritmu je předběžný program, jemuž se říká blueprint. Ten vzniká na počátku výpočtem, kdy program hraje proti starším kopiím sebe sama bez člověkem předpřipravených dat (strategií). Na počátku jsou jeho akce řízeny náhodou. Blueprint během těchto her postupně vylepšuje strategii na základě zjišťování, které jím provedené akce a s jakým rozdělením pravděpodobnosti nad nimi vedou k porážení jeho předchozích strategických verzí. Poté během samotné hry se soupeřem je blueprint vylepšován při hledání lepších strategií pro situace, v nichž se ocitá. I Pluribus

pracuje s abstrakcí akcí a abstrakcí karet. Druhou jmenovanou však používá pouze při procházení a uvažování budoucích kol sázení. Nikdy neabstrahuje v aktuálním kole sázek během hry se samotným soupeřem, jelikož by docházelo k zanedbání (podcenění) situace, což by se odchylovalo od hry na úrovní *super-human* 2.1.

Pluribus byl testován hrou s elitními hráči, z nichž každý vyhrál profesionální hrou ve své kariéře minimálně milion dolarů. Hrály se dva systémy. První byl formát 5H + 1P (pět hráčů a Pluribus), druhý 5P + 1H (jeden hráč a pětkrát Pluribus). Experimentu prvního formátu se účastnilo celkem osm hráčů a bylo v něm odehráno deset tisíc pokerových rukou ve dvanácti dnech. Každý hrací den bylo vylosováno 5 hráčů a účastníkům nebylo řečeno, kdo jsou jejich oponenti. Všichni účastníci byli finančně odměněni v závislosti na tom, jak dobře hráli, aby bylo zaručeno, že hrají nejlépe, jak dovedou. Pluribus vyhrál s průměrnou výkonností 48 mbb/g. Hry druhého formátu se zúčastnili dva z osmi hráčů a oba v něm odehráli pět tisíc pokerových rukou. Lidští soupeři byli poraženi průměrně o 32 mbb/g. Instance programu Pluribus nemohly spolupracovat za účelem poražení lidského oponenta, jelikož byla v průběhu celého testování používána falešná jména. Pluribus byl provedením tohoto experimentu ohodnocen jako umělá inteligence *super-human* [18].

2.3.2 Magic: The Gathering

Magic: The Gathering (dále *M:TG*) je karetní hra, která byla v roce 1994 oceněna Americkou Mensou v rámci každoročně pořádané události *Mind Games Competition*. M:TG je nejběžněji hráno ve dvou hráčích. Hra se od ostatních tradičních karetních her liší zejména v tom, že je hrána se speciálními kartami, kterých existují tisíce a nové bývají vydávány každým rokem. Při hře ve dvou hráčích hrají oba nejběžněji s balíčkem šedesáti karet. Ten si ze sady existujících vytvoří.

Cowling a Ward, ve své práci zabývající se touto karetní hrou, vytvořili řadu inteligentních hráčů, jejichž chování bylo řízeno třemi přístupy – náhodné chování, chování založené na pravidlech 2.4.1 a Monte Carlo prohledávání 2.4.4. Použitý na pravidlech založený přístup byl navržen herním specialistou. Všichni inteligentní hráči během provedeného testování hráli se stejným balíčkem karet. Tento balíček byl složen pouze z karet typu *creature* (stvoření). Autoři tuto zjednodušenou podobu hry rozdělili na tři podproblémy. Prvním byla volba karet *creature* k zahrání z herní ruky. Druhým volba, které z množiny vyložených karet *creature* zvolit k zaútočení. Posledním byla fáze bránění. Zkoušeli zvolit různou kombinaci přístupů pro tyto rozhodovací fáze. Autoři tento přístup k prvnímu podproblému (volba karet k zahrání) vylepšili použitím Monte Carlo prohledávacího algoritmu. Výsledky experimentu ukázaly, že takto upravený inteligentní hráč vyhrál o 4,7 až 6,7% (v závislosti na volbě přístupu pro podproblémy útočení a bránění) více her. Dále dokázali, že i malé množství Monte Carlo simulací zde postačí k významnému zvýšení síly inteligentního hráče. Celkově autoři Monte Carlo prohledávací algoritmus považují za slibný přístup k tvorbě silných inteligentních hráčů pro hru *M:TG* [49].

2.3.3 Další hry

Počítačový program *AlphaGo* vytvořený vývojáři z *Google DeepMind* dokázal v roce 2016 porazit úřadujícího osmnáctinásobného držitele titulu mistra světa v tradiční čínské hře *Go*. Velikost herní desky *Go* je 19 × 19 políček. Je tedy více než pětkrát větší oproti šachové hrací desce. Jedná se o hru s *perfect information*, která má 10^{170} možných konfigurací. Tyto hry mohou být vyřešeny rekurzivním dopočítáním optimální hodnotící funkce stavů ve vyhledávacím stromu, který obsahuje přibližně b^d možných posloupností tahů. Hodnota

b reprezentuje průměrný počet možných tahů (faktor větvení) v dané herní konfiguraci a hodnota d je průměrná délka hry (počet tahů). V případě *Šachu* je počet možných posloupností tahů roven přibližně 35^{80}, v případě *Go* 250^{150}. Řešit hry s takto rozsáhlým stavovým prostorem hrubou silou je neproveditelné [44]. Program *AlphaGo* je založen na hlubokých neuronových sítích [47] a algoritmu Monte Carlo tree search (viz. sekce 2.4.4).

Za zmínku stojí dále například ve Švýcarsku oblíbená hra *Jass*. Je hrána ve dvou týmech o dvou hráčích. Patří mezi hry s *imperfect information*, jelikož hráči mají v ruce karty, které vidí jen oni sami. *Jass* je koordinační hrou, u níž je zakázáno komunikovat. Hráči si informace sdělují prostřednictvím hraní a odhazování konkrétních karet v konkrétních situacích [37].

Úspěchů je dosahováno i na poli počítačových her. Mezi významné milníky poslední doby patří též například vytvoření počítačového programu *AlphaStar*, který byl vytrénován pomocí *Posilovaného učení* (viz. sekce 2.4.2) ve hře *StarCraft 2*. AlphaStar dosáhl v roce 2019 v rámci žebříčku hry ohodnocení lepšího než 99,8% procent reálných, aktivních hráčů [48]. Ve stejném roce počítačový program *OpenAI Five* porazil světové šampiony ve hře *Dota 2* [14]. Zmíněné události jsou přehledně vypsány v tabulce 2.1.

Název hry	**Název projektu**	**Autor**	**Dosažený milník**	**Datum události**	**Klasifikace hry**
Šach [27]	DeepBlue	IBM	super-human	11.5.1997	PIG
Dáma [42]	Chinook	J. Schaeffer a další	weakly solved	29.4.2007	PIG
Jeopardy! [4]	Watson	IBM	super-human	14.1.2011	-
Go [45]	AlphaGo	Google DeepMind	super-human	15.3.2016	PIG
Head-up no limit Texas Hold'em [17]	Libratus	Carnegie Mellon University (CMU)	super-human	leden 2017	IIG
No-limit Texas Hold'em [18]	Pluribus	CMU a Facebook AI Lab	super-human	srpen 2019	IIG
Dota 2 [6, 5]	OpenAI Five	OpenAI	super-human	duben 2019	IIG
StarCraft 2 [48, 1]	AlphaStar	Google DeepMind	high-human	srpen 2019	IIG
Bridge [26]			high-human		IIG
Jass [37]			par-human		IIG

Tabulka 2.1: Tabulka dosažení vybraných milníků, seřazených dle data události.

V případě hry *Dáma* se jednalo o slabé vyřešení (weakly-solved) – začíná-li hráč z výchozí, standardní konfigurace, při bezchybné hře je zaručeno, že neprohraje. Pokud hrají bezchybně oba hráči, je dokázáno, že hra skončí remízou nehledě na to, jaký úvodní tah začínající hráč zvolí. O plně vyřešenou hru by se jednalo, pokud by byla nalezena strategie pro každou legální herní situaci, nikoliv pouze pro počáteční [10]. Při hře proti chybujícímu hráči musí být každá chyba využita. Když byla hra *Dáma* vyřešena, jednalo se o nejrozsáhlejší vyřešenou hru [2]. V případě her *Bridge* a *Jass* je uvedeno jaké síly dosahují nejlepší počítačové programy. Ostatní uvedená data označují události, kdy došlo k významné výhře, jež znamenala dosažení daného milníku. Zkratka *PIG* v posledním sloupci tabulky znamená *Perfect information game*, viz. sekce 2.2. V případě zkratky *IIG* se jedná o *Imperfect information game*, viz. sekce 2.3.

2.4 Algoritmy a přístupy pro tvorbu umělé inteligence

Kapitola se věnuje popisu algoritmů a metod, jež jsou používány při tvorbě inteligentních hráčů se zaměřením na hry karetní. Pro správné pochopení jejího obsahu je nejprve potřeba objasnit význam několika pojmů. Většina systémů umělé inteligence, spadajících do kategorie *super-human* 2.1, byla založena na přístupu, jež spočíval ve snaze přiblížit se Nashově rovnováze.

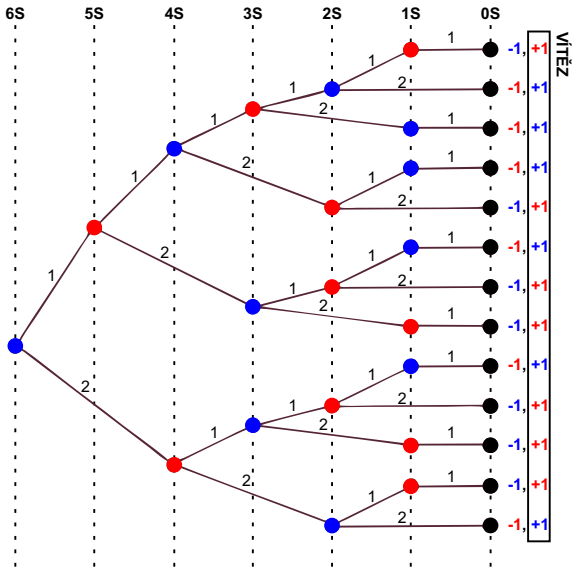
Nashova rovnováha

Nashova rovnováha je řešení hry dvou a více hráčů, kde každý zná Nashovu strategii rovnováhy všech ostatních hráčů a nikdo nemůže nic získat, změní-li pouze svoji strategii [24]. Jinými slovy — Nashova rovnováha je seznam strategií (každý hráč má jednu), kde žádný hráč, odchýlením se od té své, nemůže ve hře získat žádnou výhodu. Každý hráč, který se řídí Nashovou strategií rovnováhy, má zaručeno, že neprohraje (nejhůře remizuje) bez ohledu na to, jak bude hrát jeho soupeř [18]. Hráč hrající Nashovu strategii rovnováhy porazí každého hráče, který nepoužívá Nashovu strategii rovnováhy. Ve hrách, kde se vyskytuje náhoda, však hráč nemusí vyhrát úplně každou hru. Příkladem může být poker. Nashova strategie rovnováhy funguje dobře proti silným soupeřům, jelikož nechybuje a neposkytuje tak soupeřovi příležitosti k využití zaváhání. Jako slabina může být chápán fakt, že hráč řídící se Nashovou strategií rovnováhy nemusí nutně slabého, chybujícího hráče porazit výrazným rozdílem. Důvodem je, že se aktivně nesnaží využít soupeřových zaváhání, nýbrž se plně soustředí na to, aby sám neudělal sebemenší chybu. Při implementaci silného inteligentního hráče je cílem co nejvíce se přiblížit Nashově rovnováze [37]. Je dokázáno, že Nashova strategie rovnováhy existuje v každé konečné hře, ale může být složité ji nalézt. Speciální skupinou her, kde je možné ji nalézt pomocí efektivních, účinných algoritmů jsou takzvané *hry s nulovým součtem* [18]. Konečná hra je hra pro dva hráče, která končí po provedení konečného počtu tahů (Šach, Poker). Hra s nulovým součtem je hra, kde se součet odměn (zahrnuje kladné i negativní) rovná nule. Hra pro dva hráče má nulový součet jen a pouze pokud je čistě kompetitivní. Jednoduchým příkladem může být hra Kámen-nůžky-papír [15].

Rozšířená forma hry

Dalším důležitým termínem, kterému je třeba porozumět, je tzv. *rozšířená forma* hry. Jedná se o orientovaný graf znázorňující popis *sekvenční hry*. V sekvenční hře se (na rozdíl od her současných) na tahu jednotliví hráči střídají jeden po druhém. Příkladem sekvenční hry může být například Šach. Hra Kámen-nůžky-papír by byla příkladem hry současné. Každý nekoncový vrchol grafu rozšířené formy hry vyjadřuje, který hráč je v daný moment na tahu. U sekvenčních her se vždy jedná o právě jednoho hráče. Ten má na výběr několik tahů reprezentovaných hranami, z nichž jeden vybírá. Takovému vrcholu se říká *rozhodovací uzel*. Druhým typem vrcholů jsou *uzly náhody*. Jejich účelem je přiřadit výsledky náhodné události. Výsledkem každé hrany vedoucí z uzlu náhody je tedy případný následující stav společně s pravděpodobností, že k tomuto přechodu mezi nimi dojde. Vrcholy zároveň vyjadřují stav hry. Hra začíná v *počátečním vrcholu*. *Koncové vrcholy* jsou stavy, kde naopak končí. V koncových vrcholem je n-tice užitků, kde n je počet hráčů. Každému z hráčů je přiřazena jedna hodnota vyjadřující, jak vysoký užitek by v případě dosažení tohoto koncového stavu získal. Hra je tedy *hrou rozšířené formy*, je-li možné ji touto formou zobrazit. Cesta

z počátečního stavu do libovolného jiného konkrétního stavu v takovém grafu vyjadřuje průběh hry [38, 36].



Obrázek 2.1: Rozšířená forma hry NIM [38].

Jednoduchá hra NIM má mnoho variant. Na obrázku 2.1 je zobrazena rozšířená forma varianty, kde se hraje se šesti sirkami. Začíná modrý hráč a odebírá jednu nebo dvě sirky. Takto se oba hráči střídají a vítězem se stává ten z hráčů, který seberu poslední sirku. Barva vrcholu vyjadřuje, který z hráčů je na tahu. Osa, na níž vrchol leží, vyjadřuje, kolik ve hře zbývá sirek. Z pohledu na tuto rozšířenou formu je patrné, že začínající hráč nemůže vyhrát, hraje-li jeho soupeř racionálně. Na pravém kraji obrázku jsou hodnoty užitku pro každý koncový vrchol a barvou jsou přidruženy oběma hráčům [38].

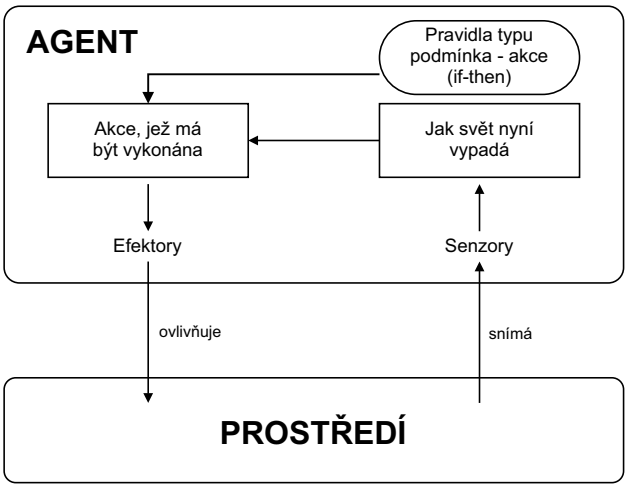
2.4.1 Metody založené na pravidlech (Rule-based systems)

Metody založené na pravidlech jsou nejstarším a nejběžnějším přístupem k tvorbě umělé inteligence ve hrách. Problém je řešen programátorem nadefinovanými pravidly. Jedná se tedy o soubor if-then-else podmínek, které napodobují uvažování a rozhodování reálného hráče. Základními komponenty takového systému jsou *množina pravidel* (báze znalostí) a *množina akcí*. Systém se chová deterministicky. To znamená, že při stejných podmínkách na vstupu vždy vykazuje stejné výstupní chování. Poskytuje spolehlivé řešení, které je předvídatelné. Tento přístup je poměrně častý, zejména u karetních her. Příkladem může být jeho aplikace pro hru Magic: The Gathering [49] nebo deskovou hru 7 Wonders [40]. Implementace je možná mimo jiné i použitím *konečných automatů* [34], či *fuzzy konečných automatů* [39, 11, 37].

2.4.2 Posilované učení

Učit se interagováním s okolím je člověku zcela přirozené. Při řešení nějakého problému se snaží pochopit jakým způsobem nejsnáze dosáhne svého cíle. Snaží se pochopit spojitost mezi důvody a důsledky. Tato podkapitola je věnována *posilovanému učení*, paradigmatu *strojového učení*, jež tento přístup aplikuje na stroje a systémy.

Posilované učení je mapování akcí na situace za účelem co nejvíce zvětšit pozitivní odměnu. Toto je v modelu posilovaného učení úkolem *agenta*, jehož lze definovat následovně. *Agent* je nezávislá entita, která pomocí senzorů vnímá své prostředí a na základě toho jej ovlivňuje prostřednictvím svých efektorů [41].



Obrázek 2.2: Agent a prostředí [41].

Agentu není řečeno jaké akce má volit, zkouší je náhodně. Odborná literatura běžně požívá pojem – metoda pokus-omyl. Informace získává zpětnou vazbou v podobě odměn znamenajících míru správnosti volby. Jakmile agent zvolí akci, okamžitě obdrží patřičnou odměnu a následující stav. Není mu řečeno, kterou akci měl zvolit, aby získal odměnu nejvyšší možnou. V komplexnějších případech užití se správnost volby může projevit až v pozdějších fázích.

Dalším paradigmatem *strojového učení* je *učení s učitelem*. Zde se používá trénovací množina příkladů. Ta se skládá z dvojic typicky vstupních vektorů příznaků a odpovídajících, očekávaných výstupních vektorů dodaných externím *učitelem*. Každá dvojice reprezentuje popis situace a zároveň štítek správné reakce, kterou by měl systém pro danou situaci zvolit. Touto reakcí je většinou kategorizace vstupu. Cílem je, aby se systém naučil vyhodnocovat a zobecňovat správný výstup i pro situace, jež se nenachází v trénovací množině vstupů. *Posilované učení* tyto výstupní štítky nepoužívá, což je hlavní rozdíl mezi těmito metodami.

Třetím paradigmatem je *učení bez učitele*, které se snaží nalézt skrytou strukturu v množině vstupních dat, které ony štítky nemají. Tomuto procesu se říká *klasterizace*. Zde se však nepoužívá systém udělování odměn, což tyto metody odlišuje. Další odlišností je, že při *posilovaném učení* probíhá souběžně učení a vyhodnocování [28, 46].

2.4.3 Counterfactual Regret Minimization (CFR)

Counterfactual Regret Minimization je algoritmus, jež se v nedávné době osvědčil při vývoji umělé inteligence v Pokeru. *Regret* (míra lítosti) je jeho základním kamenem. Regret je neformálně možné definovat jako rozdíl mezi užitkem nabytým (může být i negativní hodnotou) a největším užitkem, který bylo možné získat při onom rozhodování s ohledem na následné konání soupeřů. Jinými slovy – hlavní myšlenkou je uvažování nad uplynulými rozhodnutími ve smyslu „Kdybych byl býval věděl, rozhodl bych se ...“ a regret vyjadřuje, jak

moc hráč daného rozhodnutí lituje [36]. K popisu algoritmu je zapotřebí nejprve definovat několik pojmů. *Konečná hra rozšířené formy s imperfect information* může být definována takto:

- **N** je konečná množina hráčů.
- **H** je konečná množina historií, tedy posloupností akcí, jež byly provedeny. Prázdná posloupnost, stejně jako každý prefix historie $h \in H$, taktéž náleží H.
- Množina **Z** je podmnožinou H a jedná se o koncové (terminální) historie. Žádné $z \in Z$ není prefixem.
- $\mathbf{A(h)}$ je množina akcí, které jsou možné v neterminální historii $h \in H$.
- Hráčská funkce $\mathbf{P(h)}$ každé neterminální historii určuje, který z hráčů $p \in N$ je po provedení historie h na tahu.
- Každá neterminální historie $h \in H$, kde $P(h) = c$ (hráč c je na tahu) je spojena s funkcí $\mathbf{f_c}$, která definuje pravděpodobnost $f_c(a|h)$ pro každé $a \in A(h)$. Jedná se tedy o pravděpodobnostní rozdělení diskrétní náhodné veličiny. Toto rozdělení se u jednotlivých neterminálních historií hráče $c \in N$ liší.
- *Informační oddíl* $\mathcal{I}_\mathbf{i}$, každého hráče i, je množina všech historií $h \in H$, kde $P(h) = i$ (hráč i je na tahu). Pro každý informační soubor I_i hráče i platí, že $I_i \in \mathcal{I}_i$.
- Prvky množiny akcí $A(h) = A(h')$, kdykoliv historie h a h' spadají do stejného *informačního souboru*. Pro každé $\mathbf{I_i} \in \mathcal{I}_i$ platí, že $A(I_i)$ je množina všech akcí $A(h)$ a $P(I_i)$ vyjadřuje totéž co $P(h)$, pro kteroukoliv historii $h \in I_i$. Informační soubor I_i je jinými slovy stav hry bez ohledu na to, jakou historií k němu došlo.
- Pro každý terminální stav $z \in Z$ existuje užitková funkce $\mathbf{u_i}$ (jedna za každého hráče $i \in N$), která hráči i přiřadí hodnotu z $\Re$. Jestliže $N = \{1, 2\}$ (dva hráči) a $u_1 = -u_2$, pak se jedná o *hru rozšířené formy s nulových součtem.* [50]

Symbol	Název	Definice
σ_i	Strategie hráče i	σ_i je strategie hráče $i \in N$. Strategie v každém informačním souboru $I_i \in \mathcal{I}_i$ hráče i, definuje rozdělení pravděpodobnosti mezi jednotlivé akce z množiny $A(I_i)$.
Σ_i	Množina všech strategií hráče i	Σ_i je množina strategií hráče $i \in N$.
σ	Strategický profil	σ je strategický profil. Obsahuje právě jednu strategii pro každé $i \in N$, tedy $\sigma = \{\sigma_1,\sigma_2,\ldots,\sigma_m\}$, kde m je celkový počet hráčů.
σ_{-i}	Množina všech strategií, bez σ_i	σ_{-i} vyjadřuje množinu strategií všech hráčů, až na tu hráče i.
$\pi^\sigma(h)$	Pravděpodobnost historie h	$\pi^\sigma(h)$ je pravděpodobnost, že nastane historie h, řídi-li se hráči strategickým profilem σ.
$\pi_i^\sigma(h)$		$\pi_i^\sigma(h)$ vyjadřuje pravděpodobnost nastání historie h, řídí-li se hráč i strategií $\sigma_i \in \sigma$. Hráč i v každém prefixu h' historie h, kde $P(h') = i$ (kde je na tahu) následuje posloupnost akcí h.
$\pi_{-i}^\sigma(h)$		Produkt strategií všech hráčů (včetně náhody), až na hráče i.
$\pi^\sigma(I)$	Pravděpodobnost dosažení I	$\pi^\sigma(I)$ je pravděpodobnost dosažení informačního setu I, je-li dán strategický profil σ.

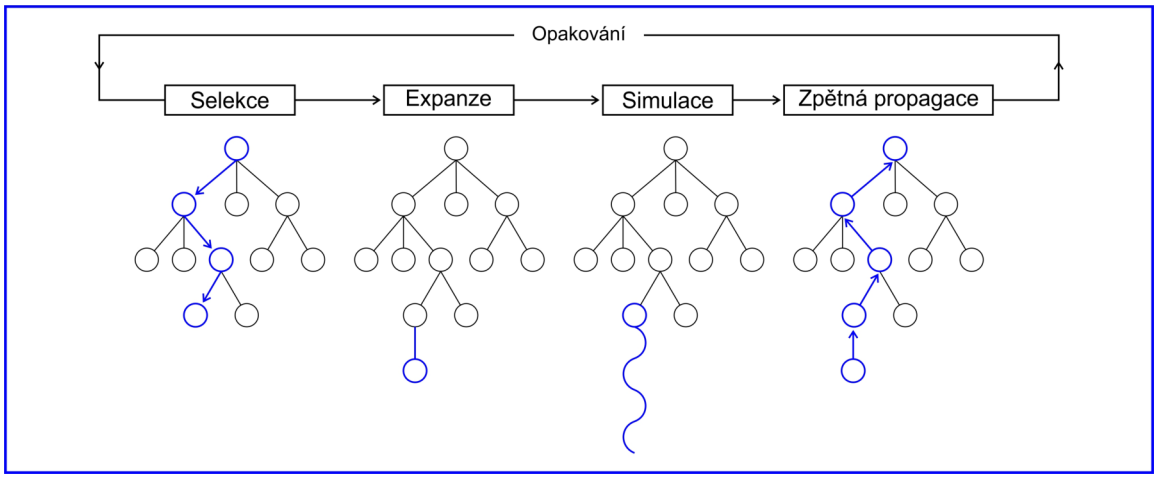
Symbol $u_i(\sigma)$ vyjadřuje celkově očekávanou výši užitku hráče i, hraje-li se podle strategického profilu σ, tedy $u_i(\sigma) = \Sigma_{h \in Z} u_i(h)\pi^\sigma(h)$. Jinými slovy se jedná o součet všech užitků terminálních historií pronásobených pravděpodobností, že k nim dojde.

CFR se učí hrát danou hru opakovaným hraním proti svým předchozím verzím. Prvotní strategií je strategie náhodná (každá akce náležící $A(h)$ má stejnou pravděpodobnost). CFR minimalizuje hodnotu *counterfactual regret* v každém informačním souboru I_i. Tato hodnota může být kladná, nulová, či záporná. Je dokázáno, že hodnota *overall regret* je menší nebo rovna sumě jednotlivých hodnot *counterfactual regret* v těchto rozhodovacích uzlech [50]. Tento proce probíhá (miliardy her), dokud nedojde k nalezení optimální strategie. Optimální strategie je aproximace Nashovy rovnováhy 2.4 – strategie, s níž je možné nejhůře remizovat. Algoritmus se tedy nesnažní využívat soupeřových chyb, ale drží se strategie hrát perfektní obranu [50, 16].

2.4.4 Monte Carlo tree search (MCTS)

Monte Carlo Tree Search (dále MCTS) je prohledávací algoritmus, který je založen na dvou základních konceptech. Prvním z nich je přesvědčení, že je možné náhodnými simulacemi (stochasticky) aproximovat hodnotu herního tahu. Druhým je předpoklad, že tyto hodnoty mohou být účinně použity k vylepšení strategie směrem k *best-first* (expanze nejslibnějších uzlů). MCTS může být aplikováno na každou hru s konečnou délkou. Základem jsou simulace hry, v níž inteligentní hráč i jeho soupeř hrají náhodné, či pseudonáhodné tahy. Odsimulování mnoha takových náhodných her totiž může vést k vzniku dobré strategie. Algoritmus postupně vytváří částečný rozhodovací strom hry, který se používá k odhadnutí hodnoty tahů. Odhad je zpřesňován v průběhu simulací. Každá iterace algoritmu se skládá ze čtyř fází (viz. obrázek 2.3).

- **Selekce** – Sestupuje se od kořenového uzlu částí stromu, která již je přítomna v paměti. V každém uzlu dochází k aplikování *selektivní taktiky* (selection policy), která vybere uzel následující.
- **Expanze** – Sestup končí, jakmile je dosaženo uzlu, který není součástí prozkoumané části stromu (neexpandovaný, neterminální uzel). Tento uzel se přidá do paměti. Odsimulováním jedné hry je tedy strom zvětšen o jeden uzel.
- **Simulace** – Simulace se spouští z nově přidaného uzlu a řídí se *předdefinovanou taktikou* (default policy). Nejčastěji náhodné vybírání uzlů.
- **Zpětná propagace** – Jakmile je dosaženo konce simulace, dochází k aktualizaci statistik všech uzlů od koncového, kde simulace skončila po počáteční kořenový.



Obrázek 2.3: Grafické znázornění jednotlivých fází algoritmu Monte Carlo tree search [22].

Selektivní taktika je algoritmus, který zachovává rovnováhu mezi dvěma protichůdnými přístupy při volbě uzlů. Prvním z nich je volit ty, které se osvědčily. Druhým je jíti cestou prozkoumávání málo prozkoumaných částí stromu. K tomu se nejčastěji používá algoritmus *UCT* (Upper Confidence Bounds applied for Trees) [51, 19, 22].

Kapitola 3

Herní Engine Unity

Herní engine je aplikační rámec, jehož účelem je urychlit a usnadnit proces vývoje her. Principem je myšlenka znovupožitelnosti kódu. Funkcemi takového softwaru běžně bývají vykreslovací engine (jádro) 2D a 3D grafiky, jednotka zpracovávající signály vstupních hardwarových zařízení, fyzikální engine, audio, 2D a 3D animace, podpora hraní po síti, skriptování, prostředky pro vývoj umělé inteligence a mnoho dalších. Herní enginy většinou nabízí i možnost výsledný program exportovat pro běh na různých operačních systémech a platformách. K tomu obyčejně nebývá potřeba zdrojový kód modifikovat. Herní enginy běžně používají vysokoúrovňové jazyky, zejména *Java*, *C#*, či *Python* [12]. Tato kapitola se zaměřuje konkrétně na herní engine *Unity*.

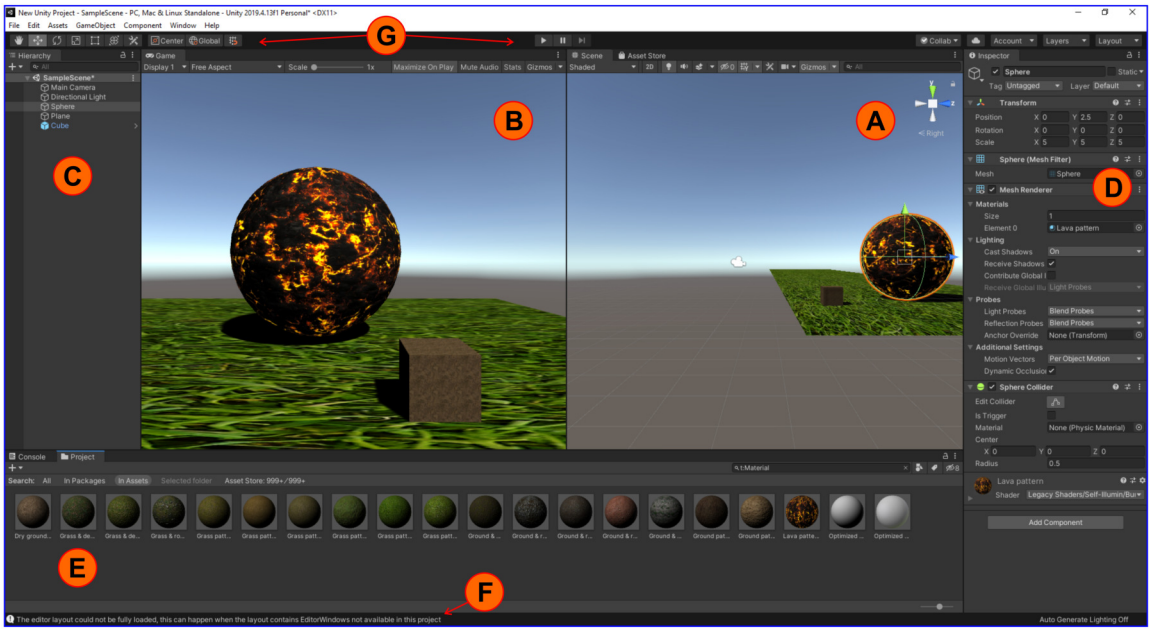
3.1 Multiplatformní vývoj

Unity podporuje vývoj aplikací pro více než dvacet pět různých platforem a technologií, od těch nejznámějších jako jsou Windows, macOS, Linux přes tradiční telefonní operační systémy Android a iOS, televizní operační systémy, herní konzole, WebGL či virtuální realitu. Jednou z nejnovějších je podpora vývoje pro konzoli PlayStation 5 dostupná ve verzi *Unity 2020 LTS*.

3.2 Unity Visual Editor

V *Unity* je možné psát skripty v jazyce *UnityScript*, což je verze *JavaScriptu*, avšak nejběžnější, doporučovanou možností je jazyk *C#*. V této kapitole se podíváme na uživatelské rozhraní editoru Unity ve výchozím rozložení 3.1. Uprostřed okna aplikace se nachází editor scény (A), kde je možné vytvářet a upravovat 2D, či 3D scénu, dle typu projektu. Pracuje se zde se základními stavebními prvky, jimž se říká *GameObject*. Tyto objekty je možné pozicovat, rotovat, měnit jejich velikost. Dále upravovat osvětlení scény a pozicovat kamery. Vedle nalevo je zobrazení, jak momentálně vypadá finální hra z pohledu kamery scény (B). Dalším klíčovým oknem je hierarchie objektů (C). Jedná se o textovou reprezentaci herních objektů *GameObject* zahrnující zmíněné kamery, dále modely a *prefabs* uspořádané do stromové struktury, v níž jsou vidět jejich vzájemné závislosti. *Prefabs* jsou konkrétní konfigurace herních objektů s nastavenými parametry, určené pro opakované používání. Oknu v obrázku označeném písmenem D se říká *Inspektor*. Při zvolení objektu skrze hierarchii či okno scény se v inspektoru zobrazí jeho konfigurace. Zde se nastavují jeho vlastnosti, které je mimo jiné možné měnit i skrze skripty. Okno projektu (E) vizualizuje do projektu naim-

portované assety, což jsou obrázky, 3D modely, materiály, audio nahrávky a další soubory podporované herním enginem. Status lišta (F) zobrazuje zprávy z konzole, stav a postup asynchronně probíhajících úloh jako je například překlad a další nastavení projektu. Nástrojová lišta (G) zahrnuje základní nástroje pro manipulaci s objekty ve scéně. Stiskem tlačítka *Spustit* uprostřed se zahajuje simulace, tedy spouští aplikace tak, jak ji vidí samotný uživatel při jejím užívání. Dále je zde možné aplikaci pozastavit či krokovat. Všechna tato okna, mimo status lištu, je možné skrýt či změnit jejich pozicování. [8]



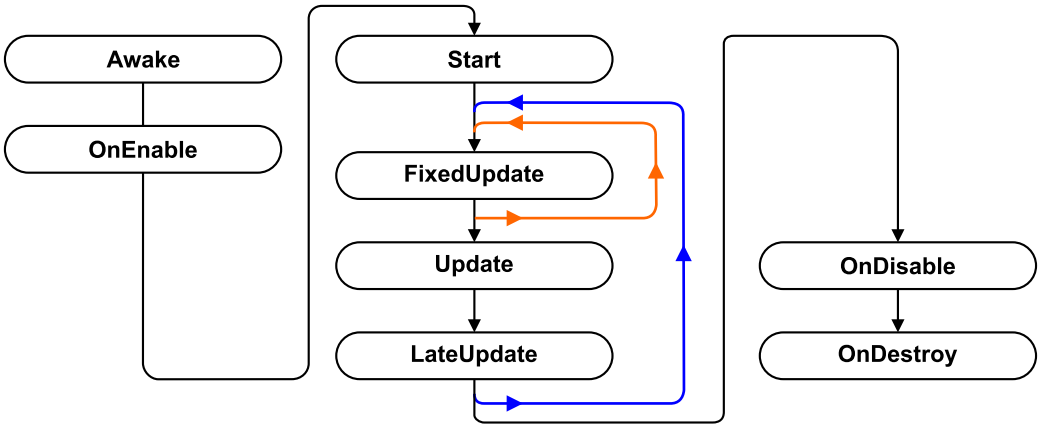
Obrázek 3.1: Uživatelské rozhraní editoru ve verzi *Unity 2019.4.13f1 LTS*.

3.3 Životní cyklus skriptu v Unity

V rámci životního cyklu skriptu Unity vykonává a opakuje v definovaném pořadí řadu událostních funkcí. [7]. Zde je vyjmenováno několik nejzákladnějších, s nimiž programátor při práci s herním enginem, v rámci jednotlivých skriptů, může pracovat (viz. obrázek 3.2).

- **Awake()** – Funkce je volána jako první, před Start funkcemi, nehledě na to, zda je skript aktivní, či nikoliv (aktivní však musí prvně být GameObject, k němuž je skript připojen).
- **OnEnable()** – Funkce je volána, jakmile je zaktivována komponenta GameObject, společně s připojeným skriptem.
- **Start()** – Funkce je volána před snímkem funkce Update(), a to právě jednou (skript musí být aktivovaný, stejně tak pro následující funkce update).
- **Update()** – Funkce je volána jednou v každém snímku.
- **LateUpdate()** – Funkce je volána jednou v každém snímku, avšak až po Update().

- **FixedUpdate()** – FixedUpdate() může být volán častěji než Update(), tedy i víckrát v rámci jednoho snímku. Funkce je volána na základě časovače nezávislém na snímkové frekvenci v základu každé dvě setiny sekundy.
- **OnDisable()** – Funkce je volána, jakmile je skript komponenta deaktivována.
- **OnDestroy()** – Funkce je volána po posledním snímku existence objektu (objekt může být zrušen voláním funkce Object.Destroy).

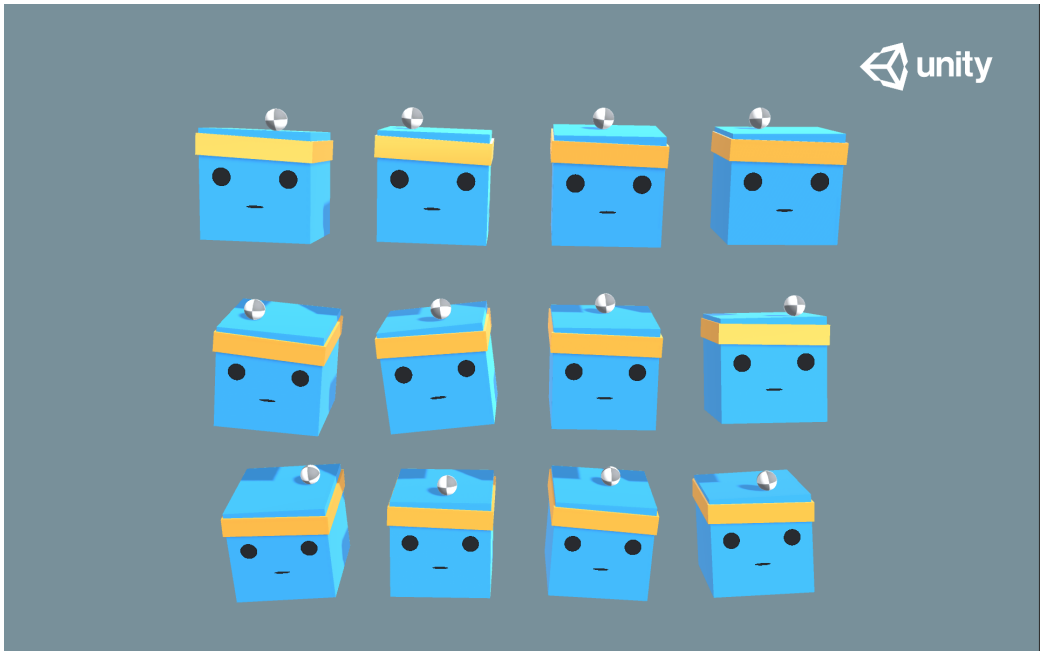


Obrázek 3.2: Zjednodušený vývojový diagram životního cyklu skriptu v Unity.

3.4 Unity Machine Learning Agents

Unity Machine Learning Agents, dále *ML-Agents*, je nástrojová sada, díky níž mohou hry a simulace vytvořené v Unity sloužit jako prostředí pro trénování inteligentních agentů. Jedná se o otevřený kód. Projekt podporuje hry ve 2D, 3D i ve virtuální realitě. Součástí *ML-Agents* je Python *API*, které uživateli umožňuje trénovat agenty prostřednictvím posilovaného učení 2.4.2, imitačního učení [32], a dalšími. Takto vytrénovaní agenti mohou být použiti pro řízení chování počítačem ovládaných charakterů ve hrách (non-playable characters), pro automatizované testování her nebo například pomáhat při rozhodování během vývoje herního designu samotného.

Třemi základními komponentami sady vývojových nástrojů ML-Agents jsou Agenti, sensory a akademie. *Agent* je v tomto kontextu chápán jako GameObject ve scéně. Může tak pozorovat prostředí, vykonávat akce a získávat za ně odměny 2.2. Může používat široké spektrum *sensorů* určených ke snímání různých forem informací, jako jsou vykreslené obrázky, paprsky ve scéně nebo vektory libovolné délky. Agenti mohou mít i společné strategie a sdílet zkušenostní data během trénování. *Akademie* je odpovědná za globální koordinaci prostředí simulace. Používá se ke sledování jednotlivých kroků simulace a správě agentů. Dále nabízí možnost měnit parametry prostředí (velikost a obecně přítomnost herních objektů, fyzika prostředí) i v průběhu trénování. Příkladem může být změna účinku gravitace v pravidelných intervalech či generování nových překážek, jakmile agent dosáhne určité míry dovedností.



Obrázek 3.3: Demonstrační aplikace *3D Balance Ball*. Agenti jsou modré objekty. Úkolem každého z nich je balancovat kuličku na své hlavě za přítomnosti gravitace co nejdéle tak, aby nespadla. [3]

Kapitola 4

Karetní hra Bang!

Bang! je italská karetní hra na motivy divokého západu, v základní verzi určena pro 4 až 7 hráčů. Hry jsou středně dlouhé, obyčejně dvacet až čtyřicet minut. V této kapitole jsou ve stručnosti popsána základní pravidla za účelem uvedení do kontextu kapitol následujících. Detailnější informace jsou k dispozici v oficiálních pravidlech [43].

4.1 Role, postavy

Každý hráč má ve hře přidělenu jednu ze čtyř různých rolí. Role hráči definuje, jaký úkol musí splnit, aby zvítězil:

- **Šerif** – Eliminovat všechny bandity a odpadlíka.
- **Pomocník šerifa (Vice)** – Eliminovat všechny bandity a odpadlíka.
- **Bandita** – Eliminovat šerifa.
- **Odpadlík (Renegade)** – Zůstat poslední ve hře. Nejprve eliminovat bandity a pomocníky šerifa, poté šerifa.

Hraje-li se ve čtyřech hráčích, do balíčku rolí se zamíchá šerif, odpadlík a dva bandité. Jako pátá role se přidává pomocník šerifa, jako šestá bandita, sedmou rolí je opět pomocník šerifa. Dále má každý hráč přiřazenu jednu z šestnácti karet postav (charakterů). Postava určuje, kolik má daný hráč na začátku hry životů. Tato hodnota je zároveň jejich maximálním počtem. Ztratí-li hráč svůj poslední život, je eliminován (přesto může být vítězem). Každá postava má navíc speciální schopnost.

4.2 Průběh hry

Na začátku hry si každý hráč vylosuje svoji roli. Šerif se odhalí, ostatní hráči svoji roli neprozrazují. Dále si každý vylosuje 2 karty postav, z nichž si jednu zvolí. Šerif má o jeden život více, než je definováno jeho postavou. Následně každý hráč obdrží z hracího balíčku 80 karet, počet odpovídající počtu jeho životů. Šerif zahajuje hru svým tahem. Hráči se na tahu střídají ve směru hodinových ručiček. Hra končí vítězstvím banditů, případně odpadlíka, je-li eliminován šerif, nebo naopak vítězstvím šerifa a pomocníků, jsou-li eliminováni všichni bandité a odpadlík.

```
while( !EndOfTheGame() )
{
    p = NextPlayer();

    // DRAWING PHASE
    p.DrawTwoCards();

    // PLAY PHASE
    p.Play();

    // DISCARD PHASE
    if( p.lives < p.CardsOnHand.Count )
    {
        p.Discard(p.CardsOnHand.Count - p.lives);
    }
}
```

Výpis 4.1: Pseudokód popisující průběh hry a jednotlivé fáze tahu.

4.3 Hrací karty

Hrací balíček obsahuje dva druhy karet. Karty s modrými okraji se vykládají před hráče (i cizí) na stůl. Jejich efekt platí, dokud nejsou odstraněny nebo přesunuty. Karty s hnědými okraji se odhazují (hrají) na odhazovací balíček, čímž se provádí jejich efekt. Nejtypičtějším příkladem může být odhození karty s názvem *Bang!* hráčem na tahu, který současně volí cíl. Napadený hráč má na výběr vyhnout se efektu zahráním karty *Vedle!* V opačném případě ztrácí život.

4.3.1 Modré karty

Každý hráč může mít před sebou na stole vyloženou maximálně jednu kartu s modrým okrajem od každého druhu (unikátní název), s výjimkou zbraně (*Schofield*, *Remington*, *Rev. Carabine*, *Winchester*). Tu může mít vyloženu jen jednu.

Kapitola 5

Návrh

Společenská hra Bang! má značnou popularitu napříč věkovými skupinami. Tato kapitola nejprve odpoví na otázku proč implementovat její počítačové zpracování a objasní jeho výhody. Bude představen návrh formátu zobrazení, základní koncept ovládání a volba cílových platforem. Dále se zaměří na síťový koncept. Na závěr bude objasněno, jaké jsou výhody, smysl a možnosti hry s umělou inteligencí.

Karetní hra je ve své původní deskové verzi nejběžněji hrána v pěti hráčích sedících okolo stolu. Uprostřed je dobírací a odhazovací balíček. Hráči vykládají lícem vzhůru karty do prostoru před sebou, kde dále mají odhalenu kartu své postavy, skrytou kartu své role (s výjimkou šerifa) a informaci o aktuálním počtu životů. Střídajíce se na tahu, hráči hrají dobírané karty ze své ruky.

Jednu z hlavních výhod počítačového zpracování shledávám v absenci manuální přípravy hry samotné. Příprava hry spočívající v zamíchání jednotlivých balíčků a rozdávání karet mezi účastníky je poměrně časově náročná. Ještě náročnější je při užití dalších rozšíření nad rámec základní varianty. Hra se často za přítomnosti velkého množství karet na stole, i bez zmíněných rozšíření, stává nepřehlednou. Pravidla jsou poměrně komplexní a často se v průběhu objevují situace, kdy mohou být méně zkušenými interpretovány nesprávně. Při nepozornosti přítomných může dojít nejen k neúmyslnému, ale i záměrnému přehlédnutí efektů karet na stole. Z tohoto úhlu pohledu tedy digitální verze hry může jasně definovat validitu tahů a správně interpretovat pravidla. Rozhodne-li se hráč pro určitý tah, není možné své rozhodnutí vzít zpět, což by hráče mělo vést k důkladnějšímu promyšlení svých akcí a obecně navyšovat kvalitu a úroveň hry. Nabízí se možnost ve hře zavést časový limit, který by udržoval tempo hry. Užití tohoto prvku by však dle mého názoru mělo převážně negativní dopad na celkový dojem a pocit ze hry. Zejména noví, méně zkušení hráči by na sobě mohli pociťovat zbytečný tlak.

5.1 Síťový koncept

Ačkoliv se zcela pochopitelně nabízí možnost koncipovat aplikaci jako online hru, v rámci své práce jsem se rozhodl ji navrhnout odlišně. Jelikož mají hráči skryté role zprvu nevědí, kdo jsou jejich spoluhráči a naopak soupeři. Mohlo by se zdát, že se u hry nekomunikuje. Opak je pravdou. Každý si na základě vývoje hry postupně skládá obrázek o tom, kdo mohou být jeho spoluhráči. Svojí hrou se jednotlivci v průběhu většinou prozradí a otevírá se možnost pro spolupráci, ale třeba i blafování. S verbální i neverbální komunikací a možností sledovat reakce ostatních může být hra zábavnější.

Z těchto důvodů jsem se rozhodl hru implementovat jako Klient-server aplikaci na lokální síti (LAN). Myšlenka spočívá v zobrazení herního pole na větším displeji a zprostředkování uživatelských interakcí skrze klientskou aplikaci na telefonu, tabletu, případně notebooku. Každý uživatel by prostřednictvím svého zařízení s operačním systémem Android, iOS, MacOS, či Windows mohl hrát karty ze své ruky, vybírat cíle, používat schopnosti své postavy a reagovat na akce ostatních.

5.2 Uživatelské rozhraní

Z pohledu uživatelského rozhraní je návrh aplikace výzvou. Důvodem je, jak již bylo zmíněno, rozsáhlost hracího pole při účasti více hráčů. Pro uživatele preferující mobilní zařízení s malými displeji by mělo být zobrazení hry na velké obrazovce komfortnější. Tato podkapitola se nejprve zaměří na existující zpracování uživatelských rozhraní karetních her pro více hráčů. Na základě takto nabytých poznatků bude následně popsán postup návrhu uživatelského rozhraní klientské a serverové aplikace.

5.2.1 Uživatelská rozhraní existujících karetních her

V rámci průzkumu a zhodnocení současného stavu řešení bych se pozastavil u dvou her. Přestože jsou postaveny na odlišném síťovém konceptu, mají s mým návrhem společný prvek, a to zobrazení několika hráčů v rámci okna aplikace. První z nich je *Governor of Poker 3*, viz. obrázek 5.1.

Obrázek 5.1: Online hra Governor of Poker 3.

Titul má dnes již mnoho verzí, tato konkrétní byla na platformě *Steam* zveřejněna v roce 2016. Jedná se dle mého názoru o velice povedené multi-platformní zpracování tradiční hry, jež má s hrou Bang! společnou westernovou tématiku. Hra je doprovázena zajímavými animacemi, audio efekty a hudbou. Vkusnou barevnou grafikou zdárně vtahuje hráče do děje

okolo pokerového stolu v salónu. Rušivým elementem zde jsou snad jen prvky uživatelského rozhraní podél horního a levého okraje. Jedná se o řadu reklam, událostí, nastavení, nápověd, žebříčků, úkolů a statistik [9].

Druhou hrou, kterou je třeba zmínit, je *Kraplow*, viz obrázek 5.2. I tento titul je prohlížečová hra. Jedná se o kopii originální verze. Řídí se oficiálními pravidly hry Bang! a obsahuje stejnou sadu karet, které jsou však jinak vyobrazeny. Hra je doplněna o chat a okno s textovým záznamem akcí ze hry. Hra byla zveřejněna v roce 2012. Kraplow neimplementuje rozšíření hry Bang! Podporuje přidání počítačových botů do hry. Celkový dojem kazí chaotické uspořádání prvků v okně. Karty, které má hráč na ruce, jsou zobrazeny na dvou místech. Chatovací okno, záznam herních událostí a duplicitní zobrazení karet na ruce ubírá prostor pro samotný herní plán. S výjimkou zbraní jsou karty s modrými okraji zobrazeny pouze textovou formou. Vlastní obrázky na kartách a portréty charakterů jsou velmi vydařené [21].

Obrázek 5.2: Online hra Kraplow je open-source kopií hry Bang!

5.2.2 Server

Uživatelské rozhraní serverové aplikace by mělo co nejvěrněji kopírovat reálnou hru. Každý, kdo ji někdy hrál, by se v něm měl zorientovat. Při návrhu jsem bral v potaz existenci mnoha rozšíření hry, přestože jejich implementace není plánována v rámci této práce. S nimi podporují oficiální pravidla hru až v osmi hráčích. Kromě nových druhů hracích karet a charakterů je v každém z rozšíření Divoký Západ, Pravé Poledne a Fistful speciální balíček. S použitím rozšíření se před hráče nevykládají jen karty s modrými okraji, ale i zelenými. S ohledem na pravidlo dovolující hráči mít před sebou na stole vyloženu jen jednu kartu téhož jména a maximálně jednu kartu zbraně není situace, kdy někdo takto vlastní deset

karet zcela nevšední. S ohledem na tato fakta mockup na obrázku 5.3 demonstruje nejhorší možnou situaci, jež by mohla nastat.

Obrázek 5.3: Ilustrační mockup zobrazující nejhorší možnou situaci na herním poli při účasti osmi hráčů s rozšířeními.

Fotografie demonstruje hru v osmi lidech. Každý z nich má před sebou vyloženo jedenáct karet. Mezi nimi i karty s hnědými okraji, které se však před hráče nevykládají (užity pouze ilustračně). Uprostřed obrázku je lícem vzhůru odhazovací balíček a lícem dolů balíček dobírací. Nalevo od nich jsou rozšíření (Divoký Západ chybí). Napravo od odhazovacího balíčku je prostor vyhrazen pro efekt karty Hokynářství. Každý hráč má před sebou kartičku s pěti náboji, která se používá jako počítadlo životů. Napravo od ní má každý svůj charakter. Ostatní karty ilustrují karty hrací.

5.2.3 Klient

Stejně tak uživatelské rozhraní klientské aplikace by mělo být v duchu snahy napodobit skutečnou hru s papírovými kartami. Hráč by měl mít možnost dohledat zde informaci o tom, jaká role mu byla přidělena a kterou postavu si na začátku hry zvolil. Zejména druhou informaci by si měl moci během hry snadno připomenout a přečíst si přesné znění schopnosti svého charakteru. Dále bude důležitý formát volby cíle efektů karet po jejich zahrání, což bude řešeno jednoduše – způsobem vertikálního menu. O stavu hry a nutnosti reagovat na cizí akce budou hráči informováni formou textových notifikací.

Obrázek 5.4: Grafický návrh uživatelského rozhraní klientské aplikace.

I v případě, že by měl hráč zrovna v ruce mnoho karet, by mělo být možné zobrazit si jednu konkrétní a přečíst si její popis či se podívat na symboly vyobrazené v témže prostoru, což je druhý způsob interpretace jejich efektů. Karty na ruce by měly býti nejdominantnějším prvkem grafického uživatelského rozhraní klientské aplikace. Funkce zahrání dané karty bude realizována patrně jejím tažením vzhůru a následným uvolněním. Na základě návrhu z obrázku 5.4 jsem se rozhodl kartami na displeji nerotovat do tvaru vějíře, ale zobrazit je s nulovým úhlem rotace. Kýženým účinkem je, aby byly na displeji co největší a zároveň, aby ve spodní části nezasahovaly do pomyslného prostoru pro situační tlačítka (na obrázku 5.4 vyznačeno vodorovnou osou).

Obecně by uživatelské rozhraní klientské aplikace mělo být intuitivní, jednoduché a graficky poutavé s dobře využitou plochou displeje a adaptací na různá rozlišení cílových zařízení. Tomu by mohly dopomoci textové notifikace.

5.3 Inteligentní hráč

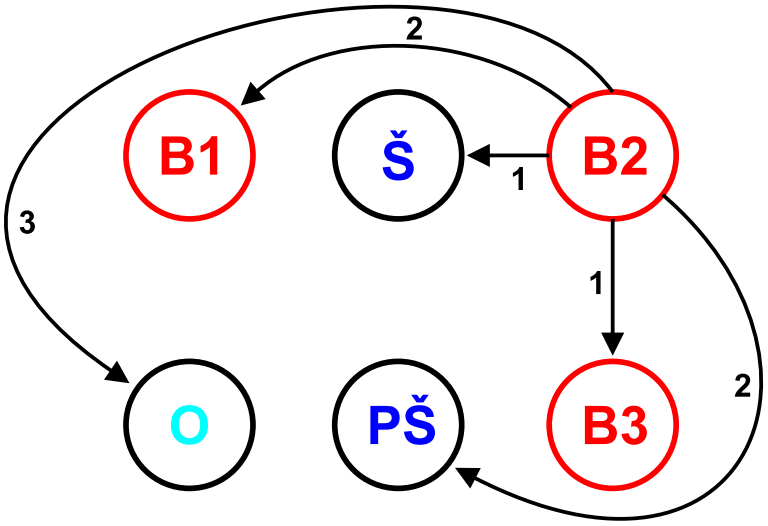
V rámci práce budu implementovat inteligentního hráče. Koncový uživatel by možnost přidat jej do hry mohl ocenit v případě, kdy pro hru není dostatek hráčů. Dalším důvodem může být rozdělování čtyř druhů rolí mezi jednotlivé účastníky. S určitým počtem hráčů může být totiž jedna ze stran mírně zvýhodněna.

Důvodem je odpadlík, který je dle mého názoru nejnáročnější rolí ve hře. Aby si totiž udržel šanci na vítězství, měl by ve hře udržovat vyrovnanost soupeření banditů se šerifem a jeho pomocníkem, viz. sekce 4.1.

Příkladem může být hra v šesti. V tomto případě si tři hráči vylosují roli bandity, jeden odpadlíka, jeden bude šerif a poslední plní úlohu pomocníka šerifa. V této situaci se předpokládá, že odpadlík bude na straně šerifa, aby byla ve hře vyrovnaná situace tři na tři. Odpadlíkovi však na rozdíl od pomocníka stačí, aby šerif nebyl eliminován. Koneckonců jeho cílem je zůstat poslední ve hře společně se šerifem a toho poté vyřadit.

Ve hře záleží i na rozsazení okolo stolu, tedy vzdálenostech mezi hráči. Důvodem je, že některé karty je možné použít pouze na hráče, kteří jsou v určitém dosahu. Nevlastní-

li například hráč zbraň (vybrané karty s modrými okraji), za normálních okolností kartou *Bang!* dostřelí pouze na hráče sedícího nalevo a na hráče sedícího napravo od něj. Následující příklad vysvětluje možné znevýhodnění strany šerifa při hře v šesti při konkrétním rozsazení.



Obrázek 5.5: Princip vzdáleností mezi hráči z pohledu bandity B2. Hodnoty u orientovaných hran jsou hodnotami vzdálenosti. Symboly *Š*, *PŠ* a *O* reprezentují postupně hráče rolí šerif, pomocník šerifa a odpadlík. Bandité jsou označeni symbolem *B* a číselným identifikátorem.

V případě brzkého otevření hry bandity, z nichž dva mají od prvního kola šerifa v dosahu, je šerif značně znevýhodněn. Přidáním sedmého hráče, jímž dle pravidel bývá druhý pomocník šerifa, se snižuje pravděpodobnost, že dojde ke konfiguraci, kdy okolo šerifa sedí tři bandité. Šerif tak zároveň získá druhého plnohodnotného spoluhráče. Odpadlík poté v sedmi hráčích zprvu bývá neutrální.

5.3.1 Klasifikace karetní hry Bang!

Z dosud popsaných informací týkajících se hry Bang! v kapitolách 4 a 5 je možné ji klasifikovat podle kritérií v kapitole 2 a najít podobnosti s tamže zmíněnými hrami. Bang! je zástupcem skupiny *Imperfect information games*, a to hned z několika důvodů. Tím nejzjevnější je fakt, že jednotliví hráči mají na ruce karty, které jsou jejich soukromou informací. Dále nevědí, kdo jsou jejich spoluhráči a naopak nepřátelé. Hráč také neví, jaké karty dostane z dobíracího balíčku. Ve hře se vyskytuje náhoda. Jedná se o kooperativní hru, určenou pro čtyři a více hráčů. Kooperativní se stává například v moment, kdy je odpadlík nucen přidat se na stranu šerifa v situaci, kdy šerifovi hrozí eliminace bandity. Je sekvenční, protože se hráči po směru hodinových ručiček střídají na tahu. Společným rysem této hry s Magic: The Gathering je fakt, že se nehraje se standardním balíčkem karet. Podobnost můžeme nalézt i s karetní hrou Jass, a to v existenci týmů. Ve Švýcarské koordinační hře jde vždy o dva dvoučlenné týmy, kdežto ve hře Bang! se jedná o týmy tři.

5.3.2 Vymezení požadavků

Vytyčení tohoto cíle vychází z nastudované teorie. Cílem mé práce v kontextu inteligentního hráče není vytvořit zcela nechybujícího hráče, nýbrž navrhnout a naimplementovat umělou

inteligenci, která bude hrát jako běžný hráč karetní hry Bang! V určitých ohledech by měla převyšovat průměrného hráče, na druhou stranu dopouštění se drobných taktických chyb by nemělo reálným účastníkům zkazit dobrý pocit ze hry. Hrát s inteligentním hráčem by mělo být v první řadě zábavné. Dle dělení podle síly umělé inteligence představeného v kapitole 2.1, by se mělo jednat o inteligentního hráče na úrovni *Par-human*. Naplnit tento cíl jsem se rozhodl formou server-side rule-based umělé inteligence.

5.3.3 Heuristika odhadu rolí

Jádrem a klíčovým prvkem implementace inteligentního hráče bude heuristika odhadu rolí. Ten by měl na základě sledování událostí ve hře odhadnout, kdo jsou jeho spoluhráči a kdo naopak jeho soupeři. Hráč může nejen ohrožovat a škodit svým oponentům, ale i pomáhat svým spoluhráčům. Způsobů, jimiž by inteligentní hráč mohl na základě dobré analýzy zvyšovat šance na svoji výhru je několik. Soupeřům může brát a odhazovat náhodné karty z ruky, ale i konkrétní karty vyložené. Jelikož existují i vykládací karty, které zapříčiňují nežádoucí efekt, může inteligentní hráč jejich odstraněním pomoci svým spoluhráčům. Mimo to je nejprostší cestou k výhře eliminace protivníků. S ohledem na implementaci by hráči měli být hodnoceni koeficienty vyjadřujícími pravděpodobnost, že jim ve hře byla přiřazena určitá role. K práci s nimi je třeba vytvořit sadu pravidel.

Kapitola 6

Implementace klientské aplikace

V rámci této kapitoly bude popsána implementace klientské aplikace. Nejprve bude představena finální podoba uživatelského rozhraní a způsob jeho realizace v Unity. Dále bude rozebrán systém hraní karet. Poslední část této kapitoly je věnována zajímavým prvkům uživatelského rozhraní, jejich implementaci a vybraným uživatelským interakcím. Klientská i serverová aplikace jsou implementovány v jazyce *C#*.

6.1 Finální podoba klientské aplikace

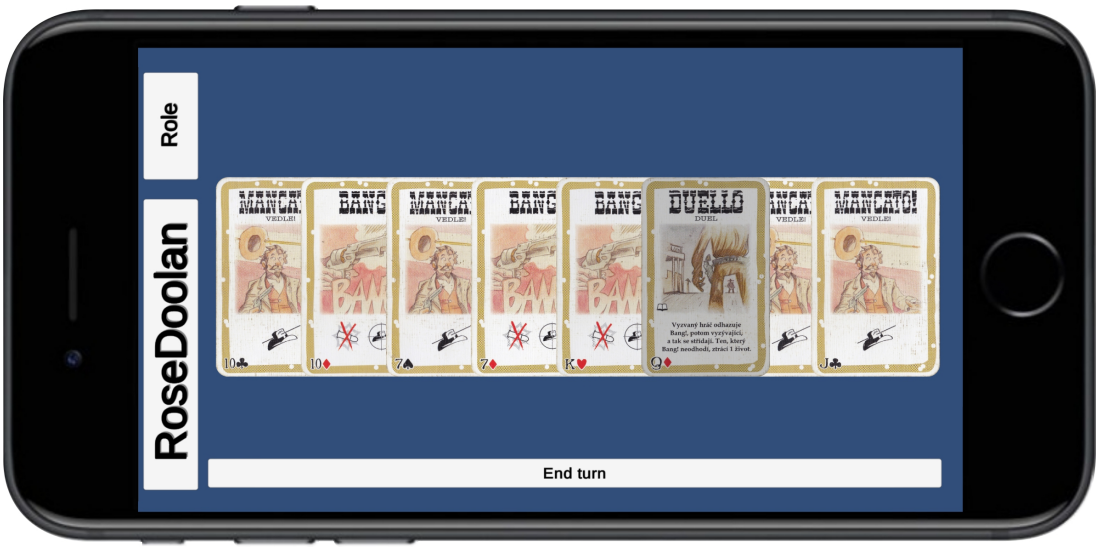
Uživatelské rozhraní je skupinou zanořených vertikálních a horizontálních rozložení prvků. Hrací karty jsou elementy horizontálního rozložení, jež je součástí této hierarchie. Vytvářeny jsou jako instance objektu prefab, jemuž je přidělen sprite (2D grafický objekt), na základě toho, o jakou kartu se jedná. Hrací karta je definována třemi hodnotami – hodnota (dva až eso), barva (srdce, káry, piky, kříže) a druhem karty (například *Bang!* nebo *Vedle!*). Tato reprezentace je používána i na serverové straně. Překrytí karet jsem docílil zápornými mezerami (spacing) v rámci horizontálního rozložení, viz. obrázek 6.1.

Obrázek 6.1: Finální implementovaná podoba uživatelského rozhraní klientské aplikace.

6.2 Princip hraní karet

Manipulace s hracími kartami je nejdůležitějším mechanismem klientské aplikace. Hráč obyčejně mívá na ruce tři až pět karet. Důvodem je zahazování přebytečných karet na konci tahu, viz. sekce 4.1. Má-li však hráč štěstí, může se stát, že bude mít v ruce během svého tahu karet třeba i dvanáct. Z tohoto důvodu je implementována funkce, umožňující jednotlivé karty zobrazit, aby si uživatel mohl přečíst popisky a symboly na kartách. Prefab hrací karty má připojenu komponentu spouštěč událostí (Event Trigger). Systémovými událostmi, které jsou v rámci každé hrací karty sledovány, jsou *Pointer Down* (Stisk levého tlačítka myši nad objektem), *Pointer Up* (Uvolnění levého tlačítka myši nad objektem) a *Pointer Drag*. Pointer Drag vyjadřuje situaci, kdy je nad objektem stisknuto levé tlačítko. Funkce je volána v každém snímku, dokud tlačítko myši není uvolněno. Na dotykových zařízeních jsou tyto systémové události vyvolány, dotýká-li se uživatel displeje. Při detekci každé z těchto událostí dochází k volání implementované funkce.

Je-li detekována událost Pointer Down, dojde k dočasnému ignorování uspořádání vrstev ve scéně a zvolená karta je zobrazena ve vrstvě nejpřednější. Toho je docíleno díky komponentě herního objektu *Canvas*, která je při výskytu této události na hrací kartě vytvořena skrze skript. Jakmile je detekována událost Pointer Up, dochází k odstranění komponenty Canvas. Tím pádem je obnoveno původní uspořádání v rámci vrstev.



Obrázek 6.2: Klientská aplikace na telefonním zařízení v průběhu hry, dotkne-li se uživatel některé z karet.

Při události Pointer Down je také uložena pozice objektu ve scéně. Při uchopení a tažení karty (událost Pointer Drag) karta v každém snímku následuje polohu kurzoru a tím mění svoji pozici ve scéně. Jakmile uživatel *A* kartu pustí, jsou rozlišovány dvě situace. Pokud hráč *A* není na tahu a není po něm vyžadována interakce v průběhu tahu soupeřů (například když hráč *B* ve svém tahu vystřelí kartou *Bang!* na hráče *A*), po uvolnění dochází k návratu karty na pozici, jež byla uložena při jejím uchopení. Pokud je vyžadována reakce, nebo je-li hráč *A* na tahu, je rozhodující, kde po tažení v rámci scény kartu pustil. Pokud k tomu došlo v horní pětině displeje (viz. obrázek 6.3), karta je odstraněna z horizontálního rozložení a poslána na server. Pokud byla uvolněna v jiné části displeje, vrací se zpět na původní pozici.

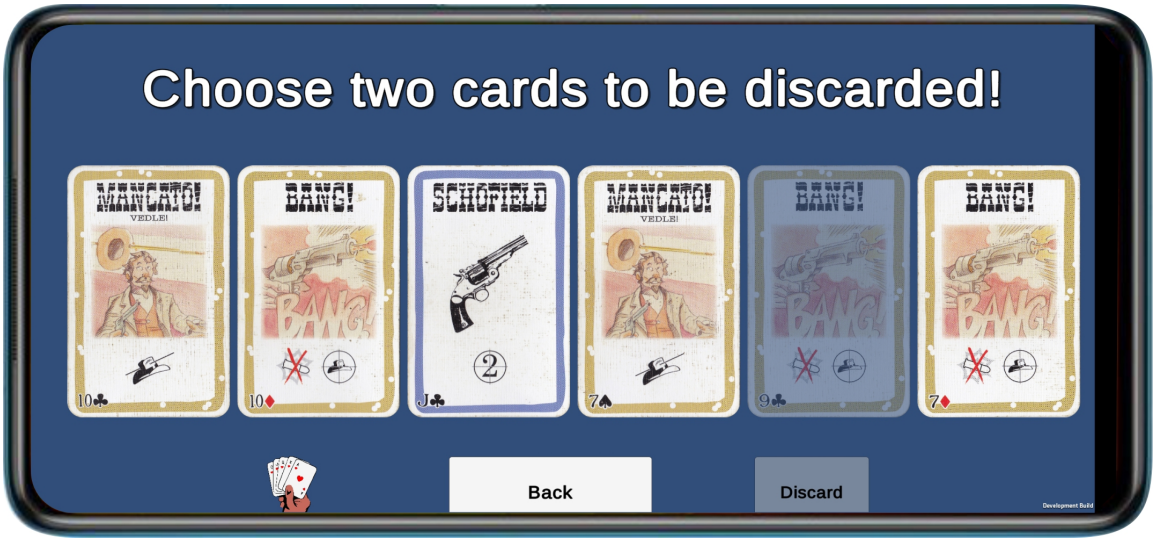
Obrázek 6.3: Tažení karty vzhůru za účelem jejího zahrání. Detekční oblast ilustračně dokreslena světle modrou barvou.

6.3 Další prvky uživatelského rozhraní a vlastnosti aplikace

Vybírá-li hráč v průběhu hry kartu v rámci provedení určitého efektu (volba v Hokynářství, zrušení nebo přivlastnění karty s modrým okrajem soupeře, a další...), nevidí na displeji karty, které má na ruce. Z tohoto důvodu je při těchto situacích k dispozici tlačítko pro zobrazení herní ruky. Je-li uživatel vyzván k učinění takové volby, je mu umožněno se na karty v ruce podívat, a to po dobu stisku onoho tlačítka. Tento prvek je taktéž implementován pomocí spouštěče událostí.

Zobrazení role a postavy je realizováno zcela dle návrhu aplikace. Při stisku odpovídajícího tlačítka se postava či role zobrazí. Následné kliknutí kdekoliv na displeji kartu zase skryje. Volby cílů jsou implementovány jako vertikální rozložení tlačítek.

Zajímavým konceptem je například způsob implementace schopnosti postavy *Sid Ketchum*. Ta spočívá v možnosti se kdykoliv během hry rozhodnout pro odhození dvou karet ze své ruky a získat tím zpět jeden život. Použitím této schopnosti může zabránit i svému vyřazení ze hry, jedná-li se o život poslední. Pokud si hráč vylosuje a zvolí postavu *Sid Ketchum* tehdy, když nemá plný počet životů, zobrazí se ve spodní části uživatelského rozhraní klientské aplikace pod kartami tlačítko pro použití jeho schopnosti. Po jeho stisku vypadá uživatelské rozhraní jako na obrázku 6.4.



Obrázek 6.4: Uživatelské rozhraní po stisku tlačítka pro použití schopnosti postavy *Sid Ketchum*. K odhození v rámci schopnosti je označena druhá karta zprava – *Bang!* Jelikož momentálně nejsou označeny dvě, na tlačítko *Discard* není možné kliknout.

Karty jsou zobrazeny celé vedle sebe v rámci horizontálního rozložení. Každé z nich je přidána komponenta herního objektu *Button*. Při kliknutí na danou kartu dojde k jejímu výběru, tedy internímu uložení. Tato akce je reflektována změnou barvy karty. Při druhém kliknutí na tutéž kartu dojde k jejímu internímu odebrání ze seznamu zvolených karet a opětovné změně barvy zpět. Toto uložení vybraných karet je realizováno pomocí *UnityEvent.AddListerner()*. UnityEvent je v tomto případě stisk tlačítka. V reakci na tuto událost je proveden kód v argumentu. Ve chvíli, kdy jsou vybrány právě dvě karty (dle definice schopnosti), se tlačítko *Use Ability* stává aktivním – *interactable*. Jeho stiskem dochází k odebrání dvou vybraných karet z ruky, hráč získává jeden život a zobrazí se zpět jeho ruka. Změna barvy při označování dané karty je realizována skrze skript, kde se mění hodnota *Alpha* komponenty barvy obrázku (sprite). Pokud si uživatel použití schopnosti v průběhu vybírání karet k zahození rozmyslí, k návratu může kdykoliv využít tlačítko *Back*.

Mezi zajímavé vlastnosti aplikace patří přítomnost vibrací mobilních zařízení. V momentu, kdy začíná tah hráče, jeho zařízení krátce zavibruje. K tomu v Unity slouží funkce *Handheld.Vibrate()*, která funguje na operačních systémech Android a iOS. Další uživatelsky přátelskou vlastností aplikace je zamezení zamykání obrazovky během hry.

Kapitola 7

Implementace serverové aplikace

Zatímco klientská aplikace pouze zprostředkovává uživatelské akce, logika hry je implementována na serveru. Jakmile se všichni hráči připojí do herní relace a hru spustí, na serveru dojde ke změně scény a vykreslí se herní plán. Zobrazí se na něm jednotliví hráči, a dojde k vytvoření a zamíchání balíčků – hrací karty, role, postavy. Každému hráči server přidělí roli a zašle dvě karty postav. Jakmile si hráč jednu z nich vybere, server ji zobrazí na herním plánu společně s patřičným počtem životů a zašle mu odpovídající počet hracích karet. Poté, co si všichni hráči zvolí postavu, server zasílá další dvě hrací karty společně s pokynem k zahájení hry tomu z nich, jemuž byla přidělena role šerifa tak, jak definují pravidla [43].

V této kapitole bude nejprve představena finální podoba uživatelského rozhraní serverové aplikace. Dále bude popsán formát komunikace mezi serverem a klienty. Poté bude vysvětlen princip střídání na tahu, zpracovávání klienty hraných karet, výpočet vzdáleností mezi hráči a na závěr budou zmíněny některé další vybrané vlastnosti aplikace.

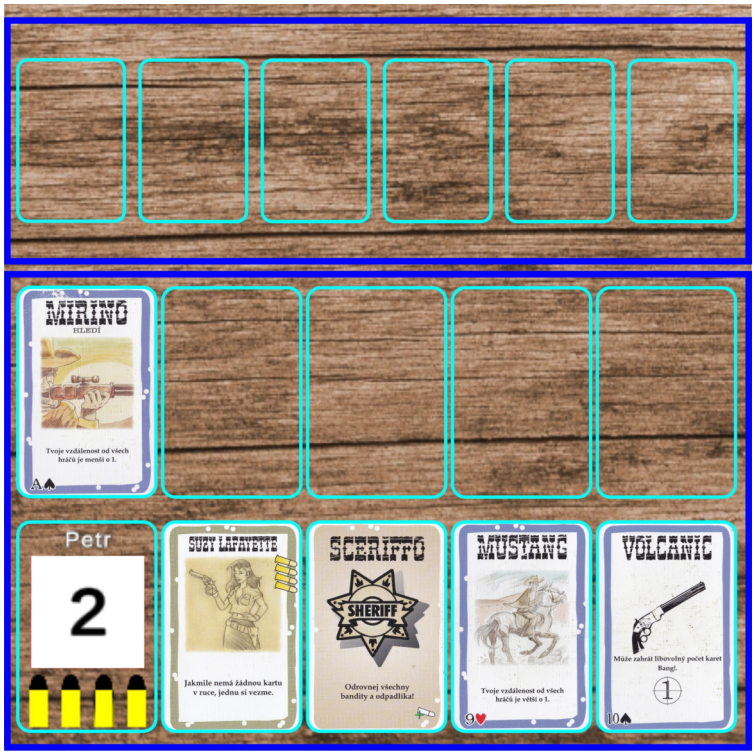
7.1 Finální podoba serverové aplikace

Rozehraná hra může na serveru vypadat tak, jako na obrázku 7.1. I zde se jedná o spletitou hierarchii rozložení. Při jejím generování jsou jednotliví hráči umísťování střídavě do vrchního a spodního horizontálního rozložení. Prostřední horizontální rozložení má uprostřed dobírací a odhazovací balíček, pod nimiž je textová informace, kolik v nich je momentálně karet. Napravo je ve hře momentálně sedm karet otočených v rámci efektu *Hokynářství*.

Obrázek 7.1: Finální implementovaná podoba uživatelského rozhraní serverové aplikace.

7.1.1 Panel hráče

Panel hráče je mřížkovým rozložením (grid layout) karet. Na obrázku 7.2 je zvětšený panel jednoho konkrétního hráče. Úplně vlevo dole je jméno hráče zadané při vytváření hry. Pod ním je čtverec s číslem, které informuje, kolik má daný hráč karet v ruce, což je důležitá herní informace. Hráč, který je momentálně na tahu, je označen červeným čtvercem. U ostatních hráčů je čtverec bílý. Níže je vyobrazen aktuální počet životů daného hráče reprezentovaný obrázky (sprites) žlutých nábojů. Dále vpravo je postava, kterou si daný hráč na začátku hry vybral. Další karta vpravo otočená lícem dolů, je skrytá role. Ta je otočena lícem vzhůru, jakmile dojde k eliminace daného hráče (ztráta posledního života). Zbylé karty jsou vyložené hrací karty s modrými okraji. Nad tímto mřížkovým rozložením je rezervní horizontální rozložení prvků, kam jsou vykládané karty přidávány, pokud je mřížka zaplněna. V případě velkého počtu vyložených karet v rámci rezervního rozložení jsou jejich rozměry zmenšovány (scale). Díky tomu nemůže nastat situace, že by pro vykládanou kartu neexistoval prostor na herním plánu.

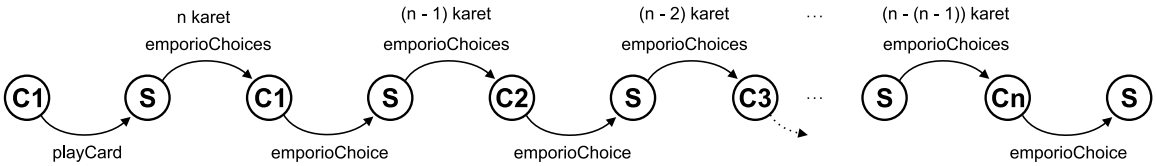


Obrázek 7.2: Detail panelu hráče.

7.2 Komunikace klient-server

Komunikováno je skrze protokol TCP (Transmission Control Protocol). Zprávy, jež si server a klienti vyměňují, obsahují povětšinou hodnoty datového typu *integer*. Na serverové i klientské straně jsou implementovány funkce pro vytváření a čtení jednotlivých druhů těchto zpráv.

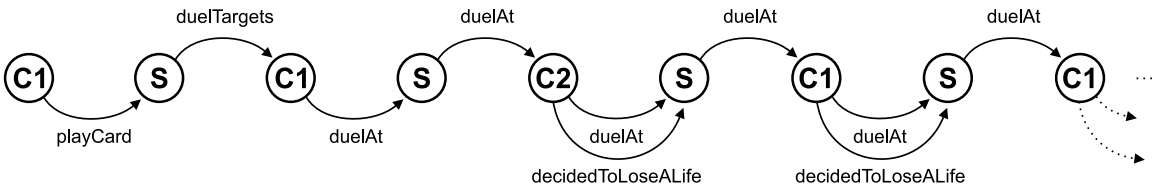
Příkladem takové výměny zpráv může být následující situace. Je-li hráčem zahráno *Hokynářství* dle pravidel [43], je z dobíracího balíčku otočeno tolik karet, kolik je ve hře hráčů (mimo eliminované). Počínaje hráčem, který Hokynářství zahrál, všichni hráči (mimo eliminované) po směru hodinových ručiček volí jednu ze zbývajících karet, a tu si vezmou do ruky. Výměna zpráv mezi serverem a klienty při zahrání této karty je demonstrována na obrázku 7.3.



Obrázek 7.3: Komunikace serveru a klientů v případě zahrání karty *Hokynářství* klientem *C1*. Symbol *S* reprezentuje server. Klienti jsou reprezentováni symbolem *C* následovaným jejich identifikační číslem. Číslo *n* je rovno celkovému počtu neeliminovaných hráčů. Názvy u šipek jsou názvy klientských a serverových zpráv.

Zpráva *playCard* obsahuje kartu *Hokynářství*. Zpráva *emporioChoices* obsahuje karty, z nichž může klient vybírat a *emporioChoice* obsahuje klientem zvolenou kartu. Jakmile poslední hráč (*Cn*) pošle zprávu *emporioChoice*, pokračuje tah hráče *C1*.

Druhým příkladem je zahrání karty *Duel* klientem *C1* na klienta *C2*, viz. obrázek 7.4. Jakmile jeden z klientů zašle zprávu *decidedToLoseALife*, server již dále zprávu *duelAt* neposílá. Duel končí a pokračuje tah hráče *C1*.



Obrázek 7.4: Komunikace serveru a klientů v případě zahrání karty *Duel*.

7.3 Střídání hráčů na tahu

Po vygenerování hráčských panelů vzniká pořadník. Realizován je pomocí abstraktního datového typu – obousměrně vázaný kruhový lineární seznam. I zde je důvodem snaha navrhnout aplikaci odolnou budoucnosti. S použitím rozšíření je totiž možné střídání tahů i proti směru hodinových ručiček.

Jakmile se hráč na tahu rozhodne svůj tah ukončit, klientská aplikace posílá zprávu *endTurn* serveru. Server v rámci funkce *GameProgress.PlayerOnTurnShift(int whosEndedTurn)* zjistí, kdo je dalším hráčem na tahu. Před tím, než je druhému hráči zaslán pokyn k zahájení tahu, je třeba zkontrolovat, zda před sebou na stole nemá vyloženy některé karty s modrými okraji – konkrétně *Vězení* a *Dynamit*. V obou případech se otáčí vrchní karta z dobíracího balíčku na balíček odhazovací. Podle hodnoty a barvy (srdce, kříže, ...) se rozhodne, zda bude proveden jejich efekt. V případě *Vězení* je efektem vynechání tahu. Hráč je tím pádem přeskočen, a pokračuje až hráč třetí. V případě *Dynamitu* je efektem ubrání tří bodů života, což často vede k eliminaci hráče. Pokud má daný hráč před sebou obě tyto karty, nejprve se otáčí na *Dynamit* [43]. Pokud má hráč například postavu *Lucky Duke*, při otáčení na každou z těchto dvou karet otáčí karty dvě a jednu si vybírá. To snižuje pravděpodobnost, že dojde k provedení těchto nežádoucích efektů. Otáčení za účelem zjištění, zda dojde k provedení efektů těchto karet či nikoliv, je prováděno na serveru automaticky, a to v rámci této funkce.

7.4 Zpracovávání hraných karet

Server při zpracovávání klienty hraných karet pracuje ve dvou režimech. Zahraje-li hráč na tahu kartu, na jejíž efekt musí reagovat ostatní hráči (například *Indiáni!*, *Gatling*, *Duel*, ...), server je přepnut do stavu *awaitingReactions*. V ten moment je tah hráče pozastaven a znova pokračuje až po reakci všech zapojených hráčů. Další obecně důležitou informací je, zda je majitel zahrané karty na tahu. Není-li, může pouze reagovat, když je k tomu vyzván. Pokud je zahrání určité hrací karty z jakéhokoliv důvodu nelegálním tahem, server klientu zahranou kartu vrací zpět.

7.5 Výpočet vzdáleností mezi hráči

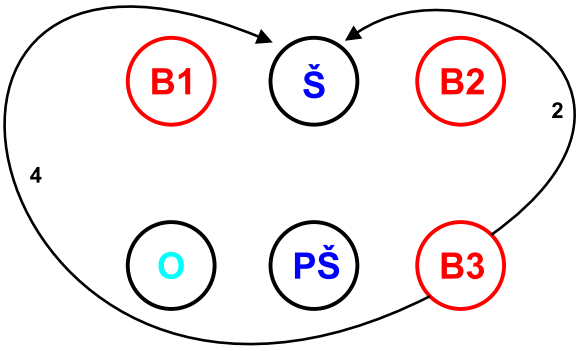
Princip vzdáleností, který byl nepřímo vysvětlen v rámci podsekce návrhu inteligentního hráče 5.5, je počítán funkcí *TargetsInRange()*. Vzdálenost, kromě rozsazení okolo stolu ovlivňují i vybrané karty s modrými okraji (*Hledí*, *Mustang* a karty zbraní) a některé postavy. Při eliminaci libovolného hráče se hráč sedící po jeho levici a hráč sedící po jeho pravici stávají sousedy, mezi nimiž je poté vzdálenost jedna.

Funkce prochází obousměrně vázaný kruhový lineární seznam neeliminovaných hráčů a vytváří z nich seznam těch, kteří jsou v dosahu. Ten poté obyčejně bývá zaslán hráči, který zahrál kartu, jež má definovánu maximální vzdálenost pro volbu cíle (karta *Panika!*, nebo *Bang!*). Jména těchto hráčů následně vidí na klientské aplikaci jako vertikální rozložení herních objektů s komponentami *Button*. Ty umožňují vybrat si jeden konkrétní cíl z množiny všech možných.

Tento kruhový seznam je procházen v obou směrech. Důvodem je již zmíněný koncept vzdáleností mezi hráči. Hráč sedící nalevo i hráč sedící napravo jsou oba z pohledu toho, kdo je mezi nimi, jeden vzdáleností bod daleko. Samotný výpočet se řídí výrazem 7.1. Daný cíl je v dosahu, pokud je vyhodnocen jako pravdivý.

$$(P_1\_gunRange + P_1\_additionalRange) - P_2\_hideCoef - distance + 1 > 0 \qquad (7.1)$$

$P_1\_gunRange$ je dosah útočícího hráče definovaný vyloženou zbraní. Výchozí hodnota je rovna jedné (pokud hráč nevlastní zbraň). $P_1\_additionalRange$ je dodatečné navýšení dostřelu útočníka prostřednictvím ostatních karet (karta *Hledí*, schopnosti postav). Výchozí hodnotou je nula. $P_2\_hideCoef$ je ostatními kartami určený koeficient dodatečného vzdálení se (navyšován kartou *Mustang* a schopnostmi postav). Výchozí hodnota je také rovna nule. Číslo *distance* je definováno rozsazením okolo stolu jako hodnota o jedna větší, než je počet hráčů sedících mezi hráči P_1 a P_2. Důvod procházení obousměrně vázaného kruhového seznamu neeliminovaných hráčů v obou směrech je znázorněn na následujícím obrázku 7.5.



Obrázek 7.5: Hodnoty u orientovaných hran jsou hodnotami *distance*. Symboly *Š*, *PŠ* a *O* reprezentují postupně hráče rolí šerif, pomocník šerifa a odpadlík. Bandité jsou označeni symbolem *B* a číselným identifikátorem.

Díváme-li se po směru hodinových ručiček z pohledu bandity *B3*, šerif je vzdálen čtyři body vzdálenosti. Výhodnější je počítat číslo *distance* v opačném směru, kde je jeho hodnota poloviční. Pokud tedy šerif má nulovou hodnotu *hideCoef*, stačí banditovi *B3* zbraň s dostřelem dva (nastavila by $P_1\_gunRange$ na hodnotu dva), případně karta *Hledí*, která zvyšuje $P_1\_additionalRange$ o jeden bod.

7.6 Další prvky uživatelského rozhraní a vlastnosti aplikace

Je-li některý z hráčů eliminován, dochází ke kontrole podmínek ukončení hry v rámci skriptu *WinningConditions*. Pokud došlo k naplnění vítězných podmínek, server zasílá zprávu s výsledkem všem klientům. Všichni účastníci následně na obrazovkách svých zařízení uvidí, zda jsou na straně vítězů, či poražených. Na serveru dochází k odhalení rolí všech hráčů.

Pokud v průběhu hry dojdou karty v dobíracím balíčku, nový je vytvořen zamícháním balíčku odhazovacího. Oba jsou implementovány jako abstraktní datový typ zásobník. Ve snaze vtáhnout účastníky do hry, jsou ve hře přítomny zvukové efekty pro jednotlivé hrací karty.

Kapitola 8

Implementace inteligentního hráče

Kapitola osmá je věnována inteligentnímu hráči, přestože je implementován v rámci serverové aplikace. Implementován je formou *rule-based umělé inteligence*, viz. sekce 2.4.1. Na základě informací ze sekce 4.1, tedy existence čtyř druhů rolí, je strategie inteligentního hráče závislá na tom, kterou z nich na začátku hry obdržel.

Hra inteligentního hráče se dá rozdělit na hru ve svém a mimo svůj tah. Řešení druhé části je z pohledu definice chování snadné a totožné pro všechny role. V těchto situacích má povětšinou hráč na výběr dvě možnosti. Typickým příkladem může být opět nejzákladnější herní situace. Je-li na hráče zahrána karta *Bang!*, má napadený dvě možnosti – vyhnout se zahráním karty *Vedle!*, nebo si ubrat život. Má-li v ruce kartu *Vedle!*, vždy je lepší ji použít. Blafovat a ubrat si život se při těchto situacích nemusí vyplatit, jelikož hráč může být následně o tuto kartu připraven kartami *Cat Balou*, či *Panika!* Navíc se hráč může ocitnout před rozhodnutím odhodit kartu *Bang!*, nebo si ubrat život (efekty karet *Indiáni!* nebo *Duel*). V takové situaci karta *Vedle!* ztrátě života nezabrání.

Hra během svého tahu je výrazně komplexnější a značně závislá na vylosované roli. Zejména proto, že cílem každého hráče je škodit svým nepřátelům a chránit se před nimi. Kdo je nepřítelem, je definováno rolemi. Ty jsou však skryty. V případě Odpadlíka záleží i na tom, kdo ve hře v daný moment vyhrává. Jak implementovaný inteligentní hráč ví, komu škodit a před kým se má chránit je vysvětleno v podkapitole 8.1.

8.1 Heuristika odhadu rolí

Každý hráč v rozehrané hře je definován trojicí koeficientů – {*V_estimate*, *B_estimate* a *R_estimate*} – postupně pomocník šerifa, bandita a odpadlík. Každý z těchto koeficientů nabývá hodnoty od nuly do sta. Celkový součet těchto tří hodnot je vždy roven stu. Na základě hráčských akcí ve hře dochází ke změnám těchto koeficientů, které je tedy možno chápat jako rozdělení pravděpodobnosti diskrétní náhodné veličiny, jíž je role. Je-li hodnota koeficientu *B_estimate* rovna osmdesáti, je dle implementované heuristiky možné situaci interpretovat tak, že daný hráč je na osmdesát procent banditou. V průběhu hry se však při střelbě do šerifa hodnota koeficientu *R_estimate* přesouvá do koeficientu *B_estimate*. Význam tohoto koeficientu se tedy později mění obecně na nepřítele šerifa a jeho pomocníků.

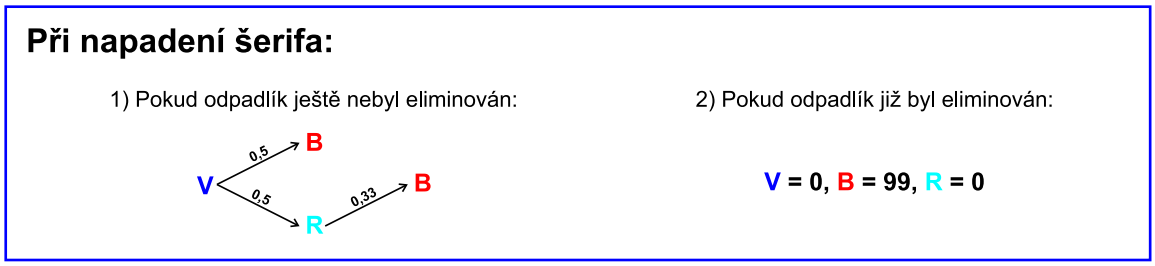
Tyto koeficienty jsou měněny na základě herních akcí daných hráčů. Tento systém ohodnocování je v rámci aplikace jeden. Herních akcí, z nichž by bylo možné vyvozovat závěry a měnit zmíněné koeficienty, je několik. Příkladem může být volba cíle pro karty *Bang!*, *Duel*, *Vězení*, *Panika!*, *Cat Balou*, ale i karty s plošnými efekty, jimiž jsou *Indiáni!* a *Kulomet*.

V rámci abstrakce problému však systém uvažuje pouze první dvě zmíněné karty – *Bang!* a *Duel*, které mají výpovědní hodnotu nejvyšší. Účelem jejich použití je snaha ubrat cíli život. Jako příklad – pomocník šerifa by nikdy nevystřelil do šerifa. Taková akce je nesmyslná. Učiní-li tak, nic nezíská. Vědomě naopak snižuje svoji šanci na výhru. Nejedná se o „blaf“, ale o nekorektní hru. Naopak sebrat ze stolu například kartu *Barel* může být taktika a nemusí nutně znamenat, že je její původní majitel nepřítelem onoho hráče. Proto užívání ostatních zmíněných karet v rámci úpravy koeficientů systém nezohledňuje.

Prvním podnětem k uvažování nad rolemi hráčů je moment, kdy některý z hráčů otevírá hru použitím *Bang!* nebo *Duel* (dále označováno jako „střelba“) na šerifa. Hráči, kteří takto učiní, jsou ukládání do *List<int> AttackedSheriffList*, který tedy sdružuje šerifovy nepřátele – bandity. V určité fázi hry do něj může být přidán odpadlík.

Změny koeficientů

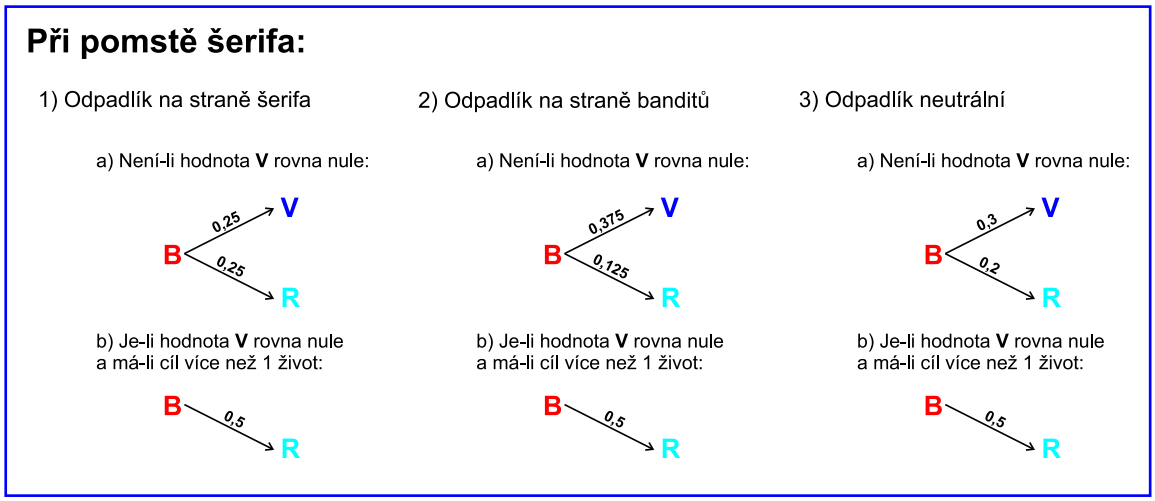
Při střelbě na šerifa je *V_estimate* daného hráče nastaven na hodnotu 0 a předešlá hodnota tohoto koeficientu je rozdělena mezi *B_estimate* a *R_estimate* v určitém poměru. Koeficient *V_estimate* již poté nemůže nabýt jiné hodnoty než 0. Situace je znázorněna na obrázku 8.1.



Obrázek 8.1: Výpočet nových koeficientů při střelbě na šerifa. Hodnoty V, B, R postupně reprezentují *V_estimate*, *B_estimate*, *R_estimate*. Číselné hodnoty nad šipkami vyjadřují jaké procento hodnoty koeficientu vlevo je převedeno do hodnoty koeficientu napravo.

Z obrázku 8.1 vyplývá, že vystřelí-li hráč na šerifa podruhé, budou přesuny hodnot z koeficientu V do koeficientů B a R nulové. Nastává pouze navýšení koeficientu B. Jakmile tedy odpadlík vystřelí do šerifa, šerif a pomocníci s ním začnou nakládat, jako by byl banditou. Důvodem je, že hodnota *B_estimate* je v průběhu hry chápána zejména jako reputace, což částečně definuje míru hrozby. Ten, kdo nejvíce uškodil šerifovi, je nejčastěji vybírán jako prioritní cíl. Odpadlík by při korektní hře měl předstírat, že je pomocníkem šerifa. Jakmile se rozhodne vystřelit do šerifa, vyzradí se a ztrácí toto krytí. Z tohoto důvodu a z definice jeho úkolu 4.1 vyplývá, že by se této akci měl vyvarovat v rané fázi hry.

Druhou situací, kdy dochází ke změně hodnot koeficientů, je střelba na hráče, který vystřelil na někoho, jehož identifikátor již je prvkem *List<int> AttackedSheriffList*. Tato situace je chápána jako snaha pomstít šerifa. Konkrétní výpočet nových hodnot koeficientů je znázorněn na obrázku 8.2.



Obrázek 8.2: Výpočet nových koeficientů při střelbě hráče, který dříve vystřelil na šerifa. Hodnoty V, B, R postupně reprezentují *V_estimate*, *B_estimate*, *R_estimate*. Číselné hodnoty nad šipkami, vyjadřují jaké procento hodnoty koeficientu vlevo je převedeno do hodnoty koeficientu napravo.

Zda-li je odpadlík neutrální, na straně šerifa nebo na straně banditů závisí na momentálním stavu hry. Pomáhá té straně, která má méně hrajících (neeliminovaných) hráčů. V případě, že již byl eliminován odpadlík, celá polovina hodnoty *B_estimate* se přesune do *V_estimate*. Pokud již byli eliminováni všichni pomocníci šerifa, celá polovina hodnoty *B_estimate* se přesune do *R_estimate* (situace 1b, 2b, 3b z obrázku 8.2). Pokud byli eliminováni jak všichni pomocníci šerifa, tak odpadlík, nedochází ke změnám hodnot při pomstě šerifa. Tato situace totiž implikuje, že jsou ve hře kromě šerifa už pouze bandité. Uvažována je však situace, kdy bandita eliminuje svého spoluhráče. Důvodem k takové akci je fakt, že za eliminaci bandity daný hráč jako odměnu dostává z dobíracího balíčku tři karty [43]. Pokud je tedy pravděpodobné, že šerif v následujícím tahu eliminuje banditu, za což by dostal tři karty, může být takticky výhodnější, aby jej ze hry vyřadil vlastní spoluhráč. S tím souvisí výpočet koeficientů označený na obrázku 8.2 jako 1b, 2b a 3c. Pokud hráč s nulovou hodnotou koeficientu V (tj. již vystřelil na šerifa) vystřelí na jiného, který již vystřelil na šerifa, polovina hodnoty *B_estimate* se přesouvá do *R_estimate*. Výjimkou, kdy ke změně koeficientů nedochází, je situace, jedná-li se o poslední život daného hráče. Bandita by za jiných okolností s ohledem na svůj úkol ve hře (viz. sekce 4.1) měl střílet na šerifa.

8.2 Tah inteligentního hráče

Tah je realizován funkcí *Play()*, jejíž základní struktura je stejná pro všechny čtyři role. Jedná se o sekvenci volaných funkcí, jejichž implementace se liší v rámci jednotlivých rolí. Každá z funkcí realizuje zahrání určité skupiny hracích karet. Část z nich je totožná pro všechny čtyři role. Implementace zbylých je specifická pro každou jednu z nich. Zkrácená sekvence volání vypadá následovně.

```
public void Play()
{
    // Joint Player class method (checks for Diligenza, Wells Fargo)
```

```
    p.CheckForDrawingCards();

    // Joint Player class method (checks for Emporio)
    if (p.CheckForEmporio())
    {
        // In case the AI actually plays Emporio. All players have to react
            one by one. This player's turn continues afterwards.
        return;
    }

    // Sheriff class method (checks for gun cards), no need for recursion
    HandleGuns();

    // Sheriff class method (checks for Panico!)
    if (CheckForPanic())
    {
        Play(id); // Recursion
        return;
    }

    ServerHandle.EndTurn(); // Sheriff ends his turn, game proceeds
}
```

Výpis 8.1: Pseudokód části tahu inteligentního hráče role šerif realizovaného funkcí *Play()*.

Každá z těchto funkcí nejprve zkontroluje, zda má inteligentní hráč některou z této množiny druhů karet v ruce. Pokud ne, vrací se návratová hodnota *false* datového typu *bool* a pokračuje se další funkcí. Pokud jsou některé z těchto karet k dispozici, v jejím těle dále probíhá proces rozhodování, zda ji inteligentní hráč zahraje. Nemusí existovat vhodný cíl nebo nemusí být její zahrání výhodné či bezpečné (například *Duel*, bude objasněno v podkapitole 8.3). Některé druhy karet jsou hrány vždy (například *Dostavník* nebo *Wells Fargo*). Jedná-li se o kartu, která je cílená na hráče (mimo sebe sama), probíhá proces rozhodování, zda ji zahrát, či nikoliv. Je-li výstupem rozhodovacího mechanismu verdikt kartu zahrát, je posléze funkce *Play()* rekurzivně volána znova. V opačném případě proběhne návrat z funkce s návratovou hodnotou *false* a postupuje se dále sekvencí volání v rámci funkce *Play()*.

V případě zahrání karty *Hokynářství* (v rámci funkce *CheckForEmporio()*) se dle pravidel z dobíracího balíčku otočí tolik karet, kolik je ve hře hráčů (mimo eliminované). Počínaje hráčem, který ji zahrál, všichni hráči (mimo eliminované) po směru hodinových ručiček volí jednu ze zbývajících karet, a tu si vezmou do ruky. Jakmile poslední hráč dostane kartu, dochází opět k volání funkce *Play()*. Důvodem je, že hráč, který kartu *Hokynářství* zahrál, mohl získat kartu *Dostavník* (originální název – Diligenza) nebo *Wells Fargo*.

8.3 Implementace dalších rozhodovacích mechanismů

V této podkapitole jsou vysvětleny vybrané mechanismy rozhodování při hraní určitých druhů karet a během tahu obecně. Tyto postupy jsou realizovány v příslušných funkcích, které jsou v průběhu tahu inteligentního hráče volány hlavní funkcí *Play()*.

Karta Duel

V rámci vybírání cíle karty *Duel* je procházen seznam hráčů, kteří nebyli eliminováni. Hráč, na nějž byl *Duel* zahrán, má na výběr ze dvou možností. Ubrat si život, nebo v duelu pokračovat odhozením karty *Bang!* Zvolí-li možnost druhou, původní hráč (autor *Duelu*) je postaven před stejnou volbu. Takto se střídají, dokud se jeden z nich nerozhodne nebo není donucen ubrat si život. Existuje tu tedy určité riziko že ten, kdo *Duel* zahrál, může ve svém tahu ztratit život. Proto hrát *Duel* například ve chvíli, kdy hráč (autor) má poslední život, není vždy rozumné. Z tohoto důvodu inteligentní hráč nehraje *Duel* vždy. Při rozhodování zohledňuje svůj počet životů, kolik karet *Bang!* má na ruce (tedy kolikrát může v duelu odpovědět) a kolik karet celkem má na ruce soupeř (počet karet *Bang!* je soukromá informace soupeře). Inteligentní hráč má odlišnou implementaci této funkce pro jednotlivé role a uvažuje i některé speciální situace. Nejběžněji však k zahrání karty *Duel* přistoupí ve chvíli, kdy je následující matematický výraz vyhodnocen jako pravdivý:

$$(2 \times AI\_PocetKaretBang + 1) \geqq (Souper\_PocetKaret + AI\_PocetChybejicichZivotu)$$

Karta Hokynářství

V případě volby z karet otočených po zahrání *Hokynářství*, se inteligentní hráč řídí několika jednoduchými pravidly. Uvažuje svoji momentální herní situaci a řídí se svým interním systémem ohodnocení cennosti druhů karet. Pokud je mezi možnostmi nejcennější karta dle tohoto systému – *Wells Fargo*, volí ji. Dále na horních místech tohoto pořadníku v rámci defenzivní hry dominují karty *Bang!* a *Vedle!*, a to pokud je inteligentní hráč nemá v ruce. Stejně tak je například karta *Pivo* volena jako jedna z nejhodnotnějších, pouze v případě, jsou-li ve hře více než dva hráči. V opačném případě totiž nemůže být zahrána a stává se bezcennou. Postup je totožný pro inteligentního hráče libovolné role.

Karta Cat Balou

Kartou *Cat Balou* je možné kterémukoliv hráči (bez ohledu na vzdálenost mezi hráči) odhodit libovolnou kartu ležící na stole před ním nebo náhodnou kartu z ruky. Implementace funkce vybírající cílovou kartu je odlišná pro každou jednu roli, jež může být inteligentnímu hráči přiřazena. V této podsekci bude vysvětleno, jak s kartou *Cat Balou* nakládá inteligentní hráč, jakožto pomocník šerifa.

Tuto kartu používá zejména za účelem útěku z dostřelu hráčů, jejichž *B_estimate* nebo *R_estimate* je vyšší než padesát. Při rozhodování tedy prochází *List* hráčů a zajímají jej ti, kteří dle heuristiky odhadu rolí jsou jeho nepřáteli. Jsou prohledávány před nimi vyložené karty s modrými okraji. Konkrétně je cíleno na karty zbraní (*Schofield*, *Remington*, *Rev. Carabine*, *Winchester*) a kartu *Hledí*, která taktéž zvyšuje dostřel. Pokud je nalezena některá z těchto karet, probíhá v rámci funkce *bool FleeFromRange(int opponentID, int viceID)* rozhodování, zda se inteligentnímu hráči vyplatí ji odhodit. Výhodu získává v případě, dostane-li se tím mimo dostřel vlastníka dané karty.

Inteligentní hráč v roli pomocníka volá funkci i ve prospěch šerifa, a to následovně: *bool FleeFromRange(int opponentID, int sheriffID)* s jeho identifikátorem – *sheriffID*. Je tedy zjišťováno, zda odstraněním karty zbraně či karty *Hledí* dostane šerifa mimo dostřel daného soupeře.

Funkce bývá volána jak pro šerifa, tak pro pomocníka samotného. Pořadí volání závisí na herní situaci. Pokud má šerif tři a méně životů, pomocník volá funkci nejprve s argumentem

sheriffID. Nedojde-li k zahrání karty *Cat Balou*, je funkce volána znova s identifikátorem samotného pomocníka. Nehrozí-li šerifovi bezprostřední nebezpečí eliminace, je pořadí volání opačné. Pokud během prvního volání není nalezena vhodná cílová karta, inteligentní hráč hledá, zda některý ze soupeřů nemá vyloženu kartu *Mustang*. V takovém případě se jedná o opačnou taktiku, tedy ofenzivní. Podobné postupy jsou použity i při hře karty *Panika!*

Odhazování přebývajících karet

Poslední fází tahu hráče dle pravidel [43], je odhazování přebytečných karet. Hráč totiž může ukončit tah maximálně s tolika kartami na ruce, kolik má momentálně životů. Před tím, než inteligentní hráč ukončí svůj tah voláním funkce *ServerHandle.EndTurn()* 8.1, je ve funkci *HandleRedundantCards()* odhozen požadovaný počet karet. Primárně si inteligentní hráč na ruce ponechává jednu kartu od každého druhu z následujícího výčtu právě v tomto pořadí:

- **Vedle!** – Pro případ odvrácení efektu karet *Bang!* nebo *Gatling*.
- **Bang!** – Pro případ odvrácení efektu karet *Indiáni!* nebo *Duel*.
- **Pivo** – Pokud jsou ve hře více než dva hráči.
- **Volcanic**

Karta *Pivo* je součástí tohoto seznamu, jelikož jejím odhozením je dle pravidel [43] možné odvrátit ztrátu posledního života. To však pouze v případě, že jsou ve hře více než dva hráči. Poslední v ruce prioritně ponechávanou kartou je zbraň *Volcanic*. Důvod bude objasněn v následující podsekci 8.3. Nedojde-li po projití nebo v průběhu procházení tohoto seznamu k naplnění maximálního povoleného počtu karet k ponechání, zbývající kapacitu inteligentní hráč doplní náhodnými kartami ze zbývajících. Toto chování je společné pro všechny role.

Výměna zbraní

Strategie hry karet zbraní se taktéž liší podle role přiřazené inteligentnímu hráči. Jako příklad bude popsána taktika, kterou používá, pokud mu byla přidělena role bandity. Pro banditu je s ohledem na jeho cíl ve hře (viz. sekce 4.1) klíčové míti v dosahu hráče, který je šerif. Klíčový význam má tedy odpověď na tuto otázku. Pokud v dosahu není, inteligentní hráč se podívá, zda v ruce nemá zbraň s větším dostřelem a tu případně zahraje. V opačném případě zbraně nemění (viz. podsekce 4.3.1). S ohledem na implementaci taktiky při odhazování přebytečných karet na konci tahu (viz. sekce 8.3) však zvažuje ještě jednu možnost. Má-li mezi kartami v ruce kartu *Volcanic*, zjišťuje, zda by se nevyplatilo jí nahradit zbraň vyloženou. Ačkoliv touto výměnou dojde ke snížení dostřelu, inteligentní hráč díky ní získává možnost během svých tahů zahrát neomezený počet karet *Bang!* Podle pravidel totiž každý hráč může během svého tahu vystřelit kartou *Bang!* pouze jednou. Pokud by šerif po této výměně měl být stále v dosahu a bandita měl na ruce například tři karty *Bang!*, má velikou příležitost šerifa eliminovat, případně výrazně oslabit.

Kapitola 9

Testování

Testování aplikace probíhalo pravidelně v průběhu jejího vývoje. Poznatky a připomínky účastníků značně ovlivnili finální podobu klientské i serverové aplikace. Největší přínos měly zejména v oblasti ovládání na klientské straně a na serverové straně v oblasti informování uživatele o průběhu hry. Tato kapitola se však zaměří na zhodnocení a závěrečné testování implementovaného inteligentního hráče.

Zhodnocení inteligentního hráče

Způsobů, jak inteligentního hráče vylepšit, se nabízí mnoho. Jednou z možností je zvolit odlišný charakter typu hráče. Zvolený přístup je v některých ohledech defenzivní. Je-li například zahrána karta *Indiáni!*, inteligentní hráč neuvažuje možnost, že by si ubral život, když má v ruce kartu *Bang!* V určitých situacích může být výhodné ubrat si život a kartou *Bang!* se ve svém tahu pokusit o eliminace soupeře. Inteligentní hráč však tuto možnost neuvažuje. Podobně v průběhu provádění efektu *Duel*.

Vzhledem ke zvolenému způsobu implementace je teoreticky možné, že by uživatel časem mohl odhalit, na základě čeho inteligentní hráč volí tahy. Vzhledem k přítomnosti čtyř druhů rolí je to však v krátkodobém horizontu značně nepravděpodobné. Důvodem je přítomnost různých způsobů rozhodování se, zda danou kartu zahrát v závislosti na přiřazené roli. Předejít by tomu bylo možné navýšením vlivu faktoru náhody. To by však mohlo být kontraproduktivní, jelikož by se objevovaly neoptimální tahy.

Experiment

Za účelem získání zpětné vazby na hru s inteligentním hráčem byl proveden experiment v podobě odehrání série her. Zvoleným formátem byla hra v pěti hráčích (čtyři reální a implementovaný inteligentní hráč). Důvodem je, že v případě hry ve vyšším počtu účastníků bývají hry dlouhé. Výhodou hry v pěti je navíc fakt, že odpadlík je lichým, neutrálním hráčem. Dobrovolníci, kteří se experimentu zúčastnili, nebyli začátečníky, ale hráči, kteří dobře znají výklad pravidel hry Bang! a odehráli desítky až stovky her. Hlavní cílem totiž bylo získat kvalitní, znalecké informace a postřehy. Účastníkům před ani v průběhu experimentu nebyly sděleny žádné informace týkající se implementace inteligentního hráče. Záznam odehraných her je zdokumentován v následující tabulce 9.1. V jednotlivých sloupcích je zaznamenáno, která role byla v rámci dané hry přidělena jednotlivým hráčům. Inteligentní hráč je v tabulce reprezentován identifikátorem *AI1*. Číselné hodnoty v závorkách znamenají, ve

kterém kole byl daný hráč eliminován. Pokud tento údaj chybí, znamená to, že hráč v dané hře eliminován nebyl. Vítězové jednotlivých her jsou zvýrazněni zeleným písmem.

	1. hra	**2. hra**	**3. hra**	**4. hra**	**5. hra**	**6. hra**	**7. hra**
Hráč1	**PŠ (15)**	**PŠ**	PŠ (6)	**B**	B (7)	PŠ	Š (15)
Hráč2	**Š**	B(3)	**B**	Š (8)	**O**	**B**	B (9)
Hráč3	B (9)	B(5)	Š (9)	PŠ	PŠ (10)	**B**	B (13)
Hráč4	B (12)	O(15)	O	**B (6)**	Š (21)	O	PŠ (13)
AI1	O (19)	**Š**	**B (8)**	O	B (12)	Š (5)	**O**

Tabulka 9.1: Záznam výsledků odehraných her v rámci experimentu.

Inteligentní hráč byl na straně vítězů třikrát z celkových sedmi odehraných her. To je možné považovat za plně uspokojivý výsledek, vzhledem k tomu, že dva ze čtyř dobrovolníků dokázali zvítězit pouze jednou. Výstupem tohoto experimentu však není vyhodnocovat úroveň hráčů. Pro podobné závěry by bylo nutné odehrát více her. Za zmínku však stojí úspěch inteligentního v podobě výhry v roli odpadlíka (7. hra). Tato role je obecně považována za nejnáročnější, což pramení z jejího herního cíle, viz. sekce 4.1. Navíc není možné stát se vítězem, je-li hráč v roli odpadlíka eliminován. Například ve třetí hře se inteligentní hráč stal vítězem, přestože byl v osmém kole eliminován. V roli šerifa má vlastní eliminace také za následek prohru, nicméně údělem odpadlíka je vyhnout se vlastní eliminaci a být tím, kdo na závěr porazí šerifa. Tento výsledek je tedy cenný.

Testující byli po skončení experimentu dotázání, zda odhalili některé principy fungování. Účastníci správně odhadli, že stane-li se inteligentní hráč cílem karty *Duel*, nesnaží se ostatní oklamat ubráním si života. V situaci, kdy se stane cílem karty *Duel*, totiž odhazuje karty *Bang!*, dokud v ruce alespoň jednu má. Ubere-li si tedy život, znamená to vždy, že již kartu *Bang!* nemá. Reaguje tedy jako méně zkušený hráč nebo jako hráč, který hraje defenzivně. Dalo by se zde uvažovat o důmyslnější implementaci chování například v situaci, kdy oponent má výrazně více karet v ruce. Odhadnout tedy, že *Duel* pravděpodobně v dalším kole efektu stejně prohraje. Testující v navazující diskuzi správně odtušili, že i v případě karty *Emporio* inteligentní hráč nepracuje s myšlenkou, kdy pro něj její zahrání nemusí být výhodné (situace z průběhu první hry). Pokud například zůstane ve hře se dvěma hráči, kteří jsou v daný moment jeho nepřáteli, důsledkem zahrání karty *Emporio* sice je, že získá jednu ze tří karet dle svého výběru, ale zároveň každý z jeho dvou soupeřů taktéž získá po jedné kartě. To opět vychází z povahy hry, kde výraznou roli hraje náhoda. Ani v tomto případě se však nejedná o tah, který by se dal klasifikovat jako vyloženě špatný. Tyto úvahy vedly k dotazům na situace, zda inteligentní hráč pracuje stejně, pokud sám vlastní kartu *Duel*. Testující po prozrazení detailů implementace ocenili postup popsaný v podsekci 8.3, tedy uvažovaní na základě navrženého vzorce.

Inteligentní hráč v průběhu her úspěšně odzbrojoval testující (odhazováním vyložených zbraní), čímž jim znemožňoval dostřelit na svoje hlavní cíle. Že se nejedná o náhodnou volbu karty, ale o výpočet založený na odhadu rolí, jim bylo prozrazeno také až po konci testování. Tento prvek byl hodnocen velice kladně, a bylo usouzeno, že se jedná o vydařeně zpracovaný postup. V průběhu hry si povšimli i toho, že inteligentní hráč má naprogramované chování pro výběr karet z voleb po zahraní karty *Emporio*, kde částečně preferuje nejcennější druhy karet (viz. podsekce 8.3).

Negativně bylo hodnoceno, že inteligentní hráč v případě hry v roli bandity brzy otevírá hru střelbou na šerifa. Testující argumentovali, že je v určitých situací výhodnější déle neprozrazovat svoji roli. V tomto případě je taktické vyčkat do momentu, kdy si bandité

vytvoří herní převahu. Tento vhodný okamžik však není snadné detekovat. Jakmile jeden z banditů zahájí ofenzivu, je v zájmu všech zbylých jej ofenzivou podpořit, protože ten kdo začal, se stává cílem šerifa a jeho spoluhráčů. Jeho nepodpořením snižují šanci na kolektivní vítězství. Na to však nemusí být připraveni v případě, že šerif není v dosahu pro zahrání karty *Bang!* Pomoci mohou jen kartami, které mají neomezený dosah – *Cat Balou*, *Gatling* a *Indiáni!* Tímto byl odhalen prostor pro možné zlepšení herního projevu.

Kapitola 10

Závěr

Cílem této práce bylo implementovat karetní hru pro více hráčů jako multiplatformní aplikaci s možností hry s inteligentním protihráčem. Byly představeny největší úspěchy ve vývoji umělé inteligence pro hry karetní, ale i hry žánrů ostatních. Byly popsány algoritmy, přístupy a metody za tímto účelem používané. Multiplatformní aplikace typu klient-server byla implementována v herních enginu *Unity*. Klientská aplikace byla testována na operačních systémech *Windows*, *macOS*, *Android* a *iOS*. Inteligentní hráč byl implementován jako *rule-based system*. Vytyčeným cílem v tomto směru bylo vytvořit umělou inteligence, s níž bude zábavné hrát. Nabídnout uživatelům při nedostatku účastníků možnost nahradit chybějícího člena inteligentním hráčem, který nebude zaostávat za běžným průměrným hráčem této karetní hry. Závěrečným testováním bylo naplnění tohoto cíle ověřeno.

Možností jak pokračovat v práci se nabízí celá řada. Tou nejprioritnější je momentálně kontaktování autorů karetní hry *Bang!*, jimž bych chtěl představit výsledek své práce a prodiskutovat možnou spolupráci v dalším vývoji. Serverová aplikace je navržena tak, aby bylo možné přidat existující herní rozšíření. Jejich implementace je krokem dalším. Co se týče implementovaného inteligentního hráče, nabízí se několik možných scénářů, jak pokračovat. Jeho závěrečné testování odhalilo několik možných způsobů, jak ještě více zlepšit jeho herní projev. Nepochybně by bylo možné dále zdokonalovat pravidla, kterými se řídí. Nabízí se však i myšlenka, zda se nepokusit aplikovat i některou z dalších představených metod ve snaze vytvořit silnější umělou inteligenci. Vzhledem k aktuálnosti tématu v kontextu *Imperfect information games* by mohlo býti zajímavé zaměřit se právě tímto směrem.

Literatura

[1] *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II* [https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii]. Accessed: 2021-04-20.

[2] *Chinook* [https://webdocs.cs.ualberta.ca/~chinook/project/]. Accessed: 2021-04-20.

[3] *Example Learning Environments* [online]. Accessed: 2021-03-28. Dostupné z: https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Learning-Environment-Examples.md.

[4] *Jeopardy! as a Modern Turing Test: Did Watson Really Win?* [https://thebestschools.org/magazine/watson-computer-plays-jeopardy/]. Accessed: 2021-04-20.

[5] *OpenAI Five* [https://openai.com/blog/openai-five/]. Accessed: 2021-04-20.

[6] *OpenAI Five Defeats Dota 2 World Champions* [https://openai.com/blog/openai-five-defeats-dota-2-world-champions/]. Accessed: 2021-04-20.

[7] *Order of execution for event functions* [online]. Accessed: 2021-03-22. Dostupné z: https://docs.unity3d.com/Manual/ExecutionOrder.html.

[8] *Unity's interface* [online]. Accessed: 2021-03-22. Dostupné z: https://docs.unity3d.com/Manual/UsingTheEditor.html.

[9] *Governor of Poker 3* [online]. 2016. V7.2.0. Dostupné z: https://store.steampowered.com/app/436150/Governor_of_Poker_3/.

[10] Allis, L. *Searching for solutions in games and artificial intelligence*. 1994. Disertační práce. Maastricht University. ISBN 9090074880.

[11] Anderson, E. F. *Playing smart - artificial intelligence in computer games*. 2003.

[12] Andrade, A. Game engines: a survey. *EAI Endorsed Transactions on Game-Based Learning*. Listopad 2015, sv. 2, s. 150615. DOI: 10.4108/eai.5-11-2015.150615. Dostupné z: https://www.researchgate.net/publication/283657797_Game_engines_a_survey.

[13] Beal, D. F. A generalised quiescence search algorithm. *Artificial Intelligence*. 1990, sv. 43, č. 1, s. 85 – 98. DOI: https://doi.org/10.1016/0004-3702(90)90072-8. ISSN

0004-3702. Dostupné z:
http://www.sciencedirect.com/science/article/pii/0004370290900728.

[14] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P. et al. Dota 2 with Large Scale Deep Reinforcement Learning. *ArXiv*. 2019, abs/1912.06680.

[15] Binmore, K. Playing for Real: Game Theory. In:. únor 2007, kap. 7. DOI: 10.1093/acprof:oso/9780195300574.001.0001. ISBN 9780195300574.

[16] Brown, N., Lerer, A., Gross, S. a Sandholm, T. Deep Counterfactual Regret Minimization. In: Chaudhuri, K. a Salakhutdinov, R., ed. *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 09–15 Jun 2019, sv. 97, s. 793–802. Proceedings of Machine Learning Research. Dostupné z: http://proceedings.mlr.press/v97/brown19b.html.

[17] Brown, N. a Sandholm, T. Libratus: The Superhuman AI for No-Limit Poker. In:. Srpen 2017, s. 5226–5228. DOI: 10.24963/ijcai.2017/772.

[18] Brown, N. a Sandholm, T. Superhuman AI for multiplayer poker. *Science*. Červenec 2019, sv. 365, s. eaay2400. DOI: 10.1126/science.aay2400.

[19] Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P. et al. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*. Březen 2012, 4:1, s. 1–43. DOI: 10.1109/TCIAIG.2012.2186810.

[20] Brunson, D. *Doyle Brunson's Super System*. 3. vyd. Cardoza Publishing, 2003. 331–340 s. ISBN 9781580420815.

[21] Carr, C. *Kraplow! Wild West Card Game* [online]. Květen 2012. Dostupné z: http://chriscarr.name:8080/westerncardgame/.

[22] Chaslot, G., Bakkes, S., Szita, I. a Spronck, P. Monte-Carlo Tree Search: A New Framework for Game AI. In:. Leden 2008, s. 216–218.

[23] Doyen, L., Raskin, J.-F. a Cachan, E. Games with Imperfect Information: Theory and Algorithms. *Lectures in Game Theory for Computer Scientists*. Leden 2011. DOI: 10.1017/CBO9780511973468.007.

[24] Duffy, J. Game Theory and Nash Equilibrium. 2015.

[25] Gilpin, A. a Sandholm, T. Lossless abstraction of imperfect information games. *Journal of the ACM (JACM)*. ACM New York, NY, USA. Říjen 2007, sv. 54, s. 25–es. DOI: 10.1145/1284320.1284324.

[26] Ginsberg, M. GIB: Imperfect Information in a Computationally Challenging Game. *Journal of Artificial Intelligence Research*. Červen 2011, sv. 14. DOI: 10.1613/jair.820.

[27] Hsu, F.-H. IBM's Deep Blue Chess grandmaster chips. *IEEE Micro*. 1999, sv. 19, č. 2, s. 70–81. DOI: 10.1109/40.755469.

[28] Kaelbling, L., Littman, M. a Moore, A. Reinforcement Learning: A Survey. *J. Artif. Intell. Res.* 1996, sv. 4, s. 237–285.

[29] Koçkesen, L. a Ok, E. A. An introduction to game theory. Červenec 2007, sv. 8. Accessed: 2-12-2020. Dostupné z: http://home.ku.edu.tr/~lkockesen/teaching/econ333/lectnotes/uggame.pdf.

[30] Koller, D. a Pfeffer, A. Generating and solving imperfect information games. In: Citeseer. *IJCAI*. 1995, s. 1185–1193.

[31] Levin, J. *Games of Incomplete Information*. http://web.stanford.edu/, únor 2002. Accessed: 2-12-2020. Dostupné z: http://web.stanford.edu/~jdlevin/Econ%20203/Bayesian.pdf.

[32] Lopes, M., Melo, F. S. a Montesano, L. Affordance-based imitation learning in robots. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, kap. III., s. 1015–1021. DOI: 10.1109/IROS.2007.4399517. Dostupné z: https://ieeexplore.ieee.org/abstract/document/4399517.

[33] Mandadi, S., Vijayakumar, S. a B., T. IMPLEMENTATION OF SEQUENTIAL AND PARALLEL ALPHA-BETA PRUNING ALGORITHM. *International Journal of Innovations in Engineering and Technology*. Srpen 2020, sv. 7, s. 98–104.

[34] Meduna, A. *Automata and Languages: Theory and Applications*. Springer Verlag: Springer Verlag, leden 2005.

[35] Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D. et al. DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker. *Science*. Leden 2017, sv. 356. DOI: 10.1126/science.aam6960.

[36] Neller, T. a Lanctot, M. An Introduction to Counterfactual Regret Minimization. In:. Palo Alto, California: AAAI Press, červenec 2013, sv. 3, s. 1602–1604. ISBN 9781577356158. Dostupné z: https://www.tib.eu/de/suchen/id/BLCP%3ACN087357685.

[37] Niklaus, J., Alberti, M., Pondenkandath, V., Ingold, R. a Liwicki, M. Survey of Artificial Intelligence for Card Games and Its Application to the Swiss Game Jass. Červen 2019.

[38] Prisner, E. *Game Theory Through Examples*. Leden 2014. ISBN Electronic ISBN: 9781614441151.

[39] Reyneri, L. An Introduction to Fuzzy State Automata. In:. červen 1997, sv. 1240, s. 273–283. DOI: 10.1007/BFb0032485. ISBN 978-3-540-63047-0.

[40] Robilliard, D., Fonlupt, C. a Teytaud, F. Monte-Carlo Tree Search for the Game of "7 Wonders". In:. Srpen 2014, s. 64–77. DOI: 10.1007/978-3-319-14923-3_5. ISBN 978-3-319-14922-6.

[41] Russell, S. J. a Norvig, P. *Artificial Intelligence (A Modern Approach)*. NJ: Prentice Hall, 1995.

[42] Schaeffer, J., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R. et al. Checkers Is Solved. *Science*. Říjen 2007, sv. 317, s. 1518–1522. DOI: 10.1126/science.1144079.

[43] Sciarra, E. *Bang - pravidla základní hry* [online]. Accessed: 2021-04-08. Dostupné z: https://eshop.albi.cz/data/files/products/24327/1603711687-bang-pravidla-zakladni-hry.pdf.

[44] Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L. et al. Mastering the game of Go with deep neural networks and tree search. *Nature*. Leden 2016, sv. 529, s. 484–489. DOI: 10.1038/nature16961. Dostupné z: https://www.researchgate.net/publication/292074166_Mastering_the_game_of_Go_with_deep_neural_networks_and_tree_search.

[45] Silver, D., Schrittwieser, J., Antonoglou, I., Huang, A., Guez, A. et al. Mastering the game of Go without human knowledge. *Nature*. Říjen 2017, sv. 550, s. 354–359. DOI: 10.1038/nature24270.

[46] Sutton, R. a Barto, A. Reinforcement learning. *Journal of Cognitive Neuroscience*. Leden 1999, sv. 11, s. 126–134.

[47] Sze, V., Chen, Y.-H., Yang, T.-J. a Emer, J. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE*. Březen 2017, sv. 105. DOI: 10.1109/JPROC.2017.2761740.

[48] Vinyals, O., Babuschkin, I., Czarnecki, W., Mathieu, M., Dudzik, A. et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*. 2019, s. 1–5.

[49] Ward, C. D. a Cowling, P. I. Monte Carlo search applied to card selection in Magic: The Gathering. In: *2009 IEEE Symposium on Computational Intelligence and Games*. 2009, s. 9–16. DOI: 10.1109/CIG.2009.5286501.

[50] Zinkevich, M., Johanson, M., Bowling, M. a Piccione, C. Regret Minimization in Games with Incomplete Information. In:. Leden 2007, sv. 2008.

[51] Świechowski, M., Godlewski, K., Sawicki, B. a Mańdziuk, J. Monte Carlo Tree Search: A Review on Recent Modifications and Applications. Březen 2021.