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

SBĚRATELSKÁ KARETNÍ HRA S UMĚLOU INTELIGENCÍ

TRADING CARD GAME WITH ARTIFICIAL INTELLIGENCE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JOSEF DOLEŽAL**
AUTHOR

VEDOUCÍ PRÁCE **TOMÁŠ MILET, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Doležal Josef, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Počítačová grafika a interakce
Název: **Sběratelská karetní hra s umělou inteligencí**
Traiding Card Game with AI
Kategorie: Počítačová grafika
Zadání:

1. Nastudujte techniky tvorby her v engine Unity. Nastudujte postupy tvorby umělé inteligence pro karetní hry.
2. Navrhněte hru s umělou inteligencí.
3. Naimplementujte navrženou hru.
4. Otestujte hru na uživatelích a vyhodnoťte obtížnost překonání umělé inteligence.
5. Vytvořte demonstrační video.

Literatura:

- Koster, Raph. *Theory of fun for game design*. O'Reilly Media, Inc., 2013.
- Schell, Jesse. *The Art of Game Design: A book of lenses*. CRC press, 2008.
- *Unity Learn*. Unity, https://learn.unity.com/.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2 a kostra aplikace.

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

Vedoucí práce: **Milet Tomáš, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Cílem práce je tvorba digitální verze sběratelské karetní hry *Flesh and Blood*. Součástí hry je umělá inteligence, která je schopna tuto hru hrát na začátečnické úrovni. Zásadním problémem jak při tvorbě této hry a její umělé inteligenci je fakt, že hra se neustále rozšiřuje o nové karty a pravidla. Práce popisuje jak je možné řešit vývoj v takto dynamickém prostředí. Ve hře je tento problém s rozšiřováním řešen pomocí zapouzdřených modulů jednotlivých herních prvků. Díky tomuto přístupu je zjednodušeno přidávání nových karet a případná úprava pravidel. Pro umělou inteligenci je možné řešení vhodné uzpůsobení vstupů. Umělá inteligence je realizována pomocí neuronové sítě a učena pomocí hry sama proti sobě, či ostatním sítím. Výstupem sítě je výběr vhodného tahu, který probíhá na základě informací o aktuálním stavu ve hře. Proto je důležité efektivní kódovaní aktuálního stavu hrací desky.

Abstract

The aim of this thesis is a digital version of a card game Flesh and Blood. The element of the game is artificial intelligence which can play the game at a beginner level. The essential issue with creating the game and its artificial intelligence is the fact that the game is constantly expanding with new cards and rules. The thesis describes how could this progress be solved in such dynamic enviroment. The issue with expansion of the game is solved by encapsulated modules for each game element. This approach simplifies the addition of new cards and possible modification of rules. A possible solution for artificial intelligence is suitable adaptation of inputs. Artificial intelligence is realized by means of a neural network and taught by self play or with other networks. The output of the network is the selection of a suitable move, which takes place on the basis of information about the current state of the game. Therefore, it is important to effectively encode the current state of the game board.

Klíčová slova

umělá inteligence, hra, karetní hra, neuronová síť, ML-Agents, Unity, Flesh and Blood

Keywords

artificial inteligence, game, card game, neural network, ML-Agents, Unity, Flesh and Blood

Citace

DOLEŽAL, Josef. *Sběratelská karetní hra s umělou inteligencí*. Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Tomáš Milet, Ph.D.

Sběratelská karetní hra s umělou inteligencí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doktora Tomáše Mileta. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Josef Doležal
15. května 2022

Poděkování

Chtěl bych poděkovat vedoucímu práce doktoru Tomáši Miletovi za vedení práce a cenné rady. Poděkování patří i členům komunity kolem hry a spolužákům za psychickou podporu při dokončování práce

Obsah

Kapitola 1

Úvod

Cílem této diplomové práce je vytvořit počítačovou hru v prostředí Unity, která bude obsahovat umělou inteligenci, schopnou hrát sběratelskou karetní hru *Flesh and Blood*. Sběratelské karetní hry jsou po celém světě velice rozšířené. Hráči si mají možnost kupovat *boostry* (balení několika náhodně vybraných karet rozdělených podle vzácnosti) a pomocí nich si rozšířit svoji kolekci. Chybějící karty je také možné měnit přímo mezi hráči. Z karet, které hráč vlastní, staví herní balíčky pomocí pravidel dané hry. S takto postaveným balíčkem může hráč hrát proti ostatním hráčům ať už pro zábavu v lokálních turnajích, nebo se zúčastnit celosvětových akcí. Mezi nejznámější sběratelskou karetní hru patří *Magic The Gathering*. Ta začala svoji historii psát už v roce 1993.[1] Následovalo plno dalších podobných her, ale pouze minimum se dokázala udržet mezi hráči déle než dva roky. *Flesh and Blood* se ve světě objevila těsně před začátkem pandemie a i přes to, že se jedná o sběratelskou karetní hru, pro kterou je klíčová hráčská komunita scházející se v místním obchůdku, stala se tato hra celosvětově rozšířená. Pro získání nových hráčů je ovšem důležité mít možnost si hru vyzkoušet a řešením v situaci pandemie je digitální podoba.

Příkladem může být již zmíněná nejznámější sběratelská karetní hra, která díky své digitální podobě formou *Magic Areny* získala mnoho nových hráčů. Případně čistě digitální sběratelská karetní hra *Heartstone*.

Hra by ovšem neměla sloužit jenom nově příchozím hráčům, ale i zkušenějším hráčům, kteří si chtějí postavit nový herní balíček a vyzkoušet jeho vlastnosti. Většina herních balíčků, které jsou na competativní úrovni, totiž obsahuje drahé karty. Cena těchto karet se může pohybovat v i tisících korun.

Další motivací bylo vyzkoušet si jak je složité vytvořit takovou umělou inteligenci, která by byla schopna hrát na slušné úrovni. Již zmíněná *Magic Arena* obsahuje bota, proti kterému je možné hrát. Tento bot je velice omezený tím, jaké balíčky může hrát a jeho rozhodování nedosahuje uspokojivé kvality ani pro začínajícího hráče.

Tato práce je mířena pro herní komunitu, aby sloužila jako nástroj pro zdokonalování se i jako možnost hru vyzkoušet. Řešení této práce nepředpokládá plnohodnotnou hru na úrovni aplikací pro digitální sběratelské karetní hry, ale vytvoření dobré základny. V rámci práce jsou vytvořená pravidla hry, spolu s herním balíčkem pro začátečníky, umělou inteligencí schopnou hrát tento balíček a rozhraním které dovoluje hru hrát. Do budoucna se počítá s rozšiřováním této aplikace.

Obrázek 1.1: Pravidelný Armory event pořádaný v brněnském Gaming and Social Hubu Black Oil. Události jsou pořádány alespoň jednou týdně s oficiální podporou LSS. Hráčská komunita se nejen v Brně, ale i po celé České i Slovenské republice rozrůstá. V Česku je k roku 2022 oficiálně 7 heren provozujících akce spojených s hrou *Flesh and Blood*.

Kapitola 2

Existující řešení

V této kapitole jsou představena existující řešení pro různé sběratelské karetní hry. Dále jsou vysvětleny základní znalosti z oblasti umělé inteligence a následně přehled s oblasti výzkumu karetních her.

2.1 Existující digitální verze her

V teto sekci je stručné představení různých nejznámějších digitálních verzí sběratelských karetních her. Je zde také uvedeno chovaní počítačem řízených protivnících, pokud takové v dané hře existují.

Flesh and Blood

Jelikož se jedná o cílovou hru je představení této hry obsaženo v samostatné kapitole. *Flesh and Blood* bohužel nemá žádnou oficiální digitální verzi hry. Neoficiální verze jsou jenom dvě.

První je možnost zahrát si hru pomocí *Tabletop simulátoru*. Ten ovšem slouží jako platforma pro různé deskové a karetní hry, proto se zde nenachází žádná umělá inteligence. Nejsou zde ani zakódovaná pravidla a hráči si musí vše hlídat sami.

Druhá možnost je aplikace *Felt Table*. Nespornou výhodou je její dostupnost, jelikož je dostupná pomocí webu[1]. Ta ovšem není přizpůsobena na mobilní zařízení. Z hlediska umělé inteligence se předpřipravené hratelné balíčky pro hrdiny z první sady chovají dobře. Chování ostatních balíčků je dosti předvídatelné a mají ještě dost prostoru k zlepšování. Při vydání nové sady je občas problém s přidáváním některých karet, kdy tyto karty jsou přidávány po částech s časovým zpožděním. Často se také objevují chyby v pravidlech. Z hlediska rozhraní pro hru se *Felt Table* velice zlepšuje a přehledně zobrazuje důležité informace uživateli.

[1] Odakz webové aplikace Felt table

Obrázek 2.1: Felt Table hra

Magic the Gathering

Hráč v této hře představuje bytost s magickou mocí (*Planeswalker*).Cílem hry je snížit protihráčovi životy na nulu. Toho lze docílit vyvoláváním kreatur za pomocí karet. Karty v této hře mají specifické vlastnosti a slouží jako zdroj, vyvolávání příšer, vyvolávání posílení či oslabení, nebo přímé sesílaní kouzel.

Magic the Gathering nabízí rovnou dvě možnosti jak si hru oficiálně zahrát v digitální podobě. Bohužel ani jedna verze neumí provázet papírovou verzi s digitální, co se týče vlastněných karet.

První možnost je starší hra *Magic Online*, ta ovšem slouží jako možnost jak hrát proti ostatním hráčům, takže neobsahuje žádnou umělou inteligenci proti které by hráč mohl hrát. Další nespornou nevýhodou je nutnost zaplatit poplatek, aby se otevřely všechny možnosti hraní této hry. Karty si v této hře je nutné samostatně dokoupit nebo pronajmout. Obrovskou výhodu je fakt, že obsahuje velké množství sad a je možné hrát všechny oficiální herní mody.

Druhá možnost je novější *Magic Arena*. Ta je plně přístupná zdarma a obsahuje i jednoduchou umělou inteligenci, proti které lze hrát. Samotná umělá inteligence ovládá pouze pět předpřipravených jednoduchých herních balíčků. Pokud si hráč vybere svůj postavený balíček, který obsahuje nějakou strategii, má umělá inteligence problém smysluplně reagovat. Hra ovšem obsahuje předscriptovaný tutoriál, který slouží jako způsob, jak si osvojit základy této hry. Karty lze získávat pomocí herní měny nebo si koupit doplňkové balíčky za reálné peníze. Nespornou výhodou je graficky upravené prostředí s mnoha animacemi. V roce 2022 byla představena i mobilní verze této aplikace. Nevýhodou je, že obsahuje pouze několik málo karet, které ve hře existují.

Obrázek 2.2: Magic Online hra

Heartstone

Cílem této hry je opět snížení protivníkových životů na nulu. Hráč představuje hrdinu ze světa Warcraftu a pomocí karet vyvolává příšery nebo různá posílení či oslabení. Karty v této hře neslouží jako zdroj pro samotné hraní útoků.

Tato hra má pouze digitální podobu a to pro počítače a mobilní zařízení. Výhodou čistě digitální verze je možnost implementace herních mechanik, které by byly v papírové verzi těžko proveditelné, jako je například vkládání karet do oponentova balíčku během hry. Z hlediska umělé inteligence hra nabízí specifické sólo kampaně, kde hráč s vlastním balíčkem bojuje proti hrdinům, kteří se ve hře nevyskytují. Chovaní takové umělé inteligence nelze porovnávat, neboť se zcela nepodobá klasické hře. Při takovémto módu hry může protihráč začínat už s několika vyvolanými příšerami nebo jsou pro něj některá pravila vynechána, jako je například placení ceny za vyvolání příšery. Chovaní samotné umělé inteligence co se týče strategie většinou nepracuje s aktuální situací na herní desce a nedokáže dostatečně detekovat hrozby.

Obrázek 2.3: Heartstone hra

2.2 Umělá inteligence

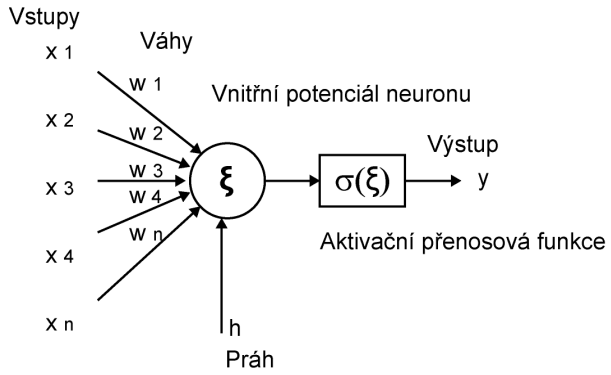
Umělá inteligence je schopnost libovolného digitálního zařízení provádět úkoly spojované s inteligentními bytostmi. Příkladem může být schopnost jednat, klasifikovat věci, či řešit úlohy na základě předchozích skutečností. Ačkoliv umělá inteligence nedokázala překonat tu lidskou co se týče všeobecného rozhodování, pro specificky zaměřené úlohy jako je třeba hraní her dokáží porazit i ty nejlepší profesionály.[2]

Dále je popsána základní architektura neuronové sítě a pojmy vztahující se k vypracování této práce.

Neuronové sítě

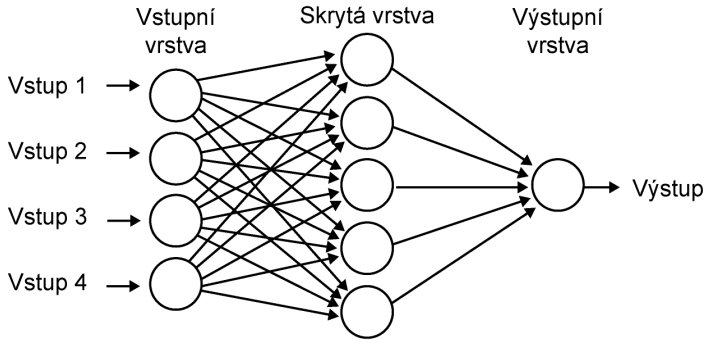
Neuron je základní prvek neuronových sítí. Grafické vyobrazení neuronu je na obrázku 2.4. Každý neuron má několik synapsí, které přenášejí vstupní informace do neuronu. Každá synapse v sobě také nese váhu, která určuje její sílu. Do neuronu ještě vstupuje konstantní hodnota zvaná práh. Práh zapříčiní posun výsledné funkce, čímž lze korigovat chování neuronové sítě.[5] Práh se většinou implementuje jako nultá synapse neuronu, kdy na vstupu je hodnota rovna 1 a hodnota váhy synapse je predikční chyba. V samotném těle neuronu se ze všech vstupů spočítá příslušná vážená suma dle synapsí. Výsledná hodnota se dále zpracuje aktivační přenosovou funkcí. Ta má za úkol upravit hodnotu příslušného intervalu a také zde slouží jako nelineární prvek. Výstup aktivační přenosové funkce je také výstup neuronu.

Neurony se poté skládají do jednotlivých vrstev, jak je vyobrazeno na obrázku 2.5. Jsou tři druhy vrstev a to vstupní, výstupní a skryté. Vstupní vrstva slouží jako vstup neuronové sítě, kdy každý neuron má přiřazen jeden vstup, který byl získán z pozorovaní. Tato vrstva musí být vždy přítomna a je pro celou neuronovou síť pouze jedna. Skrytých vrstev může být libovolný počet. Každá skrytá vrstva navazuje na vrstvu předchozí. Propojení jednotlivých vrstev závisí na architektuře samotné sítě. Výstupní vrstva se v neuronové síti nachází pouze



Obrázek 2.4: Anatomie neuronu

jedna. Je to poslední vrstva dané neuronové sítě. Na základě výstupů neuronů z této vrstvy se provádí rozhodovaní. Každá vrstva může obsahovat libovolný počet neuronů, ale vždy se v každé vrstvě musí nacházet nejméně jeden.



Obrázek 2.5: Neurony uspořádané do vrstev tvořící celou síť. V případě tohoto obrázku se jedná o plně propojenou neuronovou síť.

Proximal Policy Optimization

Proximal Policy Optimization(PPO) je rodina metod, založená na policy gradient metodách pro posilované učení, které se snaží optimalizovat policy na základě dlouhodobé kumulace odměn. PPO metody se snaží střídat mezi vzorkováním dat skrz interakci s prostředím a o optimalizací *"surrogate"* objektivní funkci s použitím stochastického gradientního vzestupu. Narozdíl od původních metod, které vykonají svoji aktualizaci vždy po jednom vzorku, mají tyto metody objektivní funkci, která dovoluje aktualizaci po více vzorcích.[7]

Self Play

Jedna z možných metod trénování neuronové sítě je hra sama proti sobě. Pří této metodě jsou proti sobě postavy dvě kopie stejného algoritmu, které proti sobě soupeří.[3] Každá takto vygenerovaná síť sdílí architekturu, ale liší se v nastavení vah. Sítě jsou ohodnoceny na základě definovaného ohodnocení jako například kladná hodnota za dosažení určitého cíle a naopak záporná za selhání.

2.3 Výzkum

Okolo výzkumu umělé inteligence, která by byla schopna hrát karetní hry, je mnoho článků. Výzkum se především vztahuje na klasické karetní hry jako je například Poker. Příkladem muže být práce [6], která se zabývá karetní hrou Batak. Jedná se o zdvihovou karetní hru, kdy hráči prvně sází kolik zdvihů jsou schopni vyhrát a následně se střídají vykládáním karet, kdy další hráč musí zahrát kartu vyšší hodnoty za dalších specifických pravidel. Po každém kole se boduje dle jejich sázek. Na konci hry vyhrává hráč s nejvíce body. Autoři článku zde zvolili přístup, kdy na každou ze dvou částí hry měli dvě odlišné neuronové sítě. První síť měla za úkol rozhodovat o sázkách a druhá síť ovládala vykládání karet. Sítě, které vykládaly karty, byly rozděleny dle jejich způsoby hry. Pro hru *Flesh and Blood* je možné toto aplikovat stylem, že každý hrdina je ovládán samostatnou neuronovou sítí, potažmo i každá varianta balíčku pro stejného hrdinu je reprezentována rozdílnou sítí.

Další práce [4] se zabývá hrou *Heartstone*. Autor zde uvádí trénovaní neuronové sítě a konvoluční neuronové sítě za cílem predikovat výhru dle stavu hry. K trénování je použita obrovská sada dat získána z mnoha her hráčů. Jsou zde uvedeny různé architektury sítí, způsob jakým lze pozorovat stav na herní desce a krátce o způsobu trénování. Z hlediska této práce je důležité především zpracování dat z pozorování a jaké údaje jsou použity k rozhodování.

Práce [8] rozebírá tvorbu umělé inteligence pro stejnou hru jako předešlá práce. Přístup je ovšem odlišný a to s tím, že se zde netrénuje za pomocí velkého množství dat ale přímo hrou. Jsou zde popsány případy pro hru proti různým hrdinům i s rozdílnými herními styly. Umělá inteligence je implementována pomocí adaptivní neuronové sítě s využitím fuzzy ART[2]. Tento přístup umožňuje využití predikce pro určení nejlepšího tahu. Pro tuto práci je zajímavý především způsob vnímaní prostředí. Pro budoucí práci by ovšem i takovýto přístup, s kterým se staví architektura sítě, byl zajímavý a využitelný.

Pro samotnou hru *Flesh and Blood* žádný výzkum neexistuje. Důvodem může být fakt, že neexistuje žádná oficiální platforma pro hraní této hry v digitální podobě. Dále se jedná o novou hru, která není tak rozšířena jako tomu je u hry *Heartstone* nebo *Magic the Gathering*. Svojí složitostí by se hra dala zařadit právě mezi již zmíněné dvě sběratelské karetní hry.

[2] Adaptive resonance theory

Kapitola 3

O Hře

Tato kapitola obsahuje informace o karetní hře *Flesh and Blood*, její základní prvky a pravidla.

Flesh and Blood je sběratelská karetní hra, která vznikla v roce 2019. Autorem hry je nezávislé studio Legends Story Studios. Přibližně každé tři měsíce přichází do hry nové sady, které přináší nové karty a případně rozšiřují hru o nové herní prvky. Ke konci roku 2021 hra obsahuje čtyři hlavní sady a jednu sadu rozšiřující.

Hra představuje souboj dvou hrdinů na život a na smrt. Hráč představuje jednoho z hrdinů z fantasy světa Rathe a za pomocí karet, které umisťuje na hrací desku, interaguje s protihráčem. Cílem hry je snížit počet protihráčových životů na nulu. Hráči za pomocí karet utočí na nepřítele, braní se útokům od nepřátel, používají je jako zdroj pro vykonávání akcí nebo získání posílení.

V následujících sekcích jsou přestaveny hrací karty a jejich anatomie a základní pravidla nutná k pochopení hry. Zdrojem obrázků karet je oficiální databáze karet, která je dostupná na oficiálních stránkách hry.

3.1 Pravidla hry

V této sekci jsou popsána základní pravidla hry, která se přímo vztahují k řešení této práce a měla by sloužit k pochopení hry jako celku. Podobně je to s popisem různých typů karet.

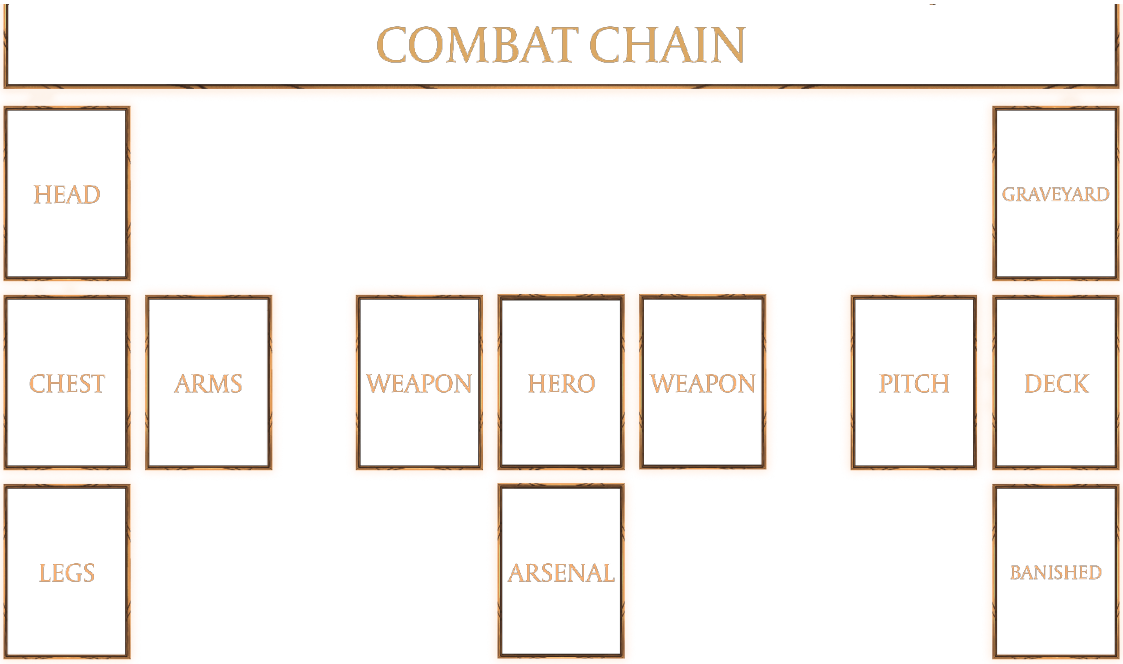
Popis těchto pravidel byl vytvořen na základě pravidel verze 1.4 [9].

Formát

Hra se dělí na dva formáty, které mají ještě další dělení. V rámci této práce je využit formát *Constructed*, kdy hráč má předem připravený balíček karet společně s hrdinou a vybavením. Tento formát se ještě dělí na další tři pod-formáty, které se od sebe liší pravidly pro stavbu balíčku a omezení výbavy. Pro začínající hráče, a tím i pro cvičení sítě bude rychlý pod-formát zvaný *Blitz* nejlepší. V tomto pod-formátu musí herní balíček obsahovat přesně čtyřicet karet, kdy se stejná karta může objevovat v balíčku pouze dvakrát, nespecifikují-li to pravidla karty jinak. Dále musí mít hráč jednu kartu hrdiny ve verzi *Young*, kdy se většinou tato verze od klasické liší nižším počtem životů. Poslední nedílnou součástí je až jedenáct karet vybavení ať už se jedná o zbraň nebo o zbroj.

Herní deska

Herní deska neboli aréna se skládá z několika zón. Oficiální rozložení těchto zón je vyobrazeno na obrázku 3.1. Toto rozložení ovšem není povinné a hráči si ho mohou upravit dle svých preferencí. Zóny slouží k usměrnění toku hry tím, že udávají rozmístění a pohyb karet v průběhu hry. Tyto zóny jsou také odkazovány v pravidlech herních karet.



Obrázek 3.1: Rozložení herní desky

Pokud není uvedeno jinak na základě pravidel, má každý hráč k dispozici každou zónu jedenkrát. Ve hře se nacházejí tyto zóny:

- **Hero:** Tato zóna může obsahovat pouze kartu hrdiny a karty, které se nacházejí v hrdinově duši
- **Weapon:** Tato zóna může obsahovat pouze karty zbraní nebo karty, které mají obsahující klíčové slovo *off-hand*. Počet těchto zón se odvyjí podle typu zbraně. Jestli se jedná o jednoruční zbraň, pak jsou tyto zóny dvě, pokud se jedná o obouruční zbraň je tato zóna pouze jedna.
- **Head:** Tato zóna může obsahovat pouze kartu vybavení s klíčovým slovem *Head*
- **Arms:** Tato zóna může obsahovat pouze kartu vybavení s klíčovým slovem *Arms*
- **Chest:** Tato zóna může obsahovat pouze kartu vybavení s klíčovým slovem *Chest*
- **Legs:** Tato zóna může obsahovat pouze kartu vybavení s klíčovým slovem *Legs*
- **Hand:** Tato zóna obsahuje karty, které má hráč v ruce. Pouze herní karty se mohou nacházet v této zóně.
- **Deck:** Místo kde se nachází hráčův balíček. Pouze herní karty se mohou nacházet v této zóně.

- **Graveyard:** V této zóně se nacházejí karty, které byly odstraněny ze hry a to zničením, zahozením, či po vyhodnocení *combat chainu* v případě, že se nejednalo o item nebo auru.
- **Banished:** V této zóně se nachází pouze karty, které se za pomocí nějakého efektu říká hráči, že má kartu do této zóny přesunout.
- **Pitch:** Tato zóna obsahuje pouze hrací karty, které byly hráčem přesunuty na tuto zónu z hráčovy ruky za účelem získání zdrojů k zaplacení ceny zahrané karty nebo aktivované akce. Tato akce se nazývá *Pitch*.
- **Arsenal:** Tato zóna obsahuje pouze hrací karty, většinou má hráč k dispozici jednu tuto zónu. Do prázdné zóny arzenálu lze lícem dolů vložit kartu na konci hráčova kola. Karty z arzenálu lze pouze hrát.
- **Combat chain:** Tato zóna slouží jako místo pro hraní karet. Zde také karty čekají na své vyhodnocení.

Začátek hry

Na začátku hry každý hráč umístí kartu svého hrdiny do *Hero* zóny lícem nahoru. Poté se náhodně rozhodne, kdo bude začínající hráč. Dále hráči vyberou z karet vybavení ty, které chtějí použít pro danou hru a umístí je do příslušné zóny lícem dolů. Následně hráči zamíchají svoje balíčky. Protože ve formátu *Blitz* hráč přichází s čtyřiceti herními kartami a herní balíček musí obsahovat přesně čtyřicet karet, není proto možné utvořit *sideboard*, který slouží pro odložení herních karet, které se nebudou v dané hře hrát. Po zamíchání hráči umístí balíček do *Deck* zóny s kartami lícem dolů. Následně hráči odhalí své karty vybavení a pokračují líznutím si karet dle hodnoty intelektu na kartě hrdiny. Hráč, který byl vybrán jako začínající, začne svoje kolo.

Průběh kola

Kolo se dělí na tři fáze, které za sebou následují v tomto pořadí:

- Začátek kola
- Akční fáze
- Konec kola

Ve fázi začátku kola se vyhodnotí všechny efekty, které se mají vyhodnotit na začátku kola, tedy na kartě je uvedeno *"at the start of turn"*. Dále se ukončí trvání všech efektů, které mají trvání do začátku kola. V této fázi nemá žádný hráč prioritu, proto není možné hrát karty, ani spouštět aktivní schopnosti.

V akční fázi hráč, který má status začínajícího hráče, získá prioritu a jeden akční bod. Akční bod je nutný k zahrání akce, čímž je míněno zahrání karty, nebo aktivace schopnosti. Některé karty dle dalších pravidel dovolují jejich zahrání i bez akčního bodu. Jedině hráč který má prioritu může provádět akce.

Pro zahrání akce je nutné ohlásit jakou akci hráč provádím tím, že danou kartu přesune na zónu *Combat chain*. Následně musí zaplatit za provedení akce příslušnou hodnotou bodů energie. Pokud hráč nemá žádné body energie a musí nějaké tyto body zaplatit, může

libovolnou kartu z ruky přesunou do *Pitch* zóny a tím získat příslušný počet bodů energie. Po zaplacení se akce stává zahranou a čeká na svoje vyhodnocení. Hráč tímto ztratí jeden akční bod. Následně má hráč možnost zahrát kartu typu *instant*. Posléze předá hráč prioritu protihráči, který může na danou akci odpovědět zahráním karty typu *instant* nebo předání priority zpět. Jakmile se podaří předat dvakrát prioritu mezi hráči aniž by se mezitím provedla akce, dojde k vyhodnocení karet v zóně *Combat chain* a navrácení priority začínajícímu hráči.

Další průběh je závislý na kartě, kterou začínající hráč zahrál. Pokud se jednalo o neútočnou akci, může začínající hráč zahrát další, pokud má příslušné zdroje, nebo předá prioritu protihráči, který může zahrát karty typu *instant*. Pokud se tak rozhodne následuje stejná procedura jako pro zahrání akce, kdy opět musí dojít k dvojí vyměně priority bez aktivace jakékoliv akce. Pokud protihráč nezahraje karty typu *instant* a opět předá prioritu, kolo se přesune do fáze konce kola.

Pokud se ovšem jednalo o útočnou kartu, protihráč má možnost umístit libovolný počet karet z ruky či příslušné zóny na zónu *Combat chain* a tím karty použít na obranu. Následně začínající hráč má možnost hrát útočné reakce nebo karty typu *instant*. Pokud hráč zahraje jednu z těchto akcí, automaticky tím předá prioritu protihráči, který má možnost zahrát obrané reakce nebo karty typu *instant*. Opět s provedením takovéto akce se předává priorita zpět začínajícímu hráči. Pokud hráč, který má prioritu nechce provést žádnou akci, předá prioritu svému protějšku. Pokud se povede dvakrát v řadě předat prioritu aniž by se provedla akce, přejde se k vyhodnocení všech karet na zóně *Combat chain*. Po vyhodnocení začínající hráč získá prioritu a hra pokračuje stejně jako po vyhodnocení neútočné akce.

Pokud se hra dostane do fáze konce kola, hráči ztratí prioritu. Poté se provedou všechny přetrvávající efekty, které mají v sobě uvedeno *"at the beginning of end phase"*. Následně jsou ukončeny všechny efekty, které mají trvání do začátku fáze konce kola. Následně všechny body energie i akční body, které hráči nestačili využít, se ztrácejí. Začínající hráč může poté umístit kartu z ruky lícem dolů do *Arsenal* zóny. Karty, které mají hráči ve své *Pitch* zóně, se umístí na spodek herního balíčku v libovolném pořadí. Začínající hráč si lízne takový počet karet, aby měl na ruce tolik karet, kolik je aktuální intelekt jeho hrdiny. Nakonec začínající hráč předá pozici začínajícího hráče svému protihráči. Při přechodu prvního kola do druhého si může hráč, který nebyl začínající, také líznout takový počet karet, aby měl na ruce stejný počet karet jako je aktuální intelekt jeho hrdiny. Pokud má hráč na ruce více nebo stejně karet jako je aktuální intelekt jeho hrdiny, tak si žádné karty nelíže a ani žádné nezahazuje. Hra dále pokračuje do dalšího kola.

Konec hry

Pokud v jakékoliv fázi hry nastane situace, že životy alespoň jednoho hrdiny klesnou na hodnotu nula nebo menší, hra tímto končí a vyhrává hráč, který má stále nějaké body života. Pokud nastane situace, kdy se oba hráči dostanou na nula a méně životů ve stejnou chvíli, je výsledek hry brán jako remíza. Při situaci, kdy hráč nemá žádnou kartu v balíčku, hra pokračuje dál a pouze v případě patové situace, kdy žádný z hráčů nemůže aktivovat žádnou schopnost či zahrát kartu, nastává remíza.

Vyhodnocení chainu

Po úspěšném dvojitém předání priority nastává samotné vyhodnocení. Ke kartám na *Combat chainu* se chováme, jako by se nacházely na zásobníku. Pro vyhodnocení karty postupujeme postupně od shora dolů a postupně vyhodnocujeme jejich efekty dle textu na kartě.

Text na kartě může obsahovat klíčová slova, která slouží ke zkrácení textu na kartě u efektů, které se objevují ve větší frekvenci, jako je například klíčové slovo *Go again*, nebo se jedná o mechaniku spojenou k určité třídě hrdinů, jako je například ochránce s klíčovým slovem *Crush*. Poté co se karta vyhodnotí celá, mají možnost hráči reagovat a hrát příslušné karty dle dané fáze hry. Vyhodnocení opět pokračuje při dalším dvojitém úspěšném předání priority. Jakmile je *Combat chain* prázdný, tak se uzavře a začínají hráč má možnost pokračovat ve svém kole.

3.2 Karty

Hlavním herním prvkem jsou hráčské karty. Ty představují způsob jakým je možno útočit na protihráče, bránit se nepřátelským útokům, ovlivňovat situaci na hrací desce a v neposlední řadě slouží i jako zdroj bodů energie pro provádění akcí. Karty mají svoji vzácnost, která je především důležitá ze sběratelského hlediska. Z herního hlediska je vzácnost důležitá v tom, že herní karty s označením *Common* a *Rare* mají tři různé verze, které se liší barevným pruhem, jak lze vidět na obrázku 3.2, a také jejich sílou. Karty s červeným pruhem mají nejsilnější efekt, ale dávají pouze jeden bod síly. Naopak modré mají nejslabší efekt, ale lze za ně získat tři body energie. Při stavění balíčku je důležité, že každá verze je odlišná kopie karty. To kupříkladu znamená, že pokud jsme limitováni maximálně dvěma kopiemi jedné karty, můžeme mít v balíčku dvě karty *Scar for a Scar* s červeným pruhem a dvě kary *Scar for a Scar* s modrým pruhem. Herní karty s vzácností *Super rare*, *Majestic* a *Legendary* mají pouze jednu verzi.

Obrázek 3.2: Barevné pruhy karet a jejich rozdíl síly. Karta s červeným pruhem poskytuje pouze jeden bod energie ale její útok má hodnotu 4. Karta s modrým pruhem poskytuje tři body energie, ale hodnota útoku je pouze 2.

Karty se dělí na několik druhů. Podle druhu se určuje, kde se daná karta může nacházet na hrací desce a jaké jsou možnosti pro její využití. Rozlišujeme kartu pro hrdinu, karty vybavení, tokeny a hrací karty. Karty vybavení dále dělíme na zbraně a brnění. Brnění se dělí na čtyři části a to hlava, ruce, hrudník a nohy, což odpovídá i zónám na herní desce. Hrací karty je označení pro karty, ze kterých se může sestavovat herní balíček. Hrací karty se pak dělí na typy, které určují jejich využití. K dispozici máme akční karty, útočné akční

karty, instanty, útočné reakce a obrané reakce. Dále následuje popis jednotlivých karet s příkladem takovéto karty.

Karta hrdiny

Karta hrdiny představuje samotného hrdinu, kterého představuje hráč. Tato karta má svoji specifickou zónu na hrací desce. Každý hráč může mít právě jednu tuto kartu ve hře. Většina hrdinů má ve hře dvě varianty a to mladou a dospělou, kdy tyto verze se převážné liší pouze menším počtem životů pro mladší verzi. V *construted* formátu se mladší verze používá v *Blitz* hře a dospělá v *Clasical constructed*.

Obrázek 3.3: Karta hrdiny - Bravo

1. Název karty - Jednoznačné odlišení hrdiny, dle jména lze odlišit normální a mladou verzi hrdiny.
2. Schopnost hrdiny - Aktivní nebo pasivní schopnost hrdiny. Zde se jedná o aktivní schopnost za jeden bod energie.
3. Intelekt - Udává počet karet, které se minimálně budou vyskytovat na konci hráčova kola na ruce.
4. Typ a podtyp karty - Klíčová slova označující typ a podtyp karty. Zde se jedná o hrdinu (*Hero*), kde jeho třída je ochránce (*Guardian*) a jedná se o mladou verzi hrdiny (*Young*).
5. Životy - Množství životů daného hrdiny.

Karty zbroje

Tyto karty představují zbroj, kterou si hrdina nese na sobě do bitvy. Každý typ zbroje má svoji zónu na hrací desce a v jeden okamžik může být maximálně jedna karta v dané zóně. Tyto karty mají dva účely a to bránění pomocí těchto karet a pomocné pasivní, či aktivní akce. Při použití karty zbroje se tyto karty vracejí zpět do příslušné zóny a nepřesunují se

na zónu *Graveyard*, jako je to v případě herních karet. Každá z těchto karet je opatřena alespoň jednou pasivní schopností, která modifikuje tuto kartu, pokud ji použijeme na obranu, a je vždy uvedena klíčovým slovem.

Obrázek 3.4: Karta zbroje - hrudník - Fyendal's Spring Tunic

1. Název karty - Jednoznačný identifikátor karty.
2. Schopnosti karty - Pravidla pro jednotlivé schopnosti karty. V tomto případě se jedná o dvě pasivní schopnosti a jednu aktivní, kdy první schopnost na začátku hráčova kola přidá na tuto kartu jeden *Counter*, pokud už na této kartě nejsou tři tyto *Countery*. Další je aktivní schopnost, kterou lze zahrát jako instant a při její aktivaci odstraníme tři *Countery* z této karty a tím získáme jeden bod energie. Poslední je pasivní schopnost, označená klíčovým slovem *Blade Break*. Jedná se právě o schopnost určující co se má stát, pokud tuto kartu použijeme na obranu. V tomto případě ji po uzavření *Combat chainu* zničíme.
3. Typ a podtyp karty - Klíčová slova označující typ a podtyp karty. Zde se jedná o generický typ, který může používat jakýkoliv hrdina. Dále že se jedná o zbroj, přesněji o hrudník.
4. Hodnota obrany - Udává hodnotu, která se odečte od soupeřova útoku, pokud budeme s touto kartou bránit.

Karty zbraní

Tyto karty představují způsob, jakým je možné opakovaně útočit na protihráče. Zbraně mají své místo ve *Weapon* zóně a pro jejich použití je potřeba provedení akce, kdy tuto kartu hráč přesune na *Combat chain* ve své akční fázi. Zbraně, stejně jako zbroj, se po vyhodnocení *Combat chainu* přesunou zpět do své původní zóny. Na rozdíl od zbroje tyto karty většinou nejsou opatřeny pasivní schopností, která by je limitovala v použití. Důležitým faktorem u zbraně je informace, zda se jedná o dvouruční zbraň (2H) nebo o jednoruční

zbraň (1H). Aktuálně hrdina může vlastnit pouze jednu dvojruční zbraň, nebo jednu či dvě jednoruční, případně s příchodem sady *Tales of Aria* může jednu jednoruční zbraň nahradit *off-hand* zbrojí.

Obrázek 3.5: Karta zbraně - Dvouruční meč - Dawnblade

1. Název karty - Jednoznačný identifikátor karty.
2. Schopnosti karty - Pravidla pro jednotlivé schopnosti karty. V tomto případě máme jednu aktivní schopnost, která říká že jednou za kolo můžeme za jeden bod energie s touto zbraní zaútočit. Dále jsou zde dva pasivní efekty. První udává že s touto zbraní způsobíme protihráči dvakrát za kolo poškození, dáme na tuto zbraň counte,r který zvýší poškození zbraně o jedna více.[1] Druhý efekt je, že pokud s touto zbraní nezpůsobíme žádné poškození ve svém kole, tak z této zbraně odstraníme všechny country, které zvyšuji poškození zbraně o jedna více.
3. Hodnota útoku - Hodnota poškození, které hráč způsobí, pokud s touto zbraní provede akci útoku.
4. Typ a podtyp karty - Klíčová slova označující typ a podtyp karty. Zde se jedná o zbraň pro bojovníka, kde typ zbraně je meč a jedná se o dvouruční zbraň.

Akční karty

Akční karty jsou první typ herních karet, které mohou být součástí herního balíčku. Tyto karty vetšinou přináší efekty, které ovlivňují další akce hráče, nebo omezují možnosti protihráče.

1. Název karty - Jednoznačný identifikátor karty

[1] Tato zbraň se dobře páruje s hrdinou Dorinthea, která dovoluje jednou za kolo použít znova zbraň pokud tato zbraň způsobila protihráči poškození

Obrázek 3.6: Akční karta - Forged for War

2. Pitch hodnota - Hodnota, která udává počet bodů energie, kterou dostaneme, když tuto kartu zahrajeme jako *pitch* akci. V tomto případě získáme 2 body energie.

3. Cena - Hodnota, která udává počet bodů energie nutný k zaplacení, abychom mohli tuto kartu zahrát.

4. Pravidla karty - Efekty a doplňující pravidla pro danou kartu. V tomto případě první efekt je *Go again*, který přidá hráči jeden akční bod. Dále dokud je karta ve hře zbroj, kterou vlastníme, získá plus jedna k hodnotě obrany. Poslední efekt říká, že na začátku další hráčovi akční fáze zničíme tuto kartu.

5. Typ a podtyp karty - Klíčová slova označující typ a podtyp karty. Zde se jedná o akční kartu, kterou může používat pouze ochránce. Dále je to Aura což je jeden z typů permanentu, který se po vyhodnocení *Combat chainu* nepřesouvá do zóny *Graveyard*.

6. Hodnota obrany - Udává hodnotu, která se odečte od soupeřova útoku, pokud budeme s touto kartou bránit.

Útočné akční karty

Útočné akční karty jsou dalším typem herních karet, na rozdíl od akčních karet slouží podobně jako zbraně k způsobování poškození protihráči.

1. Název karty - Jednoznačný identifikátor karty

2. Pitch hodnota - Hodnota, která udává počet bodů energie kterou dostaneme, když tuto kartu zahrajeme jako *pitch* akci. V tomto případě získáme 1 body energie.

3. Cena - Hodnota, která udává počet bodů energie nutný k zaplacení, abychom mohli tuto kartu zahrát.

4. Pravidla karty - Efekty a doplňující pravidla pro danou kartu. V tomto případě je zde pouze efekt *Go again*, který přidá hráči jeden akční bod.

Obrázek 3.7: Útočná akční karta - Surging Strike

5. Hodnota útoku - Hodnota poškození, které hráč způsobí pokud tuto kartu zahraje.
6. Typ a podtyp karty - Klíčová slova označující typ a podtyp karty. Zde se jedna o útočnou akční kartu, kterou může mít v balíčku pouze hrdina s třídou ninja.
7. Hodnota obrany - Udává hodnotu, která se odečte od soupeřova útoku, pokud budeme s touto kartou bránit.

Instanty

Instanty jsou speciální druh hracích karet, neboť stejně jako instantní akce je možné tyto karty hrát kdykoliv, kdy má hráč prioritu a na jejich zahrání není potřeba akčního bodu. Tyto karty nabízí ohromnou sílu ohledně vlastních efektů, avšak většinou nejsou ovlivněny efekty akčních karet, či efekty permanentů. Další výraznou nevýhodou je absence obrané hodnoty. Četnost těchto karet je také velice malá.

Obrázek 3.8: Instant karta - Sigil of Solace

1. Název karty - Jednoznačný identifikátor karty
2. Pitch hodnota - Hodnota, která udává počet bodů energie kterou dostaneme, když tuto kartu zahrajeme jako *pitch* akci. V tomto případě získáme 1 body energie.
3. Cena - Hodnota, která udává počet bodů energie nutný k zaplacení, abychom mohli tuto kartu zahrát.
4. Pravidla karty - Efekty a doplňující pravidla pro danou kartu. V tomto případě hráč získá tři body zdraví navíc.
5. Typ a podtyp karty - Klíčová slova označující typ a podtyp karty. Zde se jedna o generický instant, který může používat jakákoliv třída hrdinů.

Útočné reakce

Útočné reakce jsou hrací karty, které hráč hraje během reakční fáze ve svém kole. Za pomocí efektů z těchto karet může přidávat efekty útočné akční kartě či zbrani, nebo zvyšovat jejich útočnou sílu. Zahrání tohoto typu karty nevyžaduje akční bod.

Obrázek 3.9: Útočná reakce - Singing Steelblade

1. Název karty - Jednoznačný identifikátor karty
2. Pitch hodnota - Hodnota, která udává počet bodů energie kterou dostaneme, když tuto kartu zahrajeme jako *pitch* akci. V tomto případě získáme 2 body energie.
3. Cena - Hodnota, která udává počet bodů energie nutný k zaplacení, abychom mohli tuto kartu zahrát.
4. Pravidla karty - Efekty a doplňující pravidla pro danou kartu. V tomto případě je zde doplňující pravidlo pro specializaci a to takové, že tuto kartu může mít v balíčku pouze hrdina Dorinthea. Dále je zde efekt, který zvýší hodnotu útoku zbraně o jedna. A poslední je klíčové slovo *Reprise*, které říká pokud protihráč použil jakoukoliv kartu z ruky v rámci aktuálního *Combat chainu* spustí se další efekt. Pro tuto kartu je tento efekt, že si hráč může ve svém balíčku najít jakoukoliv útočnou reakci a tu dát do

Banish zóny lícem nahoru. Takto nalezenou kartu může během tohoto *Combat chainu* zahrát.

5. Typ a podtyp karty - Klíčová slova označující typ a podtyp karty. Zde se jedná o válečníkovu útočnou reakci.
6. Hodnota obrany - Udává hodnotu, která se odečte od soupeřova útoku, pokud budeme s touto kartou bránit.

Obrané reakce

Obrané reakce jsou hrací karty, které hráč hraje během reakční fáze v protihráčově kole. Tyto karty přidávají svou obranou hodnotu k součtu obraných hodnot karet, které byly zahrány před reakční fází. Tento typ karet nevyžaduje akční bod k jejich zahrání. Tyto karty lze použít pouze na obranu během reakční fáze.

Obrázek 3.10: Obraná reakce - Flic Flak

1. Název karty - Jednoznačný identifikátor karty
2. Pitch hodnota - Hodnota, která udává počet bodů energie kterou dostaneme, když tuto kartu zahrajeme jako *pitch* akci. V tomto případě získáme 1 body energie.
3. Cena - Hodnota, která udává počet bodů energie nutný k zaplacení, abychom mohli tuto kartu zahrát.
4. Pravidla karty - Efekty a doplňující pravidla pro danou kartu. Zde při zahrání této karty bude mít další karta, která obsahuje klíčové slovo *combo*, navýšenou hodnotu obrany o dvojku.
5. Doprovodný text - Text psaný kurzívou je na kartách jako doprovodný text, který v rámci hry nemá žádný efekt. Sám o sobě dodává informaci k příběhům ze světa Rathe.
6. Typ a podtyp karty - Klíčová slova označující typ a podtyp karty. Zde se jedná o obranou reakci, kterou může zahrát pouze hrdina s třídou ninja.
7. Hodnota obrany - Udává hodnotu, která se přidá k součtu obraných karet.

Tokeny

Jedná se o speciální druh karet, které nejsou součástí herního balíčku a ani nemají žádnou dedikovanou zónu ve které se nachází. U těchto karet se jedná o permanenty, které jsou vytvořeny díky efektu jiné karty. Tokeny, které mají být zničeny, jsou místo do zóny *Graveyard* odloženy mimo hrací desku.

Obrázek 3.11: Token - Quicken

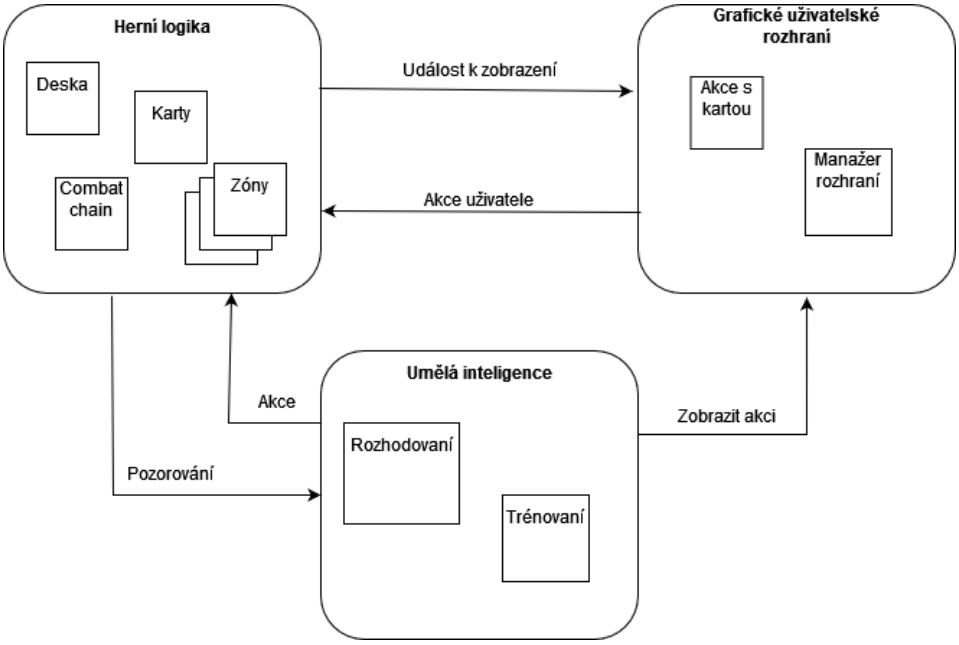
1. Název karty - Jednoznačný identifikátor karty
2. Pravidla karty - Efekty a doplňující pravidla pro danou kartu. Zde je v závorce připomínající pravidlo pro Auru. Dále je zde efekt, který se spustí při útoku s útočnou akční kartou nebo s útokem se zbraní a tento efekt říká, že dané útočící kartě přidá klíčové slovo *go again*[2]. Při spuštění se tento token zničí.
3. Typ a podtyp karty - Klíčová slova označující typ a podtyp karty. Zde se jedná o generický token, který může vytvořit jakákoliv třída hrdinů. Podtyp tohoto tokenu je Aura.

[2]Pokud karta obsahuje více instancí klíčového slova go again, jsou všechny více násobné instance tohoto slova ignorovány

Kapitola 4

Návrh

Prvně bylo nutné provést návrh implementace samotné hry, kdy se musel klást důraz na fakt, že druhého hráče bude ovládat umělá inteligence. Samotná umělá inteligence potřebuje mít přehled o aktuálním stavu hry a to nejlépe na jednom místě. Důvodem je zajištění rychlosti řešení, neboť aktuální stav hry bude sloužit jako vstup pro neuronovou síť, která bude rozhodovat jaké akce se mají provést. Dalším důležitým aspektem je striktní oddělení grafické části od logické. Pro trénování neuronové sítě není potřeba grafického výstupu, ba naopak je v celku nežádoucí, neboť by výrazně zpomaloval samotné trénování kvůli zpracovávání signálů pro grafické uživatelské rozhraní. Na obrázku 4.1 je zobrazené schéma rozdělení jednotlivých částí hry a jejich vzájemná komunikace.



Obrázek 4.1: Schéma hry. Hra se dělí na tři velké celky. Každý celek obsahuje různé podčásti, které jsou implementované samostatně v rámci celku, čímž se zajistí modularita

Další faktor, který je nutné brát v úvahu, je fakt, že se hra neustále rozšiřuje a proto musí být jednoduché přidávat nové karty do hry. Při přidávání nových karet by se nemělo nejlépe vůbec zasahovat do stávajících zdrojových kódů hry. Toto se ovšem stává nemožné

pokud se ve hře objeví nová mechanika či upraví stávající pravidla, ale při dobře sestavené implementaci se dají zásahy co nejvíce eliminovat.

Poslední důležitá věc pro počítačové hry obecně je vizuální stránka a její ovládání, kdy je nejdůležitější důkladně navrhnout grafické uživatelské rozhraní a to následně i řádně otestovat.

4.1 Návrh logické části

Pro přehlednost i dobrou modularitu je dobré rozdělit jednotlivé prvky hry na samostatné logické části. Ty spolu vzájemně spolupracují přes jedno rozhraní, které také využívá umělá inteligence pro získávání informací o aktuálním stavu hry. Logické bloky jsou rozděleny dle zón, které se vyskytují ve hře dle pravidel. Karty jsou další důležitou součásti, neboť jejich manipulace mění stav hry.

Dále jsou popsány jednotlivé logické bloky, ze kterých se skládá hra.

Herní deska

Při fyzické hře je herní deska místo, kde se nachází nejvíce informací o aktuálním stavu hry a proto je taky vhodné tento blok využít jako rozhraní pro komunikaci ostatních herních bloků. Proto se zde nachází všechny zásadní informace o stavu hry, které slouží jako vstup neuronové sítě pro rozhodovaní jaká akce se má zahrát. Dále tento blok řídí přechody mezi jednotlivými fázemi hry dle pravidel, ale i samotnou inicializaci hry, jako je například výběr začínajícího hráče, tak i samotný konec hry.

Combat chain

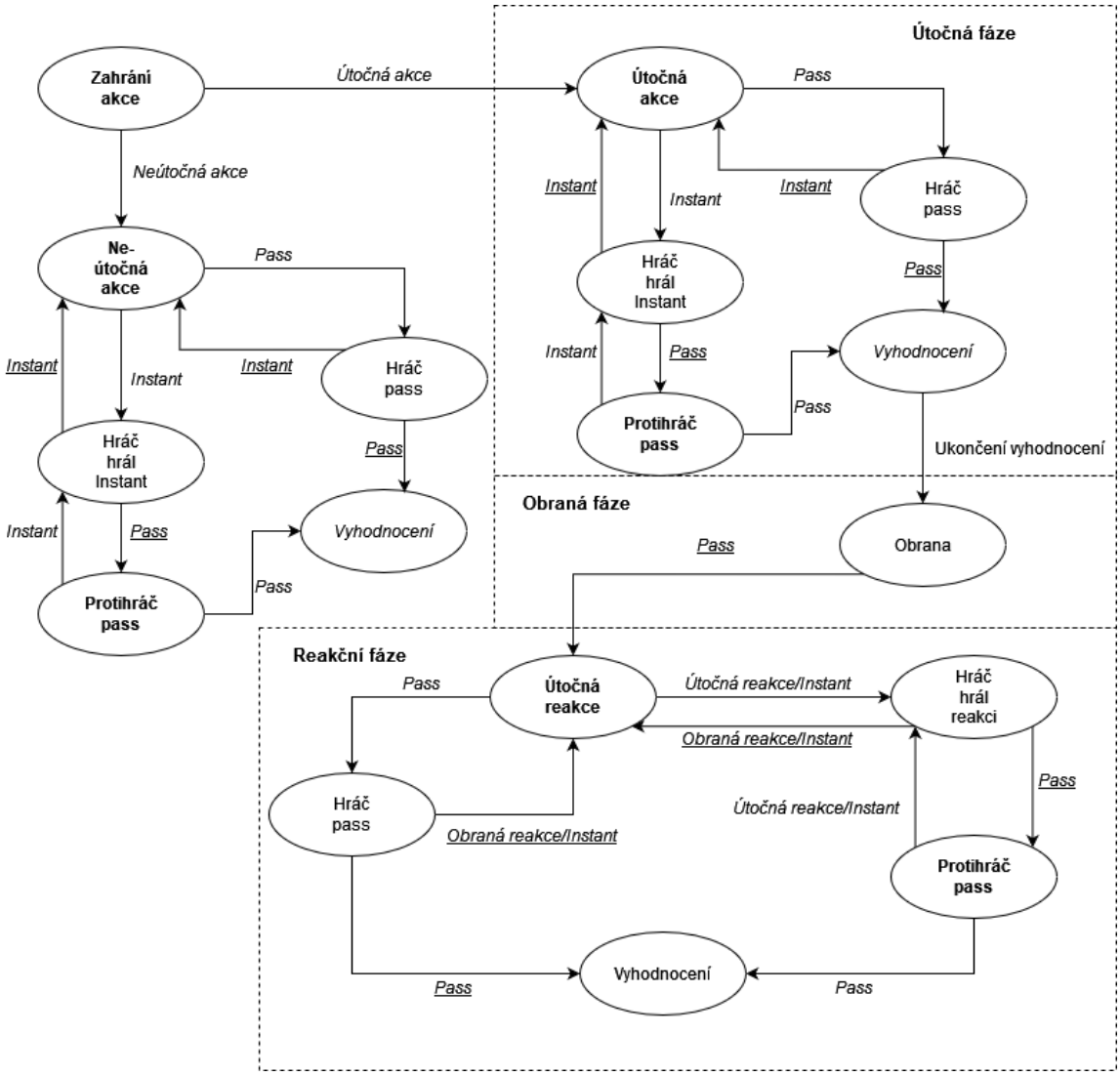
Tento blok je důležitý pro řízení boje hráčů, který odpovídá herním pravidlům. Dění v tomto bloku hry udává průběh akční fáze hry. Tento blok odpovídá téměř pravidlům samostatné hry. Diagram zobrazující chovaní *combat chainu* je zobrazen na obrázku 4.2

Pro zjednodušení a se zkušeností s touto hrou jsou vyhodnocování *combat chainu* upravena pravidla, kdy se nevyhodnocuje pouze vrchní karta, ale všechny karty, které se nachází na *combat chainu*. Při reálné hře se hráč nesetkává s přidáváním dalších karet během vyhodnocování a z hlediska hratelnosti by hráči museli s každou kartou provádět dvojité předání priority, aby bylo možné pokračovat ve vyhodnocování. Pokud by se ve hře objevila mechanika nebo karta, která by byla zvýhodněná tímto krokem, musela by se provést úprava tohoto bloku.

Karty

Vlastní logický blok tvoří karty. Většina karet má totiž nějaký efekt který přímo ovlivňuje hru. Například velice častý efekt *Go Again*, který přidá hráči jeden akční bod, čímž dovoluje hráči hrát více akcí v jednom kole. Z anatomie jednotlivých druhů karet jsou jasné některé atributy, které jsou stejné pro všechny karty. Poté jsou atributy, které jsou specifické pro herní karty nebo jenom pro vybavení.

Při vyhodnocování karty se prochází text karty od shora dolů a je zde také možnost přidávání dalších dodatečných pravidel. Z toho důvodu odstavec textu na pravidlech karty je brán jako jeden efekt. Poté lze uvažovat, že pravidla na kartě jsou složená z několika různých efektů. Proto jsou efekty programovány samostatně a karty pouze obsahují informaci o tom jaké efekty a v jakém pořadí obsahují. Je nutné správné označení těchto efektů, neboť ne



Obrázek 4.2: Diagram chování *Combat chain* pro provedení jedné akce. Tučné popisky bloků udávají úsek kdy má prioritu hráč, který je na tahu. Popisky bez úpravy jsou úseky, kdy má protihráč prioritu a kurzívou jsou části, kdy žádný hráč nemá prioritu. Podtržené popisky u přechodů znamenají akci protihráče, bez podtržení pak akci hráče. Popisek u přechodu bez úpravy je automatiky určen hrou.

všechny efekty jsou definovány přímo pravidly nějakým klíčkovým slovem jako je tomu v případě *go again*. S přidáním efektů se také zjednoduší práce, kdy některé takovéto efekty mohou přetrvávat i po vyhodnocení karty, nebo nemají okamžitý efekt.

Zóny pro vybavení

Zóny pro vybavení slouží k uložení příslušné karty vybavení dle pravidel. Dále implementují akce jako je útok se zbraní, bránění se zbrojí a aktivace schopností zbroje. Tyto zóny mají také za úkol držet a manipulovat s daty, které obsahují karty vybavení.

Herní balíček

Herní balíček se skládá z čtyřiceti herních karet a je postavený dle pravidel hry. Jeho místo je v zóně *deck*, která se stará o manipulaci s tímto balíčkem, jako je například lízání karet, či míchání balíčku.

Hráčova ruka

Tento blok se stará o manipulaci s kartami, které se nacházejí v hráčově ruce. Hlavním cílem tohoto bloku je zprostředkovat zahrání karty a kontrolovat zda zvolená akce je platná. Příkladem může být situace, kdy hráč přesunul kartu na *combat chain*, ale za tuto kartu ještě nezaplatil. Tento blok zajistí, že žádná další karta nebude zahrána na *combat chain* dokud se nezíská dostatečný počet energie. Toto rozhodnutí bylo provedeno z důvodu, že ve většině případů se energie získá přesunutím nějaké karty z ruky na zónu *pitch*.

Pitch zóna

Tato zóna tvoří blok, který musí být schopný získávat energii z karet, které byli přesunuty na tuto zónu. Dále se musí postarat o přesunutí karet na spodek balíčku na konci kola, kdy se *pitch* zóna musí vyčistit.

Graveyard a Banish zóna

Tyto bloky slouží k odkládání karet po zahrání, nebo pokud jsou karty nuceny k přesunu na jednu z těchto zón. Hráči mají možnost získání informací o tom, jaké karty se zde nacházejí.

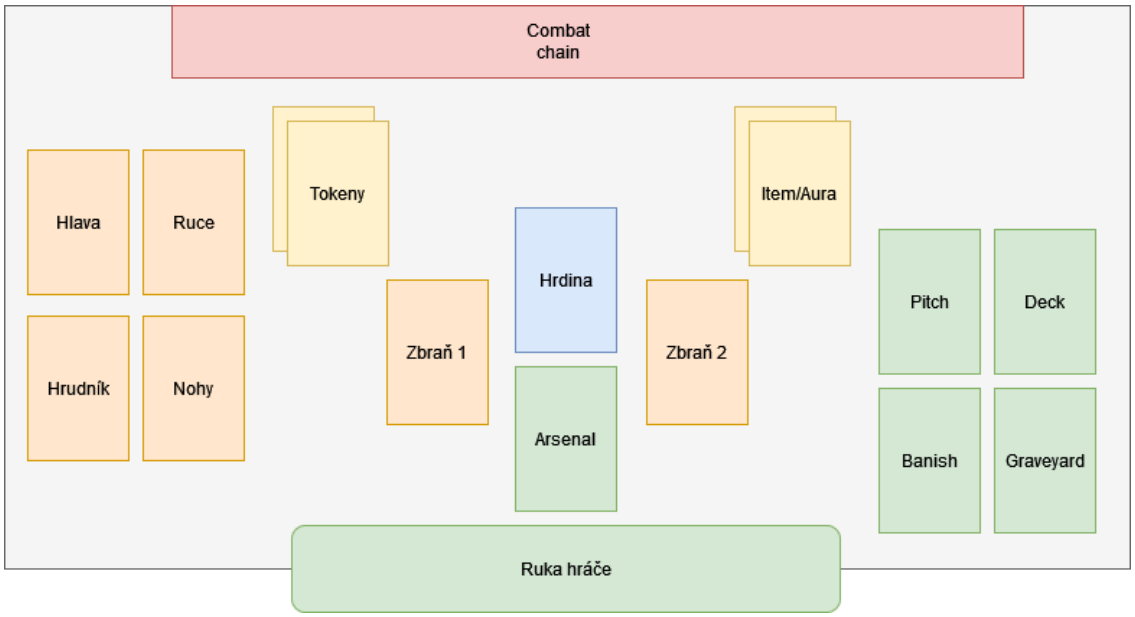
Arzenal

Tato zóna přebírá podobné chování jako herní ruka s tím rozdílem, že karty z této zóny nelze použít k blokování a k placení zahraných karet. To aby byla karta zahrána z této zóny ve správné fázi si musí také ohlídat sama.

4.2 Návrh grafické části

Grafická část slouží jako rozhraní přímo pro hráče. Pomocí přesouvání karet na příslušné zóny hráč ovládá samotnou hru. Také se mu zde zobrazují informace o aktuálním stavu hry, jako je například zahrané karty protihráčem či v jaké fázi se hra nachází.

Hra dle pravidel [9] nemá předepsané žádné pevné rozložení herní desky. Je zde pouze doporučení, které udává rozestavení zón. Se stejným rozložením jsou poskytovány hráčům podložky, které mají předtištěné zóny. Bohužel toto doporučené rozložení je dosti vertikálně rozložené, jak je zobrazeno na obrázku 3.1. Pokud vezmeme hru dvou hráčů, je takovéto rozložení nevhodné, neboť monitory mají spíše horizontální rozložení obrazu. Ideální rozložení pro jednoho hráče je možné vidět na obrázku 4.3.



Obrázek 4.3: Rozvržení poloviny hrací desky pro jednoho hráče. Zóny jsou jednotlivě rozděleny dle barev. Zeleně místa pro herní karty, oranžová pro karty vybavení, modrá pro kartu hrdiny, žlutá pro permanenty a červená pro combat chain. Combat chain je pro oba hráče sdílený.

Další problém nastává se samotnými kartami. Na jednu polovinu hrací desky se musí vejít až čtyři karty nad sebou. Proto je nutné tvar karty modifikovat, aby bylo možné požadovaným způsobem karty umístit, ale zároveň aby nedošlo ke špatnému pochopení karty nebo ztrátě informací. Možným řešením je odstranění textového pole karty. Odstranění uměleckého obrázku karty by způsobilo, že by hra vypadala velice suše. Umělecký obrázek často složí i jako rychlý způsob pro identifikaci karty zkušenými hráči. Výsledná úprava je možná vidět na obrázku 4.4. Absence textového pole s pravidly je kompenzována místem na obrazovce, kdy při najetí na miniaturu karty se na daném místě zobrazí karta v plném rozlišení.

Obrázek 4.4: Upravená karta oproti originální kartě

4.3 Návrh umělé inteligence

Umělou inteligenci lze realizovat několika způsoby. První je implementace podle přesně daných pravidel chování. Tento přístup je nejjednodušší avšak pro tento případ zcela nevhodný. Chování takového počítačem ovládaného hráče by bylo zcela předvídatelné. Implementace takového chování by z hlediska dynamičnosti neustále se rozšiřující hry byla špatně rozšiřitelná.

Druhá možnost nabízí využití prohledávaní stavového prostoru. Pro jednoduché hry, které nemají rozsáhlý stavový prostor, je tato metoda ideální. V případě této karetní hry je tato možnost opět nevhodná. Stavový prostor se s přidáváním dalších karet rapidně rozšiřuje a už základní implementace, která by obsahovala pouze jednoduchý balíček, by čítala mnoho stavů. To by bylo způsobeno nejen počtem karet, ale také počtem zón, kdy aktuální stav hry představuje rozmístění různých karet po různých zónách. Navíc hra obsahuje neurčitost v podobě tahání karet v balíčku. Z hlediska hratelnosti, pokud by byla tato možnost aplikovatelná, by hráč měl špatný požitek ze hry, protože umělá inteligence by vždy vybrala pro ni nejvhodnější tah.

Třetí možností je využití markovských procesů, které se hodí pro problémy, kdy je v část hry pod kontrolou hráče a část hry je závislá na náhodě. Opět se zde ale objevuje problém s rozsáhlým stavovým prostorem a dynamické rozšiřovaní pravidel hry.

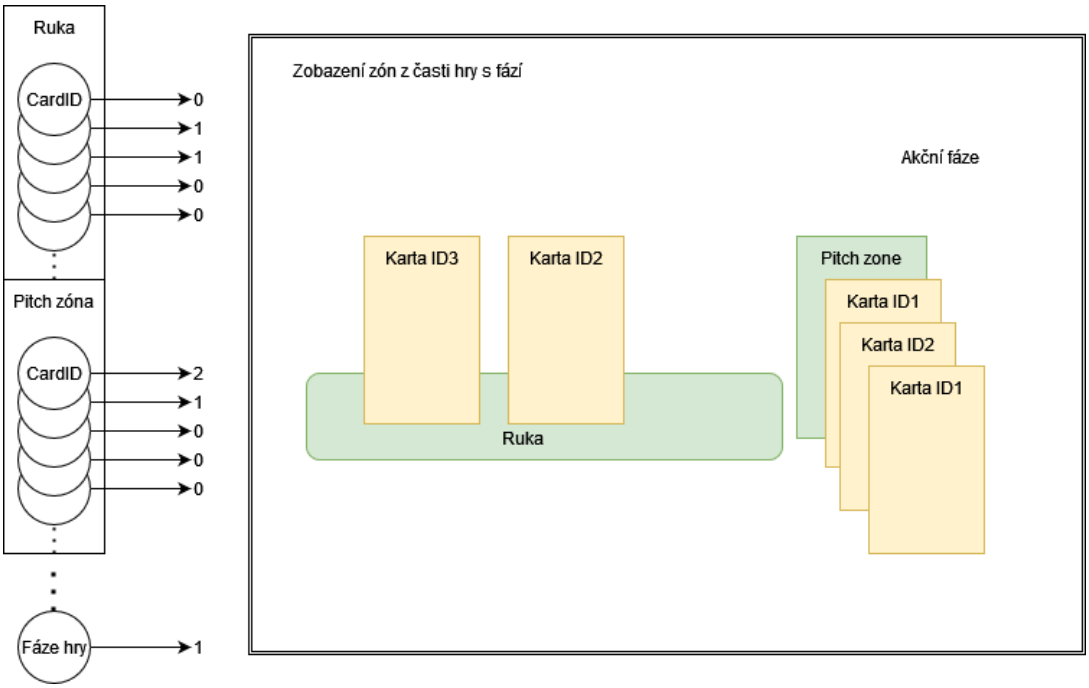
Čtvrtá možnost je použiti neuronových sítí. Zde se vyhneme problému s rozsáhlým stavovým prostorem. Dále při použití herního enginu Unity se nabízí možnost využití rozhraní ML-Agents, které dovoluje tvorbu agenta a samotné trénovaní neuronové sítě. Chovaní takového počítačem ovládaného protivníka bude více přirozené než v ostatních případech a bude možné i jednoduše nastavovat sílu protivníka. To se dá provést pomocí šumu na vstupu neuronové sítě. Bohužel je zde opět problém s dynamickým rozšiřováním pravidel, kdy při každém rozšíření bude nutné síť znovu natrénovat.

V této práci se umělá inteligence realizuje pomocí neuronové sítě, protože nabízí přirozené chovaní simulující reálného hráče a samotná implementace je částečně zjednodušena díky vybraným technologiím.

Návrh neuronové sítě

Pro tvorbu neuronové sítě jsou zde dvě možnosti. První je vytvoření architektury sítě, které je schopná hrát s libovolným herním balíčkem proti libovolnému protivníkovi. Druhá možnost je síť, která je schopna hrát za jeden vybraný a předem definovaný herní balíček. První možnost je velice robustní a při rozšíření hry nebude potřeba znovu trénovat několik sítí. Dále hráči mohou přímo přidělit jimi postavený herní balíček a samotná umělá inteligence bude schopna daný balíček ovládat. Druhá možnost zase přináší zjednodušení celkové architektury sítě a tím i zjednodušení trénovaní samotné sítě. Z hlediska charakteru práce se druhá možnost jeví jako reálnější z důvodu omezeného času a nutnosti vytvářet samotnou hru od úplného základu.

Důležité je vhodné kódovaní vstupu neuronové sítě. Vstup tvoří všechny dostupné informace o aktuálním stavu herní desky. Numerické informace jako je aktuální počet životu hrdinů, nebo počet akčních bodů se kódují napřímo jejich hodnotou. Informace o aktuální herní fázi hry a fázi kola se kóduje jako pevná hodnota přiřazená jednotlivé fázi. Například pokud je fáze kola pro počítačem ovládaném protivníkovi nastavena na hodnotu *bez priority*, je to tato hodnota kódovaná jako nula na vstupu odpovídajícímu fázi kola pro druhého hráče. Další je stav jednotlivých zón na herní desce. Zde je každá zóna reprezentována jako skupina vstupů. Každý vstup z jedné skupiny představuje jednu konkrétní kartu. Takový vstup má hodnotu, která odpovídá počtu instancí dané karty v příslušné zóně. Pro zóny, jako je například oponentova ruka, je předávaná pouze informace o počtu karet, které se v dané zóně nachází. Příklad takového kódovaní je možné vidět na obrázku 4.5.



Obrázek 4.5: Příklad kódovaní vstupu. V levé části je zobrazení jednotlivých skupin, které obsahují vstup pro jednotlivé karty s příslušnou hodnotou vstup. Hodnota odpovídá stavu části hry zobrazeném v pravé části obrázku. Žlutě jsou naznačeny jednotlivé karty a zeleně poté zóny. Aktuální fáze hry je zobrazena vpravo nad *pitch zónou*

Výstupem neuronové sítě bude rozhodnutí o provedení jedné akce. Akce, a tím pádem i výstupy, lze rozdělit do čtyřech větších skupin. První je rozhodnutí o zahrání vybrané karty. Tím je myšleno provedení akce jako je zahrání karty z ruky na *combat chain*, zaútočit zbraní, aktivovat schopnost hrdiny aj. Další skupina je rozhodnutí, jaká karta má být využita pro obranu. Třetí skupina rozhoduje, která karta má být využita pro zaplacení provedení vybrané akce. Poslední skupina je rozhodnutí o umisťování karty z ruky do *arsenalu* na konci kola. Každá skupina pak obsahuje i výstup který označuje akci *Pass*.

Agent řízený neuronovou sítí musí mít znalosti o pravidlech hry, aby mohlo začít trénování pro zdokonalení herních vlastností. Nechat agenta, aby se učil pravidla sám a dle jeho rozhodování by dostával ohodnocení, je velice zdlouhavé. Proto je nutné vhodně interpretovat tato pravidla pro vstup neuronové sítě, aby při rozhodnutí nebyl proveden neplatný tah. Toho lze docílit pomocí ML-Agents, kdy je možné vymaskovat jednotlivé výstupy a to i v rámci skupin. Neuronová síť má poté při průchodu znalost o vymaskovaných výstupech a dokáže se jim přizpůsobit. Poté už zůstávají pouze platné tahy. Jelikož síť nemá informace o pravidlech jednotlivé karty, musí se tak naučit tyto pravidla sama při trénování. To však nechává prostor pro natrénování vlastní strategie. Uvítací balíček Ira je v celku ideální pro tento případ, protože obsahuje karty, které benefitují z hraní útoků po sobě. Tím pádem větší poškození za kolo může hráč docílit dobrým výběrem karet a pořadím jakým se zahrají, tak i správným výběrem jak za dané karty zaplatit.

Trénovaní sítě zajišťuje balíček Ml-Agents, který pomocí pythonovkého scriptu řídí trénovaní na základě implementace agenta a konfiguračního souboru. Neuronová síť se učí hrou sama proti sobě. Kladné ohodnocení dostává za výhru v dané hře. Naopak za prohru získá záporné ohodnocení. Míra ohodnocení je závislá na počtu zbývajících životů vítězné neuronové sítě.

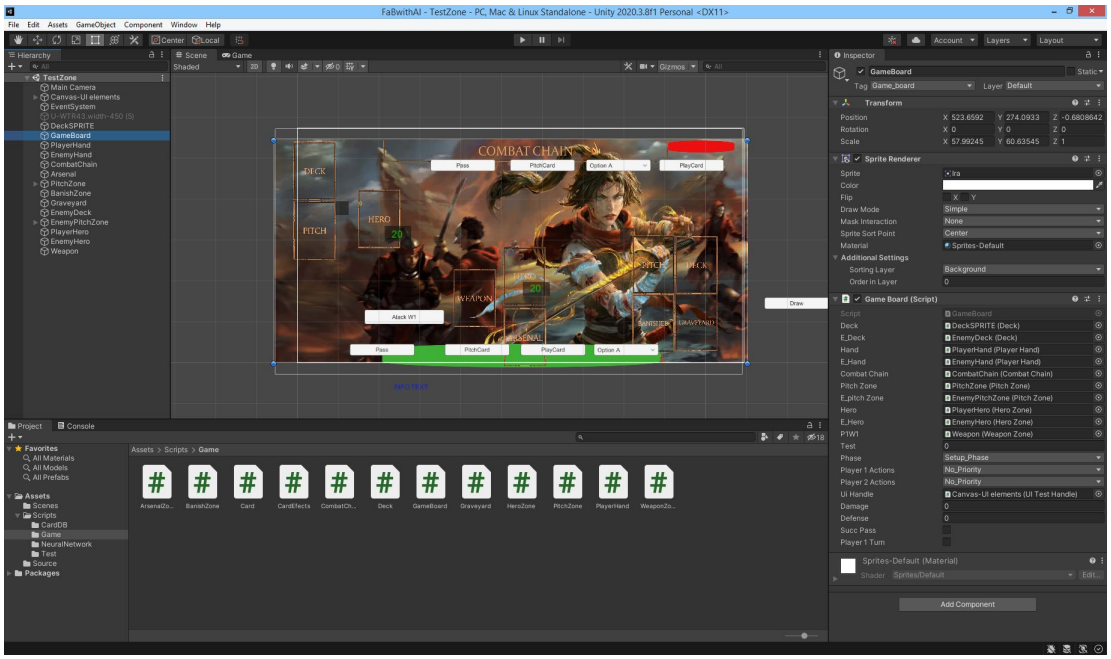
Kapitola 5

Nástroje

Hra je vytvářena v herním engine Unity ve verzi 2020.3.8. Jako rozhraní pro tvorbu umělé inteligence je využit balíček Ml-Agents 1.0.8 ve stabilní verzi pro Unity. Pro úpravu obrázků se využívá nástroje GIMP. Pro tvorbu diagramů je použit nástroj *draw.io*.

5.1 Unity

Unity je mutiplatformí volně dostupný herní engine, který složí jako nástroj pro tvorbu 2D a 3D her pro počítače, konzole a mobilní zařízení. Na těchto zařízeních podporuje mnoho operačních systémů. Tento engine také dovoluje tvorbu webových her nebo také balíčků, které mohou sloužit jako základ pro ostatní hry. Balíčky také mohou implementovat různé části her jako například realistické zobrazování vody.



Obrázek 5.1: Rozhraní unity engine - 2D scéna zobrazující tvorbu hry *Flesh and Blood*

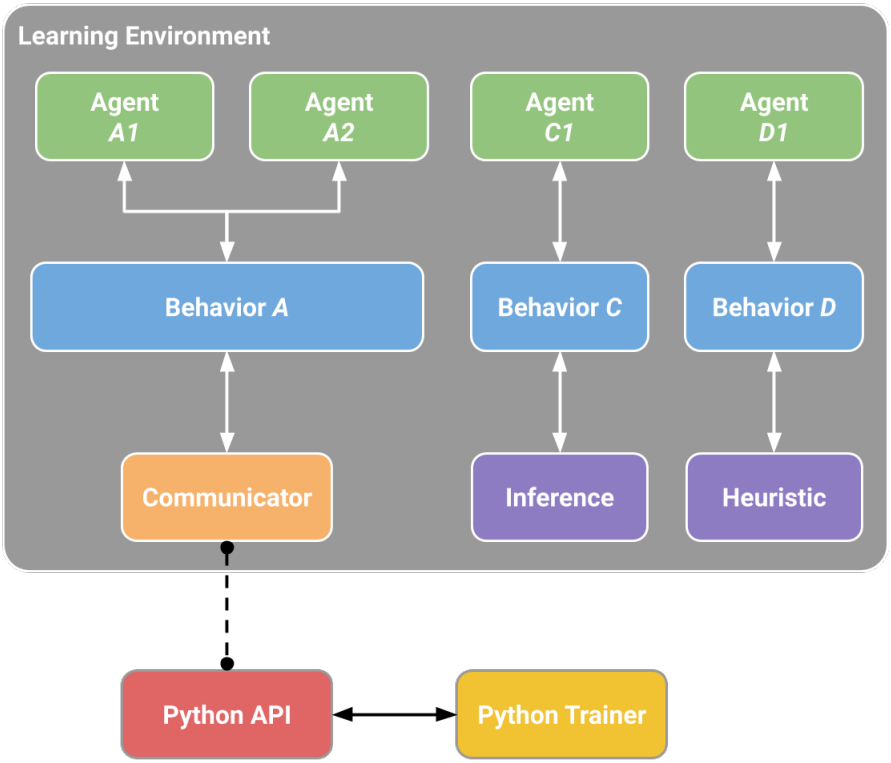
Samotné prostředí enginu je velice jednoduché na pochopení, což usnadňuje práci i méně zkušeným programátorům v oblasti práce v herním enginu. Další výhodou je možnost

spustit hru přímo v okně editoru a za běhu hry modifikovat ať už jednotlivé herní objekty nebo hodnoty veřejných proměnných připojených u scriptů. Podporovaný jazyk je pouze jazyk C#. Unity umožňuje programování vlastních shaderů, kdy prvně bylo zamýšleno použití compute shadrů v jazyce HLSL pro urychlení trénování.

Knihovna enginu Unity nabízí plno nástrojů pro programování. V této práci se často využívá takzvaných *Coroutines*. Ty dokazí rozložit výpočet přes několik snímků a tím nezpůsobí zpomalení samotné hry [11]. Ve hře jsou však *corutiny* použity pro čekání na provedení akce, protože dokáží předat řízení zpět samotnému enginu a pokračují až s dalším snímkem. Nahrazují zde použití eventů, které by kvůli režii zpomalovali hru pro umělou inteligenci. Eventy jsou využity až pro komunikaci s grafickým uživatelským rozhraním.

Unity pro implementaci herních objektů používá bázovou třídu *MonoBehavioral*. Ta nabízí sadu funkcí, které dovolují objektu pracovat s engine. Příkladem je metoda *Update*. Tato metoda se invokuje vždy před zobrazením nového snímku. Kód této metody si programátor může upravit dle potřeb objektu ke kterému je script připojen.

5.2 ML-Agents



Obrázek 5.2: Diagram prostředí pro trénování. Zeleně jsou jednotliví agenti. Agenti mohou sdílet stejné chování, jako je tomu u agentů A1 a A2. Chování A podléhá trénování které díky komunikátoru může pracovat s Pythonovským API. Chování C je už natrénované a rozhodování probíhá díky natrénované neuronové síti. Chování D je dáno přesně kódovaným chováním ve scriptu agenta v metodě *Heuristic*.

ML-Agents je balíček dostupný pro Unity engine. Jedná se o open-source projekt, který dovoluje tvorbu inteligentních agentů pro hry. Trénování může probíhat pomocí posilova-

ného učení, imitace chování, nebo libovolné metody z rodiny strojového učení a to pomocí Pythonovského API. Balíček nabízí jednoduchý, leč velice silný způsob pro tvorbu umělé inteligence. Pro tvorbu agenta je třeba definovat chování, získávání znalostí z okolí a ohodnocení. Pro samotné trénovaní je třeba definovat konfiguraci. Na obrázku 5.2 je vidět diagram ukázkového prostředí pro trénování.

Kapitola 6

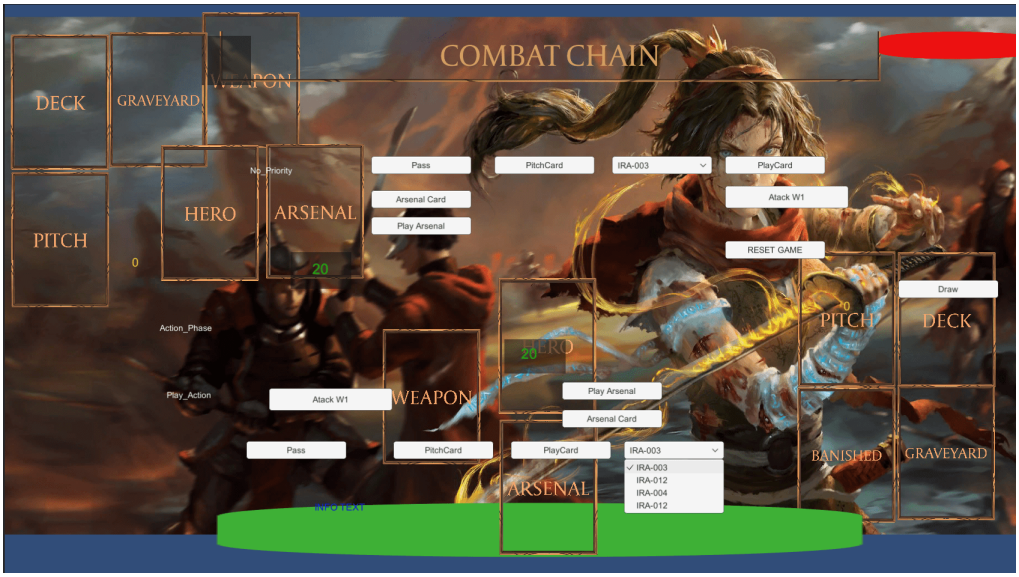
Implementace

Samotná implementace se dle návrhu skládá ze tří hlavních částí. Každá část představuje samostatný modul schopný komunikovat s ostatníma dle potřeb.

Prvně je potřeba implementovat samotná pravidla hry. Ta řídí celý průběh na pozadí a tvoří mozek celé aplikace. Pod modul pravidel patří i část implementující karty z hlediska logiky bez grafické reprezentace. Tento krok je nutný z důvodu rychlejšího trénovaní umělé inteligence, protože ta nepotřebuje interagovat s grafickým uživatelským rozhraním během trénování při hře sama proti sobě.

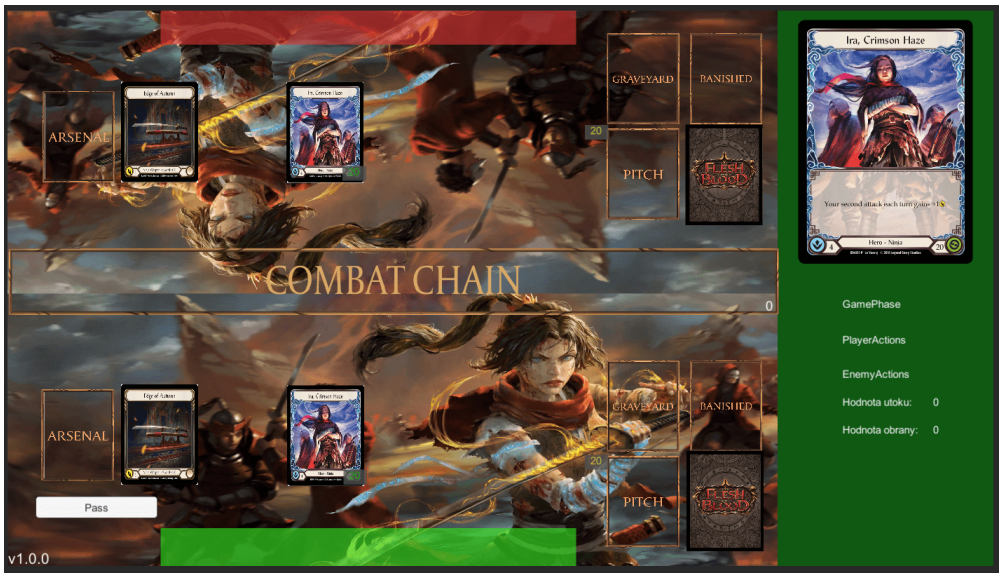
Druhý krok byla implementace umělé inteligence. Zde bylo zapotřebí implementovat pozorování hry, reagování na události a chování při trénování jako je ohodnocení chování či rozpoznání konce hry. Trénování a tvorba architektury sítě je tvořena balíčkem ML-Agent. Zde je potřeba dodat pouze konfigurační soubor.

Po navázaní komunikace mezi těmito dvěma moduly je možné spustit trénovaní sítě. Z důvodu verifikace je vytvořena samostatná herní scéna, která je na obrázku 6.1. Tato scéna obsahuje základní grafické uživatelské rozhraní pro sledovaní hry. Je zde také možné hru ovládat s pohledu obou hráčů pomocí základních prvku grafického uživatelského rozhraní. To dovolí jak verifikaci pravidel hry, tak verifikaci chování umělé inteligence.



Obrázek 6.1: Herní scéna pro verifikaci pravidel a trénování

Dále je nutné implementovat grafické uživatelské rozhraní aplikace. To slouží pro interakci uživatele se hrou. Při implementaci tohoto modulu je plně využita síla herního engine Unity. Je nutné vytvoření čistě grafické reprezentace karty, která reprezentuje kartu na pozadí aplikace. Výsledkem tohoto modulu je scéna z obrázku 6.2 reprezentující herní desku připravenou k hraní hry *Flesh and Blood*.



Obrázek 6.2: Herní scéna pro hraní hry uživatelem

V další části jsou popsány podrobnosti implementace jednotlivých modulů a jejich nejdůležitějších částí. Poslední sekci tvoří popis implementace komunikace mezi moduly.

6.1 Pravidla

Implementace pravidel hry vychází přímo z definice pravidel hry [9]. Každá zóna tvoří samostatný modul. Pro interakci mezi moduly zde slouží herní deska, která je vzájemně spojuje.

Herní deska

Herní deska tvoří základní prvek v modulu pravidel. Implementace se nachází v souboru *GameBoard.cs*. Zde jsou definovány všechny výčtové typy reprezentující herní fázi, části akční fáze a druhy priorit. Pro herní fázi jsou zde kromě fází definovaných v pravidlech přidány fáze:

- *Setup_Phase* pro nastavení hry při začátku.
- *Game_End_Phase* pro označení a detekovaní ukončení hry ze strany hrací desky.
- *Reset_Phase* pro restartovaní hry pro novou hru.

Podobně tomu je i pro části akční fáze hry, kdy kromě těch definovaných v pravidlech jsou přidány:

- *No_Priority* označující fakt, že daný hráč nemá herní prioritu.
- *Need_Energy* označující potřebu zaplatit za danou akci energii a hodnota energie v *pitch zóně* nestačí k zaplacení.

Druhy priorit jsou v momentálním kontextu nepoužity, ale jsou využitelné pro snadnější trénovaní sítí při hře sama proti sobě.

Následuje implementace třídy *GameBoard*, která dědí chovaní z *MonoBehavioral* Unity třídy. Tuto třídu lze přímo přiřadit hernímu objektu ve scéně, který se poté bude chovat jako herní deska. Tato herní deska drží reference o ostatních herních zónách, informace o herní fázi, části akční fáze pro každého z hráčů, počítadla (poškození, obrana, počet úspěšných zásahů, počet chain linků) a řízení předávání priorit.

Metody, které jsou v této třídě definovány, slouží k úpravě počítadel či zjišťovaní potřebných informací pro různé zóny. Nachází se zde corutiní funkce, která čeká na rozhodnutí o umísťovaní karty do zóny *Arsenal* na konci hráčova kola. Po dokončení této funkce se spustí automaticky kroky pro řádné ukončení kola hráče. Samostatné řízení hry z hlediska pravidel se provádí také zde. O tom jaké akce se mají spustit se rozhoduje jednou za nový snímek v metodě *Update*.

Combat chain

Combat chain je místo, kde se odehrávají všechny zahrané akce. Implementace se nachází v souboru *CombatChain.cs* a obsahuje tři třídy.

První je třída *CombatChain*, která dědí z Unity třídy *MonoBehavioral* a opět se váže přímo k hernímu objektu ve scéně. Tato třída si uchovává referenci na přáslišnou herní desku, počet akčních bodů, informace o řízení akční fáze a několik struktur typu *List* obsahující:

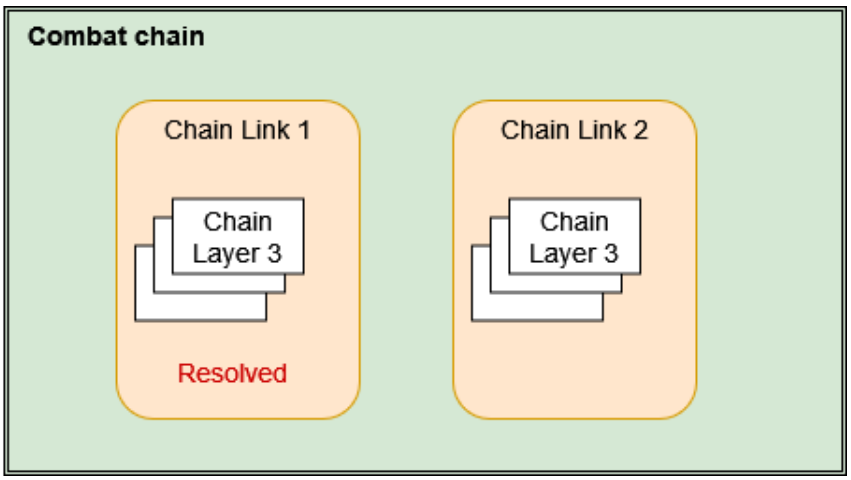
- Aktuální karty nacházející se na zóně *Combat chain*.
- Karty, které v daném kole byly zahrány prvním hráčem.
- Karty, které byly zahrány v daném kole druhým hráčem.
- *Chain linky*, které byly vytvořeny v aktuálním kole aktivním hráčem.
- Efekty karet, které jsou platné po celé kolo, či více kol.
- Dočasné efekty karet které se po své aplikaci mizí.
- Efekty vztahující se na další útok.
- Efekty, které se projeví pokud útok provede *hit*.

Dále je zde implementace podpůrných metod na řízení běhu samotného *Combat chainu*.

Zásadní je metoda pro hraní karet *AddCard*. Při invokaci této metody se spustí proces zpracování karty, která je na vstupu metody. Zóna, která tuto metodu invokovala, musí zaručit, že je karta hratelná. Jediné o co si *Combat chain* kontroluje při přidávaní karty je bránění pomocí karet z ruky. Blokování kartou totiž není zahrání karty a nevztahují se na ni další pravidla ohledně placení, či přidávání efektů. Pokud se jedná o zahrání karty, *Combat chain* určí, kterému hráči karta patří a vyhodnotí, zda je dostatek energie k zaplacení karty. Pokud ne, dostane se do fáze, kdy k dalšímu vyhodnocování je třeba kartu zaplatit. Tuto

funkci obstarává *Pitch zone*. Toto čekání je implementováno pomocí corutiní funkce. Po zaplacení dostatečného množství energie se začne karta vyhodnocovat.

Pro vyhodnocování karet jsou přítomny dvě třídy, které simulují vyhodnocování stejně jako tomu je v pravidlech hry. První třída je *ChainLink*. Ta v sobě drží provedení jedné celé akce se všemi dalšími doprovodnými kartami a efekty. Tato třída obsahuje dvě struktury typu *List*. První pro uchování karet příslušících k danému *Linku* a druhá pro uchování *ChainLayer* vrstev. *ChainLayer* je samostatná třída, která slouží pro uchování efektu, nebo samotného útoku a je schopna tento efekt, respektive útok spustit. Celá struktura složení *Combat chainu* je na obrazku 6.3.



Obrázek 6.3: Struktura combat chainu

Další důležitá metoda je *ResolveChain*. Ta se stará o vyhodnocení útoku či akce. Pokud není žádný *Chain Link* otevřený během invokace, znamená to, že hráč nechce hrát žádnou akci a je možné ukončit kolo. Pokud však je nějaký *Chain Link* otevřen, dojde k aktivování corutiny pro hraní obrany a následnému hraní reakcí, pokud se jedná o útok. Po ukončení hraní reakcí se provede celkové vyhodnocení daného *Chain Linku*. Vyhodnocení probíhá za pomocí postupného spouštění efektů patřící k danému *Chain Linku*.

Po ukončení kola se *Combat chain* postará o vyčištění všech struktur, až na přetrvávající efekty, které mohou mít platnost i na více kol.

Hráčova ruka

Implementace hráčovi ruky se nachází v souboru *PlayerHand.cs*. Zóna je implementovaná genericky, takže pouze její nastavení v editoru rozhoduje o tom, zda se jedná o uživatelovu ruku nebo ruku počítačem řízeném protivníkovi. Soubor obsahuje třídu *PlayerHand*, která dědí z Unity třídy *MonoBehavioral*. Třída si drží referenci na příslušnou herní desku, příslušnou *Arsenal zónu*, příslušnou *Pitch zónu* a příslušný herní balíček. Dále si drží informaci o počtu karet, která je důležitá především z hlediska verifikace. Je zde také struktura typu *List*, která si v sobě uchovává karty nacházející se v této zóně. Poslední je údaj o číslu hráče.

Tato třída implementuje metody, které provádějí manipulaci s kartami v této zóně. Nejdůležitější je metoda *PlayCard*, která jako parametry přijímá identifikátor karty a příslušný *Combat chain*. Tato metoda provede kontrolu zda zahrání vybrané karty je dle pravidel možné a pokud tomu tak je, zkontroluje zda je možné kartu zaplatit z dostupných zdrojů.

Pokud jedna z podmínek není splněna, metoda neprovede žádnou akci. Pokud jsou všechny podmínky splněny, je invokovaná metoda *PlayCard* z *Combat Chainu*.

Další metodou je *Pitch card*, která vybranou kartu dle identifikátor odešle na *Pitch zónu*. Pokud se karta na ruce nenachází, metoda neprovádí žádnou akci.

Poslední důležitou metodou je *ArsenalCard*. Ta podobně jako předchozí metoda odesílá kartu do *Arsenal zóny*.

Ostatní metody jsou podpůrné pro získávání informací jako je například maximální zisk energie z aktuálního stavu na této zóně, nebo zda se vybraná karta dá zaplatit ze všech dostupných zdrojů.

Pitch zóna

Implementace *Pitch zóny* se nachází v souboru *PitchZone.cs*. Zde se nachází třída *PitchZone*, která dědí z Unity třídy *MonoBehavioral*. Třída si drží referenci na příslušný herní balíček do kterého vrací karty na konci kola. Dále je zde struktura typu *List*, která v sobě uchovává aktuální karty nacházející se v této zóně. Poslední je informace o počtu bodů energie, které jsou ještě dostupné k použití v daném kole. Pro verifikaci je zde přidána reference pro textový objekt, na kterém se zobrazuje aktuální hodnota energie.

Třída implementuje metodu pro vkládání vybrané karty do seznamu karet nacházejících se na této zóně s příslušnou úpravou aktuálního počtu energie.

Neméně důležitá je metoda starající se o úklid zóny na konci kola a již zmíněného přesunu všech karet na spodek herního balíčku. V aktuální verzi není možné pořadí karet přeskládat dle zvoleného pořadí.

Poslední je metoda *PayCost*. Ta si bere jako parametr hodnotu, která představuje počet bodů energie potřebných k zaplacení akce. Pokud je v zóně dostatečný počet bodů, upraví se počet aktuální hodnoty energie a návratová hodnota metody zajistí ukončení corutiní funkce na *Combat chainu*, která brání akci od jejího vyhodnocení z důvodu nedostatku energie.

Arsenal

Arsenal zóna má svoji implementaci v souboru *ArsenalZone.cs*. Arsenal je implementován třídou *ArsenalZone*, která dědí z Unity třídy *MonoBehavioral*. Tato zóna si drží informaci o tom, zda je prázdná, případně pak ukládá příslušnou kartu. Dále jsou zde uloženy reference na několik zón:

- Herní desku aby bylo možné hrát kartu na *Combat chain* a pro zjišťování fází hry.
- Příslušná ruka pro vracení karty při špatném arsenalování na konci tahu a pro zjištění zda je možné kartu z této zóny doplatit pokud je hrána.
- *Pitch zónu* pro informace o aktuálním počtu dostupné energie.

Důležitou metodou je *PlayCard*, ta zajišťuje hraní karty na *Combat chain*. Je velice podobná stejnojmenné metodě z třídy *PlayerHand*, až na to že z této zóny nelze použít kartu na zaplacení energie, nebo s ní bránit během obrané fáze.

Metoda *IsDecoded* řídí dávání karty na konci tahu do arsenalu. Pokud se hráč rozhodne na konci tahu dát kartu z ruky do této zóny, tato metoda zajistí přerušení corutiní funkce, která pozdržuje průběh hry pro rozhodnutí, zda hráč chce umístit kartu do *Arsenalu* či nikoliv.

Dále je zde metoda na vložení karty *AddCard* a metoda na vyčištění zóny při restartování hry *Restart*.

Zóna pro zbraň

Zóna, určená pro zbraň, má svoji implementaci v souboru *WeaponZone.cs*. Zde se nachází třída *WeaponZone*, která jako ostatní zóny dědí z Unity třídy *MonoBehavioral*. Zóna si udržuje informaci o tom zda se zbraní, která se zde nachází, bylo zaútočeno a k tomu si udržuje reference na herní desku, hráčovu ruku a *Pitch zónu*. Reference těchto zón mají stejně opodstatnění jako v případě *Arsenal zóny*. Jako poslední je zde uložena samotná karta zbraně.

Nachází se zde implementace metody *Attack*, která zaručuje útok se zbraní. Zbraň ovšem nesdílí kompatibilní typ karty, které se mohou hrát na *Combat chainu*. Proto se pro její zahrání vytvoří dočasná reprezentace karty se správným typem.

Zóna pro herní balíček

Zóna pro herní balíček je implementovaná v souboru *Deck.cs*. Zde se nachází stejnojmenná třída *Deck*, která dědí z Unity třídy *Monobehavioral*. Třída si v sobě udržuje informaci o velikosti a strukturu typu *List*, která obsahuje karty nacházející se na této zóně.

Hlavní úlohou této zóny je, že pomocí předem definované metody v sobě vytvoří herní balíček připravený ke hře. Aktuální implementace obsahuje implementaci *Ira Welcome decku* pomocí metody *IraWlcomeDeck*.

Lízaní karet obstarává metoda *DrawCard*, která jako parametr přijímá vybranou ruku na kterou se má karta přesunout. Poté provede přesun karty na vybranou hráčovu ruku. Tento přístup je zvolen, protože některé karty umožňují podívaní se na herní balíček oponentem.

Třída obsahuje další metody, které obstarávají míchání balíčku nebo restartování zóny.

Ostatní zóny

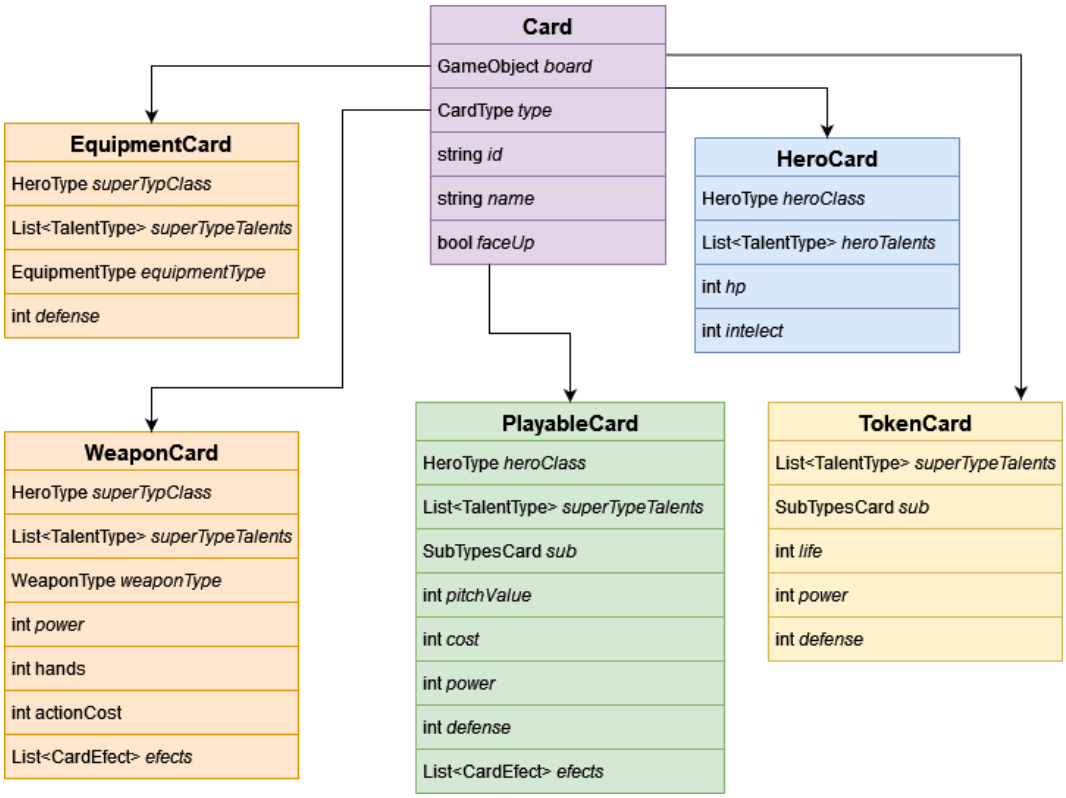
Ve hře se momentálně nenachází implementace pro zóny obsahující brnění. Vybraný herní balíček je totiž neobsahuje a jejich využití by přidalo další úroveň složitosti, zvláště pak při hraní obrany.

Ostatní zóny mají především úlohu držet informaci o kartě či kartách, které se na nich nachází:

- Zóna pro hrdinu je implementována v souboru *HeroZone.cs*. Příslušná třída udržuje kartu hrdiny a zaručuje počítání aktuálních životů a intelektu.
- Zóna pro hřbitov je implementována v souboru *Graveyard.cs*. Úlohou této třídy je udržovat v sobě zahrané karty.
- *Banish zóna* je implementovaná v souboru *BanishZone.cs*. V aktuální verzi má podobnou implementaci jako zóna pro hřbitov. Při implementaci karet ze sady *Monarch* dochází k upravě pravidel a je nutno tuto zónu pozměnit.

6.2 Karty

Implementace rodičovských tříd pro vytvoření karet se nachází v souboru *Card.cs*. Výpis těchto tříd a jejich hierarchie je vyobrazena na obrázku 6.4. Základem pro každou kartu je třída *Card*. Zde je důležité, že při vytváření si tato třída získává referenci na herní desku. Tento přístup dovoluje kartě interagovat s herní deskou. Negativní dopad nastává, že při trénování nelze vytvořit více instancí herní desky pro urychlení trénování. Z této třídy podtřídy představující více specifickou kartu co se typu týče. Zásadní je třída *PlayableCard*. Ta je nadřazena všem kartám, které se nacházejí v balíčku a představuje herní karty. Takto implementované karty jsou čistě virtuální a nemají žádnou fyzickou podobu ve hře tj. neváží se k žádnému hernímu objektu na přímo.



Obrázek 6.4: Hierarchie rodičovských tříd pro torbu karet. Fialově je vyobrazena základová třída. Modrá je pro karty které představují hrdinu, oranžová představuje karty vybavení, žlutá patří tokenům vytvořených po zahrání efektů a zelená představuje herní karty.

Soubor *Card.cs* dále obsahuje všechny výčtové typy pro jednotlivé označení jednotlivých typu a supertypů karet ve hře.

Efekty karet

Obecné efekty karet mají svoji implementaci v souboru *CardEfects.cs*. Bázová třída je *CardEfect*, která v sobě opět drží referenci na herní desku. S touto referencí lze přistupovat k metodám jednotlivých zón a tím provádět aplikaci těchto efektů. Každý efekt se rozlišuje svým typem, který udává kdy se má daný efekt zpracovat a kam se má zařadit v rámci

vyhodnocení *Combat chainu*. Výčtový typ označující zařazení má svoji implementaci ve stejném souboru. Typy a jejich zařazení je následující:

- *Special* je pro efekty, které mají využití definované přímo v sobě a *Combat chain* je nerozlišuje.
- *Turn* je pro efekty, které mají své trvání po celé kolo a lze je aplikovat i vícekrát.
- *Next_attack* je pro efekty, které se mají aplikovat na další útok.
- *On_hit* je pro efekty, které se spustí pokud karta ke které jsou vázány způsobí hit[1].
- *Inmidiete* je pro efekty, které mají okamžité spuštění.

Pro spuštění efektu je využita virtuální metoda *Process*. Ta v sobě implementuje chování každého efektu.

Tvorba karet a jejich databáze

Každá karta v papírové podobě je reprezentována samostatnou třídou ve hře. Třída vždy dědí z vybrané třídy dle svého typu. Tyto rodičovské třídy jsou na obrázku 6.4. V konstruktoru se vždy vyplní všechny údaje která daná karta obsahuje. Dále se naplní struktura *List* všemi efekty. Důležité je správné vyplnění identifikátoru karty. Pomocí něj se totiž na karty odkazuje při jejich manipulaci. Jako identifikátor se používá téměř identický identifikátor jako pro karty v papírové podobě. Tvarem takového identifikátoru jsou tři písma označující sadu následované pomlčkou a trojmístnou číslicí označující pořadí v sadě. Originální identifikátor lze vidět na obrázku 6.5.

Obrázek 6.5: Identifikátor fyzické karty. V digitální podobě je jeho formát ELE-129

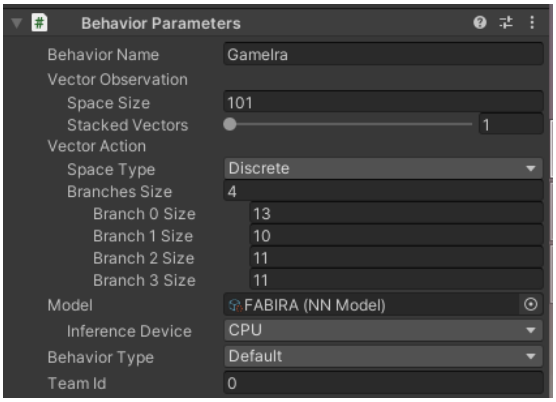
Implementace jednotlivých karet a jejich unikátních efektů je v samostatných souborech pojmenovaných zkratkou sady. Karty tvořící *Ira Welcome deck* jsou implementovány v souboru *IRACards.cs*. Hierarchie těchto souborů tvoří databázi pro virtuální reprezentaci karty.

6.3 Umělá inteligence

Umělá inteligence tvoří samostatný blok v rámci aplikace. Její implementace je silně závislá na balíčku ML-Agents. Ten poskytuje základy pro tvorbu umělé inteligence. K implementaci byla použita oficiální dokumentace [10]. Pro správné fungování je třeba implementovat pozorování, chovaní a ohodnocení agenta. Důležité je také nastavení parametrů samotného agenta v editoru hry. Aktuální nastavení takového agenta je zobrazeno na obrázku 6.6.

Počítačem ovládaný protivník má svoji implementaci v souboru *AIPlayer.cs*. Zde se nachází stejnojmenná třída *AIPlayer* která dědí z ML-Agents třídy *Agent*. Třída v sobě uchovává reference na své a nepřátelské zóny tak i na hrací desku. Dále je zde uchována

[1]Karta způsobí hit pokud po vyhodnocení celého *Chain linku* je hodnota útoku vetší než hodnota obrany



Obrázek 6.6: Parametry agenta nastavované v editoru. Název chování je důležitý z hlediska trénování pro rozpoznání agenta. Vektor pozorování určuje velikost vstupu neuronové sítě. Vektor akcí určuje velikost výstupu neuronové sítě a jeho typ. Model představuje natrénovaný model neuronové sítě. Typ chování určuje zda se má využít neuronové sítě pro rozhodování, či se agent bude rozhodovat podle definovaného chování uživatelem. Týmový identifikátor je důležitý z hlediska trénování pro rozlišování agentů

informace zda se jedná o hráče na pozici jedna nebo dva. Toto je důležité z hlediska trénování, neboť třída *AIPlayer* představuje libovolného hráče. Poté jsou zde drženy dvě důležité informace a to o tom, zda se umělá inteligence rozhoduje o svém tahu a maximální počet kroků, které může zahrát v jedné hře. Počet kroků je důležitý z hlediska trénování, aby se vyhnulo situaci kdy, počítačem řízení hráči pouze předávali prioritu a nehráli žádné akce.

Počítačem řízený protivník si při každé aktualizaci herní fyziky zjistí zda není po ni vyžadováno rozhodnutí. Pokud je vyžadováno rozhodnutí, provede se pozorování stavu na hrací desce. Neuronová síť má 101 vstupů. Tyto vstupy jsou naplněny pomocí metody *CollectObservations*. Zde jsou postupně předány informace o výskytech karet na zónách patřící agentovi. Posléze jsou předány informace o stavu protihráčových zón. Poslední jsou generické informace o stavu hry, které jsou dostupné oběma hráčům jako například počty energie na *Pitch zónách* nebo jaká je zrovna fáze hry. Dále se zamaskují výstupy, které v dané fázi hry ztrácejí smysl. Příkladem může být zamaskování výstupů, které jsou pro zahrání karet, ale dané karty se na ruce nenachází. Proto aby bylo možné provést maskování, musí neuronová síť počítat v diskrétním prostoru. Masky se počítají pro každou skupinu výstupů zvlášť a zajišťují je implementované metody v rámci stejné třídy. Metoda *OnActionReceived* poté na základě výstupu neuronové sitě provede vybraný tah.

Architektura neuronové sítě je 101 vstupů, 45 výstupů rozdělených do čtyřech skupin a obsahuje 2 skryté vrstvy o velikosti 256 neuronů. Skupiny výstupů jsou následovně i s počtem odpovídajících výstupů:

- Zahrání akce jako je karta z ruky, útok zbraní nebo hraní karty z arsenalu. Celkem 13 výstupů.
- Zaplacení akce kartou z ruky. Celkem 10 výstupů.
- Bránění kartou z ruky. Celkem 11 výstupů
- Dání karty z ruky do arsenalu. Celkem 11 výstupů.

Konfigurační soubor

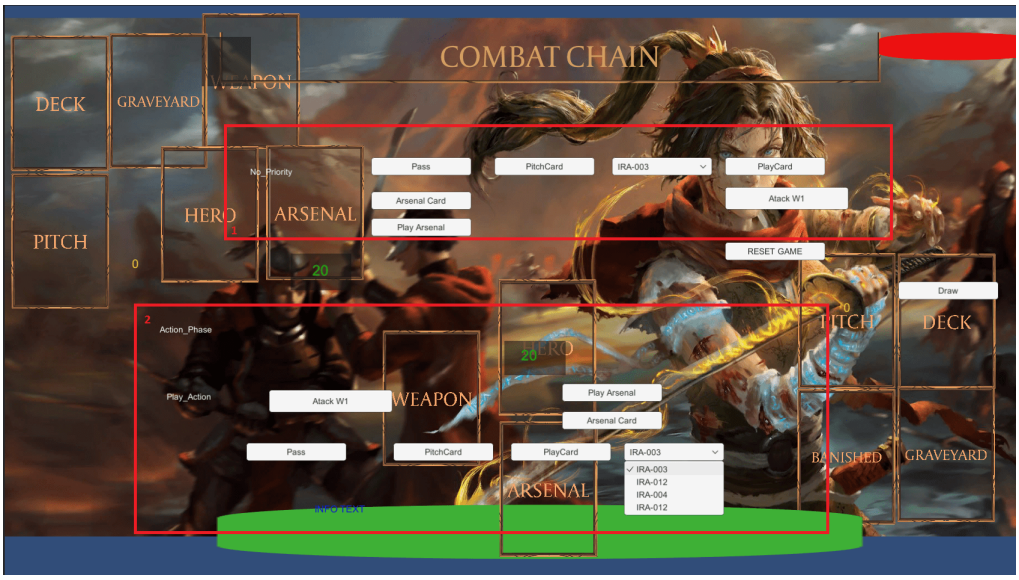
Pro trénování a sestavení architektury sítě je nutné definovat konfigurační soubor. Pro využitou neuronovou síť FABIRA je konfigurační soubor s názvem *trainingParam.yaml*. Z hlediska trénování je důležité vybrat trénovací algoritmus, který byl v tomto případě zvolen PPO. Dále se zde nastavuje počet kroků které, budou v jednom trénovacím cyklu provedeny, změna hodnot parametrů kde je možnost výběru mezi lineární a konstantní. Další parametry jsou už závislé na výběru trénovacího algoritmu nebo slouží pro architekturu sítě. Parametry pro PPO byly navoleny doporučeně pro stabilní trénování sítě.

6.4 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní je poslední velký blok tvořící aplikaci. Hra v sobě nese dvě scény, kdy první je určená čistě pro trénování a verifikaci pravidel. Druhá poté slouží jako reprezentace samotné hry určená uživatelům k hraní. Dále jsou popsány jednotlivé scény z hlediska uživatelského rozhraní.

Scéna pro testování

Testovací a trénovací scéna složí pro rychlé a jednoduché ovládání. Její vyobrazení s rozdělením je na obrázku 6.7.



Obrázek 6.7: Herní scéna pro verifikaci pravidel a trénování. Tlačítka jsou dělena na dvě skupiny. První skupina slouží pro ovládání hráče, který ve hře představuje umělou inteligenci. Skupina číslo dva představuje uživatele. Tlačítka mimo tyto skupiny slouží pro verifikaci různých částí hry jako je například okamžité restartování hry

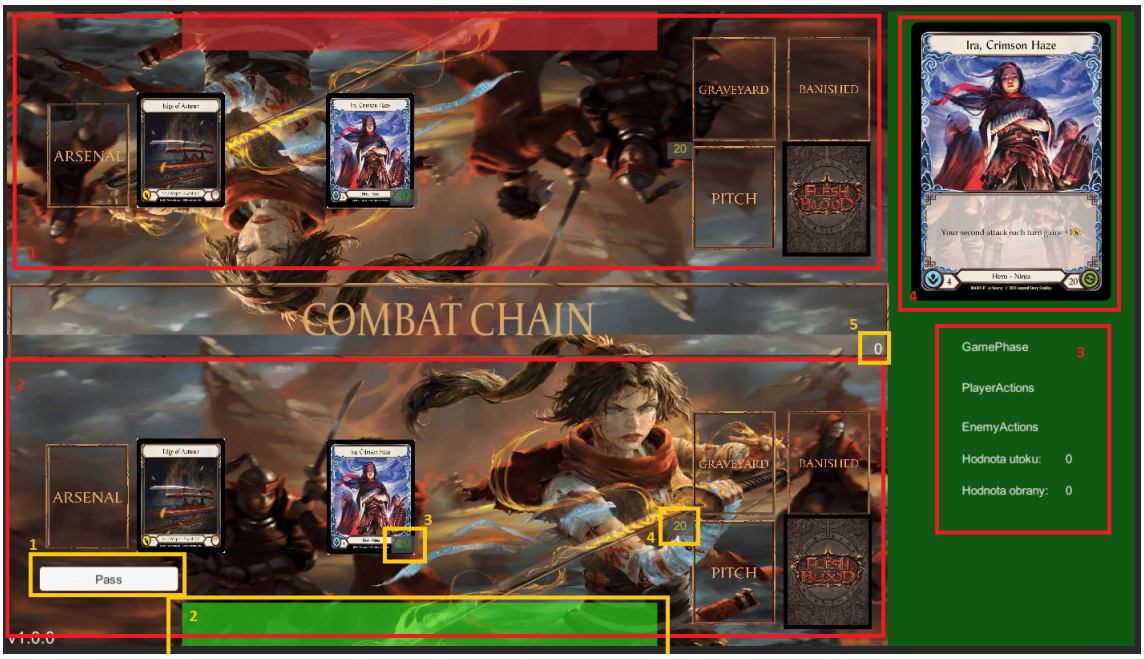
Z této scény lze ovládat oba hráče za pomocí tlačítek. Každé tlačítko slouží pro provedení požadované akce. Pro výběr zvolené karty souží dropdown menu. Toto menu obsahuje identifikátory karet, které se nacházejí na ruce daného hráče. Scéna dále obsahuje textová pole pro zobrazení aktuálních životů, hodnotu energii dostupných v příslušných *Pitch zónách* a označení různých fází hry.

Manager pro ovládání rozhraní je implementován v souboru *UITestHandle.cs*. Stejnojmenná třída *UITestHandle*, která dědi z Unity třídy *Monobehavioral*, si drží referenci na textová pole a na tlačítka pro prvního hráče. Je zde také reference pro herní desku aby bylo možné získávat informace o aktuálním stavu hry a ty následně vyobrazit. Tento manager má za úkol udržovat popisky aktuální a blokovat či odblokovávat tlačítka. Blokace tlačítek simuluje nedostupnost některých tahů během různých fází. Aktualizace textových polí a blokací tlačítek probíhá s každou aktualizací snímku.

Implementace funkčnosti jednotlivých tlačítek a dropdown menu se nachází v samostatných souborech. Metoda zodpovědná za provedení akce má vždy název *Execute*.

Scéna pro hru

Tato scéna reprezentuje digitální podobu hry v papírové podobě. Její rozhraní pak slouží pro interakci se scénou. Implementace tohoto rozhraní se skládá ze dvou částí. První je samotné ovládání hry a zobrazování informací uživateli. Druhou část tvoří grafická reprezentace jednotlivých karet. Na obrázku 6.8 lze vidět popis rozhraní.

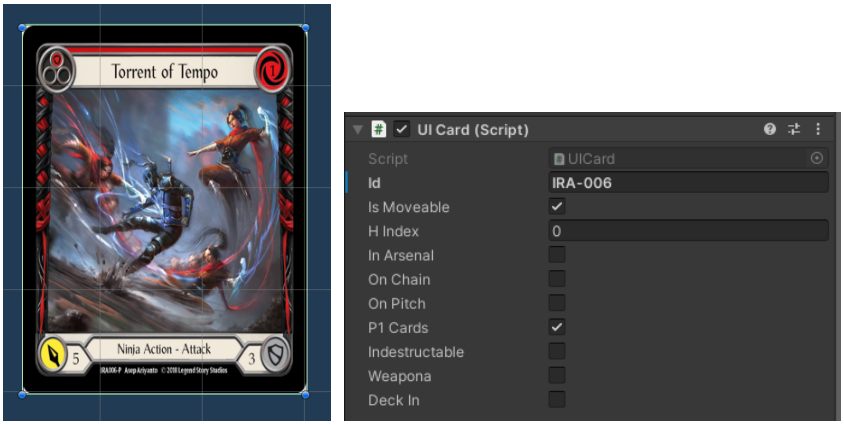


Obrázek 6.8: Herní scéna. Červeně jsou vyznačeny části patřící pod jeden celek a zobrazující důležité části průběhu hry. Červená jedna označuje část hrací desky, která patří počítačem řízenému protivníkovi. Červená dva je část desky náležící uživateli. Červená tři označuje část pro zobrazování informací, které si během papírové hry hráč musí udržovat v hlavě a nemají reprezentaci. Červená čtyři je část, kde se zobrazuje karta v plném rozlišení. Obrázek se mění v závislosti na pozici kursoru myši. Žlutě jsou pak označené prvky, které slouží pro informace, či interakce s hrou. Žlutá jedna představuje tlačítko pro předání priority oponentovi. Žlutá dva pak představuje hráčovu ruku. Žlutá tři je aktuální počet životů hrdiny. Žlutá čtyři je hodnota použitelné energie. Žlutá pět je počet akčních bodů aktivního hráče.

První část rozhraní je tvořena dvěma managery. První je využit manager z předešlé scény pro aktualizovaní textových polí ze souboru *UITestManager.cs*. Ostatní tlačítka, která jsou

použita v testovací zóně, ale nenacházejí se ve scéně, jsou svedena pod jedno tlačítko, které je mimo kameru. Tím je toto tlačítko nedostupné uživatelům a Unity nemá problém s referencí. Druhý manager má svoji implementaci v souboru *UIManager*. Tento manager má za úkol zpracovávat akce a ty následně zobrazit nebo také provést. Pro stranu oponenta manager přijímá informace o akcích od umělé inteligence a ty následně zobrazit uživateli ve scéně. Ze strany uživatele manager hlídá kolize miniatur karet s příslušnými zónami. Pokud by umístění karty znamenalo validní tah, invokuje se příslušná metoda ze strany pravidel s identifikátorem karty.

Druhou část tvoří implementace grafické reprezentace karet. Každá karta je reprezentována pomocí prefabrikátu a do hry je vždy vytvářena jeho instance. Obrázek prefabrikátu pro kartu a popis atributů je na obrázku 6.9.



Obrázek 6.9: Prefabrikát karty a jeho atributy. Vlevo se nachází prefabrikát otevřený v editoru Unity. Zeleně lze vidět kolizní box. Karta také obsahuje *Rigidbody 2D* pro rozeznávání kolizí se zónami. Těleso je nastavené na kinematické, čímž se sníží vytížení výpočtu fyziky pro každou kartu. Vpravo lze vidět nastavení atribut pro prefabrikát. Nejdůležitější je atribut *Id*, který se využívá pro identifikaci karty ve hře při kolizích a pro aktivování akcí. Další atributy drží informaci o pozici karty a její chování jako je možnost pohybu s danou kartou nebo zda se jedná o kartu která se nemá při restartu hry ničit.

Implementace chování pro grafickou reprezentaci karty je v souboru *UICard.cs*. V tomto souboru je implementován pohyb karty pomocí myši a změna zobrazení karty v plném rozlišení při najetí myši na vybranou kartu.

6.5 Komunikace mezi moduly

Jelikož všechny tři hlavní moduly jsou implementovány jako samostatné části je nutné mít způsob jak zajistit jejich interakci.

První způsob takovéto interakce je pomocí invokace metod cílového modulu. Tento způsob se využívá při komunikaci směrem k pravidlům hry. Pokud chce modul grafického uživatelského rozhraní dát informaci že byla určitá karta zahrána na vybranou zónu, invokuje metodu přidání karty pro vybranou zónu. Tyto metody jsou vždy veřejně přístupné a přijímají identifikátor karty. Stejným způsobem funguje komunikace umělé inteligence směrem do pravidel.

Pravidla komunikují s modulem grafického uživatelského rozhraní pomocí *Eventů*. Grafické uživatelské rozhraní ve svém manažeru implementuje odpovědi na tyto události. Pří-

kladem může být líznutí karty. Strana pravidel provede akci líznutí karty a pomocí události *OnDrawCard* z herního balíčku dá signál grafickému uživatelskému rozhraní o tom že byla líznuta karta o určitém identifikátoru. Identifikátory karty je pro všechny moduly vždy stejný. Stejným způsobem funguje komunikace ze strany umělé inteligence do Grafického uživatelského rozhraní. Tento přístup zajistí méně zátěže pro modul s herními pravidly.

Oproti návrhu pozorování ze strany pravidel do umělé inteligence není implementováno. Umělá inteligence musí tyto informace získávat sama když provádí rozhodování.

Kapitola 7

Trénování a testování

Po dokončení implementace pravidel hry a umělé inteligence se přešlo na trénování sítě, která by reprezentovala mozek agenta. Testování se provedlo po dokončení grafického uživatelského rozhraní a bylo prováděno pod dohledem.

7.1 Trénovaní

Samotné trénování probíhalo pomocí ML-Agents balíčku, kdy hra musela být připravena v editoru ke spuštění s nastavenými agenty. Bylo nutné správné pojemnování agentů pro vybrání správné konfigurace trénování. Pro tuto konfiguraci byl použit konfigurační soubor *trainingParam.yaml*. Trénování zajišťoval pythonovský script, který po své inicializaci čekal na spuštění hry v Unity engine. Během trénování bylo zařízení nepoužitelné, neboť při přepnutí okna obsahující editor se trénování pozastavilo. Průběh trénování šlo sledovat na grafech dostupných přes prohlížeč připojením na vybraný port a přes výpisy do příkazové řádky s informací o průměrné odměně za definovaný počet kroků.

Byla provedena celkem tři různá trénování. Hra probíhala formou zápasu dvou agentů ovládající *Ira Welcome deck*. Výsledkem je pět neuronových sítí.

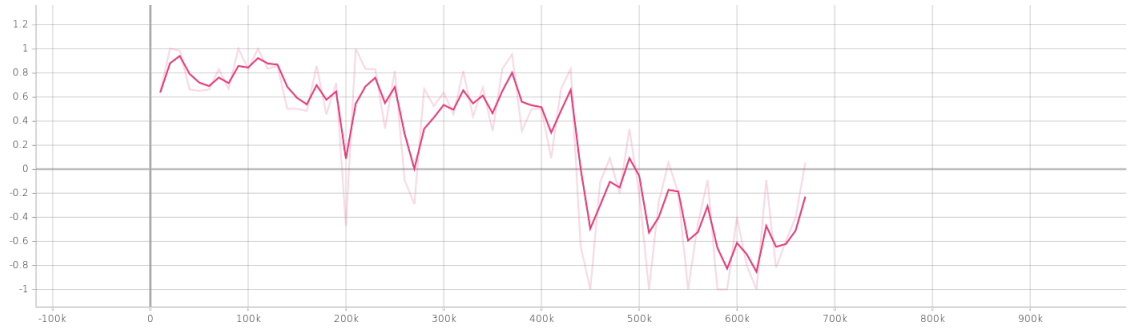
První trénování probíhalo hrou dvěma stejnými agenty proti sobě. Výsledkem byla jedna neuronová síť. Celkový počet kroků byl nastaven na padesát tisíc. Tento experiment spíše sloužil jako zjištění, zda se při trénování neobjeví nějaké chyby, nebo jestli se hra nedostane do nekonzistentního stavu. Proto byl zvolen menší počet kroků, neboť bylo zapotřebí hru neustále sledovat. Výsledná síť si za tuto dobu neutvořila žádné chování a její hraní bylo zcela náhodné.

Druhé trénování už probíhalo za hry dvou odlišných agentů se stejnou architekturou sítě a odlišným nastavením koeficientů. Ohodnocení bylo nastaveno kladně za výhru a záporně za prohru či remízu. Remíza nastala pokud počet kroků na kolo překročil hodnotu 1000 kroků. Trénování probíhalo na 5 milionu kroků. Výsledné sítě si vytvořili velice obraný postoj, kdy se velice často bránili útokům a docházelo velmi často k remízám.

Pro třetí sadu trénování byla provedena změna v ohodnocení agentů. Zůstalo kladné ohodnocení za výhru a záporné ohodnocení za prohru, ale hodnota byla vždy odstupňovaná dle zbývajících počtů životů výherce. Když došlo k těsné prohře nebyla ztráta kumulovaných bodu tak vysoká. Aby se předešlo pasivnímu chování, bylo za remízu vždy hodnocení maximální záporná hodnota odměny. Výsledné sítě se dosti lišily. První síť hrála velice defenzivně od začátku a spíše využívala útoku zbraně. Druhá síť naopak velice často útočila pokud měla možnost a snažila se zahrát několik útoků v kole. Naopak pokud se její hodnota

životů blížila více k nule zaujala obraný postoj. Tato síť nakonec byla vybrána jako mozek pro agenta který reprezentuje umělou inteligenci ve hře. Její chování představuje určitý vzor a není náhodné. Ovšem její chování není zcela ideální, neboť se stává že na malý útok použije většinu karet čímž blokuje výrazně více než by bylo nutné.

Graf kumulativní odměny agenta který představuje umělou inteligenci ve hře lze vidět na obrázku 7.1. Hodnota entropie je vyobrazena na obrázku 7.2.



Obrázek 7.1: Hodnota kumulativní odměny. Osa x představuje jednu epochu (odehranou hru). Osa y pak představuje hodnotu kumulativní odměny kde možné maximum bylo 1,2 a minimum -1,2. Světlá barva představuje surové hodnoty, tmavá pak hodnoty vyhlazené. První dva pády hodnot byli způsobeny větším počtem remíz v odehraných hrách. Větší propad v hodnotě kolem 450k odehraných her byl způsoben zvýšením defenzívy druhého agenta, kdy hry trvali déle, ale bez remíz.

Obrázek 7.2: Hodnota entropie. Z klesající hodnoty entropie v čase lze říci, že se síť postupně zlepšovala, ale její hodnota je ještě vcelku vysoká

Jelikož samotné trénování bylo velice náročné časově, kdy bylo nutné dokončit grafické uživatelské rozhraní, byl tento výsledek uznán za vhodný pro prezentování umělé inteligence k testování uživatelům. Pro lepší výsledky by bylo nutné více času, aby se dalo experimentovat s různým nastavením koeficientů.

7.2 Testování

Testování probíhalo na dvou skupinách uživatelů. První skupinu tvořili hráči se zkušenostmi se hrou a jsou znalí pravidel. Druhá skupina lidí byla tvořena lidmi bez zkušeností a znalostmi hry v papírové podobě.

Skupina zkušených hráčů nedostala žádné instrukce a byla jim prezentována hra s celým grafickým rozhraním. Při hraní hráči postupovali dle pravidel a prováděli validní kroky. Po dohrání byl každý tázán na:

- Nejsilnější stránku aplikace.
- Nejslabší stránku aplikace.
- Postřehy ohledně sílu a chování umělé inteligence.
- Vzhled grafického uživatelského rozhraní.
- Co jim ve hře chybí.

Nejsilnější stránkou bylo především přehledně dané jaká akce se dá hrát a to že hra hlídá neplatné tahy. Slabou stránkou byla absence historie tahů a pár chyb jako přetrvávání efektu z *Whirling Mist Blossom* a nefunkčnost efektu hrdiny. Umělá inteligence byla hodnocena celkem kladně, kdy měla určité náznaky strategie, avšak nebylo tak těžké vyhrát. Pro grafické uživatelské rozhraní zobrazovat více informací jako například aktuální hodnotu útoku a obrany, či počet akčních bodů. Dále pak nějaká informace proč byl proveden neplatný tah. Z posledního dotazu vycházelo že by bylo dobré mít více karet na hraní, i přidání zajímavých animací pro karty.

Skupina nezkušených hráčů dostala první hru bez vysvětlení pravidel jen pouze s radou že se hra ovládá pomocí přesunování karet na vyznačené zóny. Pravidla byla vysvětlena až pro druhou hru. Tento přístup se velice vyplatil, neboť nezkušení hráči zkoušeli různé karty z různých zón přesouvat na jiné zóny a tím objevit nestandardní chování. Žádný tah nezpůsobil pád hry, neboť ze strany pravidel bylo vše správně ošetřeno. Několik problémů nastalo s grafikou, kdy se karty špatně přichytávali na špatné zóny. Jednalo se hlavně o placení energie kdy se například karta reprezentující zbraň přichytila na *Pitch zónu*, což zapříčinilo že hráč posléze neměl možnost se zbraní zaútočit jelikož zbraň zůstala přichycena v této zóně.

Po ukončení několika her byli hráči tázáni na stejné otázky jako předešlá skupina s přidáním několika dalších:

- Jak pochopili různé části rozhraní.
- Co by jim pomohlo pro zdokonalení se ve hře.

Silnou stránkou byla často vybírána umělá inteligence a hlavně pro provedení několika po sobě jdoucích útoků. Slabší stránka byla absence nějakých tipů jako například které karty lze aktivovat. Grafické rozhraní by mohlo být více barevné, ale řešení karet bylo dobře provedené. Kladně byla hodnocena zobrazovaná karta v plném rozlišení po najetí na vybranou kartu. Pochopení různých částí nebyl větší problém až na význam různých fází hry a odůvodnění proč daný tah nelze provést. Pro zdokonalení by bylo dobré hlavně zobrazovat tipy co je možné hrát a nějaký tutoriál.

Kapitola 8

Závěr

Výsledkem práce je aplikace, které dává možnost hráčům zahrát si sběratelskou karetní hru *Flesh and Blood* proti umělé inteligenci. Je zde dostupný herní balíček *Ira Welcome deck*, který složí jako ideální vstup pro nové hráče. Z programátorského hlediska je zde možnost využití trénovací scény pro trénování neuronové sítě a k dalšímu rozšiřování hry. Modulárně programovaná pravidla lze jednoduše rozšiřovat bez zbytečných zásahů do už existující implementace. Přidávání karet za přidání pouze jejich implementace a vytvoření prefabrikátu pro grafickou reprezentaci, tvoří jednoduchý způsob jak hru rozšiřovat o nové karty. Výsledná implementace může sloužit pro další rozšiřování této aplikace. Umělá inteligence prezentovaná v aktuální verzi vykazuje známky určité strategie a její kroky nejsou zcela náhodné. Pro lepší výsledky bude potřeba více času pro trénování, které by bylo ideální proti různým balíčkům.

Aplikace představuje dobrý potenciál pro rozšiřování. Do hry 27.5.2022 přibude sada dvou herních balíčků ideálních pro začátečníky. Balíčky z tohoto produktu představují ideální možnost pro začlenění nových karet do této aplikace. Implementace těchto balíčků dobře podpoří možnosti v trénování neuronové sítě pro toto nové prostředí. Další plánované rozšíření je přidání ostatních existujících karet do hry a začlenit hru s vybavením do pravidel hry. Poté je potřeba do hry přidat možnost tvorby vlastních balíčku s dostupných karet a přidání dalších herních modů. Pro umělou inteligenci se zde otevírá prostor pro trénování agenta který by byl schopný hrát i sealed formáty hry, kdy si hráč staví balíček jako součást hry. V neposlední řadě inovovat grafiku aplikace o přidání animací a lepší modelování herní scény. Je zde také možnost implementace hru dvou hráčů online proti sobě.

Literatura

[1] *THE HISTORY OF MAGIC* [online]. Wizards of the Coast LLC [cit. 2022-01-12]. Dostupné z: https://magic.wizards.com/en/content/history.

[2] Copeland, B. *Artificial intelligence* [online]. Encyclopedia Britannica, 2022 [cit. 2022-05-10]. Dostupné z: https://www.britannica.com/technology/artificial-intelligence.

[3] DiGiovanni, A. a Zell, E. C. *Survey of Self-Play in Reinforcement Learning* [online]. Departments of Statistics and Mathematics University of Michigan, 2021 [cit. 2022-05-10]. Dostupné z: https://arxiv.org/pdf/2107.02850.pdf.

[4] Grad, L. Helping AI to play hearthstone using neural networks. In: *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*. 2017, s. 131–134 [cit. 2022-05-11]. DOI: 10.15439/2017F561. ISBN 978-83-946253-7-5. Dostupné z: https://ieeexplore.ieee.org/abstract/document/8104525.

[5] Luca, G. D. *Bias in Neural Networks* [online]. 2021 [cit. 2022-05-10]. Dostupné z: https://www.baeldung.com/cs/neural-networks-bias.

[6] Omer, B. a Ferda, A. Reinforcement Learning in Card Game Environments Using Monte Carlo Methods and Artificial Neural Networks. In:. Září 2019, s. 1–6 [cit. 2022-05-11]. DOI: 10.1109/UBMK.2019.8907235. ISBN 978-1-7281-3964-7. Dostupné z: https://ieeexplore.ieee.org/document/8907235.

[7] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. a Klimov, O. *Proximal Policy Optimization Algorithms* [online]. OpenAI, 2017 [cit. 2022-05-10]. Dostupné z: https://arxiv.org/pdf/1707.06347.pdf.

[8] Silva, A. R. da a Goes, L. F. W. HearthBot: An Autonomous Agent Based on Fuzzy ART Adaptive Neural Networks for the Digital Collectible Card Game Heartstone. *IEEE Transactions on Games*. 2018, sv. 10, č. 2, s. 170–181, [cit. 2022-05-11]. DOI: 10.1109/TCIAIG.2017.2743347. ISSN 2475-1510. Dostupné z: https://ieeexplore.ieee.org/document/8015141.

[9] Studio, L. S. *Flesh and Blood Comprehensive Rules v1.4* [online]. 2021 [cit. 2021-12-30]. Dostupné z: https://storage.googleapis.com/fabmaster/media/documents/Comprehensive_Rules_v1.4_PUBLISH.pdf.

[10] Technologies, U. *Unity ML-Agents Toolkit Documentation* [online]. 2021 [cit. 2022-05-14]. Dostupné z: https://github.com/Unity-Technologies/ml-agents/blob/release_2_verified_docs/docs/Readme.md.

[11] *Unity Documentation - Coroutines* [online]. Unity Technologies, 2022 [cit. 2022-01-14]. Dostupné z: https://docs.unity3d.com/Manual/Coroutines.html.

Příloha A

Odkazy na zdroje obrázků

2.2 https://www.cmus.cz/images/bgr.jpg
2.3 https://i0.wp.com/nextrift.com/wp-content/uploads/2021/12/hearthstone-return-alterac-2021-3.jpg?resize=1024%2C576&ssl=1
2.4 https://portal.matematickabiologie.cz/res/image/Umela%20inteligence/obr-4-5-model-neuronu.png
2.5 https://portal.matematickabiologie.cz/res/image/Umela%20inteligence/obr-4-4-usporadani-neuronu-do-vrstev.png
3.1 https://storage.googleapis.com/fabmaster/media/images/base_playmat.width-10000.png
3.2 https://storage.googleapis.com/fabmaster/media/images/U-WTR191.width-450.png
3.2 https://storage.googleapis.com/fabmaster/media/images/U-WTR192.width-450.png
3.2 https://storage.googleapis.com/fabmaster/media/images/U-WTR193.width-450.png
3.3 https://storage.googleapis.com/fabmaster/media/images/U-WTR39.width-450.png
3.4 https://storage.googleapis.com/fabmaster/media/images/U-WTR150.width-450.png
3.5 https://storage.googleapis.com/fabmaster/media/images/U-WTR115.width-450.png
3.6 https://storage.googleapis.com/fabmaster/media/images/U-WTR46.width-450.png
3.7 https://storage.googleapis.com/fabmaster/media/images/U-WTR107.width-450.png
3.8 https://storage.googleapis.com/fabmaster/media/images/U-WTR173.width-450.png
3.9 https://storage.googleapis.com/fabmaster/media/images/U-WTR121.width-450.png
3.10 https://storage.googleapis.com/fabmaster/media/images/U-WTR92.width-450.png
3.11 https://storage.googleapis.com/fabmaster/media/images/U-WTR225.width-450.png
4.4 https://storage.googleapis.com/fabmaster/media/images/IRA003-P_BrjByqK.width-450.png
5.2 https://raw.githubusercontent.com/Unity-Technologies/ml-agents/release_2_verified_docs/docs/images/learning_environment_example.png
6.5 https://storage.googleapis.com/fabmaster/media/images/ELE129.width-450.png

Příloha B

Obsah přiložené SD karty

Přiložená sd karta obsahuje krátké video s názvem *FaBDigital*, složku *FaBwithAI* obsahující celý projekt, složku *FleshAndBlood* obsahující spustitelnou hru pro PC a složku *Doc* obsahující textovou zprávu. Složka *FaBwithAI/Assets/Scripts* obsahuje zdrojové kódy a má následující strukturu.

CardDB Obsahuje implementaci samotných karet.

Game Obasahuje implementaci herních pravidel.

NeuralNetwork Obsahuje implementaci umělé inteligence.

Test Obsahuje implementaci grafického rozhraní pro testovací zónu.

UIs Obsahuje implementaci grafického uživatelského rozhraní hry.