Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra matematiky a informatiky

Diplomová práce

Návrh a realizace 3D hry

Vypracoval: Bc. Štěpán Mudra
Vedoucí práce: doc. Ing. Ladislav Beránek, CSc., MBA.
České Budějovice 2023

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Bc. Štěpán MUDRA
Osobní číslo:	E19125
Studijní program:	N6209 Systémové inženýrství a informatika
Studijní obor:	Ekonomická informatika
Téma práce:	Návrh a realizace 3D hry
Zadávající katedra:	Katedra aplikované matematiky a informatiky

Zásady pro vypracování

Cílem práce je návrh a realizace 3D hry, kde postavy budou mít určitou formu inteligence, založenou na implementaci rozhodovacího stromu případně jiné metody. Pro grafickou část budou využity předpřipravené balíčky, tzv. assety. Praktická část bude zahrnovat programování hry ve vhodném enginu, např. Unity. Součástí bude naprogramování inteligence postav pro rozhodování v různých situacích, navržené v teoretické části.

Metodický postup:

1. Studium odborné literatury.
2. Teoretický popis implementace inteligence v počítačových hrách, jejich porovnání.
3. Výběr vhodné metody pro inteligentní chování postav a výběr vhodného frameworku pro implementaci.
4. Naprogramování a implementace hry pomocí vybraného frameworku.
5. Vypracování doporučení a závěrů.

Rozsah pracovní zprávy:	50 – 60 stran
Rozsah grafických prací:	dle potřeby
Forma zpracování diplomové práce:	tištěná

Seznam doporučené literatury:

1. Bourg, D. M., & Seemann, G. (2004). *AI for game developers*. Sebastopol, CA: O'Reilly.
2. Hocking, J. (2018). *Unity in action: multiplatform game development in C#*. Shelter Island, NY: Manning Publications.
3. *Machine Learning* [online]. Dostupné z: <https://unity3d.com/machine-learning>.
4. Okita, A. (2019). *Learning C# programming with Unity 3D*. Second edition. Boca Raton, FL: CRC Press/Taylor & Francis Group.
5. *Unity for all* [online]. (2020). Dostupné z: <https://unity.com/>.

Vedoucí diplomové práce: doc. Ing. Ladislav Beránek, CSc.
Katedra aplikované matematiky a informatiky

Datum zadání diplomové práce: **17. ledna 2020**
Termín odevzdání diplomové práce: **16. dubna 2021**

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval/a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 14.4.2023 Podpis studenta

Poděkování

Rád bych poděkoval doc. Ing. Ladislavu Beránkovi, CSc., MBA. za vedení diplomové práce, konzultace problematiky a odborný dohled.

Obsah

1 Úvod a cíl práce

Důvod pro volbu tohoto tématu diplomové práce byl, že tvorba hry může zahrnovat širokou škálu oblastí, které je potřeba brát v patrnosti od samotného programování po rozhodovací modely. Toto téma tedy umožnilo propojení co možná nejvíce získaných znalostí. Někdy může zahrnovat i tvorbu dialogů postav, či vyprávění příběhu. V současné době totiž není výjimkou, že hry vyprávějí příběhy. Příběhy obecně pak mají pro lidstvo také velký význam viz sekce Příběhy a lidé.

Struktura této práce je rozvržena do několika kapitol. V této, úvodní, kapitole je popsán cíl práce společně s nástroji, které byly při tvorbě práce použity. Kapitola končí slovníkem pojmů a zkratek využívaných v této práci.

Kapitola Přehled řešené problematiky shrnuje teoretické poznatky o hrách nejen počítačových, ale i o deskových. Dále rozebírá roli příběhů a teorii ohledně umělé inteligence. Na dané problematiky nahlíží z různých úhlů pohledu.

V kapitole Metodika se práce zabývá převedením teoretických znalostí do praxe. Obsahuje tedy ustanovení funkcí počítačové hry společně s návrhy rozhodovacích modelů.

Kapitola Implementace obsahuje vlastní proces tvorby počítačové hry.

Závěr shrnuje výsledky práce.

1.1 Cíl práce

Cílem práce je návrh a realizace 3D hry, kde postavy budou mít určitou formu inteligence založené na implementaci rozhodovacího stromu, případně jiné metody. Pro grafickou část budou využity předpřipravené balíčky, tzv. assety. Praktická část bude zahrnovat programování hry ve vhodném enginu, např. Unity. Součástí bude naprogramování inteligence postav pro rozhodování v různých situacích navržené v teoretické části.

1.2 Využitý software

Tato sekce je věnována popisu jednotlivých programů, s jejíchž pomocí byla tato práce zhotovena. Některé sloužily pro psaní poznámek, jiné pro vytváření textu a další pro praktickou část, jako je programování a vlastní tvorbu hry.

1.2.1 Audacity

Tento software byl využit pro nahrání a modifikaci hlasových záznamů, které pak byly přiřazeny jednotlivým postavám.

1.2.2 Draw.io

Jde o software používaný pro tvorbu grafického zobrazení procesů, UML diagramů, nebo rozhodovacích stromů. V této práci byl využit právě pro vytvoření zobrazení rozhodovacích stromů.

1.2.3 Obsidian

Software Obsidian umožňuje vytvářet poznámky. Využívá značkovací jazyk Markdown, který umožňuje základní formátování. Poznámky lze třídit do složek a lze je exportovat do formátu PDF. Další funkce, kterou Obsidian nabízí, je graf vytvořený z poznámek. Každá poznámka pak vytvoří vrchol grafu a lze ji propojit s další poznámkou. Propojení lze vytvořit pomocí odkazu z jedné poznámky na druhou a je zobrazeno prostřednictvím hrany. Můžeme se odkazovat, ale i na příslušné části poznámky, například na podkapitoly označené druhou úrovní nadpisu. Poznámkám můžeme přiřadit i tzv. tagy. Ty vytváří další vrchol, na který jsou napojeny poznámky obsahující příslušný tag. Díky grafu si tak lze udělat přesnější představu o souvislostech jednotlivých témat. Někteří takto tvořené poznámky označují pojmem „druhý mozek“. Myslím si, že toto označení je celkem výstižné. Za předpokladu, že si jednotlivé poznámkové soubory, vrcholy grafu, představíme jako neurony v mozku a hrany pak jako synapse mezi neurony, tak skutečně můžeme jistou analogii najít. Obzvláště, pokud hrany „rostou“ s počtem odkazů. Můžeme tedy vidět asociace. S příslušným rozšířením lze vytvářet i tabulky a s využitím značkovacího jazyku LaTeX umí i matematické vzorce. Dalším, velmi užitečným rozšířením, je Excalidraw, které uživateli zprostředkuje jednoduché kreslení. Pro osobní využití lze program využívat zdarma. Existují i předplatná, které nabízí hlavně dřívější přístup k novým verzím a přístup do chatu s vývojáři. Pak existuje i verze předplatného pro komerční využití. Další možnost předplatného je synchronizace mezi zařízeními prostřednictvím účtu, která zahrnuje i historii verzí. Následující možnost předplatného slouží pak pro publikování.

1.2.4 Unity

Unity už je herní engine, který umožňuje primárně tvorbu her. Dají se však prostřednictvím tohoto software vytvářet i jiné aplikace nežli hry. Aplikace pak mohou být kompatibilní s celou řadou zařízení, protože Unity umožňuje tvorbu mobilních aplikací pro Android i iOS, v rámci počítačů pak umožňuje kompatibilitu s operačními systémy Windows, Mac OS, i Linux. Dokonce s prostřednictvím Unity lze vytvářet i webové aplikace.

Využití tohoto herního engine se však nemusí omezovat pouze na aplikace. Existuje i krátký seriál, který je vytvořen prostřednictvím Unity. Seriál se jmenuje Adam a je jej možné shlédnout prostřednictvím YouTube.

Unity pak umí pracovat s několika programovacími jazyky. Standartně je to programovací jazyk C# a JavaScript. K programování v C# Unity nativně využívá Visual Studio od společnosti Microsoft, ale je možné využít i Rider od společnosti JetBrains.

Základní vlastnosti a funkce herního engine Unity jsou pak stručně popsány v sekci Unity.

1.2.5 Visual Studio

Visual Studio sloužilo pro vytváření programu, C# scriptů. Bylo využito, jak již bylo zmíněno výše, pro svou úzkou provázanost s herním engine Unity. Nebylo tedy třeba nic dalšího nastavovat, ani stahovat další pluginy, které by spolupráci Unity s jiným IDE podporovaly. Nicméně v průběhu vývoje bylo Visual Studio vyměněno za Rider od společnosti JetBrains.

1.2.6 Rider

Během vývoje byl vyzkoušen i Rider pro vytváření scriptů. Na základě osobních preferencí byl dále používán tento nástroj místo Visual Studia.

1.3 Využité assety

- Basic Motions
- Free – 32 RPG Animations
- POLYGON Dungeons
- POLYGON Knights
- Ultimate Game Music Collection
- Universal Sound FX

1.4 Seznam pojmů a zkratek

1.4.1 Počítačové hry

- PC – hráčská postava, vychází z anglického *Playable Character*.
- NPC – nehráčská postava. Jde o jakoukoli postavu ve hře, která není ovládána hráčem. Vychází z anglického *Non-Playable Character*.
- FPS – jde o zkratku pro střílečky z pohledu první osoby (*First Person Shooter*).

- TPS – tato zkratka značí střílečku z pohledu třetí osoby (*Third Person Shooter*).
- RPG – zkratka pro hru na hrdiny (*Role Playing Game*).
- HP – zkratka pro zdraví postavy. Vychází z anglického *Hit Points*.

1.4.2 Genetické algoritmy

- Genom – jde o formální předpis daného jedince (Mařík & Štěpánková, & Lažanský, 2001)
- Jedinec – vyjádření konkrétního formální předpisu (genomu) (Mařík & Štěpánková, & Lažanský, 2001)
- Populace – soubor jedinců (Mařík & Štěpánková, & Lažanský, 2001)
- Selekce – proces, který umožní vybrat nejúspěšnější jedince v populaci prostřednictvím zdatnosti (Mařík & Štěpánková, & Lažanský, 2001) (Berka, 2003)
- Mutace – náhodná změna genomu při tvorbě nového jedince (Mařík & Štěpánková, & Lažanský, 2001)
- Křížení – proces vytvoření nového jedince z dvou předchozích (rodičů) (Mařík & Štěpánková, & Lažanský, 2001)
- Zdatnost – hodnota vyjadřující úspěšnost daného jedince v populaci (Mařík & Štěpánková, & Lažanský, 2001)
- Fitness funkce – funkce, která udává zdatnost jedince (Mařík & Štěpánková, & Lažanský, 2001)

1.4.3 Neuronové sítě

- Neuron – základní stavební jednotka neuronové sítě ať už umělé, nebo biologické (Berka, 2003) (Mařík & Štěpánková, & Lažanský, 2001)
- Axon – dlouhý výběžek z neuronu, který přenáší výstup neuronu (Berka, 2003) (Mařík & Štěpánková, & Lažanský, 1993)
- Dendrit – krátký výběžek neuronu (Berka, 2003) (Mařík & Štěpánková, & Lažanský, 1993)
- Tělo – část neuronu, která reaguje na vstupy a vysílá výstupy (Berka, 2003) (Mařík & Štěpánková, & Lažanský, 1993)
- Práh – minimální hranice, kterou musí nabýt výstupní funkce neuronu k jeho aktivaci (Berka, 2003) (Mařík & Štěpánková, & Lažanský, 1993)

- Excitace – aktivace daného neuronu v reakci na vstup (Berka, 2003) (Mařík & Štěpánková, & Lažanský, 1993)
- Inhibice – deaktivace neuronu v reakci na vstup (Berka, 2003)

2 Přehled řešené problematiky

Tato kapitola je zaměřena na shrnutí teoretických poznatků o všem, co může počítačová hra obsahovat.

2.1 Příběhy a lidé

Příběhy obecně hrají v historii lidstva velkou roli. Jordan B. Peterson roli příběhů rozvinul ve své knize *Mapy smyslu* v souvislosti s mýty a ideologiemi. Dle jeho názoru nás mají příběhy svým způsobem učit pravidla a předávat zkušenosti. Šíři a stáří příběhů pak přirovnává k hradbám. Starší příběhy přirovnává k hradbám města, protože nesou obecnější poselství/pravidlo, které chrání/ovlivňuje více lidí. (Peterson, 2023) Dále autor rozebírá téměř totožnou strukturu těchto starých příběhů. Hlavní aktéři příběhů často zastupují jeden ze třech pilířů. Pilíře jsou následující:

- Chaos – symbolizuje cokoliv neznámého. V některých příbězích, například *Epos o Gilgamešovi*, je chaos reprezentován divokou přírodou a Enkiduem. (Peterson, 2023) (Sedláček, 2017)
- Hrdina – hlavní aktér příběhu, který vyvolá změnu. (Peterson, 2023)
- Řád – symbolizuje striktní pravidla. Ve starších příbězích jej reprezentuje civilizace, velká města, ve kterých se lidé ukryjí před nástrahy přírody, ale musí dodržovat pravidla pro klidné soužití. (Peterson, 2023)

Vzorec těchto příběhů je ten, že s pomocí hrdiny se buď z chaosu tvoří řád, nebo hrdina řád svrhne (protože se z řádu stala tyranie), na nějakou dobu zavládne chaos, než hrdina vytvoří nový řád. (Peterson, 2023)

Podobný pohled na příběhy popsal i autor ve své knize *Sapiens*. Přičemž také vnímá příběhy jako velmi důležité pro lidstvo z velmi podobného důvodu. Tvrdí totiž, že lidská řeč se vyvinula pravděpodobně kvůli „drbům“, protože zvířata mají menší „slovní zásobu“ a přesto se dokážou varovat před nebezpečím a pokojně soužít v malých skupinách. Tvrdí tedy, že květnatost naší řeči nám skrze drby a potažmo i příběhy umožňuje vytvářet velké fungující celky městy počínaje a aliancemi států konče. Samozřejmě i zvířata, například mravenci nebo včely, dokážou spolupracovat v početných skupinách. Tam je toho dosaženo přesným určením rolí a genetickým přizpůsobením, kdyžto lidé nejsou geneticky přizpůsobeni pro život v početných společenstvích. Například opice jsou schopny udržet skupiny o několika desítkách členů. Jako horní hranici pro udržitelnost uvádí 150 jedinců

ve společenství. Nám lidem tedy tyto genetické předpoklady vynahrazuje řeč, příběhy a pravidla. Dále autor zavádí pojem kolektivní mýtus. S tímto termínem ztotožňuje například i spravedlnost, právní systém i firmy. Svou myšlenku vysvětluje tvrzením, že nic z toho reálně neexistuje, jen se všichni chovají, jako kdyby to existovalo. (Harari, 2018)

Vzorec příběhů, který definoval Peterson, můžeme pozorovat nejen v mytologických příbězích, ale i u příběhů současné kultury Jeden takový příklady by mohla být série *Star Wars*. Tato příběhová série je v současnosti zpracována několika způsoby, ať už filmově, seriálově, knižně či videoherně.

2.1.1 Příběhy v počítačových hrách

Příběhy mohou být součástí počítačových her. Nejsou však nutností. Příběhy obvykle chybí v počítačových hrách, které jsou počítačovou verzí některé deskové hry, například *Šachů*. Dalšími hrami, které nepotřebují příběh jsou pak některé vzdělávací hry, které jsou zaměřeny na zkoušení znalostí, například *Dobyvatel*. Nicméně mnoho počítačových her nějaký příběh obsahuje. Stejně tak i vzdělávací hry mohou obsahovat příběh. Jde například o hru *Attentat 1942*, což je výpravná historická hra z prostředí nacistické okupace. Hra je vyprávěna prostřednictvím svědectvích a historických záznamů. Hra vznikla na Matematicko – fyzikální fakultě a Filosofické fakultě Univerzity Karlovy ve spolupráci s Ústavem pro soudobé dějiny Akademie věd ČR.

Ačkoliv příběh počítačových her může být vymyšlený, tak existují hry, které smyšlený příběh zasazují do konkrétního historického prostředí. Jde například o starší díly herní série *Assassin's Creed*, které jsou zasazeny v historickém období, využívají i historické postavy a události, ale konkrétní vyprávěný příběh je smyšlený a nelze je považovat za věrnou kroniku dějin. Dalším příkladem hry zasazené do historie může být *Kingdome Come: Deliverance* od studia WarHorse.

Naproti tomu existují hry, které jsou kompletně zasazeny do smyšleného světa. Tento svět pak může existovat jen pro danou herní sérii, nebo se může opírat o sérii knih, či filmů. Existuje nespočet her z prostředí filmů *Star Wars*. Stejně tak mezi úspěšnými hrami nalezneme i takové, které čerpaly předlohu z knih. Jedna z velmi známých herních sériích čerpajících z prostředí knih je *Zaklínač*.

2.2 Hraní a hry

V souvislosti s hrami se můžeme setkat i s pojmem teorie her. Tento obor toho ale příliš společného s hrami a herním designem nemá. Zato více souvisí s matematikou, psychologií a své uplatnění najde i v oblasti ekonomie (Koster, 2013) viz sekce Teorie her.

Hraní ve smyslu klasického hraní, nikoliv ve smyslu hraní her, umožňuje lidem a nejenom jim, ale i dalším tvorům, naučit se reagovat na svět kolem sebe. Jde o to, že když si hrajeme, tak vlastně poznáváme, reakce okolí na naše chování. Získáváme tedy zpětnou vazbu na naši akci. Typicky jde o interakce typu tohle se stane, když udělám tohle. (Huberman, 2022)

2.2.1 Definice

V literatuře se můžeme setkat s několika definicemi hry. Nicméně většina z těchto definic zcela přehlíží podíl zábavy v této aktivitě. Přeci jen, pokud kdokoli nehraje hru pro zábavu, pak to z jeho subjektivního pohledu přestává být hra. Jeho chování se změní, dokonce i neurochemie jeho mozku bude jiná, než když hraje pro zábavu. (Huberman, 2022)
Definice hry dle literatury:

- Roger Callois – „*aktivita, která je dobrovolná, nejistá, neproduktivní, řízená pravidly, předstíraná*“ (Koster, 2013)
- Johan Huizinga – „*volná činnost mimo ‚všední‘ život*“ (Koster, 2013)
- Jesper Juul – „*Hra je formální systém založen na pravidlech s proměnnou a kvantifikovatelným výstupem, kde jsou rozdílné výstupy přeřazeny různým hodnotám. Hráč vynakládá úsilí za účelem ovlivnění výsledku. Hráč se cítí vázán na výsledek a důsledky činnosti jsou volitelné a lze o nich diskutovat*“ (Koster, 2013)

Takto definují hry game designéři:

- Sid Meier – „*série smysluplných rozhodnutí*“ (Koster, 2013)
- Katie Salen a Eric Zimmerman – „*systém, ve kterém hráči čelí umělému konfliktu, který je definován pravidly a ústí v kvantifikovatelný výsledek*“ (Koster, 2013)
- Ernest Adams a Andrew Rolldings – „*jedna, nebo více kauzálně propojené série výzev v simulovaném prostředí*“ (Koster, 2013)

Ačkoliv tato práce využívá definice hry v kontextu zábavním, je třeba si uvědomit, že pojem hra se vyskytuje i ve zcela odlišných kontextech. Ekonomie definuje hru jako

konflikt mezi hráči, kteří se snaží zvítězit prostřednictvím strategických interakcí. (Hořejší & Soukupová & Macáková, & Soukup, 2018)

Dokonce i v kontextu psychologie/psychiatrie se můžeme setkat s pojmem hra. Pro tento kontext hru definoval Eric Berne ve své knize *Jak si lidé hrají* jako „*Souvislý sled druhotných doplňkových transakcí, jež směřují k jasně definovanému, předem známému výsledku. Dá se vyjádřit jako obměňující se soustava často opakovaných, zdánlivě racionálních transakcí se skrytou motivací; anebo populárněji jako řada tahů s různými léčkami nebo fintami.*“ (Berne, 2011)

Všechny tyto definice ať už pocházejí od akademických pracovníků, nebo od game designérů, mají jednu věc společnou. Totiž, že hru označují jako simulaci, nebo jiný svět, který je naprosto separován od reálného světa. V tomto oddělení se definice ukazují jako pravdivé jen částečně. I když se události ve hře dějí jen „jako“ mozek i tělo na ně reaguje velmi podobně, ne-li stejně, jako kdyby se děly ve skutečnosti. Dokonce existují zmínky, že pouhé představování si dané činnosti vyvolá v mozku stejné reakce, jako kdyby danou činnost prováděl. Sice představováním si zdvihání činek nezískáme svalovou hmotu, ale představování si podání při tenisu, nebo odpal míčku v golfu může zlepšit naši techniku při příštím pokusu. (Koster, 2013) (Brown & Roediger, & McDaniel, 2017) Tato myšlenka byla i zdokumentována v dokumentární sérii „Tělo nezná hranic s Chrisem Hemsworthem“ od National Geogrhraphics, kde herci připevnili na hlavu i na tělo elektrody a prostřednictvím virtuální reality se procházel po úzké plošině ve výšce asi 300 metrů nad zemí. A opravdu, jeho tělo reagovalo, jako kdyby prožitek byl skutečný. Jeho tep se zvýšil, začal se potit a v mozku se aktivovala stejná centra, jako kdyby situaci zažíval v reálném světě. Samozřejmě zde mohl být efekt posílen virtuální realitou.

Některé hry vyžadují více fantazie, jiné méně. Nicméně hry jsou s námi, jako živočišným druhem spojeny již několik tisíciletí. Jejich účel je kulturní, ale i biologický. V každém z těchto účelů nás ale hry v jistém slova smyslu učí. (Koster, 2013) (Huberman, 2022) (Harari, 2018) Z pohledu kultury, nás i hra na policii a zloděje může učit, že špatný skutek (krádež) bude potrestán a že dobro vítězí.

Hra na schovávanou nás učí hledat vhodný úkryt pod časovým tlakem, sebeovládání, jako je nehybná pozice, když hledající hráč jde okolo a hledá nás, nebo i časování kdy můžeme vyjít ze skrýše, aniž bychom byli zpozorováni. Skákáním panáka zase zlepšujeme naše dovednosti v rovnováze na jedné noze. Dalo by se tedy říci, že dětské hry nás učí se

správně hýbat a ovládat svoje tělo. Podobně, jako si koťata hrají, bez úmyslu jedno druhému ublížit, tak takto získané dovednosti později využívají při lovu.

2.2.2 První hry

Při zkoumání dávných her se nejčastěji vychází z archeologických nálezů. Nejstarší takovýto nález je starý přibližně 4900 let. Zde je seznam nejlepších deseti historických deskových her dle Britského muzea.: (The British Museum, 2021)

- Královská hra z města Ur
- Šachy
- Wari
- Senet
- Mahjong
- Hra husy
- Ajaxova a Achillova hra v kostky
- Sugoroku
- Pachisi
- Mehen

2.2.3 První videohry a jejich stručná historie

V této práci pojmem „videohra“ rozumějme kteroukoliv elektronickou hru, kterou lze hrát prostřednictvím počítače, herní konzole, videoherního automatu atd. Nejde tedy jen o hry, které lze hrát jen a pouze na počítači.

První vznikající videohry čerpaly inspiraci z deskových her, sportu, či vojenstvím. Tyto typy her nevyžadovaly příliš náročné výpočty, proto vyhovovaly technickým možnostem. V roce 1951 byl představen Nimrod, specializovaný stroj pro hraní hry Nim. („Nimrod, the World's First Gaming Computer”, 2010) O rok později byla představena hra OXO. Ta považována za jednu z nejstarších videoher pro počítač a vůbec první videohru umožňující hraní piškvorek formátu 3x3. Hraní zprostředkovával počítač Cambridge EDSAC a vytvořil ji Alexander Douglas. Roku 1955 zas byl představen Hutspiel, který simuloval válku mezi USA a SSSR. 1958 byla představena hra, která ke svému zobrazení využívala osciloskop a ovládala se pomocí potenciometrů. Šlo o hru Tenis for Two. Vytvořil ji William Higinbotham pro zábavu. V roce 1962 vyšla první válečná hra. Šlo o hru Spacewar!,

která simulovala pohyb dvou vesmírných lodí v gravitačním poli hvězdy, které po sobě střílely. (Alza, n. d.)

Ačkoliv videoherní historie sahá téměř sedmdesát let do minulosti, tak trvalo více než deset let, než se videohry staly dostupné široké veřejnosti. Zásluhu na tomto mají velké arkádové (videoherní) automaty. Ty umožňovaly hrát vybranou hru po vhození určitého peněžního obnosu. Ty se objevili v osmdesátých letech minulého století. 1971 to byl videoherní automat Galaxy Game, který umožňoval hrát verzi Spacewar!. Většího přijetí se videoherní automaty dočkaly o rok později, v roce 1972, kdy firma Atari vydala videoherní automat pro hru Pong. Začaly vznikat videoherní herny, kam si lidé mohli přijít zahrát tyto arkádové hry. (Alza, n. d.) Mezi nejznámější arkádové hry patří Pac-Man, Donkey kong a Space invaiders. Tyto tři hry byly i využity ve filmu PIXELY z roku 2015.

Arkádové stroje nebyly však jedinou možností, jak tyto videohry hrát. Začaly vznikat i mnohem menší tzv. herní konzole, které umožňovaly hrát arkádové hry z pohodlí domova. Tyto herní konzole se nijak zvlášť nelišily od dnešních videoherních konzolí. Jejich balení také obsahovalo ovladače a konzole se připojily k televizi. (Alza, n. d.)

2.2.4 Žánry počítačových her

V prostředí počítačových her rozlišujeme několik hlavních žánrů. Nicméně hry nemusí výlučně spadat pouze do jednoho z nich, ale mohou být kombinovány.

- Adventury – pro tento žánr je typické, že hráči jsou předkládány úkoly, nebo hádanky a prostřednictvím jejich plnění se pak hráč posouvá hrou dál. (BASLER, 2016) Do tohoto žánru by pak spadaly hry jako *Deponia*, *Polda*, nebo *Tomb Raider*.
- Akční – zástupci tohoto žánru jsou především střílečky a účelem hry nejčastěji bývá prostřílet se skrze protivníky. Tyto akční střílečky lze dále rozdělit na FPS a TPS. (BASLER, 2016) Pro PFS pak můžeme uvést jako zástupce *Far Cry* a pro TPS hru *Mafia*.
- Bojové – jde o hry, ve kterých proti sobě nastoupí dvě postavy, obě mohou být ovládány hráči, v aréně a smyslem hry je zvítězit v souboji v určitém bojovém umění. (BASLER, 2016) Do tohoto žánru pak patří hry jako *Mortal Kombat*, *Tekken*, nebo *Soulcalibur*.

- Hudební – jde o hry, které hráče učí hrát na hudební nástroj, nebo tančit. (BASLER, 2016) Typickým zástupci hudebních her jsou *Guitar Hero*, nebo *Just Dance*.
- Logické – hry tohoto žánru předkládají hráči logické hlavolamy, které musí vyřešit. Obtížnost hlavolamů se zpravidla stále stupňuje. (BASLER, 2016) Tato kategorie zahrnuje například *Tetris, Angry Birds*, nebo *Portal*.
- RPG – tento videoherní žánr umožňuje hráči prozkoumávat svět, který hra obsahuje čímž hráč potkává protivníky, které musí na své cestě porazit, nebo jiná NPC, která pro hráče mají úkoly a za jejich splnění získá odměnu. Odměnou mohou být různé předměty, které hráči umožní vylepšení stávající výzbroje, nebo mu propůjčí magické schopnosti, ale odměna může mít i podobu určitého obnosu videoherní měny. (BASLER, 2016) Jako klasické RPG hry mohou být považovány *World of Warcraft*, *The Elder Scrolls V: Skyrim*, nebo série *Zaklínač*.
- Simulátory – jak již z názvu vyplívá, simulátory si dávají za cíl co nejvěrohodněji zobrazit určitý jev, nebo činnost a pak jím provést hráče. (BASLER, 2016) V tomto žánru můžeme nalézt hry jako *The Sims*, nebo *Farming simulator*.
- Sportovní – v těchto hrách se hráč ujme role sportovců/sportovce a snaží se v daném sportu vyhrát. Avšak nejde o přesné simulace, protože to by ztěžovalo hratelnost. (BASLER, 2016) Do sportovního žánru videoher řadíme například hru *FIFA*.
- Strategické – při hraní těchto her hráči uplatňují strategické myšlení. Takovéto hry lze dělit ještě do několika subžánrů. Hráč se stará o určité město, nebo zemi, kterou musí spravovat a rozvíjet ať už po ekonomické, vojenské, či technologické stránce. Budovatelské strategie simulují chod určitého města, nebo státu. Můžeme zde najít zástupce jako *SimCity*, nebo *Civilization*. Dále pak existují tahové strategie, které jsou velmi blízké deskovým hrám. Hráči mají omezené akce, které mohou v rámci svého tahu provést, přičemž čas na tah bývá zpravidla neomezený. Klasickými zástupci tahových strategií je pak herní série *Heroes of Might and Magic*. Dalším typem strategických her pak je real – time strategie, jde o strategické hry, ve kterých hráči hrají paralelně bez nutnosti využívat tahy. (BASLER, 2016) Jako příklady tohoto subžánru lze uvést herní sérii *Age of Empires*, nebo hru *Warcraft*.

- Vzdělávací – tyto hry, jak již žánr napovídá si kladou především za cíl hráče vzdělat. (BASLER, 2016)
- Závodní – cílem pro hráče těchto her je dostat se do cíle dříve než ostatní. (BASLER, 2016) V tomto žánru můžeme nalézt hry jako *Need for Speed*.

2.3 Umělá inteligence

Již v 17. století si filozofové kladli otázku, zda mohou stroje myslet. Jejich úvahy však zůstávali jen ve filozofické oblasti. Fakticky nevymysleli žádný postup, jak myslící stroj sestrojit. Teprve ve 30. letech minulého století se objevily první významné pokroky v souvislosti s umělou inteligencí, a to zejména díky matematikovi K. Godela a jeho práci v oblasti matematické logiky a teorii algoritmů. Po 2. světové válce výpočetní technika zaznamenala prudký vývoj, čímž bylo umožněno některé tyto navrhované postupy ověřit. Postupem času byly navrhovány stále komplexnější a složitější postupy a algoritmy, které umožňovaly strojům určité inteligentní chování. Nicméně všechny tyto postupy čerpají ze stejné inspirace a tou je příroda. Některé však využívají analýzu živých organismů. To jsou například neuronové sítě, genetické algoritmy, mravenčí kolonie atd. Další využívají matematickou abstrakci fungování lidského mozku vycházejícím ze zkoumání psychologie a kognitivity. Jde o metody reprezentace a využívání znalostí, metody učení založené na modelech atd. Pro chování strojů, které alespoň trochu připomíná inteligentní chování živého organismu se vžil termín umělá inteligence. (Mařík & Štěpánková, & Lažanský, 1993)

2.3.1 Definice

Pokusů o definici umělé inteligence bylo mnoho. Jen na 8. mezinárodní konferenci IJCAI v roce 1983 bylo předloženo téměř 200 pokusů o definici umělé inteligence. Problém s vytvořením definice pro pojem umělá inteligence spočívá v tom, že doposud není nikde přesně vymezen pojem inteligence ani pro živé organismy. Sice existují metody pro měření některých aspektů, například IQ test. Jenže žádný test není zcela objektivní. (Mařík & Štěpánková, & Lažanský, 1993) Navíc většina těchto testů, ne-li všechny, měří aktuální stav lidského mozku, jenže díky neuroplasticitě, což je schopnost vytvářet nová nervová spojení a některá starší rušit se mozek může měnit. Díky rušení nervových spojení zapomínáme a naopak, když se vytváří nová spojení, tak se učíme. Ať už jde o znalost slov cizího jazyka, vzorců pro matematické výpočty, nebo nový druh pohybu (styl plavání,

jízda na lyžích, malování). Všechny tyto činnosti vyžadují určitá nervová spojení a pokud naše aktuální nejsou dostačující, musí se vytvořit.

- Minského definice – Minsky při tvorbě definice umělé inteligence vychází z Turingova testu. Ještě v roce 1993 neexistovala taková umělá inteligence, která by byla schopna tímto testem projít. Pro takovouto umělou inteligenci se užívá termín silná umělá inteligence. (Mařík & Štěpánková, & Lažanský, 1993) Úroveň umělých inteligencí se však stále zvyšuje. Příkladem mohou být tzv. coboty neboli kooperativní roboty. Ty se využívají hlavně v továrnách a jejich princip zaučení se na danou pozici vypadá velmi podobně, jako zaučování lidského kolegy. Cobot totiž pozoruje svého lidského mentora při vykonávání určitého úkolu prostřednictvím senzorů, které má člověk na sobě připevněny a analyzuje daný úkon. Tím si osvojí základní postup a reakce na neobvyklé situace, až dokáže daný úkon vykonávat zcela samostatně. (Zandl, 2022) Pokud tedy lidské chování při výkonu úkonu považujeme za inteligentní, dle Turingova imitačního testu můžeme považovat stejné chování stroje při stejném úkonu také za inteligentní.
- Definice Richové – tato definice spíše definuje umělou inteligenci jako obor než jako pojem. Definice zní takto: „*Umělá inteligence se zabývá tím, jak počítačově řešit úlohy, které dnes zatím zvládají lidé lépe*.“ (Mařík & Štěpánková, & Lažanský, 1993)
- Kotkova definice – chápe umělou inteligenci jako vlastnost technických systémů a definuje ji takto: „*Umělá inteligence je vlastnost člověkem uměle vytvořených systémů vyznačujících se schopností rozpoznávat předměty, jevy a situace, analyzovat vztahy mezi nimi a tak vytvářet vnitřní modely světe, ve kterých tyto systémy existují a na tomto základě pak přijímat účelná rozhodnutí, za pomoci schopností předvídat důsledky těchto rozhodnutí a objevovat nové zákonitosti mezi různými modely nebo jejich skupinami.*“ (Mařík & Štěpánková, & Lažanský, 1993)

Inspirace pro většinu umělých inteligencí vychází z přírody. Ať už jde o optimalizační algoritmy na bázi mravenčí kolonie, neuronové sítě, nebo optimalizace hejn částic. Předlohu mají stejnou a tou je příroda. (Mařík & Štěpánková, & Lažanský, 1993)

V souvislosti s inspirací přírodou vyvstává otázka, co znamená pojem inteligence pro živé organismy. Chování živých organismů se řídí základními potřebami pro přežití a přenos své genetické informace dál. To znamená, že živý tvorové se snaží nasytit, utéct, nebo

čelit bezprostřednímu nebezpečí a rozmnožit se. Pokud se jim to podaří přežijí a předají svou genetickou informaci dál.

2.3.2 Rozhodovací modely

Rozhodovací modely slouží pro rozhodování se v určité situaci. Každý rozhodovací model obsahuje:

- Množinu možných rozhodnutí – veškerá rozhodnutí, která jsou v dané rozhodovací situaci přípustná. (Mařík & Štěpánková, & Lažanský, 2001)
- Množinu výsledků – jde o všechny možné výsledky, které rozhodovací situace dovoluje. (Mařík & Štěpánková, & Lažanský, 2001)
- Výsledkovou funkci – výsledková funkce přiřadí každému rozhodnutí strukturu z množiny výsledků. (Mařík & Štěpánková, & Lažanský, 2001)
- Preferenční relaci – slouží rozhodovateli k ohodnocení a seřazení možných výsledků podle preferencí od nejvíce preferované po nejméně preferovanou. (Mařík & Štěpánková, & Lažanský, 2001)

Jednokritériové

Jde o modely využívané pro nejjednodušší rozhodovací situace. Rozhodnutí je tvořeno na základě jediného kritéria. Rozlišujeme několik typů: (Mařík & Štěpánková, & Lažanský, 2001)

- Rozhodování za určitosti – každému rozhodnutí je přiřazen jediný výsledek, rozhodovatel vybírá rozhodnutí vedoucí k nejlepšímu výsledku. (Mařík & Štěpánková, & Lažanský, 2001)
- Rozhodování za rizika – jde o situace, při které je každému rozhodnutí přiřazeno několik výsledků. Každý výsledek pak má svou pravděpodobnost, že nastane. (Mařík & Štěpánková, & Lažanský, 2001)
- Rozhodování za nejistoty (neurčitosti) – v tomto případě jsou každému rozhodnutí přiřazeny výsledky, ke kterým může rozhodnutí vést. Avšak rozhodovatel nemá informaci o tom, s jakou pravděpodobností, který výsledek nastane. Pro tuto rozhodovací situaci se často volí metoda minmaxu. (Mařík & Štěpánková, & Lažanský, 2001)

Vícektritériové

Pro takovéto rozhodování rozhodovatel sleduje několik kritérií. Rozhodovatel se snaží o zlepšení dle každého z kritérií. Jenže některá kritéria mohou jít i proti sobě tzn. zlepšením se podle jednoho z kritérií se rozhodovatel může zhoršit podle jiného z nich. V literatuře se rozlišuje několik přístupů: (Mařík & Štěpánková, & Lažanský, 2001)

- Lexikografický – neboli slovníkový, využívá uspořádání jednotlivých kritérií dle důležitosti a následně volí rozhodnutí, podle kritéria s nejvyšší prioritou. Pokud kritérium splněno není, postupuje na další kritérium v pořadí. (Mařík & Štěpánková, & Lažanský, 2001)
- Vektorová optimalizace – respektuje všechna kritéria současně (Mařík & Štěpánková, & Lažanský, 2001)
- Agregační – k tomuto je potřeba znát hodnoty užitkových funkcí jednotlivých kritérií. Poté rozhodovatel sestrojí agregovanou funkci užitku. Vytvořením této užitkové funkce dojde k převedení rozhodování z vícekritériového na jednokritériové. V této agregované funkci užitku jsou hodnoty citlivé na hodnoty dílčích užitkových funkcí. (Mařík & Štěpánková, & Lažanský, 2001)

2.3.3 Teorie her

Jak již bylo řečeno, tato oblast je známá napříč několika vědními obory. Teorie her předpokládá znalost důsledků provedené činnosti. Některé metody teorie her lze uplatnit i při tvorbě umělé inteligence počítačových her. V tomto ohledu jsou dané metody vhodné spíše pro deskové, nebo tahové počítačové hry a už méně vhodné pro akční střílečky. Hráči vybírají akci, která nabízí nejlepší možný důsledek.

Výplatní matice

Můžeme se setkat i s pojmem „normální forma zápisu výsledků“. Jde o tabulku, ve které se zobrazují výsledky alternativních strategií hráčů. Předpokládá se, že hráči jsou schopni uspořádat výsledky podle preferencí a jednají racionálně, tj. chtějí dosáhnout co nejlepšího výsledku. (Hořejší & Soukupová & Macáková, & Soukup, 2018)

Rozhodovací strom

Také jde o formu zápisu výsledků, někdy se rozhodovacímu stromu říká „extenzivní forma zápisu výsledků“. Jde tedy o alternativu k výplatní matici. Rozdíl je však v tom, že se lépe hodí pro sekvenční hry, protože je v něm lépe vidět chronologičnost jednotlivých rozhodnutí. (Hořejší & Soukupová & Macáková, & Soukup, 2018)

Vězňovo dilema

Jde o typický příklad uplatnění teorie her. Vězňovo dilema předkládá důsledky možností spolupráce/nespolupráce s policií pro dva lupiče. Jinými slovy, jsou známy důsledky jak při spolupráci s policisty, tak pro nespolupráci. Vězni společně spáchali trestný čin a jsou vyslýcháni samostatně, takže ani jeden neví, jestli druhý spolupracuje či nikoliv. (Hladký, & Leitmanová, 2000)

Tabulka 1 zobrazuje vězňovo dilema pomocí výplatní matice. Výsledné tresty, v měsících, jsou od sebe barevně rozlišeny. Výplatní matice tedy říká, že pokud se rozhodnou oba vězni spolupracovat, jejich trestem bude 7 měsíců. Pokud ale vězeň 1 se rozhodne spolupracovat a vězeň 2 nikoliv, trest pro vězně 1 nebude žádný a pro vězně dva bude trest 17 měsíců. Pro případ, že vězeň 1 nespolupracuje a vězeň 2 spolupracuje budou tresty obráceně. Pokud však nebude spolupracovat ani jeden z vězňů, budou mít trest stejný, a to tři měsíce.

		Vězeň 2	
		Spolupracovat	Nespolupracovat
Vězeň 1	Spolupracovat	7 / 7	17 / 0
	Nespolupracovat	0 / 17	3 / 3

Tabulka 1: vězňovo dilema – výplatní matice

2.3.4 Genetické algoritmy

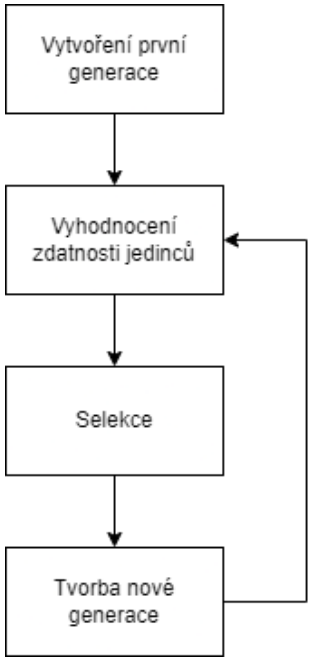
Jsou součástí evolučních výpočetních technik. Jde o nejčastější využívanou formu EVL. Do EVL rovněž patří:

- Evoluční strategie
- Genetické programování

Genetické algoritmy jsou výrazně inspirovány přírodou. Veskrze vychází z Darwinovi Evoluční teorie. Řadí se mezi tzv. optimalizační algoritmy. Což znamená, že jejich účelem je zjistit nejlepší, optimální, nastavení řešitele určitého problému. Genetické algoritmy samy o sobě **neřeší** zadanou úlohu. (Mařík & Štěpánková, & Lažanský, 2001)

Průběh genetického algoritmu pak vyjadřuje Obrázek 1. Prvním krokem je vytvoření první generace. Ta bývá zpravidla tvořena náhodně. Následuje část, kdy se jedinci chovají dle svého genomu a za úspěchy se jim zvyšuje jejich zdatnost /fitness. Proces selekce

vybere na základě zdatnosti nejúspěšnější jedince, které předá pro tvorbu vhodné generace. Tvorba nové generace pak v přírodě může probíhat několika způsoby, ale i jejich kombinací. Jedinci pro následující generaci jsou tvořeni buď předáním původního genomu dál (probíhá jen u vybraných organických druhů), křížením a mutacím (sama o sobě neprobíhá, proběhne buď jako součást přenosu genomu do další generace, nebo jako součást křížení). Mutace slouží jako prvek náhody. U tvorby jedince pro novou generaci můžeme v počítačem simulovaném prostředí využít náhodnou tvorbu jedince, jako tomu bylo u tvorby jedinců pro první generaci. (Bourg, & Seemann, 2004)



Obrázek 1: Průběh genetického algoritmu (Bourg, & Seemann, 2004)

Pro formální vyjádření jedince slouží tzv. genom. Jde o předpis, díky kterému je možné vytvořit příslušného jedince. Informace v genomu mohou být zapsány binárně, abecedně, ale i prostřednictvím reálných čísel. (Mařík & Štěpánková, & Lažanský, 2001)

Proces křížení jedinců může probíhat pomocí jednoho, nebo více bodů. Bod, nebo body určují, kde v genomu dojde k přehození předka, od kterého nový jedinec dědí. (Bourg, & Seemann, 2004) Obrázek 2 ilustruje průběh jednobodového křížení dvou jedinců. Délka jejich genomu je rovna osmi a bod je umístěn za pátým elementem genomu. Prvních pět elementů tedy nový jedinec (C) zdědí od jedince A a zbývající tři od jedince B.

Genom jedince A	Genom jedince B	Genom jedince C
1	0	1
1	0	1
1	0	1
1	0	1
1	0	1
1	0	0
1	0	0
1	0	0

Obrázek 2: Křížení jedinců v jednom bodě

Křížení probíhající prostřednictvím dvou bodů zobrazuje Obrázek 3. Délka genomu jedinců je zde, stejně jako u předchozích, rovna osmi. První bod se nachází za čtvrtým elementem genomu a druhý pak za pátým. Jedinec C tedy první čtyři elementy získá od jedince A, pátý element od jedince B a zbývající tři od jedince A.

Genom jedince A	Genom jedince B	Genom jedince C
1	0	1
1	0	1
1	0	1
1	0	1
1	0	0
1	0	1
1	0	1
1	0	1

Obrázek 3: Křížení jedinců ve dvou bodech

Mutace je pak čistě náhodná a může se projevit při tvorbě nového jedince ať už je tvořen křížením viz Obrázek 4, nebo přechodem jedince do další generace viz Obrázek 5.

Genom jedince A	Genom jedince B	Genom jedince C
1	0	1
1	0	1
1	0	0
1	0	1
1	0	1
1	0	0
1	0	0
1	0	0

Obrázek 4: Mutace v průběhu křížení

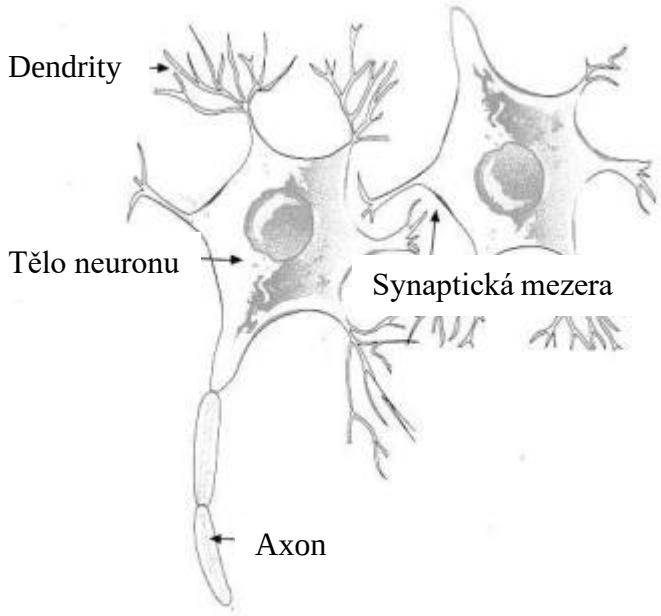
Obrázek 5 zobrazuje náhodnou mutaci jedince, který se přesouvá do další generace bez zamýšlených změn.

Genom jedince A	Genom jedince B
1	1
1	1
1	0
1	1
1	1
1	1
1	1
1	1

Obrázek 5: Mutace při přechodu jedince do další generace

2.3.5 Neuronové sítě

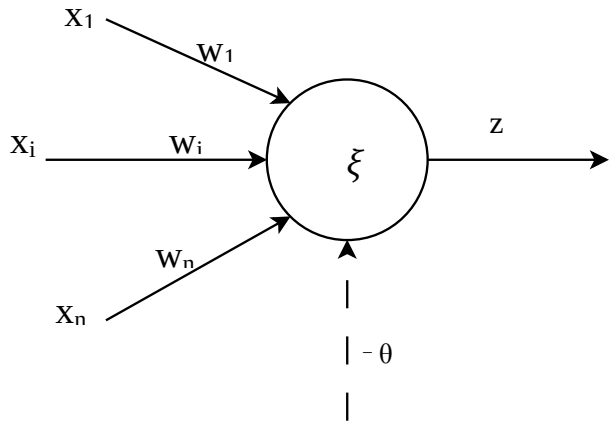
Jsou další formou umělé inteligence inspirované přírodou. Jsou založeny na modelování chování skutečných neuronů, které jsou mezi sebou propojeny do sítí. (Mařík & Štěpánková, & Lažanský, 1993) Obrázek 6 pak zobrazuje ilustraci biologického neuronu.



Obrázek 6: Biologický neuron (Bourg, & Seemann, 2004)

Neurony se tedy skládají z dendritů, které slouží jako vstupy neuronu, těla, které provádí proces myšlení samotného neuronu a axonu, což je výstupní výběžek neuronu. Mezi spoji jednotlivých neuronů pak bývá synaptická mezera, která pomocí chemikálií určuje, zda

jde o excitační, nebo inhibiční vstup neuronu. Obrázek 7 pak zobrazuje model jednoho neuronu. Tomuto modelu se z historických důvodů říká Perceptron. (Mařík & Štěpánková, & Lažanský, 1993)



Obrázek 7: Perceptron (Mařík & Štěpánková, & Lažanský, 1993)

V Perceptronu jsou vstupy značeny písmenem x, ve spodním indexu je číselné označení vstupu. W pak značí váhu příslušného vstupu. -θ symbolizuje práh, který neuron musí překonat pro svou aktivaci. ξ je potenciál neuronu, jehož matematické vyjádření představuje následující rovnice: (Mařík & Štěpánková, & Lažanský, 1993)

$$\xi = \sum_{i=1}^{n} x_i w_i - \vartheta$$

Z pak představuje výstup neuronu.

$$z = S(\xi) = \frac{1}{1 + e^{-\lambda\xi}}$$

Výstupní funkcí Perceptronu je tedy sigmoida. Jde o monotónně rostoucí funkci mezi dvěma asymptotickými body, nejčastěji jsou tyto body 0 a 1. (Mařík & Štěpánková, & Lažanský, 1993)

Neurony se pak spojují do větších celků, neuronových sítí. Tyto sítě jsou složeny z několika vrstev. Pokud počet vrstev je roven alespoň číslu tři, tak o vrstvách mezi vrstvou vstupů a vrstvou výstupů hovoříme jako o skrytých vrstvách. Jejich počet může být libovolný stejně tak jako počet neuronů, které síť obsahuje.

2.3.6 Využití umělé inteligence

Umělou inteligenci lze využít i v jiných oblastech než ve videohrách. Například genetické algoritmy byly využity i při tvorbě proudového motoru pro Boeing 777. Jeho základ byl sice navržen dle klasických postupů s důrazem na minimalizaci spotřeby, ale drobné detaily byly vytvořeny pomocí genetických algoritmů. Oproti původnímu návrhu, bez využití genetických algoritmů, došlo k úspoře paliva asi 2,5 %. (Mařík & Štěpánková, & Lažanský, 2001)

2.3.7 Současnost umělé inteligence

Umělé inteligenci se v současné době věnuje hodně pozornosti. Je to z toho důvodu, že se stává čím dál dostupnější a schopnější. Koncem loňského roku (2022) byl spuštěn *Chat GPT3*, což je jazykový model, který je schopen souvisle odpovídat na otázky a hovořit (myšleno písemně) o mnoho tématech. Dokonce je schopen psát i příběhy. Bohužel tento model si dokáže i vymýšlet. I když model prezentuje chybnou myšlenku je ji schopen podat velmi přesvědčivě. Když je model požádán, aby předložil zdroje, ze kterých čerpal, vymyslí si i ty. Předkládá i vymyšlené studie akademických pracovníků, kteří skutečně existují a dané oblasti se věnují. Studie, ale nejsou pod předloženým DOI dohledatelné.

Nicméně *Chat GPT3* není jediným úspěchem umělé inteligence současnosti. Dalším současným úspěchem je *DEXA*. Ta není tak komplexní jako *Chat GPT3*, ale to souvisí s jejím využitím. *DEXA* totiž slouží pro vyhledávání informací v podcastech. Prozatím umí podcasty od dvou tvůrců, Adnrewa Hubermana a Jordana Harbingera.

2.4 Unity

Jak již bylo řečeno v úvodu, Unity je jedním z několika nástrojů, které lze využít pro tvorbu počítačových her. V této sekci jsou stručně popsány některé aspekty tohoto herního enginu.

2.4.1 Scéna

Scénou lze nazvat úroveň hry. Uvnitř scén jsou uloženy objekty, tvoří prostředí dané úrovně. (Hardman, 2020)

2.4.2 Prefab

Slouží jako předpis určitého typu objektu. Například nepřátelských postav. V momentě, kdy by nebyl Prefab použit by bylo nutné pro změnu chování jednoho typu nepřátel

upravovat v každé scéně každého nepřítele zvlášť, ale s využitím Prefabu stačí upravit Prefab a změna se propíše do všech objektů napříč scénami. (Hardman, 2020)

2.4.3 MonoBehaviour

Je základní třídou, ze které dědí každý script už po vytvoření. MonoBehaviour umožňuje spouštět i zastavovat Coroutine. Každý vytvořený script také v základu obsahuje napojení na dvě události, Start a Update. Start se provede na začátku spuštění programu, nebo načtení scény. Update se pak provádí každý jednotlivý snímek. Právě proto je vhodné některé akce, jako například načítání scén, spouštět v Coroutine. Ta danou akci prování postupně během několika snímků. (UNITY TECHNOLOGIES, 2022b)

2.4.4 ScriptableObject

Využívá se zejména pro ukládání dat. Její hlavní využití nastává v případě, když si více objektů žádá data. Se ScriptableObjectem jsou data uložena na jednom místě a všechny objekty k nim mohou získat přístup. Pokud by data byla uložena prostřednictvím MonoBehaviour, zvýšila by se paměťová náročnost, protože by každý objekt měl data vlastní. (UNITY TECHNOLOGIES, 2022c)

2.4.5 Timeline

Timeline je nástroj pro tvorbu krátkých videoherních filmů (cut-scén). Umožňuje přehrávat zvuk, aktivovat objekty, hýbat s objekty přehrávat animace, přepínat mezi kamerami a další. (DocFX, 2023d)

2.4.6 Cinemachine

Cinemachine je nástroj, díky kterému lze snadněji ovládat kamery, přepínat mezi nimi a upravovat jejich zaměření na postavu, nebo objekt. (DocFX, 2023a)

2.4.7 Audio Source

Využívá se pro přehrávání zvukové nahrávky ve scénách. Audio Source nabízí mnoho nastavení. Lze nastavit například, zda se má zvuk přehrávat ve smyčce dokola. Dalším užitečným nastavením pak je, minimální a maximální vzdálenost. Minimální vzdálenost udává, jak moc blízko musí být Audio Listener blízko zdroje zvuku, aby slyšel zvuk v plné hlasitosti. Maximální vzdálenost zase udává, maximální vzdálenost, při které je zvuk alespoň minimálně slyšet. (UNITY TECHNOLOGIES, 2022a)

3 Metodika

Pro vyvíjenou počítačovou hru byl zvolen žánr adventura.

3.1 Herní lokalizace

Hra byla tvořena pro plnou podporu českého jazyka a částečnou podporu anglického jazyka. Oba jazyky podporuje uživatelské rozhraní, titulky k hlasovým nahrávkám a nápověda ve hře. Hlasové nahrávky vznikly pouze v českém jazyce.

Pro změnu textů na základě volby jazyka byl využit package *Localization*, který byl právě pro potřeby lokalizace vytvořen. Kromě textů lze pomocí tohoto package lokalizovat i obrázky, nebo zvuky. (DocFX, 2023c)

3.2 Ovládání hry

Hra je primárně tvořena pro ovládání prostřednictvím klávesnice a myši, ale podporuje i ovládání prostřednictvím herního ovladače. Hráčova postava může ve hře provádět několik akcí:

- Pohyb.
- Skok.
- Zaměření se na nepřátelskou postavu.
- Změna zbraně.
- Léčení se.
- Úskok.
- Interakce s NPC, nebo okolím.
- Obrana.
- Útok.
- Zobrazení úkolu.
- Otáčení kamery.
- Pozastavení hry.

Pro vytvoření systému ovládání hry byl využit *Input System*, který je k tomuto určený. Ze seznamu podporovaných herních ovladačů pak byl vybrán ovladač *DualSense*. (DocFX, 2023b)

3.3 Akce postav

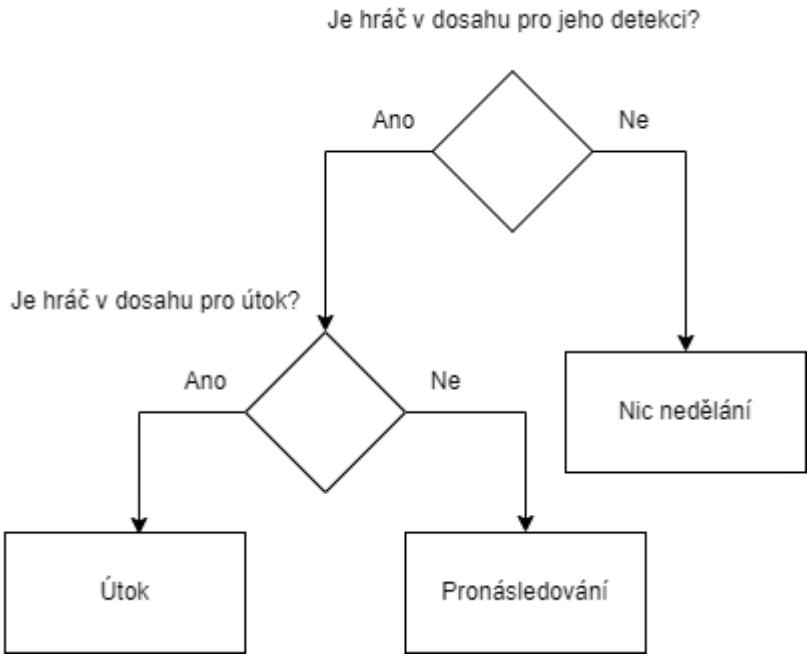
Jednotlivé akce postav jsou řešeny prostřednictvím stavů. Každý stav, ve kterém se postava ať už hráčova, nebo ovládaná počítačem může dostat obsahuje tři metody. Jednu metodu pro zahájení činnosti, například spuštění příslušné animace). Další metoda pak běží, dokud stav neskončí. Poslední metoda je volána na výstupu ze stavu.

3.4 Inteligence nepřátelských postav

Pro tuto hru byly vytvořeny tři různé inteligence postav.

3.4.1 Jednokritériový rozhodovací model

Kritériem pro rozhodování postav, které se řídí tímto modelem, je vzdálenost hráče od nepřátelské postavy. Z hlediska hratelnosti byly těmto postavám upřeny některé stavy. Rozhodovací strom tohoto modelu zobrazuje Obrázek 8.

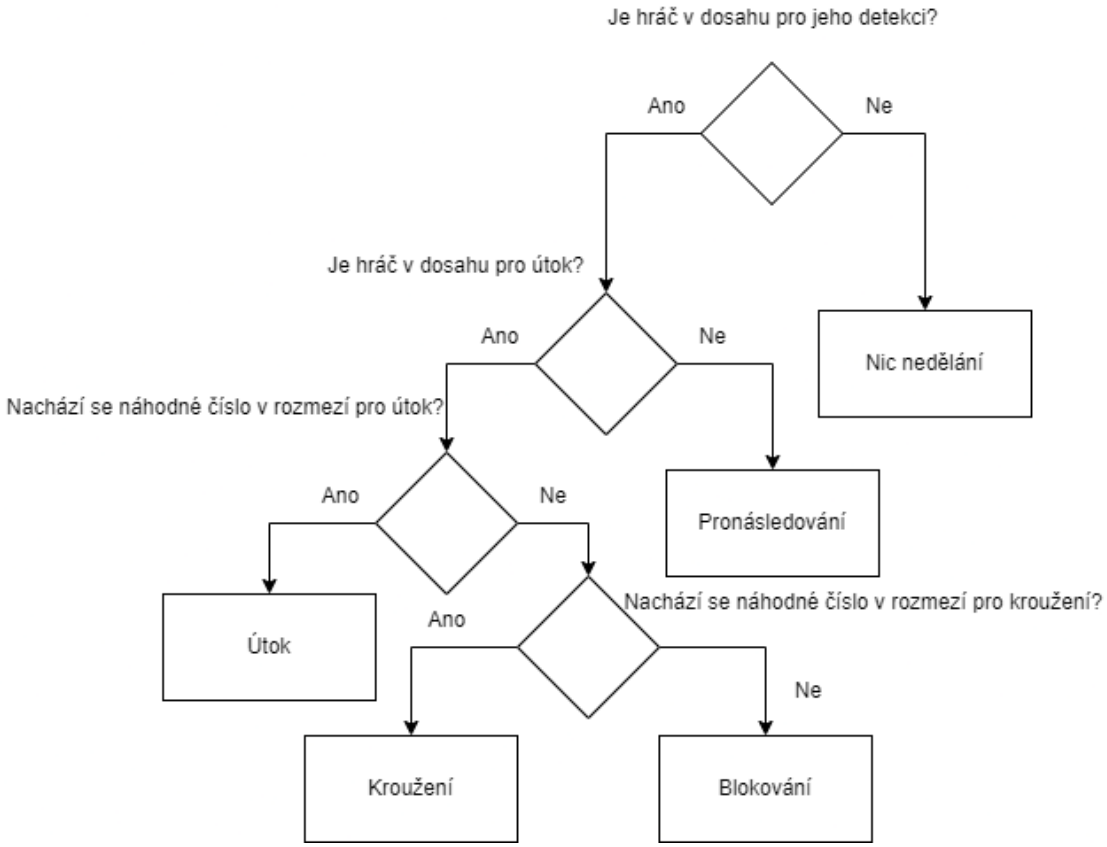


Obrázek 8: Návrh jednokritériového rozhodovacího modelu

3.4.2 Vícekritériový rozhodovací model

Tento rozhodovací model byl navržen jako rozšíření jednokritériového. Rozhoduje se ale i na základě svého aktuálního zdraví. Rozhodování na základě zdraví se začíná uplatňovat po rozhodování na základě vzdálenosti. Jakmile je tedy dostatečná vzdálenost pro to, aby si postavy mohly vzájemně ublížit, nastává volba akce. Akce tohoto modelu byly rozšířeny o akci blokování hráčových úderů a akci kroužení kolem. Model se částečně řídí i náhodou. Vygeneruje náhodné číslo a zkoumá, zda spadá do hranic pro jednotlivé akce

(útok, kroužení a blokování). Tyto hranice se mění s počtem životů postavy a jejích priorit pro dané akce tzn. postava má míň životů -> bude se chtít víc bránit. Rozhodovací strom reprezentuje Obrázek 9.



Obrázek 9: Návrh vícekritériového rozhodovacího modelu

3.4.3 Neuronové sítě

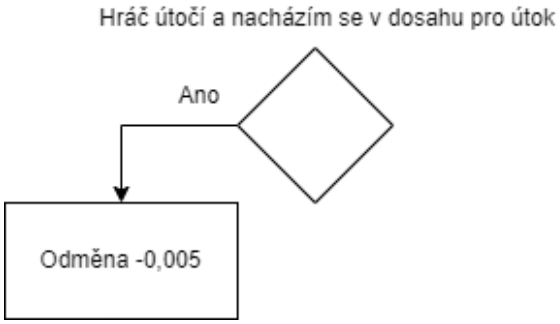
Unity umožňuje práci s neuronovými sítěmi prostřednictvím assetu Unity-ML-Agents.

Tento asset dovoluje trénovat neuronovou síť prostřednictvím posilovacího učení, které zprostředkovává algoritmus *Proximal polici optimalization*. Lze jej ale i rozšířit prostřednictvím systému *Curiosity*, který odměňuje zkoušení nových akcí, nebo učením dle vzoru, *Apprenticeship learning*. (Juliani, 2020) Právě tento asset byl využit.

Neuronové síti jsou pro její učení předány vstupy, na jejichž základě vygeneruje svůj výstup. Pro různé výstupy jsou pak prostřednictvím výsledkové funkce přiřazeny příslušné akce. Učení sítě probíhá na základě odměn, kdy je síť odměňována za žádoucí chování (výstup). Může být také odměněna za událost, kterou její chování vyvolalo.

Využitá neuronová síť byla odměňována na základě kombinace odměn chování a událostí, které byly chováním vyvolány. Některé akce a události byly odměňovány i zápornou hodnotou.

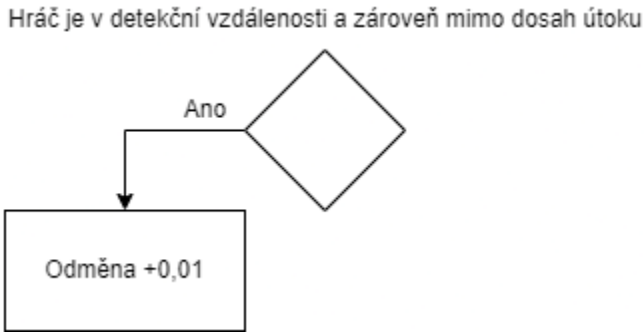
Akce nicnedělání



Obrázek 10: Odměňování akce nic nedělání

Akce nicnedělání ovlivňuje odměnu jen záporně. Tato záporná odměna je přidělena jen v případě, že je postava pod hráčovým útokem a je v dosahu hráčovi zbraně.

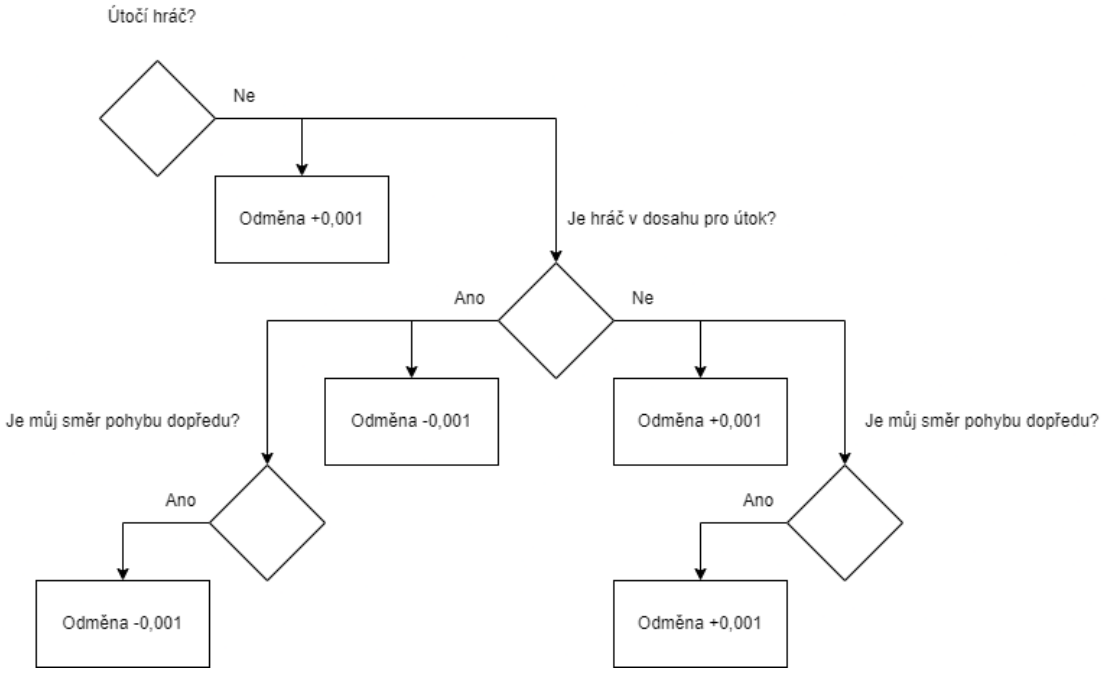
Akce pronásledování



Obrázek 11: odměňování akce pronásledování

Akce pronásledování je odměněna jen za předpokladu, že se hráč nachází v detekční vzdálenosti a zároveň není dostatečně blízko pro útok.

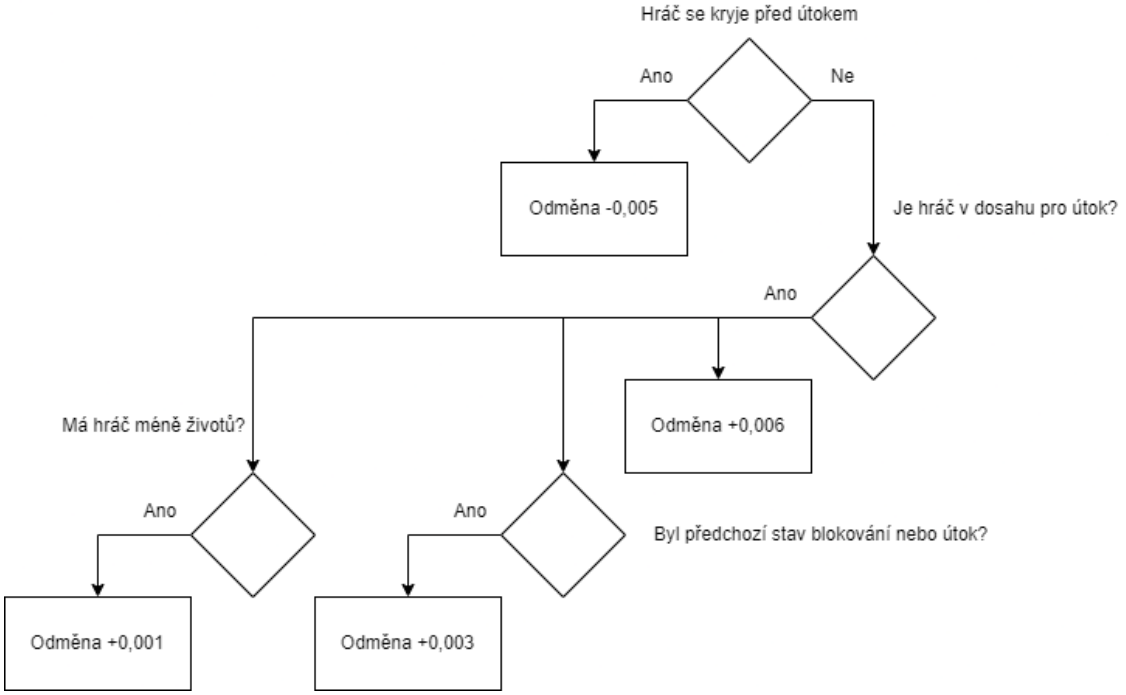
Akce pohybu kolem



Obrázek 12: Odměňování akce pohybu kolem

Odměnění pohybu kolem může nastat jen pokud hráč neútočí. V takovém případě dojde k navýšení odměny. Dále pak dochází k rozhodování, zda je hráč v dosahu pro útok. Pokud ano, je odměna snížena a dochází ke zkoumání, zda je směr pohybu dopředu. Pokud je směr pohybu dopředu dochází tak ke snížení odměny. Pokud hráč v dosahu útoku není, dojde k navýšení odměny a dochází k ověření, zda je směr pohybu postavy dopředu. Pokud ano, také dojde k navýšení odměny.

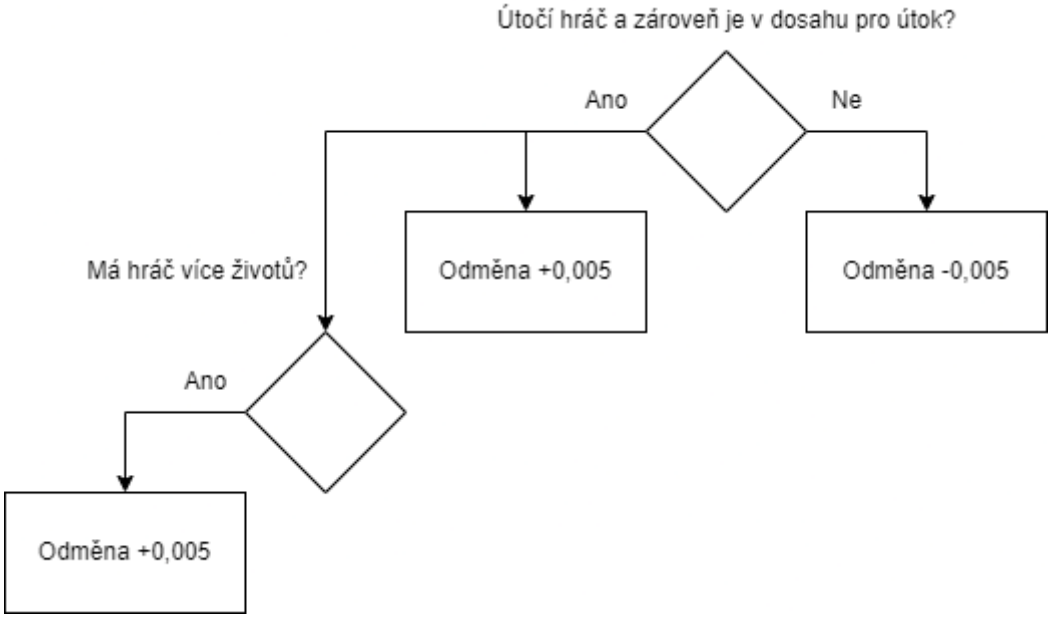
Akce útoku



Obrázek 13: Odměňování akce útoku

Jako první se při odměňování této akce zkoumá, zda se hráč kryje před útokem. Pokud ano, tak dojde ke snížení odměny. Pokud ne, probíhá vyhodnocení, zda je hráč v dostatečné blízkosti pro útok. Pokud ano, dojde k navýšení odměny. Dále pak proběhne vyhodnocení předchozího stavu. Pokud byl předchozím stavem útok, nebo blokování, tak je odměna navýšena. Další odměna může být přičtena v případě, že má hráč méně životů.

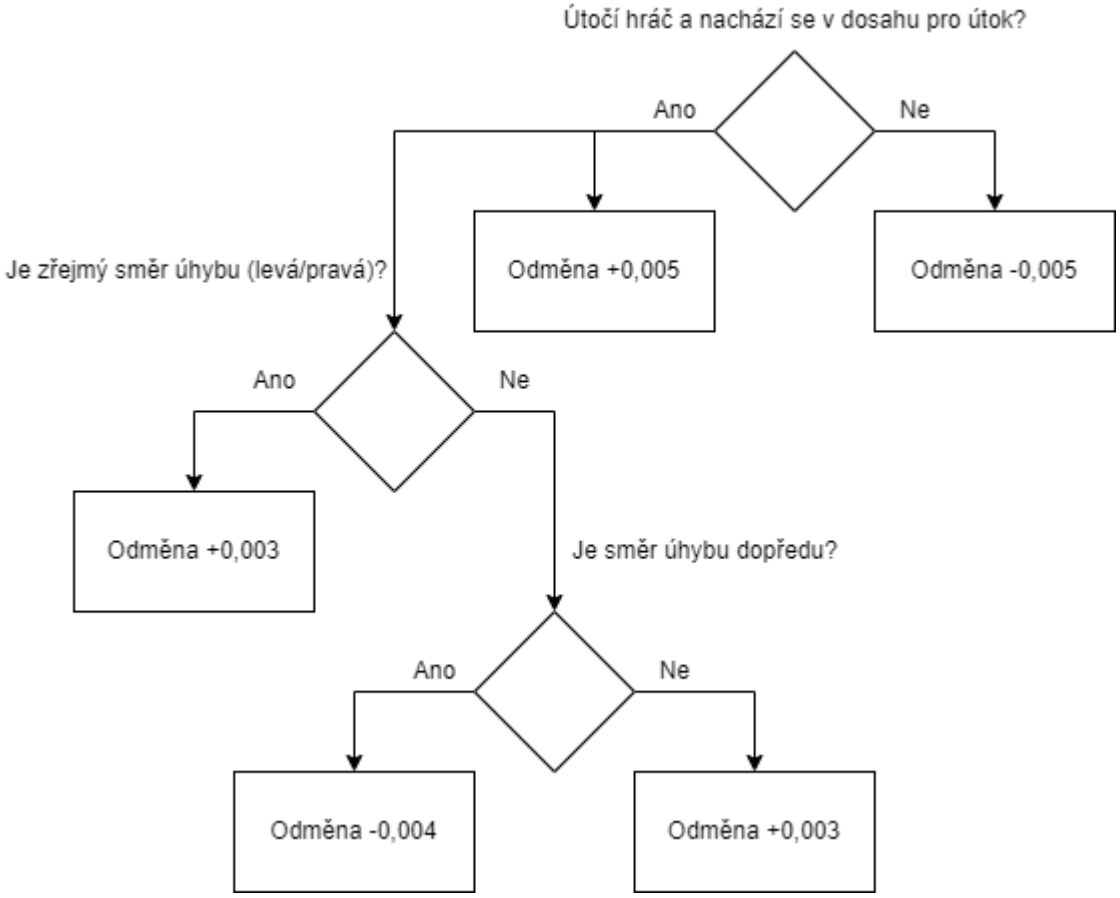
Akce blokování



Obrázek 14: Odměňování akce blokování

Akce je odměňována záporně, pokud hráč neútočí, nebo není v dosahu pro útok. V opačném případě dojde ke kladnému odměnění akce. Zároveň pak dojde ke zkoumání počtu životů. Pokud má hráč více životů, také dojde ke kladnému ohodnocení akce.

Akce úhybu



Obrázek 15: Odměňování akce úhybu

Pokud hráč útočí a zároveň se nachází v dosahu pro útok, dojde k odměnění akce. Dále pak dojde k rozpoznání směru pohybu. Pokud je zřejmý směr úhybu (levá/pravá), dojde k navýšení odměny. Pokud směr zřejmý není, dojde k rozlišení, zda je směr úskoku dopředu. Pokud je směr úskoku dopředu, tak dojde ke snížení odměny. V opačném případě dojde k jejímu navýšení. Pokud není první podmínka splněna, dojde ke snížení odměny.

Odměňování událostí

Dále byly odměňovány následující události:

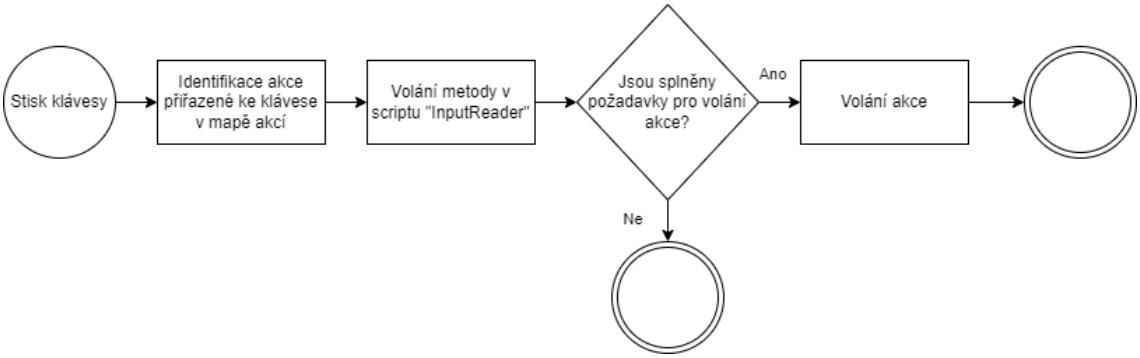
- Hráč byl zasažen (odměna +0,04).
- Hráč zasáhl postavu (odměna -0,03).
- Postava narazila do virtuální zdi (odměna -1).
- Postava vyblokovala útok (odměna +0,03).
- Postava vyhrála (odměna +1)
- Postava prohrála (odměna -1).

4 Implementace

4.1 Ovládání hry

Hra je koncipována pro hraní na počítači pomocí klávesnice a myši. Původně byla zamýšlena i podpora herního ovladače od společnosti *Steam*. Při mapování kláves ovladače na akce nastaly problémy a posléze bylo zjištěno, že *Input System*, který je využíván tento ovladač nepodporuje. Jako náhrada byl zvolen ovladač pro *PlayStation 5*.

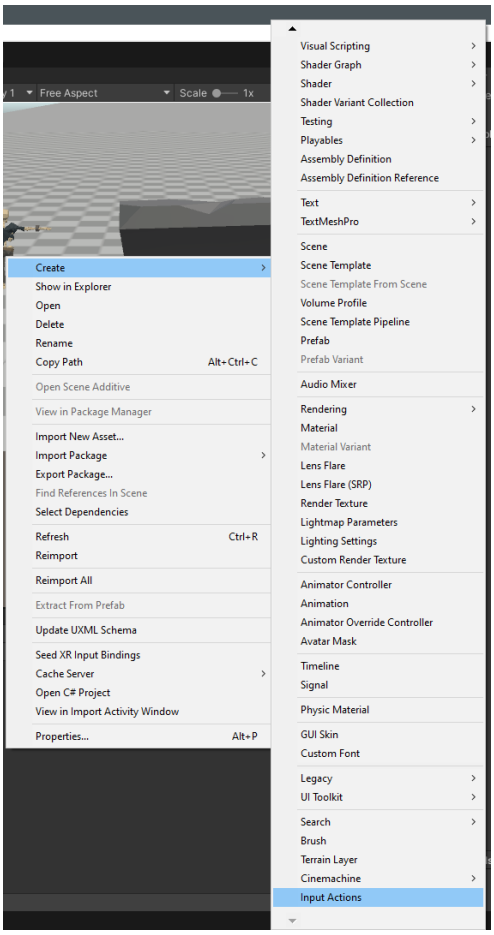
Proces od stisknutí klávesy po volání akce zachycuje Obrázek 16.



Obrázek 16: Proces reakce na stisk klávesy

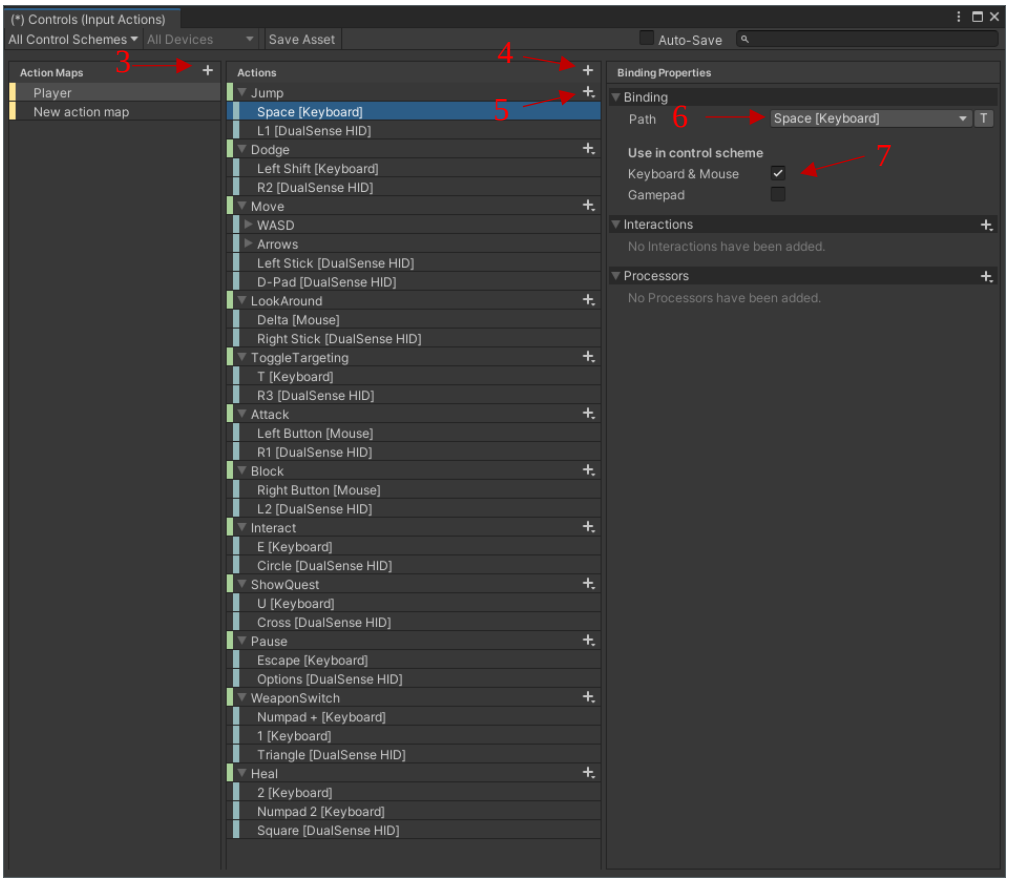
Pro tvorbu ovládacího rozhraní herní engine Unity obsahuje zvláštní package *Input System*, který umožňuje jednoduché mapování ovládání. To funguje prostřednictvím mapy akcí a kláves. Obrázek 18 zobrazuje mapu akcí a kláves včetně číselných popisků kroků tvorby. Proces tvorby mapy akcí lze popsat následujícími kroky.:

1. Nejprve je potřeba naistalovat a importovat package. To lze provést v okně *Package manager*, kde je nutné vyhledat package *Input System*, následně kliknout na tlačítko *Install*, pak na tlačítko *Import*.
2. Tvorba souboru, pro uchování map akcí viz Obrázek 17.



Obrázek 17: Tvorba souboru InputSystem

3. Vytvoření nové mapy akcí prostřednictvím + v levé části (mapu akcí lze přejmenovat).
4. Střední část zobrazuje jednotlivé akce. Ty lze přidat prostřednictvím + ve střední části.
5. Nyní je potřeba vytvořit vazbu mezi akcí a tlačítkem. Vazbu přidáme prostřednictvím + ve střední části **tentokrát však na úrovni jednotlivé akce**.
6. Dalším krokem pak je přiřazení příslušného tlačítka vazbě. Což lze udělat v pravé části. Tam je nabídka *Path* a je potřeba vybrat příslušnou klávesu.
7. Následně už jen stačí vybrat ovládací schéma, pro které se má vazba využít. Schéma stačí vybrat jednoduše zaškrtnutím.



Obrázek 18: Mapa akcí

Těmito kroky bylo vytvořeno mapování akcí na klávesy. Aby vše fungovalo, musel vzniknout script, který volá příslušné akce po stisknutí klávesy. Script byl pojmenován *InputReader*.

4.1.1 InputReader

InputReader obsahuje akce, a metody. Metody jsou volány akcí kláves. Vytvořený script rozlišuje tři akce kláves. Blokování využívá dvě z nich viz Blok kódu 1.

```
public void OnBlock(InputAction.CallbackContext context)
{
    if (context.started)
    {
        startBlocking?.Invoke();
    }

    if (context.canceled)
    {
        stopBlocking?.Invoke();
    }
}
```

Blok kódu 1: InputReader – OnBlock

Metoda *OnBlock* při stisku a držení příslušné klávesy volá akci *startBlocking* a při puštění klávesy volá akci *stopBlocking*.

Další využitou akcí kláves je pak *performer*. Tu využívá například metoda volající akci ro změnu zbraně viz Blok kódu 2.

```
public void OnWeaponSwitch(InputAction.CallbackContext context)
{
    if (context.performed)
    {
        weaponSwitched?.Invoke();
    }
}
```

Blok kódu 2: InputReader – OnWeaponSwitch

Tato metoda volá akci *weaponSwitched* při stisku příslušné klávesy.

Podobným způsobem funguje i volání ostatních akcí s výjimkou zaměření se na nepřátelskou postavu a útoku. Obě tyto metody obsahují ještě kontrolu, zda se hráč nenachází ve scéně, kde jsou tyto akce zakázány viz Blok kódu 3.

```
public void OnToggleTargeting(InputAction.CallbackContext context)
{
    if(SceneManager.GetActiveScene().name == "Throne Room")
    {
        notAllowed?.Invoke();
    }
    else
    {
        if (isTargeting)
        {
            isTargeting = false;
            stopTargeting?.Invoke();
        }
        else
        {
            isTargeting = true;
            startTargeting?.Invoke();
        }
    }
}
```

Blok kódu 3: InputReader – OnToggleTargeting

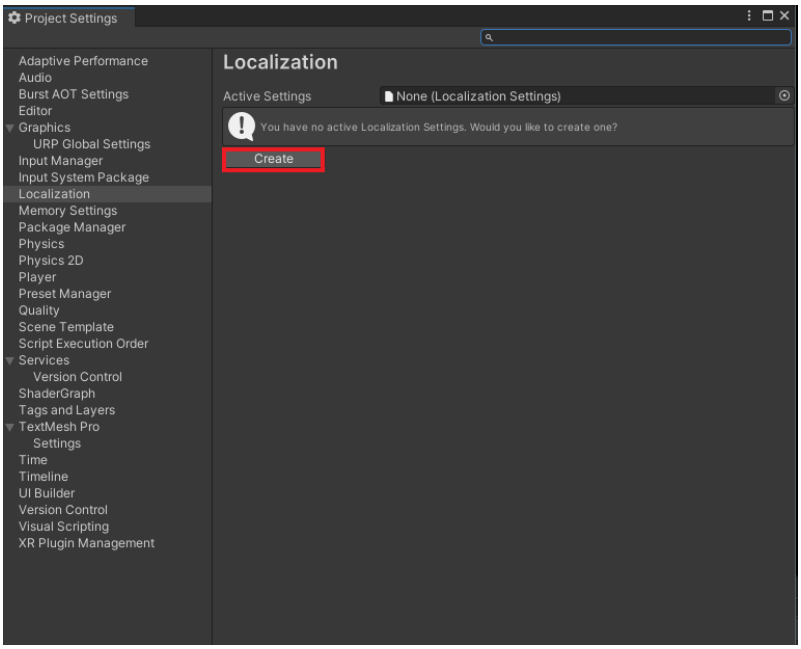
Metoda pro zacílení nepřátelské postavy navíc ještě obsahuje booleovskou proměnnou, *isTargeting*, která vyjadřuje aktuální stav (je zacíleno/není zacíleno). Pokud zacíleno je, volá se akce *stopTargeting*, v opačném případě dojde k volání akce *startTargeting*.

4.2 Lokalizace hry

Hra je plně lokalizována pro český jazyk a částečně podporuje anglický jazyk. V anglickém jazyce lze zobrazit Uživatelské rozhraní UI a titulky. Mezi oběma jazyky jde libovolně přepínat. Pro tvorbu lokalizací byl využit oficiální package *Localization* pro Unity engine.

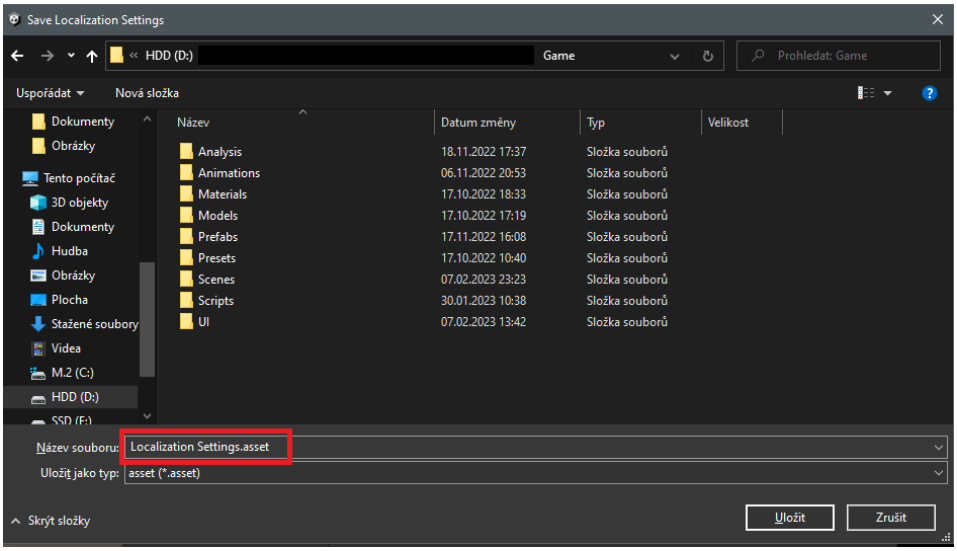
Proces tvorby lokalizace lze popsat v několika krocích:

1. Stáhnout a importovat package *Localization* prostřednictvím *package manager*.
2. Přejít do *Edit-> Project Stettings* a vybrat možnost *Localization*.
3. Vytvoření nastavení pro lokalizaci prostřednictví tlačítka *Create*.



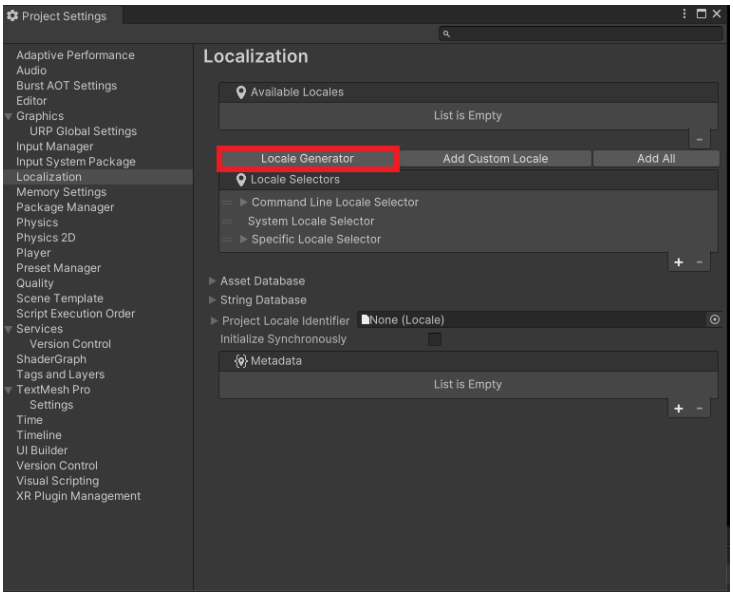
Obrázek 19: Vytvoření nastavení lokalizace

4. Soubor byl uložen do složky *Game* a jméno bylo ponecháno defalutní.



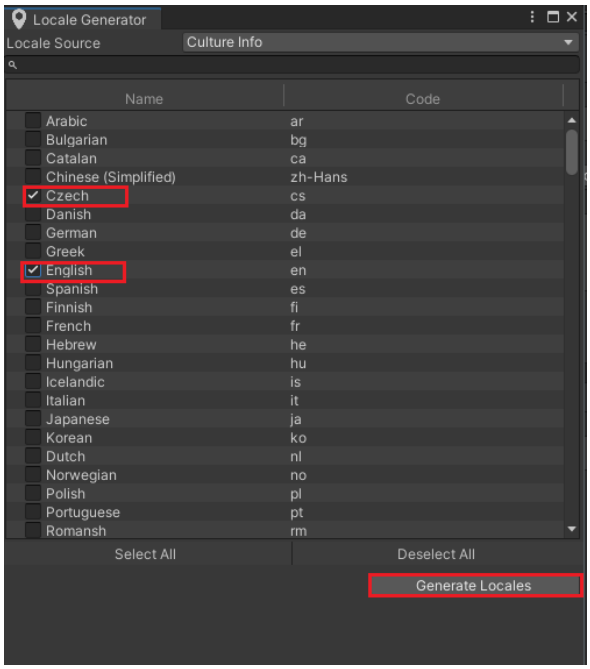
Obrázek 20: Volba umístění nastavení lokializace

5. Vygenerovat požadované lokalizace prostřednictvím tlačítka *Locale Generator*.



Obrázek 21: Vstup do nabídky pro generování lokalizací

6. Vybrat požadované jazyky z nabídky a nechat je vygenerovat tlačítkem *Generate Locales*.

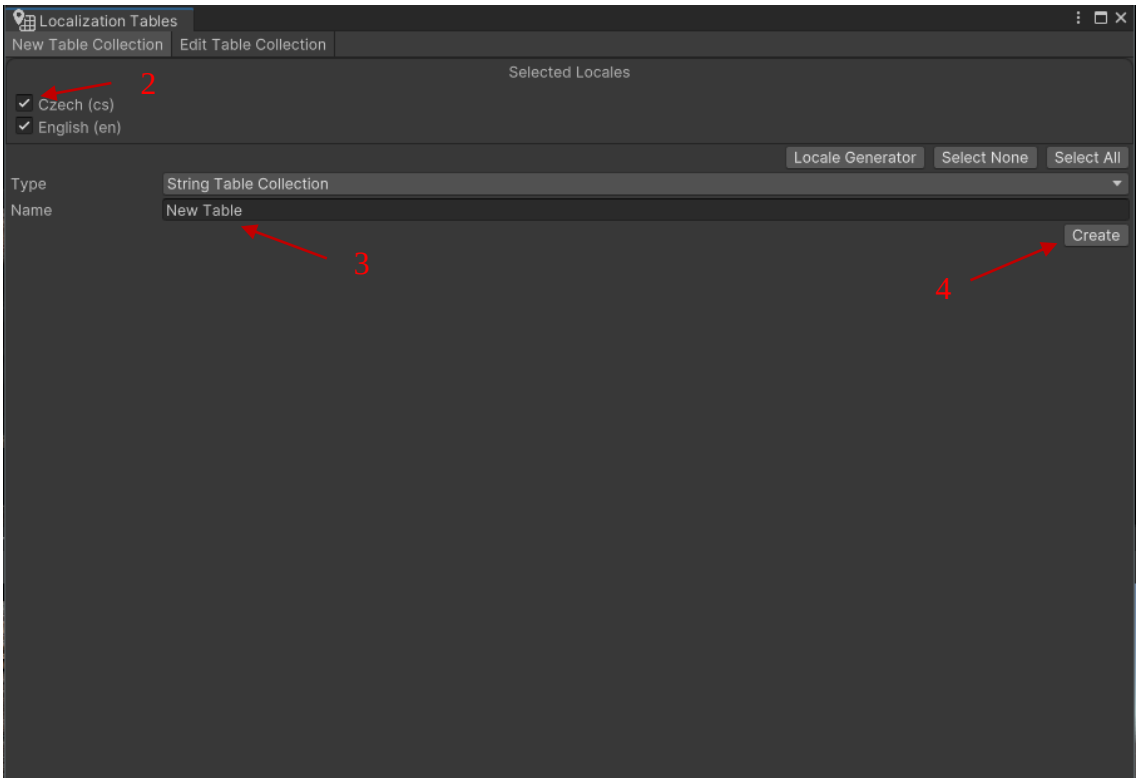


Obrázek 22: Výběr lokalizací pro generování

7. Zvolit složku, kam se mají soubory lokalizace uložit.

Tím byly vytvořeny soubory lokalizace. Teď je nutné vytvořit tabulku/tabulky lokalizací, které určí, který text se pro danou lokalizaci zobrazí. V této práci bylo vytvořeno několik takových tabulek. Každá pro příslušný účel. Proces tvorby tabulky překladů je následující.:

1. Rozkliknout nabídku *Window-> Asset management-> Localisation Tables*.
2. Vybrat lokalizace, které chceme v tabulce použít.
3. Pojmenovat tabulku.
4. Vytvořit tabulku tlačítkem *Create*.



Obrázek 23: Tvorba tabulky lokalizace

Tabulka lokalizací obsahuje sloupce pro texty ve vybraných jazycích, navíc v tabulce existuje ještě jeden sloupec. Sloupec určen pro identifikátor.

4.3 Uživatelské rozhraní

Pro uživatelské rozhraní byly vytvořeny obrázky prostřednictvím programu *Corel Painter 2020*. Obrázek 24 je ve hře využíván jako podklad pro texty a tlačítka uživatelského rozhraní.

Obrázek 24: Podklad

Dalším vytvořeným prvek uživatelského rozhraní je ukazatel životů. Hráčův ukazatel životů a ostatních postav se trochu liší. Oba ukazatele mají společné, že jde o rámeček, který je navrch posuvníku. Hráčův ukazatel, však není prázdný rámeček, ale uvnitř jsou nakresleny skvrny.

Obrázek 25: Hráčův ukazatel životů

Obrázek 26: Ukazatel životů nehráčských postav

4.4 Animace

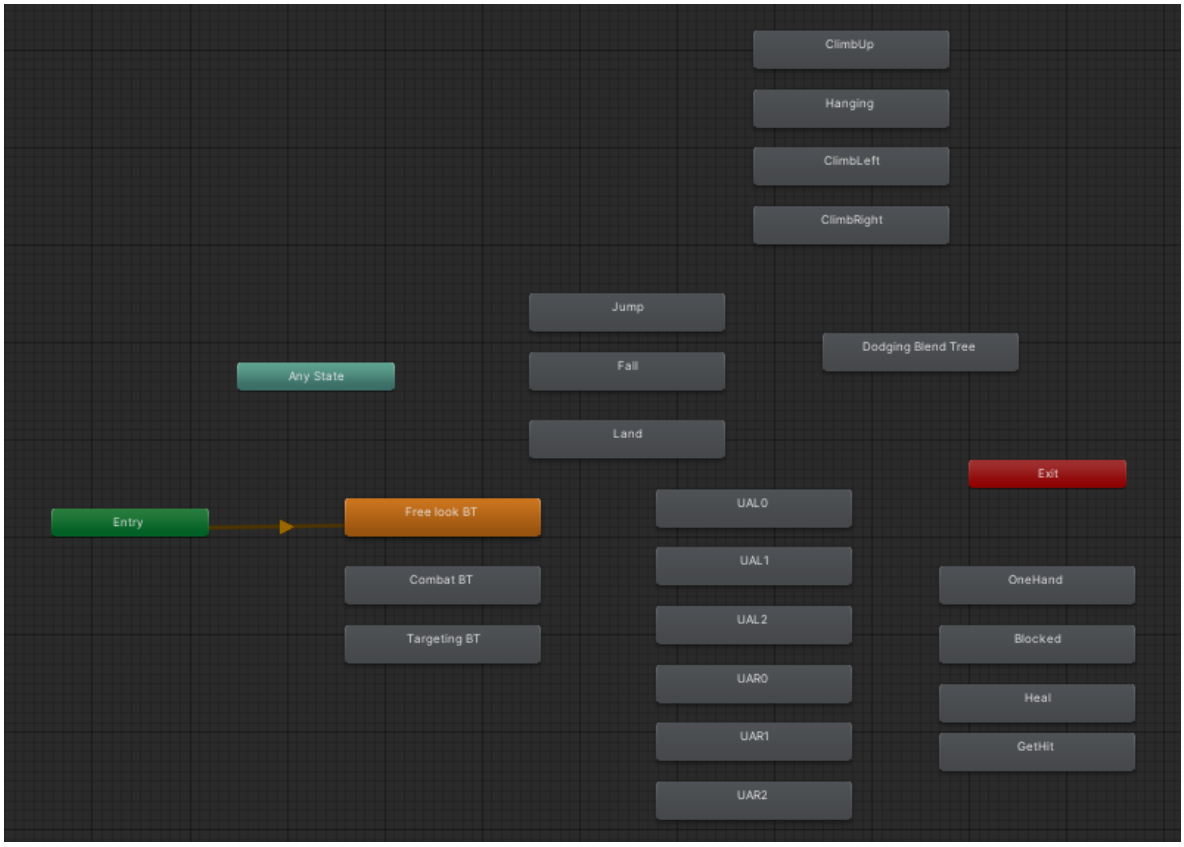
Pro práci s animacemi poskytuje Unity vlastní nástroje, *Animator* a *Animation*. Pomocí nástroje *Animator* lze nastavit přechody mezi animacemi.

4.4.1 Animator

Animator umožňuje i vytvoření *BlendTree*, což je struktura složena z dvou a více animací, které se prolínají na základě parametrů. Parametry pro *BlendTree* mohou být číselné, ale i booleovské.

Animator dále také umožňuje právě nastavení přechodů mezi animacemi. Tímto fungováním by se dal přirovnat k deterministickému automatu, který považuje jednotlivé animace za stavy a pokud jsou splněny podmínky pro přechod do jiné animace, přechod uskuteční. V této práci, ale tato funkcionalita využita nebyla. Přechody mezi animacemi zajišťují jednotlivé stavy postav viz sekce Stavy postav.

Animator hráče zobrazuje Obrázek 27.

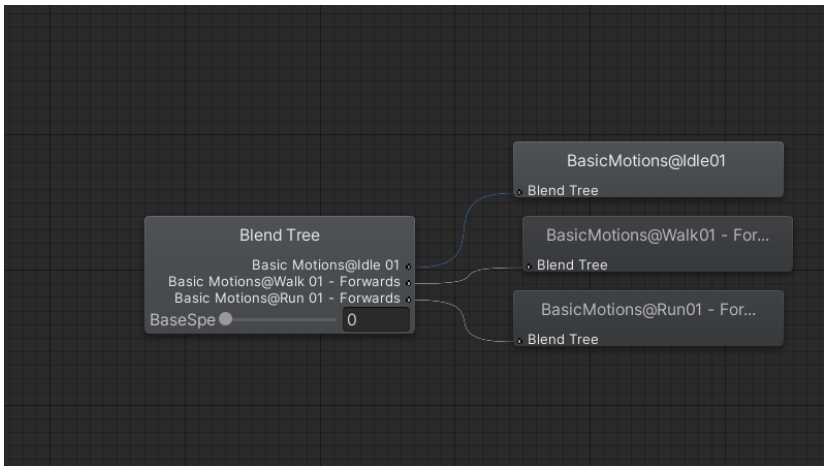


Obrázek 27: Animator hráče

Hráčův *Animator* obsahuje šest animací útoku beze zbraně. Tři pro každou ruku. Dále obsahuje animaci pro stav, když utrží poškození. Dalšími animacemi pak jsou animace pro pád, skok, útok jednoruční zbraní, léčení a animace pro šplh. Dále obsahuje Několik *Blend tree*.

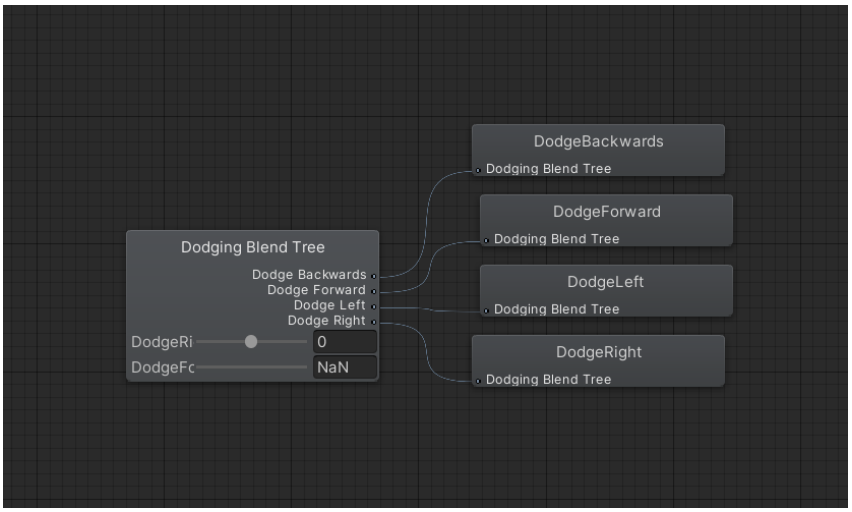
Free look Blend tree

Pro volný pohyb byl vytvořen *Blend tree*, který postavě umožňuje jen tak stát, chodit a běžet v závislosti na rychlosti, kterou se postava pohybuje. K tomu byly použity tři animace. Jedna pro stav *IDLE*, kdy postava jen stojí, druhá pro chůzi a třetí pro běh viz Obrázek 28



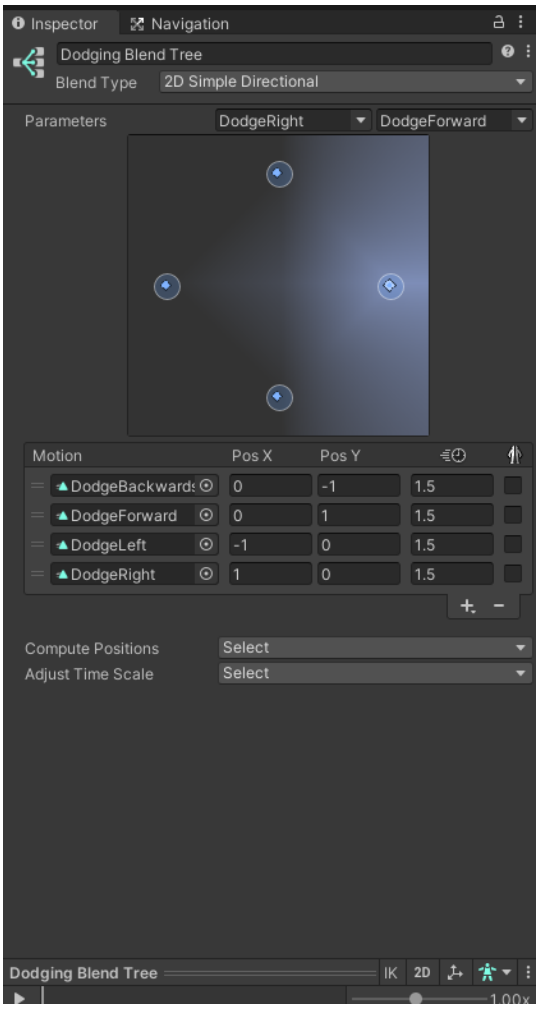
Obrázek 28: Free look Blend tree

V podobném stylu byly vytvořeny i ostatní *Blend tree*. Například *Blend tree* pro úskok při souboji je velmi podobný, ale zde už záleží na dvou parametrech. Jak moc se hráč uhýbá dopředu/dozadu a doleva/doprava viz Obrázek 29.



Obrázek 29: Blend tree úskoku

Další možný náhled na *Blend tree* pak poskytuje inspektor, ve kterém je vidět i rozmezí hodnot pro jednotlivé animace viz Obrázek 30.



Obrázek 30: Inspektor pro blend tree úskoku

4.4.2 Animation

Prostřednictvím *Animation* lze měnit, ale i tvořit vlastní animace. Dokonce můžeme i pomocí animace volat různé události. Například v animacích pro útok jsou volány události, které aktivují a deaktivují detektory kolizí (*collidery*) na jednotlivých zbraních. Tímto krokem hra předchází nechtěnému útočení. Například pokud by detektor kolize na zbrani byl zapnut při běhu, mohlo by dojít k odebrání životů počítačem ovládané postavě jen tím, že hráč kolem ní projde a zbraní „zavadí“ o tuto postavu. Obrázek 31 a Obrázek 32 pak zobrazují momenty, ve kterých je voláno zapnutí a vypnutí *collideru* zbraně.

Obrázek 31: Zapnutí collideru zbraně

Obrázek 32: Vypnutí collideru zbraně

Animace volají metodu ze scriptu *WeaponActivator*. Ten obsahuje metody pro aktivaci a deaktivaci příslušných *colliderů*. Dále pak obsahuje metodu, která je schopna deaktivovat kterýkoliv z *colliderů* viz Blok kódu 4. Ta je volána v případě, že postava utrží poškození. V takovém případě totiž animace útoku nemusí dospět do potřebného bodu pro deaktivaci *collideru*. Jenže v animaci utrženého poškození nelze zjistit, která zbraň je aktuálně aktivní, proto bylo nutné vytvořit tuto metodu pro deaktivaci kteréhokoliv z nich.

```
public class WeaponsActivator : MonoBehaviour
{
    [SerializeField] private GameObject RightFistLogic;
    [SerializeField] private GameObject LeftFistLogic;
    [SerializeField] private GameObject Weapon;

    public void EnableRightFist()
    {
        RightFistLogic.SetActive(true);
    }
    public void DisableRightFist()
    {
        RightFistLogic.SetActive(false);
    }
    public void EnableLeftFist()
    {
        LeftFistLogic.SetActive(true);
    }
    public void DisableLeftFist()
    {
        LeftFistLogic.SetActive(false);
    }
    public void EnableWeapon()
    {
        Weapon.SetActive(true);
    }
    public void DisableWeapon()
    {
        if (Weapon != null)
        {
            Weapon.SetActive(false);
        }
    }
    public void DisableActualWeapon(){
        if (Weapon != null && Weapon.activeSelf)
        {
            Weapon.SetActive(false);
        }
        if(RightFistLogic != null && RightFistLogic.activeSelf)
        {
            RightFistLogic.SetActive(false);
        }
        if(LeftFistLogic != null && LeftFistLogic.activeSelf)
        {
            LeftFistLogic.SetActive(false);
        }
    }
}
```

Blok kódu 4: WeaponActivator

4.5 Scény hry

Hra obsahuje celkem pět scén.

4.5.1 Hlavní nabídka

Hlavní nabídka obsahuje celkem čtyři tlačítka viz Obrázek 33.

Obrázek 33: Hlavní nabídka

Příslušné akce tlačítek jsou pak řešeny prostřednictvím volání příslušných metod.

```
public class MainMenu : MonoBehaviour
{
    public Progress progress;
    public Book book;
    public PlayerHP hp;
    public QuestProgress questProgress;


    public void PlayGame()
    {
        StartCoroutine(Play());
    }
    private IEnumerator Play()
    {
        ResetProgress();
        AsyncOperation asyncLoad = SceneManager.LoadSceneAsync("Narrator");
        while (!asyncLoad.isDone)
        {
            yield return null;
        }
    }
    private void ResetProgress()
    {
        questProgress.Id = 1;
        questProgress.Phase = 1;
        hp.HP = 100;
        progress.pKilled = false;
        progress.scene = SceneManager.GetActiveScene().name;
        book.used = false;
        book.pickedUp = false;
    }

    public void Quit()
    {
        Application.Quit();
    }
}
```

Blok kódu 5: Hlavní nabídka

Hlavní nabídka obsahuje metodu pro tlačítko *Hrát hru* a tlačítko *Vypnout hru* viz Blok kódu 5. Metoda *PlayGame* volá metodu *Play* v *Coroutine*. Což umožňuje rozložit načítání scény během několik snímků. Před samotným načtením scény ještě proběhne metoda *ResetProgress*, která nastaví výchozí hodnoty.

Nastavení

Nastavení umožňuje změnit hlasitost zvuků ve hře a jazyk hry viz Obrázek 34.

Obrázek 34: Nastavení hry

Kódově je pak při startu hry nastavena hlasitost buď na hodnotu -70, nebo na hodnotu získanou z *PlayerPrefs*, následuje nastavení hodnoty posuvníku a zneaktivnění nabídky nastavení. Metoda *SetVolume* nastavuje hlasitost dle hodnoty posuvníku a zároveň hodnotu ukládá do *PlayerPrefs*.

```
public class SettingsMenu : MonoBehaviour
{
    public AudioMixer audioMixer;
    public Slider volumeSlider;
    public Slider scaleSlider;
    public InputActionReference lookAction;

    void Start()
    {
        audioMixer.SetFloat("Volume", PlayerPrefs.GetFloat("Volume", -70));
        volumeSlider.SetValueWithoutNotify(PlayerPrefs.GetFloat("Volume", -
70));
        gameObject.SetActive(false);
    }
    public void SetVolume(float volume)
    {
        audioMixer.SetFloat("Volume", volume);
        PlayerPrefs.SetFloat("Volume", volume);
    }
}
```

Blok kódu 6: Nastavení

Pro změnu jazyka pak byla vytvořena zvláštní třída, která přepíná zvolený jazyk mezi anglickým a českým viz Blok kódu 7.

```
public class Language : MonoBehaviour
{

    public void SetLanguage(Locale locale)
    {
        LocalizationSettings.SelectedLocale = locale;
    }

    public void ChangeLanguage()
    {
        int hashLangCze = LocalizationSettings.AvailableLocales.Lo-
cales[0].GetHashCode();
        int cze = 0;
        int eng = 1;

        SetLanguage(LocalizationSettings.AvailableLocales.Locales[Locali-
zationSettings.SelectedLocale.GetHashCode() == hashLangCze ? eng : cze]);
    }
}
```

Blok kódu 7: Přepínání jazyka

4.5.2 Scéna vypravěče

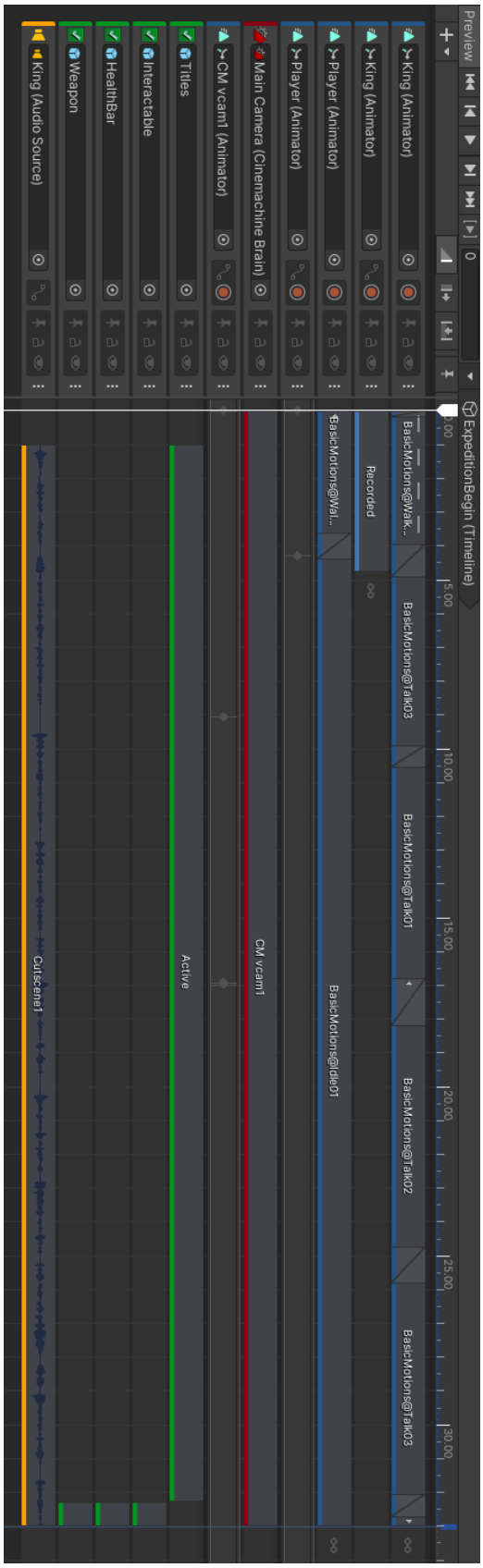
Tato scéna obsahuje podkladový obrázek a text, který se mění v závislosti na postupu hrou. Pro prolog vypravěče se spustí i animace, kdy text postupně vyjíždí na obrazovku. Následně se zobrazí i možnost přeskočit tuto scénu. Znovu se hráč na tuto scénu dostane v epilogu. Podle jeho rozhodnutí se nahraje příslušný text a následně seznam využitých assetů. Poté se hra vypne. Proces výběru textu pro načtení probíhá na základě jména předchozí scény, postupu hrou a hráčovým rozhodnutím viz Blok kódu 8. Tyto informace jsou uloženy v souborech, které byla za účelem přesunu informací mezi scénami vytvořeny.

```
private void LoadText(){
    string _tableName = SceneManager.GetActiveScene().name;
    Debug.Log(_tableName);
    string _entryReference = "Prologue";
    switch (progress.scene)
    {
        case "Main Menu":
            _entryReference = "Prologue";
            break;
        case "Arena":
            if (progress.pKilled)
            {
                _entryReference = "ArenaPLost";
            }
            else
            {
                _entryReference = "ArenaPWon";
            }
            break;
        case "Dungeon":
            if (!Book.used)
            {
                _entryReference = "BookNotUsed";

            }
            break;
    }
    Debug.Log(_entryReference);
    var op = LocalizationSettings.StringDatabase.GetLocalizedStrin-
gAsync(_tableName, _entryReference);
    if (op.IsDone)
    {
        Debug.Log(op.Result);
        Text.text = op.Result;
        if (_entryReference == "Prologue")
        {
            Prologue.SetActive(true);
            //70
            Invoke("SceneSwitchUIActivation", 3f);
        }else
        {
            Debug.Log("Cast");
            Invoke("Cast", 10f);
            Debug.Log("Cast");
        }
    }
    Debug.Log(op.Result);
}
```

Blok kódu 8: NarratorController – LoadText

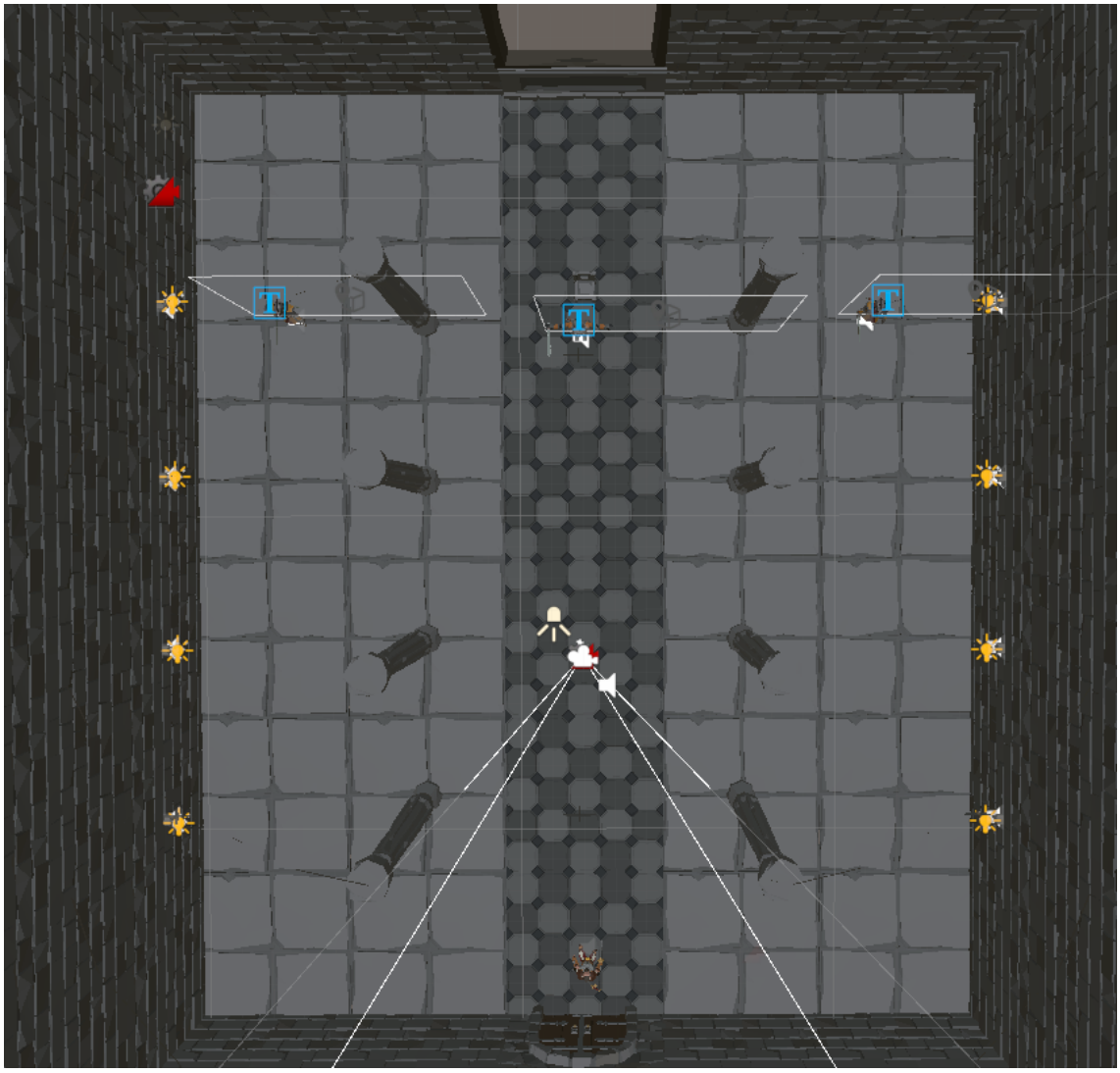
4.5.3 Úvodní scéna

Po načtení scény se spustí krátký film. V průběhu tohoto filmu k sobě přichází postava krále a postava hráče. Dále se spustí hlasová nahrávka repliky krále. Ta byla nahrán a upravena prostřednictvím program Audacity. Film obsahuje i aktivaci a pohyb jiné kamery než kamery hráče viz Obrázek 35. V závěru filmu se aktivují prvky uživatelského rozhraní jako je například ukazatel životů.



Obrázek 35: Úvodní timeline

Dále scéna obsahuje dva strážné, kteří se pohybují mezi dvěma body. Na stěnách jsou pak rozmístěny pochodně, které jsou vlastním zdrojem světla a zvuku viz Obrázek 36. Také obsahují efekt simulující oheň viz Obrázek 37.



Obrázek 36: Úvodní scéna

Obrázek 37: Pochodeň

4.5.4 Scéna jeskyně

V této scéně se hráč ocitá v jeskyni, kde je spolu s ním pět kostlivců a kniha. Kniha ve filmu, kterým scéna začíná mizí a objeví se až když je hlavní kostlivec poražen. S knihou pak hráč může interagovat prostřednictvím dvou tlačítek. Pomocí prvního hráč vybere volbu přečíst a doručit. Druhé tlačítko pak představuje volbu pouze doručit. Pokud hráč knihu přečte zobrazí se ikona léčení společně s nápovědou, že byla získána schopnost léčení. Nápověda za sedm vteřin sama mizí. Tyto funkce knize zprostředkovává script *BookController* jehož funkci pro přečtení knihy zobrazuje Blok kódu 9.

```
public void Read()
{
    Book.used = true;
    HealUI.SetActive(true);
    BookUI.SetActive(false);
    Time.timeScale = 1f;
    Cursor.visible = false;
    HealInfo.SetActive(true);
    Invoke("HideHealInfo", 7f);
}
```

Blok kódu 9: BookController – Read

První kostlivci, se kterými se hráč může setkat nejsou agresivní. Což se mění ve chvíli, kdy je na ně zaútočeno. Jakmile je na jednoho z kostlivců zaútočeno, ostatní kostlivci se to dozvědí a stávají se agresivními. Pokud jsou agresivní a hráč se dostane dostatečně blízko, začnou jej pronásledovat a útočit. Takovéto chování je zprostředkováno díky tomu, že obsahují *StateMachine* pro neutrální NPC, ale i pro nepřátelské postavy.

Po spuštění scény se tito kostlivci řídí stavy pro neutrální postavy, protože jejich *StateMachine* pro nepřátelské postavy není aktivní. *StateMachine* NPC obsahuje pole objektů, které představují jeho spojence. Na začátku každý kostlivec své pole projde a začne hlídat, jestli nějaký z jeho spojenců dostal zásah viz Blok kódu 10.

```
if (friendList.Length > 0)
{
    foreach (EnemyStateMachine friend in friendList)
    {
        friend.Health.OnTakeDamage += ChangeFitstAttack;
    }
}
```

Blok kódu 10: Hlídání, zda nějaký ze spojenců utržil poškození

Tato akce volá metodu *ChangeFirstAttack*, která nastaví proměnnou pro první útok na true.

V každém běhu metody *Tick* se u NPC kontroluje nejen, zda je hráč v dosahu a má mu NPC zpřístupnit interakci, ale i zda je hráč v dosahu pro detekci a zda může postava být agresivní viz Blok kódu 11.

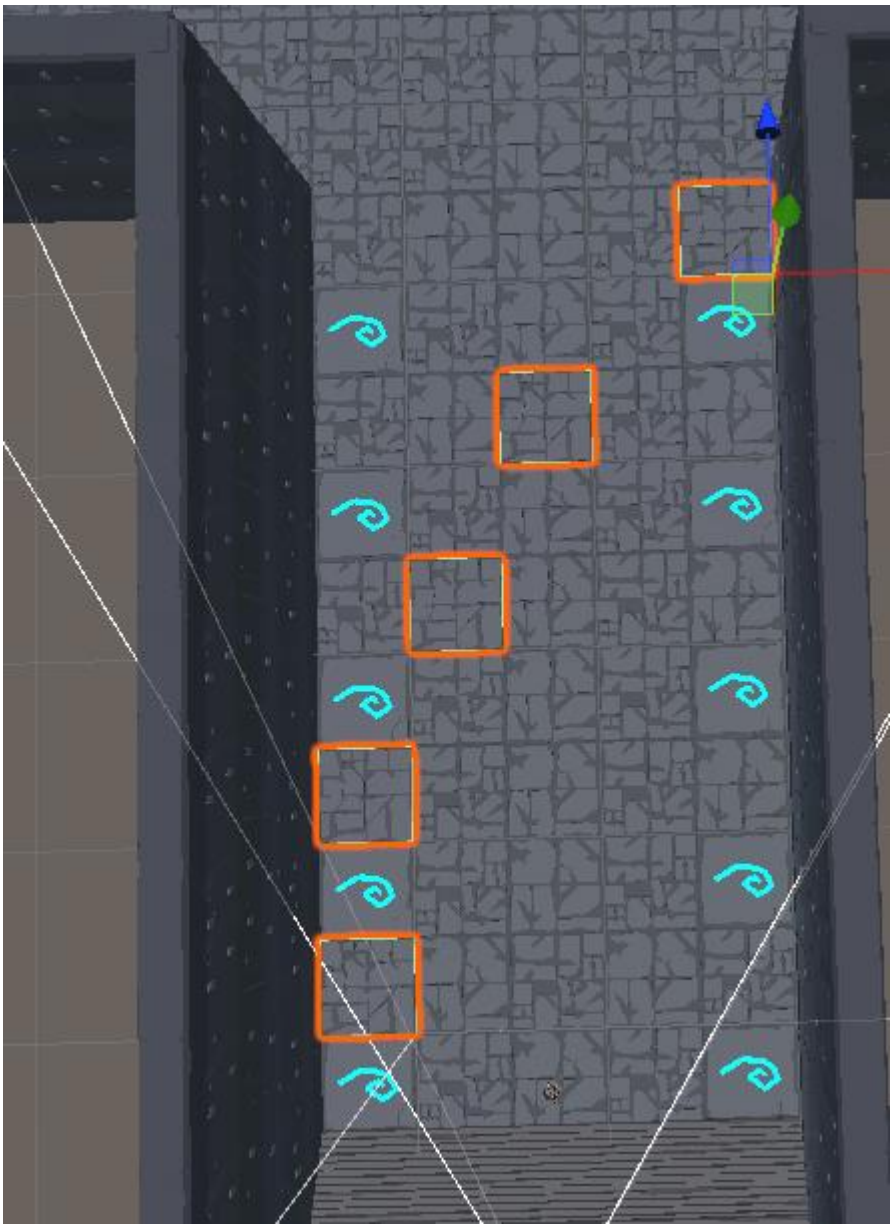
```
public override void Tick(float deltaTime)
{
    if (nsm.GetRangeController().IsInInteractableRange(nsm) && nsm.Que-
sts.Any() && !ena && !nsm._clicled)
    {
        nsm.Interactabe.enabled = true;
        //ena = !ena;
    }
    else
    {
        nsm.Interactabe.enabled = false;
        ena = !ena;
    }

    if (nsm.GetRangeController().IsInDetectionRange(nsm.GetComponent<Ene-
myStateMachine>()) && nsm.canAttackFirst)
    {
        nsm.enabled = false;
        EnemyStateMachine esm = nsm.GetComponent<EnemyStateMachine>();
        esm.enabled = true;
        esm.Health.SetHealth(nsm.Health.GetHealth());
    }
}
```

Blok kódu 11: NPCBaseState – Tick

4.5.5 Podmíněná scéna

Zda se hráč do této scény dostane záleží na jeho rozhodnutí ohledně přečtení knihy. Tato scéna obsahuje pouze jednoho protivníka a pasti. V krátkém filmu na úvod scény se hráči ukáže informace o propadlištích v podlaze společně s nápovědou pro bezpečný průchod: „*Chodba je plná pastí. Řady označené symboly spirály jsou bezpečné. V ostatních jsou propadliště. Vždy jen jedna dlaždice je pravá. Pro úspěšné vyhnutí se propadlištím je potřeba následovat sekvenci čísel spirály*.“

Bezpečný průchod hráči zajistí dlaždice jejichž pořadí zleva je shodné s čísly Fibonnaciho posloupnosti. Bylo zde vytvořeno pět řad a v každé jedna takováto dlaždice. Bezpečně lze tedy přejít po dlaždicích jejichž pořadí zleva je 1,1,2,3,5 viz Obrázek 38.



Obrázek 38: Bezpečné dlaždice

4.6 Stavy postav

Každá postava ve hře je ovládána prostřednictvím stavů. Jednotlivé stavy se od sebe liší dle účelu, ale jejich struktura je stejná. Obsahují metody *Enter*, *Tick* a *Exit*. Některé využívají i metodu *GetNormalizedTime*. Obecnou abstraktní třídu stavu zobrazuje Blok kódu 12. Metoda *Enter* je volána při vstupu do stavu. *Tick* probíhá po celou dobu trvání stavu a *Exit* probíhá při výstupu z daného stavu.

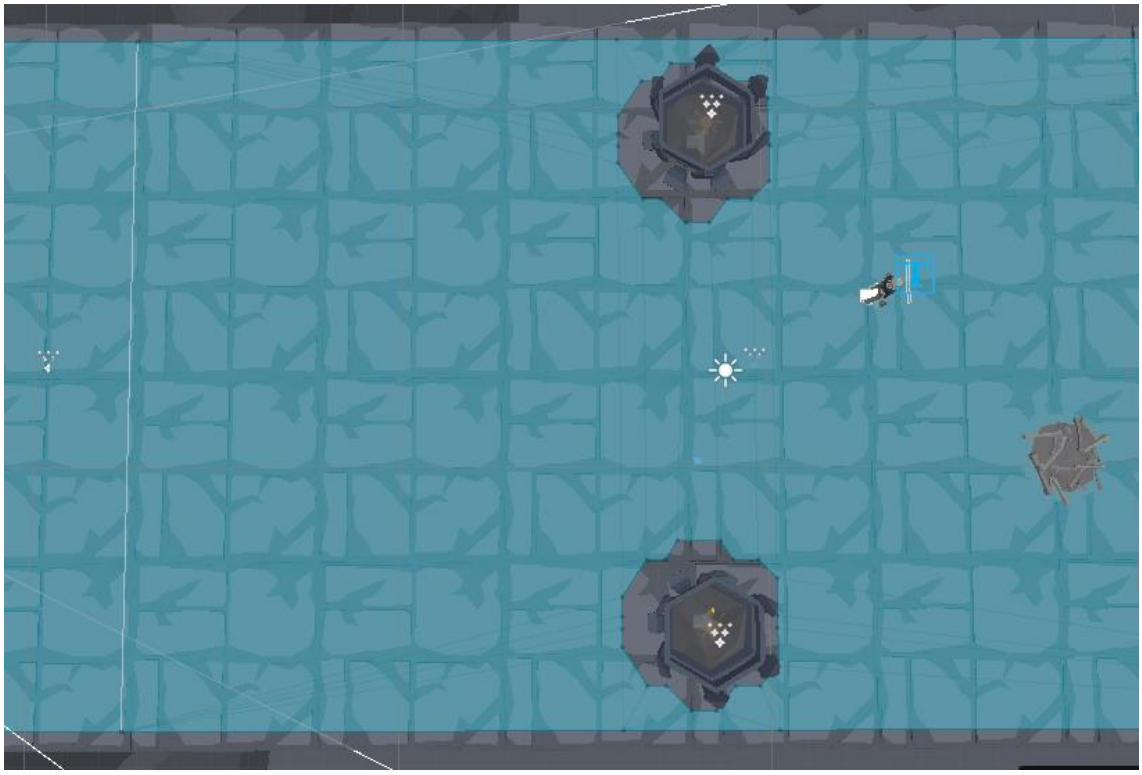
```
public abstract class State
{
    public abstract void Enter();
    public abstract void Tick(float deltaTime);
    public abstract void Exit();
    protected float GetNormalizedTime(Animator animator)
    {
        AnimatorStateInfo currentState = animator.GetCurrentAnimatorSta-
teInfo(0);
        AnimatorStateInfo nextState = animator.GetNextAnimatorStateInfo(0);
        if (animator.IsInTransition(0) && (nextState.IsTag("Attack") ||
nextState.IsTag("Blocked")))
        {
            return nextState.normalizedTime;
        }
        else if (!animator.IsInTransition(0) && (currentState.IsTag("At-
tack")|| currentState.IsTag("Blocked")))
        {
            return currentState.normalizedTime;
        }
        else
        {
            return 0f;
        }
    }
}
```

Blok kódu 12: Reprezentace základního stavu postav

Z této abstraktní třídy pak dědí třídy základních stavů hráče, nepřátelských postav a neutrálních postav. Ty obsahují rozšiřující metody, které jsou společné pro všechny stavy hráče, nepřátelských postav a neutrálních.

4.7 Navigace nehráčských postav ve hře

Pro navigaci nehráčských postav byl využit nástroj *NavMesh*. Všechny herní objekty, které představují nepohyblivé překážky byly označeny jako statické, aby se jim postavy vyhnuly.



Obrázek 39: Mapa NavMesh

Obrázek 39 zobrazuje mapu pro *NavMesh*. Modře je vyznačena plocha, po které se nehráčské postavy mohou pohybovat. Je vidět, že okraje, ani oblast v okolí překážek modré nejsou. To, aby nedošlo k narážení postav do objektů.

4.8 Implementace rozhodovacích modelů

V metodice byl navržen jednokritériový model pro chování nepřátelské postavy a vícekritériový. Obě programové řešení těchto modelů mají společného předka. Tím je abstraktní třída *DecisionModels*. Tato třída obsahuje pouze jednu abstraktní metodu, a to metodu *MakeADecision* viz Blok kódu 13.

```
public abstract class DecisionModels : MonoBehaviour
{
    public abstract void MakeADecision(EnemyStateMachine stateMachine);

}
```

Blok kódu 13: DecisonModels

4.8.1 Jednokritériový rozhodovací model

Pokud se hráč nachází mimo vzdálenost určenou pro detekci, nachází se nepřátelská postava ve stavu IDLE. Jakmile se hráčská postava dostatečně přiblíží, nepřátelská postava ji začne pronásledovat. Jakmile jsou hráčská a nepřátelská postava dostatečně blízko pro útok, nepřátelská postava útočí.

Tento model tedy neposkytuje nepřátelské postavě možnost blokování, nebo pohybu kolem. Je tedy vhodnější pro protivníky, kteří **jen a pouze** pronásledují hráče a útočí, pokud je v dosahu.

```
public class ModelRange : DecisionModels
{
    public override void MakeADecision(EnemyStateMachine stateMachine)
    {
        if (stateMachine.RangeController.IsInDetectionRange(stateMachine))
        {
            if(stateMachine.RangeController.IsInRange(stateMachine))
            {
                stateMachine.switchStates(new EnemyAttackingState(stateMa-
chine));
            }
            else
            {
                stateMachine.switchStates(new EnemyChasingState(stateMa-
chine));
            }
        }
        else
        {
            stateMachine.switchStates(new EnemyIdleState(stateMachine));
        }
    }
}
```

Blok kódu 14: Logika jednokritériového rozhodovacího modelu

4.8.2 Vícekritériový rozhodovací model

V rámci této práce byl vytvořen jeden vícekritériový rozhodovací model. Model je založen na stavu HP dané, vůči hráči nepřátelské, postavy a vzdálenosti od hráče. Rozhodnutí modelu je dále ovlivněno náhodou. Model vygeneruje pseudonáhodné číslo a pak pomocí hranic příslušnosti určí, následující stav. Vlastní program rozhodování reprezentuje Blok kódu 15.

```
public class HP_NPC_Range : DecisionModels
{
    [field: SerializeField] public HP_NPC_Range_Settings Settings { get;
set; }

    public override void MakeADecision(EnemyStateMachine stateMachine)
    {
        int difference = stateMachine.Health.GetMaxHealth() - stateMa-
chine.Health.GetHealth();
        float attackMax = stateMachine.Health.GetHealth() * Settings.At-
tack;
        //float CB = Settings.Circle + Settings.Block;
        float circleMax = difference * Settings.Circle + attackMax;
        float blockMax = difference * Settings.Block + circleMax;
        float value = Mathf.Round(Random.Range(1.0f, blockMax));
        //Debug.Log(value);
        //print(AttackMax);
        if (stateMachine.RangeController.IsInDetectionRange(stateMachine))
        {
            if(stateMachine.RangeController.IsInRange(stateMachine))
            {
                if (value >= 1.0f && value <= attackMax)
                {
                    //print("utok");
                    stateMachine.switchStates(new EnemyAttackingState(sta-
teMachine));
                }
                else if (value > attackMax && value <= circleMax)
                {
                    //Debug.Log("krouzim");
                    stateMachine.switchStates(new EnemyMoveAroundTar-
get(stateMachine));
                }
                else
                {
                    //print("blokuji");
                    stateMachine.switchStates(new EnemyBlockingState(state-
Machine));
                }
            }
            else
            {
                stateMachine.switchStates(new EnemyChasingState(stateMa-
chine));
            }
        }
        else
        {
            stateMachine.switchStates(new EnemyIdleState(stateMachine));
        }
    }
}
```

Blok kódu 15: Logika vícekritériového rozhodovacího modelu

Rozhodovací model obsahuje pouze jednu veřejnou proměnnou *Settings*. Ta slouží pro uložení nastavení modelu. Dále byla uvnitř modelu vytvořena metoda *MakeADecision*, jejímž úkolem je učinit rozhodnutí a zavolat metodu pro přepnutí do stavu dle rozhodnutí.

V této metodě existuje pět proměnných:

1. *difference* – reprezentuje rozdíl mezi maximálním a aktuálním počtem HP.

2. *attackMax* – horní hranice příslušnosti pro stav útoku. Počítá se jako počet HP krát vůle útoku.
3. *circleMax* – horní hranice příslušnosti pro stav kroužení. Výpočet je proveden jako *difference* krát vůle kroužení plus *attackMax*.
4. *blockMax* – horní hranice příslušnosti pro stav krytí se před útoky. Její hodnota je pak *difference* krát vůle krýt se před útoky plus *circleMax*.
5. value – pseudonáhodně vygenerovaná hodnota z intervalu <1; *blockMax*>

Na základě těchto hranic pak probíhá kontrola, zda *value* spadá do intervalu <1; *attackMax*>, v tomto případě přejde do stavu pro útok. Pokud *value* náleží intervalu (*attackMax*; *circleMax*>, proběhne přesun do stavu kroužení, jinak automaticky hodnota patří intervalu (*circleMax*; *blockMax*> a je spuštěn stav krytí se před útoky.

```
public class HP_NPC_Range_Settings : ScriptableObject
{
    [field: SerializeField] public int Number { get; set; }
    [field: SerializeField] public float Attack { get; set; }
    [field: SerializeField] public float Circle { get; set; }
    [field: SerializeField] public float Block { get; set; }
}
```

Blok kódu 16: Nastavení vícekritériového rozhodovacího modelu

Nastavení modelu obsahuje pořadové číslo a nastavení vůlí pro jednotlivé akce (stavy). Hledání optimální nastavení pro tento model proběhlo prostřednictvím genetického algoritmu viz Genetické algoritmy.

4.9 Genetické algoritmy

V rámci této práce proběhlo šlechtění modelu za pomoci genetického algoritmu. V rámci každé generace byla vytvořena populace o dvaceti jedincích. Před vytvořením nového jedince vždy proběhala kontrola unikátnosti daného jedince v rámci generace. Kontrola unikátnosti byla prováděna pro zajištění rozmanitosti populace a aby nedošlo k uváznutí v některém z lokálních extrémů. Hodnoty pro nastavení pak byly generovány v intervalu <1;5>. Tento interval tedy umožnil vzniknout až 5^3, tedy 125, různých možností. Existují sice možnosti, které by mohly být pokládány za stejné, například nastavení, pro která jsou všechny priority stejné (1,1,1|2,2,2|3,3,3|4,4,4|5,5,5). Tyto možnosti nebyly nijakým způsobem vyňaty a bylo jim umožněno vzniknout. Indexace generací probíhala od nuly. První, nultá, generace byla vytvořena čistě náhodně.

4.9.1 Zdatnost jedinců

Každý z jedinců v rámci generace odehrál pět zápasů se všemi jedinci v generaci včetně sebe sama, vždy v režimu jeden proti jednomu. Jedinec tedy odehrál sto her v generaci. Počet výher jedince byl použit jako jeho zdatnost. Čím vyšší měl tedy počet výher, tím vyšší byla jeho šance postoupit do další generace.

4.9.2 Tvorba nové generace

Tvorba nové generace je rozložena do tří částí. První částí je přesun pěti nejlepších do následující generace. Druhá část spočívá v jejich křížení, případně mutací. Třetí část tvoři tvorba čistě náhodných jedinců. Implementaci tvorby nové generace pak zobrazuje Blok kódu 17.

```
private void CreateNewGeneration()
{
    Directory.CreateDirectory("Assets/Game/Scripts/AI/Decision models/Mul-
tipleCriteriums/HP_NPC/Gen" + genCounterProperty + "/");
    string actual;
    float attack;
    float circle;
    float block;
    for (int i = 0; i < 5; i++)
    {
        attack = Results[i].Attack;
        circle = Results[i].Circle;
        block = Results[i].Block;
        actual = attack.ToString() + circle.ToString() + block.ToString();
        if (!used.Contains(actual))
        {
            CreateAnIndividual(attack, circle, block);
            used.Add(actual);
            AgentNumber++;
        }
    }
    for (int i = 0; i <= 9; i++)
    {
        CreateCrossover(Results[i], Results[i==9?0:i+1]);
    }
    CreateRandomIndividuals(5);
}
```

Blok kódu 17: Tvorba nové generace

Nejdříve dojde k vytvoření složky, pro uložení jedinců. Následuje deklarace proměnných. Proměnná *actual* je typu *string* a ukládá řetězcový součet jednotlivých přetypovaných nastavení jedince. Následují proměnné pro jednotlivá nastavení citlivostí (*attack*, *circle* a *block*). Následuje for cyklus, který ze **seřazeného** pole výsledků vybere prvních pět (indexy 0-4). Do deklarovaných proměnných uloží příslušná nastavení. Jejich přetypovaný součet pak uloží do deklarované proměnné *actual*. Následně dochází k preventivní kontrole, zda již pole použitých nastavení neobsahuje tento řetězec. Pokud by řetězec

obsahovalo, znamenalo by to, že jedinec pro danou generaci již existuje. Za předpokladu, že jedinec je generaci unikátní, proběhne jeho vytvoření a uložení do souboru s koncovkou .asset, což je, v tomto případě, uložený *ScriptableObject*. Po vytvoření a uložení jedince dojde k uložení jeho řetězcové reprezentace do pole použitých/existujících jedinců. Inkrementuje se proměnná *AgentNumber*, která ukládá pořadové číslo vytvářeného jedince v generaci.

Křížení

```
private void CreateCrossover(NPC_HP_Results first, NPC_HP_Results second)
{
    string actual;
    actual = first.Attack.ToString() + first.Circle.ToString() + se-
cond.Block.ToString();
    if (!used.Contains(actual))
    {
        CreateAnIndividual(first.Attack, first.Circle, second.Block);
        AgentNumber++;
        used.Add(actual);
    }
    else
    {
        int elementForMutation = UnityEngine.Random.Range(1, 3);

        float attack = elementForMutation == 1 ? Mathf.Round(UnityEn-
gine.Random.Range(1f, maxValue)) : first.Attack;
        float circle = elementForMutation == 2 ? Mathf.Round(UnityEn-
gine.Random.Range(1f, maxValue)) : first.Circle;
        float block = elementForMutation == 3 ? Mathf.Round(UnityEn-
gine.Random.Range(1f, maxValue)) : second.Block;

        actual = attack.ToString() + circle.ToString() + block.ToString();
        if (!used.Contains(actual))
        {
            CreateAnIndividual(attack, circle, block);
            AgentNumber++;
            used.Add(actual);
        }
        else
        {
            CreateRandomIndividuals(1);
        }
    }
}
```

Blok kódu 18: Křížení

Metoda pro křížení vytváří nové jedince do generace pomocí křížení. Následně kontroluje, zda už jedinec je v generaci. Pokud se jedinec již v generaci nachází, dochází k náhodné mutaci. Pokud i tento jedinec už v generaci existuje, dojde k vytvoření náhodného jedince.

Vytvoření náhodného jedince

```
private void CreateRandomIndividuals(int count)
{
    int i = 0;
    float randomAttack;
    float randomCircle;
    float randomBlock;
    string actual;
    while (i < count)
    {
        randomAttack = Mathf.Round(UnityEngine.Random.Range(1f, maxValue));
        randomCircle = Mathf.Round(UnityEngine.Random.Range(1f, maxValue));
        randomBlock = Mathf.Round(UnityEngine.Random.Range(1f, maxValue));
        actual = randomAttack.ToString()+randomCircle.ToString()+random-
Block.ToString();
        if (!used.Contains(actual))
        {
            CreateAnIndividual(randomAttack, randomCircle, randomBlock);
            AgentNumber++;
            i++;
            used.Add(actual);
        }
    }
}
```

Blok kódu 19: Tvorba náhodného jedince

Tento jedinec je tvořen zcela náhodně, dokud se nenalezne takový, který v generaci ještě není.

4.9.3 Výsledky genetického algoritmu

V rámci této práce byl genetický algoritmus spuštěn dvakrát. Vždy na třicet tři generací. Každá generace obsahovala dvacet jedinců, který hráli systémem každý s každým pět her. Jedinec tedy odehrál pět her v rámci jedné generace.

Oba běhy byly spuštěny za jiných podmínek. Poprvé jedincům byla odepřena možnost tzv. protiútoku, tj. možnost po úspěšnému vykrytí úderu protivníka přejít do útoku, který by měl navíc výhodu. Podruhé jedincům protiútok byl umožněn. Na protiútok měli šanci 1:3 a poškození bylo oproti základnímu útoku navýšeno o 50 procent (z deseti na patnáct). Bylo očekáváno, že možnost protiútoku s navýšeným poškození zatraktivní stavu krytí proti nepřátelským útokům.

Celá výsledková data pak jsou v příloze A.

Pro vyhodnocení nejlepších jedinců bylo možné použít dvě kritéria:

1. Průměrný počet výher na generaci.
2. Šance jedince na přesun do další generace **bez** křížení, náhodné mutace, nebo náhodným vytvořením.

Bez možnosti protiútoku

Kombinace			Průměrný počet výher
Attack	Circle	Block	
5	1	2	77,04
5	1	1	76,56
5	1	3	74,39
5	1	4	71,33
4	1	2	70,86

Tabulka 2: Pět nejlepších nastavení z hlediska počtu výher

Tabulka 2 zobrazuje pět nejlepších nastavení z hlediska průměrného počtu výher na generaci. Z výsledků je zřejmé, že vůle útočit je hlavní parametr, který ovlivní výsledek souboje. Na druhou stranu, vliv parametru kroužení, je v tomto modelu spíše na škodu. Parametr vůle chránit se v reakci na stav HP před útoky je poněkud zajímavější. Vůle chránit se se zdá být užitečnějším parametrem než vůle kroužení. Avšak touha chránit se nesmí být příliš vysoká. Pak už je spíše na škodu. Nicméně rozdíl v průměrném počtu výher není příliš markantní, obzvlášť pro první tři místa.

Kombinace			Šance na přesun do další generace
Attack	Circle	Block	
5	1	3	75,76 %
5	1	1	62,50 %
5	1	2	60,00 %
4	1	1	51,72 %
4	1	2	46,43 %

Tabulka 3: Pět nejlepších nastavení z hlediska šance na přesun do další generace

Při zaměření se na šanci přenést svůj genom do další generace, vznikají rozdíly markantnější viz Tabulka 3. Přesun do další generace v tomto případě znamená, že se jedinec **musí** umístit v první čtvrtině nejlepších z generace. Pokud tuto podmínku splní je automaticky přesunut do následující generace viz Tvorba nové generace. Zde se projevuje parametr vůle pro útok v souvislosti s HP také jako primární a parametr chránění se jako sekundární, nicméně je patrné, že důležitost chránění se oproti předchozímu kritériu stoupla.

Snížení parametru vůle chránit se z hodnoty 3 na hodnotu 2 přinese zvýšení průměrného počtu výher o 2,65, ale toto zvýšení bude na úkor snížení šance umístit se v první čtvrtině nejlepších o 15,76 %. Parametr chránit se tedy není vůbec špatný, protože umožňuje přežití jedince do další generace.

S možností protiútoku

Kombinace			Průměrný počet výher
Attack	Circle	Block	
5	1	1	72,50
4	1	1	68,04
5	1	2	67,04
5	1	4	66,63
5	1	3	64,87

Tabulka 4: Pět nejlepších nastavení z hlediska počtu výher

Přidáním jedincům šance na protiútok došlo ke snížení pozitivního vlivu vůle bránit se na výhru viz Tabulka 4. K tomuto došlo i přes zvýšení poškození, které by bylo tímto protiútokem uděleno. Pravděpodobně to souvisí s tím, že tento protiútok byl vynucený a nešlo o rozhodnutí modelu jako takového. Začala se zde více projevovat logika „prašť protivníka, než praští on tebe“.

Kombinace			Šance na přesun do další generace
Attack	Circle	Block	
5	1	1	96,67 %
5	1	2	77,78 %
4	1	1	62,07 %
3	1	1	44,83 %
5	1	4	40,63 %

Tabulka 5: Pět nejlepších nastavení z hlediska šance na přesun do další generace

Jak zobrazuje Tabulka 5, šance na přesun do další generace se tímto krokem výrazně zvýšila. U nastavení 511 rozdíl činí 34,17 % a u nastavení 512 17,78 %. Nicméně průměrný počet výher poklesl. Pro nastavení 511 pokles činí 4,06 a u nastavení 512 10. Nastavení 513 (vítěz předchozího běhu dle hlediska šance na přesun do další generace) se do prvních pěti v tomto běhu z hlediska šance na přesun do další generace ani nedostalo. Navíc pokles tohoto nastavení průměrného počtu výher činil 9,52.

Na základě těchto zjištění byla funkce šance na protiútok nepřátelským postavám odebrána.

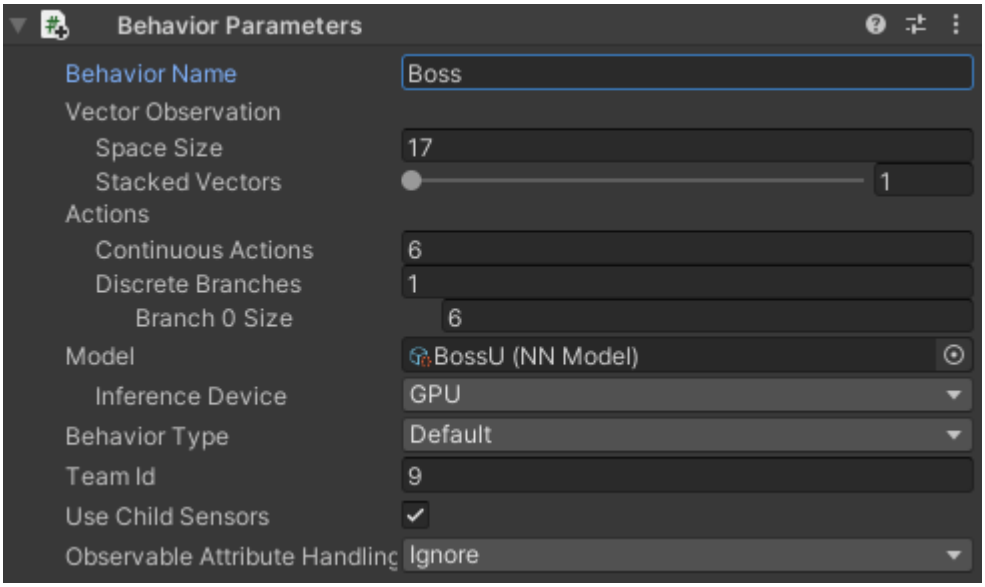
4.10 Neuronové sítě

Unity umožňuje práci s neuronovými sítěmi prostřednictvím assetu Unity-ML-Agents.

Tento asset dovoluje trénovat neuronovou síť prostřednictvím posilovacího učení, které zprostředkovává algoritmus *Proximal polici optimalization*. Lze jej ale i rozšířit prostřednictvím systému *Curiosity*, který odměňuje zkoušení nových akcí, nebo učením dle vzoru, *Apprenticeship learning*.

Pro chod package je však nezbytné mít nainstalovaný programovací jazyk Python. Dle dokumentace je podporována verze 3.8.13 a vyšší. V rámci této práce byla nainstalována verze 3.9.xy. Dále bylo nutné nainstalovat, knihovnu pro *Python*, *PyTorch* ve verzi 1.7.1.

Postava, která toto chování implementuje musí mít připojena script *Behavior Parameters*. Tomuto scriptu lze nastavit velikost sledovaného prostoru *Sapace Size*. Byla použita hodnota 17. Dále byl model nastaven, aby jeho výstupem bylo 6 akcí ve formátu desetinného čísla od -1 do 1. A jeden výstup ve formátu celočíselném od 0 do 5 viz Obrázek 40.



Obrázek 40: Behavior Parameters

Model pak byl trénován prostřednictvím soubojů. Agenti se utkávali s třemi různými nastaveními vícekritériového rozhodovacího modelu (5|1|1, 5|1|2 a 5|1|3). Dále se pak utkávali i mezi sebou. Dle metody byl implementován systém odměňování výsledkové funkce.

```
public override void OnActionReceived(ActionBuffers actions)
{
    float dodgeX = actions.ContinuousActions[0];
    float dodgeY = actions.ContinuousActions[1];
    float blockDuration = actions.ContinuousActions[2];
    float moveForward = actions.ContinuousActions[3];
    float moveRight = actions.ContinuousActions[4];
    float moveDuration = actions.ContinuousActions[5];
    int action = actions.DiscreteActions[0];



    switch (action)
    {
        case 0:
            if(_psm.ToAttack == 1 && _rc.IsInRange(_asm))
            {
                AddReward(-0.005f);
            }
            _asm.ToChase = -1;
            _asm.ToIdle = 1;
            _asm.ToAttack = -1;
            _asm.ToBlock = -1;
            _asm.ToMoveAround = -1;
            _asm.ToDodge = -1;
            _asm.switchStates(new EnemyIdleState(_asm));
            break;
        case 1:
            if (_rc.IsInDetectionRange(_asm) && !_rc.IsInRange(_asm))
            {
                AddReward(0.01f);
            }
            _asm.ToChase = 1;
            _asm.ToIdle = -1;
            _asm.ToAttack = -1;
            _asm.ToBlock = -1;
            _asm.ToMoveAround = -1;
            _asm.ToDodge = -1;
            _asm.switchStates(new EnemyChasingState(_asm));
            break;
        case 2:
            if(_psm.ToAttack != 1)
            {
                AddReward(0.001f);
                if (_rc.IsInRange(_asm))
                {
                    AddReward(-0.001f);
                    if(moveForward > 0.5f)
                    {
                        AddReward(-0.001f);
                    }
                }
                else
                {
                    AddReward(0.001f);
                    if(moveForward > 0.5f)
                    {
                        AddReward(0.001f);
                    }
                }
```

```
            }
            _asm.ToChase = -1;
            _asm.ToIdle = -1;
            _asm.ToAttack = -1;
            _asm.ToBlock = -1;
            _asm.ToMoveAround = 1;
            _asm.ToDodge = -1;
            _asm.switchStates(new EnemyMoveAroundTarget(_asm, moveForward,
moveRight, moveDuration));
            break;
        case 3:
            if (_psm.ToBlock == 1)
            {
                AddReward(-0.005f);
            }
            else if (_rc.IsInRange(_asm))
            {
                AddReward(0.006f);
                if (_asm.Health.GetHealth() > _psm.Health.GetHealth())
                {
                    AddReward(0.001f);
                }
                if(_previousState == 4 || _previousState == 3)
                {
                    AddReward(0.003f);
                }
            }

            _asm.ToChase = -1;
            _asm.ToIdle = -1;
            _asm.ToAttack = 1;
            _asm.ToBlock = -1;
            _asm.ToMoveAround = -1;
            _asm.ToDodge = -1;
            _asm.switchStates(new EnemyAttackingState(_asm));
            break;
        case 4:
            if (_psm.ToAttack == 1 && _rc.IsInRange(_asm))
            {
                AddReward(0.005f);
                if (_asm.Health.GetHealth() < _psm.Health.GetHealth())
                {
                    AddReward(0.005f);
                }
            }
            else
            {
                AddReward(-0.005f);
            }

            _asm.ToChase = -1;
            _asm.ToIdle = -1;
            _asm.ToAttack = -1;
            _asm.ToBlock = 1;
            _asm.ToMoveAround = -1;
            _asm.ToDodge = -1;
            _asm.switchStates(new EnemyBlockingState(_asm, blockDuration));
```

```
            break;
        case 5:
            if (_psm.ToAttack == 1 && _rc.IsInRange(_asm))
            {
                AddReward(0.005f);
                if (dodgeX > -0.5f && dodgeX < 0.5f)
                {
                    if (dodgeY > 0f)
                    {
                        AddReward(-0.004f);
                    }
                    else
                    {
                        AddReward(0.003f);
                    }
                }
                else
                {
                    AddReward(0.003f);
                }
            }
            else
            {
                AddReward(-0.005f);
            }

            _asm.ToChase = -1;
            _asm.ToIdle = -1;
            _asm.ToAttack = -1;
            _asm.ToBlock = -1;
            _asm.ToMoveAround = -1;
            _asm.ToDodge = 1;
            _asm.switchStates(new EnemyDodgingState(_asm, new
Vector3(dodgeX, dodgeY, 0)));
            break;
    }
    AddReward(-0.001f);
    //AddReward(-370f / MaxStep);

}
```

Obrázek 41: Odměňování akcí

Dále byly odměňovány i události navržené v metodice.

Trénování bylo spuštěno s následujícím nastavením:

```
behaviors:
  MLAgentv42Boss:
    trainer_type: ppo
    hyperparameters:
      batch_size: 4096
      buffer_size: 40960
      learning_rate: 3.0e-4
      beta: 5.0e-4
      epsilon: 0.2
      lambd: 0.93
      num_epoch: 7
      learning_rate_schedule: linear
      beta_schedule: constant
      epsilon_schedule: linear
    network_settings:
      normalize: true
      hidden_units: 1024
      num_layers: 10
      memory:
        memory_size: 16
    reward_signals:
      extrinsic:
        gamma: 0.96
        strength: 0.9
        network_settings:
          normalize: true
          hidden_units: 512
          num_layers: 4
      curiosity:
        gamma: 0.99
        strength: 0.1
        network_settings:
          normalize: true
          hidden_units: 16
          num_layers: 1
    keep_checkpoints:       30
    checkpoint_interval:    100000
    max_steps: 3000000
    time_horizon: 1000
    summary_freq: 10000
```

Obrázek 42: Nastavení trénování agenta

Tréning byl spuštěn pro 3‘000‘000 kroků, ale byl předčasně ukončen. Důvod ukončení byl ten, že agenti záměrně v prvním kroku uskočili dozadu a tím vyskočili z detekční vzdálenosti druhého agenta.

4.11 Příběh

Pro tuto práci byl vytvořen fiktivní příběh, odehrávající se ve fiktivním světě. Herní svět je podobný našemu ve středověkém období s výjimkou připuštění existence kouzel, ale nemá za účel odrážet realitu.

4.11.1 Prolog

Vypravěč v úvodu vypráví o období před válkou mezi magickými bytostmi a ostatními:

„*Vše začalo před několika staletími, za vlády Richarda I. Svět jej znal jako přítele pokroku. Velice rád zaváděl novinky, například desátek. K jeho dvoru přijížděli alchymisté a astrologové z celého světa s příslibem vytvoření zlata ze železa.*

Jednou si vyžádal u krále audienci muž, jehož obličej zahalovala kápě. Sňal ji teprve před králem. Jeho lebka se leskla, bradu s čelistmi zdobil plnovous. Pravil, že jen on může králi poskytnout, co žádá. Král zavelel přinést železo. Cizinec dostál svému slovu a železo přeměnil ve zlato. Richard I. přijal cizince do svých služeb. Jednoho dne požádal cizince o vytvoření stříbra. Holohlavý však odmítl. Král s cizincovou pomocí nesmírně zbohatl. I armádu vybavil zlatým brněním a zlatými zbraněmi. Ovšem ocelí vybaveni lapkové hravě prorazili zlaté brnění. Pobili mnoho královských rytířů, vybrali královské pokladnice a zlato se dostalo mezi lidi.

Uplynul rok a obydlí rolníků zdobilo mnoho zlata. Problém byl v tom, že museli nosit stále více a více zlata. Pekař požadoval za chléb zlatou cihličku. Rolníci měli více zlata než jídla. Rozprchli se tedy do sousedních zemí, kde se stali zámožnými. Richard I. chtěl, aby jeho království zůstalo nejbohatší a postupem času z holohlavého vymámil, kde žije jeho lid. Pak jej chladnokrevně zavraždil stříbrnou vidličkou. Svolal všechny své zbývající vojáky, poddané a vydali se do lesů smlouvat s lapky. Někteří se ke králi přidali. Společně pak vyzbrojeni stříbrem a ocelí vytáhli do boje proti čarodějům a magickým bytostem. Válka to byla krutá a nemilosrdná. Vzhledem k nedostatku oceli a stříbra se zbraně předávali z generace na generaci.

Nyní je potřeba dokončit, co Richard I. začal.“

4.11.2 Příběh hry

Po úvodu je hráči králem zadán úkol, který spočívá ve výpravě do jeskyně nemrtvých a vrátit se s ní. Jako odměna je postavě hráče slíbeno povýšení do šlechtického stavu a hrad. Jakmile hráč přijde ke knize, dostane na výběr ze dvou možností:

1. Přečíst knihu a vrátit se s ní.
2. Jen se s knihou vrátit.

Na základě hráčovi volby je pak učiněn výběr následující fáze příběhu. Pokud hráč zvolí první volbu, je konfrontován králem a musí se spolu utkat. Následně je hra přesunuta do epilogu, který je volen podle vítěze souboje. Pokud ale hráč vybere druhou volbu, hra se přesune do epilogu, kde je hráčova postava odměněna dle slibu krále.

4.11.3 Epilog

Hra nabízí celkem tři možnosti zakončení.

Hráč knihu nepřečetl

„*Podařilo se ti knihu v pořádku doručit a odolal jsi i pokušení ji přečíst. Kniha byla zničena, načež všichni nemrtvý padli. Dle královského slova, jsi byl povýšen do šlechtického stavu a nyní jako odměnu za své služby vlastníš hrad.*“

Hráč knihu přečetl a zvítězil v souboji

„*Pokušení knihu přečíst bylo silnější než ty. Vesnice za vesnicí se ocitá pod útokem armády nemrtvých, kterou vedeš. Podařilo se ti i přivést zpět ostatní magické bytosti, ale jen jako nemrtvé. Ty teď zaplavují celý svět.*“

Hráč knihu přečetl a prohrál v souboji

„*Knihu jsi doručil, ale přečetl jsi ji. Král tě v souboji přemohl, následně knihu zničil a zabránil tak návratu magických bytostí.*“

5 Závěr

Vytvořená hra umožňuje ovládání prostřednictvím klávesnice a myši, nebo herního ovladače. Obsahuje i krátký příběh a hráč ji je schopen odehrát v řádu minut. Inteligence nepřátelských postav je založena na třech variantách.

První variantou je jednokritériový rozhodovací model, který má omezená akce.

Dalším modelem je pak model vícekritériový, který má širší spektrum akcí a jehož optimální nastavení bylo hledáno prostřednictvím genetického algoritmu.

Poslední inteligence, se kterou se hráč může setkat, je založena na neuronových sítích a strojovém učení. Tato inteligence má nejširší škálu možných akcí.

Cílem práce bylo navrhnout a realizovat 3D hru, a to se podařilo.

Chování inteligence vytvořené s pomocí neuronové sítě není dokonalé a mohlo by být dosaženo lepšího prostřednictvím jiného nastavení odměn nebo jiné hodnotící funkce.

Hra by také mohla být zlepšena přidáním interakcí s více postavami a vytvořením více úkolů pro hráče.

I Summary and keywords

I.1 Summary

The goal ofthis thesis was to design and create a 3D game with the use of graphical assets. In particular, th Unity game engine was used to Create the game. Visual Studio and Rider were used to write the program. In the game, the player can encounter enemies whose decison making is based on different methods. The first on eis making decisions according to single criterion. The next is a multi-criteria based decision model. Preferences for each action can be set to the model. The search for the optimal setting was performer using a genetic algorithm. The decison making of the third enemy that the player may encounter is based on the use of neural networks, whose output is then converted into the action that the enemy character performs.

I.1 Keywords

Game, Unity, decison trees, genetic algorithm, neural networks

II Citovaná literatura

Alza. (n. d.). *ALZA Muzeum: Historie počítačového hraní.* (1). Praha.

BASLER, J. (2016). COMPUTER GAMES, THEIR DIVISION, CONTEMPORARY DEVELOPMENT TENDENCIES AND BASIC INVESTIGATIONS FROM THE FIELD OF COMPUTER GAMES. *Trends in Education*, *9*(1), pp. 20-27.

Berka, P. (2003). *Dobývání znalostí z databází.* (1.). Praha: Academia.

Berne, E. (2011). *Jak si lidé hrají.* (1). Praha: Portál.

Bourg, M., & Seemann, G. (2004). *AI for game developers.* (1). Cambridge: O´Reilly.

Brown, P., Roediger, H., & McDaniel, M. (2017). *Nauč se to!: jak se s pomocí vědy efektivněji učit a více si pamatovat.* (). V Brně: Jan Melvil Publishing.

DocFX. (2023a). . Retrieved from: https://docs.unity3d.com/Packages/com.unity.cinemachine@2.9/manual/index.html

DocFX. (2023b). . Retrieved from: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.5/manual/index.html

DocFX. (2023c). . Retrieved from: https://docs.unity3d.com/Packages/com.unity.localization@1.4/manual/index.html

DocFX. (2023d). . Retrieved from: https://docs.unity3d.com/Packages/com.unity.timeline@1.8/manual/index.html

Harari, Y. (2018). *Sapiens: stručné dějiny lidstva.* (V nakladatelství Leda vydání třetí). Voznice: Leda.

Hardman, C. (2020). *Game Programming with Unity and C#: A Complete Beginner's Guide.* (1). USA: Apress.

Hladký, J., & Leitmanová, I. (2000). *Mikroekonomie II.* (1.). České Budějovice: Jihočeská univerzita.

Hořejší, B., Soukupová, J., Macáková, L., & Soukup, J. (2018). *Mikroekonomie.* (6. aktualizované a doplněné vydání). Praha: Management Press.

Huberman, A. (2022). Using Play to Rewire & Improve Your Brain. In: *YouTube*. Retrieved from: https://youtu.be/BwyZIWeBpRw

Juliani, A. (2020). Unity: A general platform for intelligent agents.

Koster, R. (2013). *A Theory of Fun for Game Design.* (2.). United States of America: O'Reilly.

Mařík, V., Štěpánková, O., & Lažanský, J. (1993). *Umělá inteligence*. (1). Praha: Academia.

Mařík, V., Štěpánková, O., & Lažanský, J. (2001). *Umělá inteligence (3).* (1.). Praha: Academia.

Nimrod, the World's First Gaming Computer. In: *Wired*. (2010). Retrieved from: https://www.wired.com/2010/06/replay/

Peterson, J. (2023). *Mapy smyslu: Architektura přesvědčení.* (1). Praha: Argo.

Sedláček, T. (2017). *Ekonomie dobra a zla: po stopách lidského tázání od Gilgameše po finanční krizi.* (3. vydání). Praha: 65. pole.

UNITY TECHNOLOGIES. (2022a). . Retrieved from: https://docs.unity3d.com/2022.1/Documentation/Manual/class-AudioSource.html

UNITY TECHNOLOGIES. (2022b). . Retrieved from: https://docs.unity3d.com/2022.1/Documentation/Manual/class-MonoBehaviour.html

UNITY TECHNOLOGIES. (2022c). . Retrieved from: https://docs.unity3d.com/2022.1/Documentation/Manual/class-ScriptableObject.html

Zandl, P. (2022). *Mýty a naděje digitálního světa: vše, co potřebujete vědět o kryptoměnách, umělé inteligenci a dalších převratných technologiích.* (). V Brně: Jan Melvil Publishing.

The British Museum. (2021). Top 10 historical board games [Online]. Retrieved from https://www.britishmuseum.org/blog/top-10-historical-board-games

III Seznam obrázků, tabulek a bloků kódu

III.1 Obrázky

III.2 Tabulky

III.3 Bloky kódu

IV Seznam příloh

A Výsledky genetického algoritmu

A.1 Bez možnosti protiútoku

Kombinace			První generace	Počet možných generací	Počet generací	Počet výher na generaci	Počet přesunů	Počet promarněných možností	Šance na přesun do další generace
Attack	Circle	Block							
1	1	1	0	33	5	32,80	0	33	0,00 %
1	1	2	10	23	4	30,25	0	23	0,00 %
1	1	3	22	11	2	27,00	0	11	0,00 %
1	1	4	12	21	6	17,17	0	21	0,00 %
1	1	5	26	7	1	15,00	0	7	0,00 %
1	2	1	0	0	0	0,00	0	33	0,00 %
1	2	2	3	30	3	15,00	0	30	0,00 %
1	2	3	0	33	3	14,33	0	33	0,00 %
1	2	4	6	27	2	3,00	0	27	0,00 %
1	2	5	0	0	0	0,00	0	33	0,00 %
1	3	1	9	24	3	4,33	0	24	0,00 %
1	3	2	25	8	1	15,00	0	8	0,00 %
1	3	3	0	0	0	0,00	0	33	0,00 %
1	3	4	26	7	2	3,00	0	7	0,00 %
1	3	5	25	8	1	6,00	0	8	0,00 %
1	4	1	0	0	0	0,00	0	33	0,00 %
1	4	2	2	31	2	5,00	0	31	0,00 %
1	4	3	0	0	0	0,00	0	33	0,00 %
1	4	4	11	22	1	0,00	0	22	0,00 %
1	4	5	8	25	1	1,00	0	25	0,00 %
1	5	1	23	10	1	0,00	0	10	0,00 %
1	5	2	24	9	1	7,00	0	9	0,00 %
1	5	3	19	14	1	1,00	0	14	0,00 %
1	5	4	1	32	2	1,50	0	32	0,00 %
1	5	5	14	19	1	0,00	0	19	0,00 %
2	1	1	1	32	6	62,00	1	31	3,13 %
2	1	2	5	28	10	58,20	2	26	7,14 %
2	1	3	1	32	7	43,43	0	32	0,00 %
2	1	4	16	17	5	41,40	0	17	0,00 %
2	1	5	17	16	2	31,50	0	16	0,00 %
2	2	1	11	22	3	26,33	0	22	0,00 %
2	2	2	1	32	7	25,14	0	32	0,00 %
2	2	3	1	32	4	32,00	0	32	0,00 %
2	2	4	0	33	4	32,75	0	33	0,00 %

2	2	5	30	3	1	26,00	0	3	0,00 %
2	3	1	2	31	3	19,67	0	31	0,00 %
2	3	2	0	33	5	23,60	0	33	0,00 %
2	3	3	4	29	3	25,67	0	29	0,00 %
2	3	4	19	14	3	25,00	0	14	0,00 %
2	3	5	23	10	1	35,00	0	10	0,00 %
2	4	1	3	30	4	19,75	0	30	0,00 %
2	4	2	3	30	3	16,00	0	30	0,00 %
2	4	3	3	30	4	17,25	0	30	0,00 %
2	4	4	23	10	2	5,50	0	10	0,00 %
2	4	5	0	33	1	18,00	0	33	0,00 %
2	5	1	24	9	1	1,00	0	9	0,00 %
2	5	2	4	29	2	13,00	0	29	0,00 %
2	5	3	6	27	1	20,00	0	27	0,00 %
2	5	4	0	33	4	7,25	0	33	0,00 %
2	5	5	6	27	1	26,00	0	27	0,00 %
3	1	1	3	30	22	66,86	8	22	26,67 %
3	1	2	1	32	19	65,47	4	28	12,50 %
3	1	3	2	31	18	60,22	5	26	16,13 %
3	1	4	0	33	12	58,58	3	30	9,09 %
3	1	5	29	4	1	59,00	0	4	0,00 %
3	2	1	2	31	4	60,00	1	30	3,23 %
3	2	2	2	31	9	41,11	0	31	0,00 %
3	2	3	8	25	7	45,00	0	25	0,00 %
3	2	4	10	23	4	27,75	0	23	0,00 %
3	2	5	19	14	1	37,00	0	14	0,00 %
3	3	1	2	31	2	42,00	0	31	0,00 %
3	3	2	0	33	1	53,00	0	33	0,00 %
3	3	3	0	33	5	33,80	0	33	0,00 %
3	3	4	1	32	4	35,75	0	32	0,00 %
3	3	5	1	32	2	29,50	0	32	0,00 %
3	4	1	5	28	5	29,80	0	28	0,00 %
3	4	2	0	33	5	21,00	0	33	0,00 %
3	4	3	0	33	5	30,20	0	33	0,00 %
3	4	4	0	33	3	27,33	0	33	0,00 %
3	4	5	19	14	1	51,00	0	14	0,00 %
3	5	1	0	33	3	30,33	0	33	0,00 %
3	5	2	4	29	2	23,00	0	29	0,00 %
3	5	3	27	6	2	8,00	0	6	0,00 %
3	5	4	14	19	2	6,00	0	19	0,00 %
3	5	5	19	14	1	11,00	0	14	0,00 %

4	1	1	4	29	27	70,78	15	14	51,72 %
4	1	2	5	28	22	70,86	13	15	46,43 %
4	1	3	6	27	22	70,50	10	17	37,04 %
4	1	4	6	27	12	68,58	5	22	18,52 %
4	1	5	27	6	2	56,50	0	6	0,00 %
4	2	1	0	33	7	55,29	1	32	3,03 %
4	2	2	7	26	7	53,71	1	25	3,85 %
4	2	3	9	24	8	49,25	0	24	0,00 %
4	2	4	1	32	4	52,50	1	31	3,13 %
4	2	5	0	0	0	0,00	0	33	0,00 %
4	3	1	9	24	2	43,00	0	24	0,00 %
4	3	2	3	30	6	35,00	0	30	0,00 %
4	3	3	2	31	6	30,17	0	31	0,00 %
4	3	4	0	33	6	39,00	0	33	0,00 %
4	3	5	0	0	0	0,00	0	33	0,00 %
4	4	1	2	31	5	43,00	0	31	0,00 %
4	4	2	5	28	5	25,20	0	28	0,00 %
4	4	3	1	32	10	39,80	0	32	0,00 %
4	4	4	0	33	4	33,25	0	33	0,00 %
4	4	5	3	30	3	27,33	0	30	0,00 %
4	5	1	20	13	1	10,00	0	13	0,00 %
4	5	2	7	26	3	17,00	0	26	0,00 %
4	5	3	1	32	3	37,67	0	32	0,00 %
4	5	4	5	28	3	23,33	0	28	0,00 %
4	5	5	0	0	0	0,00	0	33	0,00 %
5	1	1	1	32	27	76,56	20	12	62,50 %
5	1	2	3	30	25	77,04	18	12	60,00 %
5	1	3	0	33	31	74,39	25	8	75,76 %
5	1	4	3	30	15	71,33	8	22	26,67 %
5	1	5	5	28	6	64,83	1	27	3,57 %
5	2	1	3	30	20	64,65	6	24	20,00 %
5	2	2	6	27	13	64,38	4	23	14,81 %
5	2	3	2	31	14	62,00	3	28	9,68 %
5	2	4	7	26	8	60,13	2	24	7,69 %
5	2	5	5	28	4	53,00	0	28	0,00 %
5	3	1	11	22	7	45,71	0	22	0,00 %
5	3	2	0	33	5	60,20	1	32	3,03 %
5	3	3	1	32	10	54,80	1	31	3,13 %
5	3	4	3	30	5	43,20	0	30	0,00 %
5	3	5	16	17	1	51,00	0	17	0,00 %
5	4	1	5	28	6	40,67	0	28	0,00 %

5	4	2	0	33	5	46,40	0	33	0,00 %
5	4	3	21	12	4	38,25	0	12	0,00 %
5	4	4	30	3	1	35,00	0	3	0,00 %
5	4	5	4	29	1	62,00	0	29	0,00 %
5	5	1	2	31	6	31,83	0	31	0,00 %
5	5	2	31	2	1	28,00	0	2	0,00 %
5	5	3	0	33	4	48,50	1	32	3,03 %
5	5	4	1	32	2	37,50	0	32	0,00 %
5	5	5	30	3	1	41,00	0	3	0,00 %

A.2 S možností protiútoku

Kombinace			První generace	Počet možných generací	Počet generací	Počet výher na generaci	Počet přesunů	Počet promarněných možností	Šance na přesun do další generace
Attack	Circle	Block							
1	1	1	9	24	4	33,25	0	24	0,00 %
1	1	2	12	21	3	39,00	0	21	0,00 %
1	1	3	2	31	1	40,00	0	31	0,00 %
1	1	4	18	15	1	32,00	0	15	0,00 %
1	1	5	0	0	0	0,00	0	33	0,00 %
1	2	1	0	0	0	0,00	0	33	0,00 %
1	2	2	1	32	2	25,00	0	32	0,00 %
1	2	3	14	19	5	16,80	0	19	0,00 %
1	2	4	0	0	0	0,00	0	33	0,00 %
1	2	5	0	33	4	20,25	0	33	0,00 %
1	3	1	0	33	2	15,50	0	33	0,00 %
1	3	2	31	2	1	17,00	0	2	0,00 %
1	3	3	2	31	2	12,00	0	31	0,00 %
1	3	4	0	33	2	13,00	0	33	0,00 %
1	3	5	13	20	1	11,00	0	20	0,00 %
1	4	1	11	22	1	8,00	0	22	0,00 %
1	4	2	18	15	1	13,00	0	15	0,00 %
1	4	3	19	14	1	9,00	0	14	0,00 %
1	4	4	0	33	6	10,00	0	33	0,00 %
1	4	5	18	15	2	14,50	0	15	0,00 %
1	5	1	26	7	1	10,00	0	7	0,00 %
1	5	2	0	0	0	0,00	0	33	0,00 %
1	5	3	5	28	3	10,00	0	28	0,00 %
1	5	4	0	0	0	0,00	0	33	0,00 %
1	5	5	0	0	0	0,00	0	33	0,00 %
2	1	1	7	26	11	55,45	0	26	0,00 %
2	1	2	5	28	9	54,00	1	27	3,57 %

2	1	3	3	30	9	47,22	0	30	0,00 %
2	1	4	2	31	5	48,00	0	31	0,00 %
2	1	5	0	0	0	0,00	0	33	0,00 %
2	2	1	5	28	4	38,25	0	28	0,00 %
2	2	2	0	33	7	41,14	0	33	0,00 %
2	2	3	3	30	5	32,00	0	30	0,00 %
2	2	4	5	28	1	38,00	0	28	0,00 %
2	2	5	1	32	6	34,50	0	32	0,00 %
2	3	1	2	31	1	34,00	0	31	0,00 %
2	3	2	0	33	3	39,33	1	32	3,03 %
2	3	3	1	32	4	26,25	0	32	0,00 %
2	3	4	7	26	5	25,00	0	26	0,00 %
2	3	5	0	33	3	24,00	0	33	0,00 %
2	4	1	0	33	3	25,00	0	33	0,00 %
2	4	2	0	33	4	26,25	0	33	0,00 %
2	4	3	22	11	2	12,00	0	11	0,00 %
2	4	4	7	26	6	19,83	0	26	0,00 %
2	4	5	25	8	1	9,00	0	8	0,00 %
2	5	1	21	12	2	16,00	0	12	0,00 %
2	5	2	5	28	3	14,67	0	28	0,00 %
2	5	3	13	20	2	13,00	0	20	0,00 %
2	5	4	14	19	3	15,67	0	19	0,00 %
2	5	5	0	0	0	0,00	0	33	0,00 %
3	1	1	4	29	22	62,95	13	16	44,83 %
3	1	2	6	27	17	57,29	3	24	11,11 %
3	1	3	4	29	8	60,50	2	27	6,90 %
3	1	4	3	30	10	54,40	0	30	0,00 %
3	1	5	3	30	5	54,60	1	29	3,33 %
3	2	1	1	32	8	51,63	0	32	0,00 %
3	2	2	2	31	4	47,00	0	31	0,00 %
3	2	3	0	33	5	46,60	0	33	0,00 %
3	2	4	2	31	5	39,80	0	31	0,00 %
3	2	5	0	33	3	59,00	2	31	6,06 %
3	3	1	0	33	4	42,75	0	33	0,00 %
3	3	2	2	31	4	35,25	0	31	0,00 %
3	3	3	9	24	4	37,75	0	24	0,00 %
3	3	4	7	26	3	31,00	0	26	0,00 %
3	3	5	17	16	3	35,67	0	16	0,00 %
3	4	1	21	12	2	44,00	0	12	0,00 %
3	4	2	13	20	4	29,25	0	20	0,00 %
3	4	3	0	33	4	34,50	0	33	0,00 %

3	4	4	5	28	4	26,25	0	28	0,00 %
3	4	5	2	31	1	41,00	0	31	0,00 %
3	5	1	31	2	1	25,00	0	2	0,00 %
3	5	2	11	22	4	25,75	0	22	0,00 %
3	5	3	0	33	1	43,00	0	33	0,00 %
3	5	4	7	26	3	24,67	0	26	0,00 %
3	5	5	11	22	2	19,00	0	22	0,00 %
4	1	1	4	29	25	68,04	18	11	62,07 %
4	1	2	4	29	24	63,17	11	18	37,93 %
4	1	3	4	29	14	63,50	6	23	20,69 %
4	1	4	2	31	11	57,64	2	29	6,45 %
4	1	5	0	0	0	0,00	0	33	0,00 %
4	2	1	2	31	11	55,18	2	29	6,45 %
4	2	2	5	28	10	56,20	1	27	3,57 %
4	2	3	6	27	6	50,67	0	27	0,00 %
4	2	4	5	28	7	50,14	0	28	0,00 %
4	2	5	0	33	5	58,20	1	32	3,03 %
4	3	1	1	32	7	51,29	0	32	0,00 %
4	3	2	3	30	5	46,00	0	30	0,00 %
4	3	3	1	32	6	52,67	1	31	3,13 %
4	3	4	1	32	9	42,78	0	32	0,00 %
4	3	5	0	33	2	59,00	1	32	3,03 %
4	4	1	6	27	6	37,67	0	27	0,00 %
4	4	2	1	32	4	47,75	0	32	0,00 %
4	4	3	2	31	6	41,00	0	31	0,00 %
4	4	4	0	33	3	45,33	1	32	3,03 %
4	4	5	17	16	1	33,00	0	16	0,00 %
4	5	1	0	33	1	59,00	0	33	0,00 %
4	5	2	1	32	4	38,00	0	32	0,00 %
4	5	3	5	28	5	27,80	0	28	0,00 %
4	5	4	0	33	2	50,00	0	33	0,00 %
4	5	5	0	0	0	0,00	0	33	0,00 %
5	1	1	3	30	30	72,50	29	1	96,67 %
5	1	2	6	27	27	67,04	21	6	77,78 %
5	1	3	8	25	15	64,87	7	18	28,00 %
5	1	4	1	32	16	66,63	13	19	40,63 %
5	1	5	2	31	3	57,67	2	29	6,45 %
5	2	1	1	32	24	63,04	11	21	34,38 %
5	2	2	4	29	19	62,84	5	24	17,24 %
5	2	3	0	33	13	56,92	2	31	6,06 %
5	2	4	3	30	8	51,13	0	30	0,00 %

5	2	5	1	32	7	53,71	2	30	6,25 %
5	3	1	8	25	11	53,55	0	25	0,00 %
5	3	2	10	23	7	50,86	1	22	4,35 %
5	3	3	16	17	2	45,00	0	17	0,00 %
5	3	4	11	22	3	48,00	0	22	0,00 %
5	3	5	9	24	1	42,00	0	24	0,00 %
5	4	1	2	31	6	50,33	0	31	0,00 %
5	4	2	16	17	5	44,20	0	17	0,00 %
5	4	3	6	27	4	42,50	0	27	0,00 %
5	4	4	6	27	3	44,00	0	27	0,00 %
5	4	5	0	0	0	0,00	0	33	0,00 %
5	5	1	5	28	2	43,00	0	28	0,00 %
5	5	2	18	15	2	44,00	0	15	0,00 %
5	5	3	27	6	1	42,00	0	6	0,00 %
5	5	4	10	23	2	44,50	0	23	0,00 %
5	5	5	13	20	2	38,50	0	20	0,00 %