TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Vývoj umělé inteligence v interaktivních médiích

Diplomová práce

Studijní program:	N6209 Systémové inženýrství a informatika
Studijní obor:	Manažerská informatika
Autor práce:	**Bc. David Vejvoda**
Vedoucí práce:	Ing. Petr Weinlich, Ph.D. Katedra informatiky

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Zadání diplomové práce

Vývoj umělé inteligence v interaktivních médiích

Jméno a příjmení: **Bc. David Vejvoda**
Osobní číslo: E18000359
Studijní program: N6209 Systémové inženýrství a informatika
Studijní obor: Manažerská informatika
Zadávající katedra: Katedra informatiky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Historie vývoje umělé inteligence ve videohrách
2. Analýza trendů a současných technologiích pro tvorbu umělé inteligence
3. Aplikace UI pro postavy ve 3D hře
4. Zhodnocení a doporučení

Rozsah grafických prací:
Rozsah pracovní zprávy: 65 normostran
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

1. BARTÁK, Roman. 2017. *Co je nového v umělé inteligenci*. Praha: Nová beseda. Co je nového. ISBN 978-80-906751-2-4.
2. FENG, Jie a Peter L. NEWTORN. 2016. *Unreal Engine 4 AI Programming Essentials: Jie Feng, Peter L. Newton*. Birmingham, Velká Británie: Packt Publishing Limited. ISBN 9781784393120.
3. DILLON, Roberto. 2011. *The golden age of video games: the birth of a multi-billion dollar industry*. Boca Raton, FL: A K Peters/CRC Press. ISBN 978-143-9873-236.
4. STANTON, Richard. 2015. *A brief history of video games*. Philadelphia, PA: Running Press Book Publishers, a member of the Perseus Books Group. ISBN 978-076-2456-154.
5. PROQUEST. 2019. Databáze článků ProQuest [online]. Ann Arbor, MI, USA: ProQuest. [cit. 2019-09-26]. Dostupné z: http://knihovna.tul.cz

Konzultant: Ing. Dana Nejedlová, Ph.D.

Vedoucí práce: Ing. Petr Weinlich, Ph.D.
Katedra informatiky

Datum zadání práce: 31. října 2019
Předpokládaný termín odevzdání: 31. srpna 2021

L.S.

prof. Ing. Miroslav Žižka, Ph.D.
děkan

doc. Ing. Klára Antlová, Ph.D.
vedoucí katedry

V Liberci dne 31. října 2019

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

7. května 2020 Bc. David Vejvoda

Anotace

Diplomová práce se soustředí na vývoj umělé inteligence ve videohrách. To zahrnuje historický vývoj napříč několika dekádami od prvních her s umělou inteligencí na bázi instrukcí až po hry využívající neuronové sítě. Dále se práce zabývá několika metodami pro řešení umělé inteligence ve hrách. V dalších částech jsou předmětem práce herní enginy, kde jsou rozebírány jednotlivé části enginu a následně se práce věnuje dostupným enginům na trhu, které podrobněji analyzuje. Hlavní náplní diplomové práce je návrh a následná implementace umělé inteligence do 3D hry.

Klíčová slova

Herní engine, počítačové hry, umělá inteligence, videohry

Annotation

Development of artificial intelligence in interactive media

The diploma thesis focuses on the development of artificial intelligence in video games. This includes historical developments over several decades from the first artificial intelligence based on instruction logic to neural network. Furthermore, the work deals with several methods for solving artificial intelligence in games. In the following parts, the subject of the work are game engines, where the individual parts of the engine are analyzed and then the work deals with the available engines on the market, which are analyzed in more detail. The main content of the diploma thesis is the design and subsequent implementation of artificial intelligence in 3D game.

Key Words

Artificial intelligence, computer game, game engines, videogames

Poděkování

Tímto bych chtěl poděkovat Ing. Petru Weinlichovi, Ph.D. za odborné vedení, pomoc, trpělivost a cenné rady při tvorbě této práce. Dále bych chtěl poděkovat Ing. Daně Nejedlové, Ph.D. za inspiraci a předané znalosti během předmětu Umělá inteligence, které dopomohly ke vzniku této práce. V neposlední řadě bych chtěl poděkovat své rodině bez jejíž podpory by tohle všechno nebylo možné.

Obsah

Seznam obrázků

Seznam použitých zkratek

2D	dvourozměrný
3D	trojrozměrný
AI	Artificial Intelligence (umělá inteligence)
BB	BlackBoard
CEO	Chief executive officer (výkonný ředitel)
CPU	centrální procesorová jednotka
DT	decision tree (strom rozhodování)
ESRB	Entertainment Software Rating Board
FMS	Finite State Machine
FPS	first-person shooter (střílečka z pohledu první osoby)
GDC	Game Developers Conference
GPU	graphics processing unit (grafický procesor)
HDR	High Dynamic Range
MCST	Monte Carlo Search Tree
MMO	Massively Multiplayer Online
MS-DOS	Microsoft Disk Operating System
NPC	Non Playable Character (postava řízená umělou inteligencí)
RPG	role playing game (hra na hrdiny)
RTS	real-time strategy (realtimová strategie)
SDK	software development kit (sada vývojových nástrojů)
VR	virtuální realita

Úvod

Videohry a umělá inteligence jsou dnes neodlučitelné pojmy. Již od padesátých tet dvacátého století je umělá inteligence nedílnou součástí všech počítačových her a v dnešních dnech i našich životů. Počítačové hry za bezmála sedmdesát let ušli dlouho cestu. První hry sice vznikli pár let po konci druhé světové války, ale trvalo ještě několik dekád, než vznikl herní průmysl a z her se stalo médium pro kreativní vyjadřování. Narozdíl však od knih, filmů či dalšího druhu umění jsou hry jediná umělecká díla založená na interakci a interpretaci. Lidé se mohou častokrát přít o tom co daný obraz vyobrazuje, ale stále ve výsledku vždy koukají na jeden a tentýž fyzický objekt. U her je to jiné. Herní zážitek každého hráče se liší v závislosti na jeho interakci. Oscar Wilde kdysi řekl, že skrze interpretaci se člověk sám stává umělcem. Toto je jádro všech videoher. Každá jednotlivá kopie hry je identická a přesto, si každý hráč odnese jiný zážitek. K budování těchto zážitků tak v mnoha případech dopomáhá právě i umělá inteligence.

Tato práce si mimo jiné klade za cíl seznámit čtenáře se stručnou historii videoher a vývojem umělé inteligence ve hrách napříč několika desetiletími. Zabývá se počátky herního průmyslu a vznikem prvních her. Nástupem herních automatů, prvních konzolí a domácích počítačů. Dále příchodem prvních 3D her, prvních botů ovládaných umělou inteligencí, neuronovými sítěmi, procedurálním generováním v oblasti videoher a především metodami pro tvorbu umělé inteligence ve hrách. Tohle vše by v dnešní době nebylo možné bez patřičných nástrojů pro tvorbu videoher. K tomu slouží herní enginy, kterým se tato práce také věnuje. Hlavním cílem této práce je návrh a následná implementace umělé inteligence pro 3D hru.

1 Historie videoher a vývoj umělé inteligence

Tato kapitola se věnuje stručné historii videoher a vývoji umělé inteligence ve hrách napříč několika desetiletími. Zabývá se počátky herního průmyslu a vznikem prvních her. Nástupem herních automatů, prvních konzolí a domácích počítačů. V druhé polovině této kapitoly se práce věnuje příchodu prvních 3D her, prvních botů ovládaných umělou inteligencí, neuronovými sítěmi a procedurálním generováním v oblasti videoher.

1.1 Základní vymezení

Oxfordský slovník definuje videohru takto: *„ a game in which you press buttons to control and move images on a screen hru*“, v překladu tedy *„ hra kde hráč ovládá dění na obrazovce pomocí několika tlačítek“* (Oxford Dictionary, 2020) . U nás se však spíše vžil pojem počítačová hra. Oba názvy však znamenají totéž. Interaktivní médium.

Co se týče umělé inteligence tak tu definuje oxfordský slovník následovně: *„ the study and development of computer systems that can copy intelligent human behaviour“*, což můžeme přeložit jako *„studium a vývoj počítačových systémů, které mohou kopírovat inteligentní lidské chování“*(Oxford Dictionary, 2020). Jednoznačně však definovat tento pojem není možné, a to zejména z důvodu víceoborového zaměření této disciplíny. Nejvíce užívanou definicí je definice Marvina Minskyho z roku 1967, jenž vychází Turingova testu a zní takto: „*Umělá inteligence je věda o vytváření strojů nebo systémů, které budou při řešení určitého úkolu užívat takového postupu, který, kdyby ho dělal člověk, bychom považovali za projev jeho inteligence.“* (Minsky, 1967). V rámci této práce se však budeme převážně bavit o umělé inteligenci, která užívá takového postupu, který ji vnuknul její stvořitel.

1.2 Počátky

Počítačové hry jsou tu s námi již bezmála sedmdesát let. Počátky datujeme do roku 1952, kdy Alexander S. Douglas, tou dobou student na univerzitě v Cambridge, dokončil svoji disertační práci zabývající se interakcí mezi člověkem a počítačem. Součástí této práce byl i program, jenž Douglas naprogramoval na počítači EDSAC. V podstatě šlo o jednoduchou verzi piškvorek s hrací plochou 3x3 políčka. Tak vznikla i první počítačová hra, která nesla název OXO. Samotná hra se ovládala velmi sofistikovaným způsobem, a to pomocí telefonního otočného číselníku. Pro grafický výstup sloužila upravená obrazovka osciloskopu. Obrazovka, jež byla

připojená k počítači EDSAC umožnovala zobrazení bitmapy [1] o rozměrech 35x16 pixelu, což se může zdát jako velmi málo, ale pro tehdejší potřeby hry to bylo naprosto dostačující. Spolu s první počítačovou hrou spatřilo světlo světa i první umělá inteligence, která se na bázi instrukcí vytvořených Douglasem, snažila porazit svého lidského protivníka. (Švára, 2013)

O šest let později, roku 1958, vzniká další důležitý projekt, na kterém pracuje William Higinbotham. Člověk, který se během druhé světové válce podílel na vývoji radaru a následně pracoval i na projektu Manhattan[2]. Projekt, který Higinbotham spolu se svými kolegy vytvořili nese název Tennis for Two. Jak již název napovídá, jedná se o tenis pro dva hráče. Tenisový kurt jde zde zobrazen z pohledu z boku a je složen ze dvou statických čar. Jedna z čar, ta vodorovná, zobrazuje povrch a druhá, svislá, představuje síť. Jediným aktivním prvkem tedy byl míček, který se odrážel od země a pokud narazil do sítě, tak se ztratil část své rychlosti. Hráči mohli ovládat virtuální neviditelnou tenisovou raketu pomocí speciálně sestaveného ovladače. Stisknutím tlačítka na ovladači tak mohli odrážet míček zpět na soupeřovu polovinu. Tennis for Two běžel na speciálně pro účely hry sestaveném analogovém počítači. Z tohoto důvodu nebylo technický možné implementovat jakýkoliv druh umělé inteligence, která by se zhostila role protivníka. Pro zobrazování byl opět použit osciloskop. Hra pak byla předvedena v rámci dnu otevřených dveří Brookhaven Lab, kde sklidila velký úspěch. (Tišnovský, 2014)

Hra, které nepřímo odstartovala vznik herního průmyslu byla vytvořena v roce 1962 na minipočítač PDP-1 a jmenovala se SpaceWar! Na svědomí ji měl Stephen Russel spolu se svými kolegy z tehdejších laboratořích umělé inteligence na MIT. Zatímco předchozí dvě hry byli ve své podstatě simulacemi. OXO bylo ztvárnění reálné stolní hry ve virtuálním prostředí a a v případě Tennis for Two šlo o simulaci sportu. Spacewar! však zašel mnohem dál a vytvořil si svět vlastní. Na kruhové obrazovce byla vyobrazena plocha, která znázorňovala vesmír s hvězdami. Na herní ploše se pohybovala dvojice vesmírných lodí, které byly ovládány samotnými hráči, kteří po sobě mohli střílet rakety. V tu dobu se jednalo o velmi náročnou hru, protože bylo nutné počítat trajektorii vesmírných lodí, raket a několik druhů kolizí. Spacewar! tak slaví na univerzitě obrovský úspěch a je velmi populární hlavně mezi studenty. (inventions et al. 2019) Jedním z těchto studentů je i Nolan Bushnell, který o několik let později zakládá společnost Atari a vydává svou verzi hry Spacewar! kterou nazve Computer Space. (Donovan, 2010)

[1] reprezentace rastrového obrazu v paměti počítače ve formě pixelů
[2] krycí název pro utajený americký vývoj atomové bomby za 2. světové války

Obrázek 1: Spacewar!
Zdroj: (inventions et al., 2019)

Ještě však než se dostaneme ke zlaté éře herních automatů a prvních konzolí, je třeba zmínit několik dalších her, které stály u zrodu žánrů. Jedna z těchto her je Hamurabi, kde se hráč pomocí textového rozhraní ujme role babylonského krále Chammurapiho. Jako král pak následně spravuje zdroje, jako jsou lidé, půda a obilí. Každý člověk může obhospodařovat stanovené množství půdy, která produkuje obilí. Zrno může být zase použito ke krmení lidí, kteří jinak zemřou hlady nebo k zasetí plodin na následující rok. Hráč může také nakupovat nebo prodávat pozemky svým sousedům, výměnou za obilí. Všechny tyto zdroje jsou spravovány v průběhu několik kol, z nichž každé představuje rok. Každá nová hra byla odlišná, a to díky tomu, že přírůstek zdrojů ať už záporný nebo kladný, byl ovlivňován generátorem náhodných čísel. Samotná hra pak běžela na dvanácti bitovém minipočítači PDP-8. Hamurabi vznikl původně jako naučná počítačová hra pro žáky základní školy. Autorkou byla učitelka Mabel Addis, jenž se tak stala první ženou, která se podílela na tvorbě počítačové hry. Spolu s programátorem této hry Williamem McKayem stáli u zrodu zcela nového žánru, budovatelských strategií. (Willaert, 2019)

Další hrou, kterou je nutné zmínit je Space Travel. Hra samotná nebyla nijak revoluční, spíše, než o hru šlo o simulaci cestování vesmírem. Cílem hry, bylo pokusit se přistát svou vesmírnou

lodí na různých planetách či měsících. Space Travel bylo vytvořeno Kenem Thompsonem a jeho kolegy z Bellových laboratoří, a to na sálovém počítači GE-635. Nicméně vedení firmy, se nelíbilo, že takto výkonný počítač jako je GE-635, je využíván k pro hraní hry namísto seriózní práce. Ken Thompson se nehodlal vzdát práce na Space Travel a proto se pokusil o konverzi na méně výkonné PDP-7. V rámci konverze byl Thompson donucen vytvořit vlastní operační systém. Systém, ze kterého následně vzešel Unix. (Švára, 2013)

V roce 1972 však vznikla ještě jedna důležitá hra, která následně definovala celý nový žánr. Tato hra se jmenovala Empire. Stál za ní Peter Langston, který se nechal inspirovat stejnojmennou deskovou hrou. Empire je tahová 4X[3] hra, kde hráč velí jednotkám, které se pohybují po herním světě, který je rozdělen do čtvercové sítě. Každý čtverec značí určitý druh políčka ať už jde o zemědělské políčko, průmyslové nebo i vodní plochu. Tyto políčka následně generují různé suroviny, které pak slouží k výrobě jednotek. Jednotky mohou prozkoumávat herní svět, obsazovat území a útočit na soupeře. Empire tak položil základní kameny žánru tahových strategií, a především vydláždil cestu dnes již legendární Civilizaci. (Tišnovský, 2014)

1.3 Příchod arkádových automatů a konzole první generace

Počátkem 70. let 20 století započal vznik arkádových automatů, které postupně začali zaplavovat bary, restaurace, kina až pro ně nakonec vznikla samostatné arkádové herny. S příchodem arkádových automatů datujeme také vznik videoherního průmyslu jako součást zábavního segmentu, kam se zařadil bok po boku k velikánům jako je ten filmový nebo hudební.

1.3.1 První arkády

První arkádovou hrou, tj. hrou, k jejímu spuštění je potřeba vhodit minci do arkádového automatu, byla Galaxy Game. Arkádový automat byl poprvé nainstalován v září 1971 na Stanfordské univerzitě v budově Tresidder Union, a to dva měsíce před vydáním Computer Space, první masově vyráběné videohry pro arkádové automaty. Za Galaxy Game stáli dva programátoři Bill Pitts a Hugh Tuck. Stejně jako v případě Computer Space šlo o vylepšenou verzi videohry Spacewar! (Stanton, 2015)

[3] 4X – explore, expand, exploit, exterminate

Vývoj hry stál okolo 20 000 dolarů, drtivá většina všech výdajů však padla pouze na sestrojení arkádového automatu, který se skládal z počítače DEC PDP-11 a vektorové zobrazovací jednotky. Pro hraní hry bylo nutné zaplatit 10 centů za jednu hru, popřípadě 25 centů za hry tři. Původně byl postaven pouze jeden arkádový automat, ale v roce 1972 po hardwarovém vylepšení bylo umožněno hraní více uživatelům, kteří tak mohli hrát proti sobě. Samotná hra zůstala v kampusu Stanfordské univerzity velmi populární a každý kdo si ji chtěl zahrát musel v průmětu čekat až hodinu, než na něho vyšla řada. Automat byl z půdy univerzity odstraněn až v roce 1979 a to z důvodu mechanického poškození obrazovky. (Glancey, 1996)

Jak již bylo zmíněno, dva měsíce po vydání Galaxy Game, spatřilo světlo světa další arkádová verze Spacewar! s názvem Computer Space. Za ní stálo duo Nolan Bushnell a Ted Dabney. Společnost Nutting Associates od nich vykoupila práva a vyrobila přes 1500 kusů arkádových automatů se hrou Computer Space. Hra samotná však byla neúspěšná, a to především kvůli náročnosti, kterou kladla na tehdejší hráče. Arkádové automaty se hrou Computer Space mizí tak pohořeli, nicméně si zasloužili prvenství alespoň v tom, že se jedná o první masově vyráběnou videoherní arkádu, která byla nabízena pro komerční užití.
Bushnell a Dabney se však nevzdávají a spolu zakládají společnost Atari. (Glancey, 1996)

1.3.2 Videoherní konzole

V roce 1972 doráží na americký trh první komerčně prodávaná videoherní konzole Magnavox Odyssey. Se samotným nápadem na vytvořit herní konzoli přišel její tvůrce Ralp Henry Baer již v roce 1966. Další tři roky spolu se svými dvěma spolupracovníky Billem Harrisonem a Billem Ruschem vytvořili několik prototypů a až s jejich sedmým prototypem přesvědčili společnost Magnavox, která následně svolila k masové výrobě. Konzole se skládala z několika boxů, které byli připojeny do televize a také ze dvou obdélníkových ovladačů připojených pomocí kabelů. Konzole byla schopna pomocí televizní obrazovky zobrazovat tři pixely na obrazovce v černobíle obměně dle černého nebo bílého podkladu. Chování pixelů se lišilo v závislosti na hrané hře. Hry byli samozřejmě bez zvuku, neboť chyběl jakýkoliv audio výstup. Hráči byli následně i nuceni na obrazovku umísťovat plastové krycí vrstvy, aby tak mohli vzniknout vizuální prvky potřebné pro danou hru. Hry se ovládali pomocí třech otočných knoflíků a jednoho tlačítka umístěných na ovladači. Součástí balení bylo také několik fyzických doplňků nezbytných pro hraní několika her jako například kostky, papírové peníze atd. (Stanton, 2015)

Konzole Magnavox Odyssey se prodalo přes 350 000 kusů. Cena, za kterou se prodávala byla tehdejších 100[4] dolarů. Na samotnou konzoli během jejího životního cyklu, který trval 3 roky, bylo vydáno 28 her. (Stanton, 2015) Jedna z těchto her byla Table Tennis.

1.3.3 Vzestup Atari

V roce 1972 se v hlavě Nolana Bushnella tvoří další nápad na hru, tentokrát opět inspirovanou již vytvořeným dílem. V tomto případě právě Table Tennisem na Magnavox Odyssey. Kontaktuje tedy Allana Alcorna s nápadem na vytvoření podobné hry. Samotný Alcorn neměl s vytvářením her žádné zkušenosti, nicméně to byl velmi zdatný elektrotechnik. Bushnell požadoval, aby hra obsahovala realistické zvuky, což se zprvu zdálo jako nemožné, neboť Alcorn zjistil, že mu dochází místo na desce plošných spojů[5] a nevěděl, jak vytvořit takové zvuky pomocí digitálních obvodů. Alcorn si dokázal poradit a objevil jeden obvod, který by mu umožnil generovat potřebné zvuky. Po třech měsících tedy hra uměla zvuky, požadované Bushnellem. Ke zkonstruování prototypu Alcornovi posloužila černobílá televize, kterou pořídil v místním obchodě, jež následně umístil do 1,2 metru vysoké dřevěné skříně a následně propájel jednotlivé obvody. Tak se zrodil PONG. Ten zaujal Bushnella s Dabneym natolik, že se ho rozhodli otestovat na veřejnosti. V září 1972 tak byl první arkádový automat se hrou PONG nainstalován v baru Andy Capp's Tavern, v severní kalifornii (Glancey 1996). PONG se stal instantním hitem a jeho popularita začala raketově růst. Pro Bushnella a Dabneyho to byl dostatečný důvod začít masovou výrobu. V první roce se jim podařilo prodat 2500 automatů a v roce následujícím to bylo již 8000 kusů. Atari tak následně prodalo přes 35 000 arkádových automatů se hrou PONG, který odstartoval zlatou éru herních automatů a položil základní kámen videoherního průmyslu. (Stanton, 2015)

1.4 Zlatá éra arkádových videoher a první domácí počítače

Zlatá éra arkádových automatů byla doba velkých technických pokroků a růstu kreativity v herním designu. Herní vývojáři museli pracovat s omezeným výkonem procesoru a také s malou pamětí, proto také museli být mnohdy velmi kreativní. Videohry pro arkádové automaty se rozprostírali žánrově do mnoha směrů, což také vedlo k obrovskému rozšíření arkád a

[4] Při započtení inflace by se dnešní cena pohybovala okolo 611 dolarů.
[5] Deska určená k osazení elektronických součástek, které propojují vodiče vytvořené v tenké kovové vrstvě na povrchu desky.

speciálních gameroomů po Severní Americe, Evropě a také po Japonsku. Zároveň se arkádové automaty začaly častěji objevovat po obchodních centrech, restauracích, barech, hospodách, a dokonce i na čerpacích stanicích. Během této éry také došlo k mnoho experimentům s novým herním hardwarem a několika druhy vstupních zařízení pro videohry. Kromě klasických joysticků a tlačítek tak vznikl i první volant nebo světelné pistole. S obrovským úspěchem raných videoherních arkád naskočili do herního průmyslu další vývojáři a velké firmy jmenovitě například japonské Namco, SEGA, Capcom nebo Nintendo. Někteří jednoduše zkopírovali stávající hry a jednoduše je vydávali za vlastní, zatímco jiní zkoušeli nové a unikátní věci a tím definovali nové žánry.

1.4.1 Atari vs. Apple

Atari po obrovském úspěchu se hrou PONG, začíná nabírat nové zaměstnance a značně se rozšiřuje. Jedním z nových zaměstnanců je i sedmnácti letý vysokoškolský odpadlík Steve Jobs, který se k Atari přidává v roce 1974. Na pozici technika pracuje na basketbalové videohře. Jedním z přátel Steva Jobse je také další velmi zdatný technik Steve Wozniak, který byl velkým fanouškem her od Atari a kterého po večerech Jobs pašoval do práce, aby zde mohli spolu zdarma hrát závodní hru Gran Trak. Gran Trak vyšel v roce 1974 a jedná se o první video herní arkádový automat, který se ovládá volantem. Wozniak, kromě hraní Gran Traku pomáhal Jobsovi po večerech i s různými technickými problémy což následně vedlo k tomu, že začal pro Atari pracovat na plný úvazek. V roce 1976 Jobs a Wozniak dokončuji arkádový automat se hrou Breakout. (Glancey, 1996)

Společná láska k technologiím vedla Jobse a Wozniaka k založení vlastní společnosti. Po nocích pracovali v garáži na prvním osobním počítači, který následně pojmenovali Apple I. Samotný počítač nebyl nijak zvlášť výkonný, nicméně, než se nadáli měli na stole několik objednávek. Steve Jobs si v tu chvíli uvědomil, že v rukách mají hotový zlatý důl. Takže zatímco Steve Wozniak pracoval na technické stránce počítače, Jobs se snažil najít cestu, jak získat co nejvíce potenciálních zákazníků pro jejich produkt. Věděl, že nemůže cílit jen na technické nadšence, ale že musí mířit na širší okruh lidí. Proto při navrhování počítače Apple II museli brát tento fakt v potaz. Jobs také věděl, že jestli se budou chtít více rozrůst, budou potřebovat kapitál. Obrátil se tedy na svého bývalého šéfa Nolan Bushnella s nabídkou podílu ve firmě. Ten však odmítl, neboť on sám již měl plány na to, jak přivést hry od Atari do domácností. (Glancey, 1996)

Bushnell se totiž nechal inspirovat první domácí konzolí Magnavox Odyssey, která byla již několik let na trhu. V roce 1977 tak vychází Atari 2600. Konzole uměla zobrazit barevnou grafiku, umělá přehrávat zvuk a co bylo důležité, hry byli uloženy na tzv. cartridgích[6]. Takže kdokoliv mohl začít vytvářet hry pro jejich konzoli a prodávat je na cartridgich. Nicméně 2600 byla extrémně drahá na výrobu a Atari samotné nemělo prostředky k tak masové výrobě jakou by si přáli (Stanton, 2015). Nolan Bushnell se tedy rozhodl sehnat investory. Nakonec se dohodl se společností Warner Communications (nyní Warner Media), kteří se nabídli, že odkoupí celé Atari. To zprvu Bushnell odmítá, ale nakonec s odkupem s těžkým srdcem souhlasí. Bushnell sice na odkupu obrovsky zbohatne, a dokonce zůstává ve vedení Atari, ale záhy se ukáže, že jeho vize a vize nových majitelů jsou naprosto odlišné. Jednou z vážných rozepří mezi Bushnellem a Warner Communications byla otázka týkající se vstupu Atari na trh s osobními počítači. Bushnell chtěl s jejich počítačem Atari 800 trumfnout Apple II. Atari 800 tak byla hardwarově navržena tak, aby strčila počítač od Applu hravě do kapsy. Grafická a zvuková vybavenost počítače Atari tak byla mnohem lepší, než měl Apple II. Co však 800 zlomilo vaz bylo rozhodnutí Warner Communications, že nikdo další nebude moci psát software pro jejich počítač. Narozdíl od Applu, který vehementně tvorbu softwaru třetích stran pro jejich zařízení podporoval. Takže zatímco Apple II oplýval velkým množstvím softwaru všeho druhu, Atari 800 po této stránce velmi střádala. Bushnell také požadoval, aby Warner snížili cenu samotné konzole 2600, protože se mu zdála přemrštěná. Chtěl, aby se místo toho spoléhali na příjem z prodeje cartridgů, který by rostl, právě díky většímu počtu lidí, kteří by si konzoli mohli dovolit. Warneři to však razantně odmítli. Bushnell, zdrcen nemožností něco změnit ve firmě, kterou sám založil, tak roku 1978 odchází z Atari. (Glancey, 1996)

1.4.2 Mainframe scéna

Zatímco arkádové automaty a domácí konzole zažívali boom, stále se našlo několik nadšenců, kteří vytvářeli hry pro mainframové počítače. Během dekády se začali totiž ustalovat programovací jazyky jako BASIC nebo C, které práci značně usnadňovali. Sálové počítače umožňovaly práci více uživatelů a také došlo k rozšíření ARPANETu, což vedlo k tomu, že mnoho univerzitních studentů začalo vytvářet hry, které následně mezi sebou sdíleli. Zatímco arkádové hry a hry pro konzole byli zaměřené na rychlou akci tak mainframové hry byli zcela

[6] paměťové médium, které obsahuje paměťové obvody typu ROM. V současnosti stále využívány například společnosti Nintendo.

odlišné. Bylo to dáno především omezeností hardwaru sálových počítačů. Ačkoliv dosahovaly většího čistého výkonu než arkády nebo konzole té doby, nutnost rozdělit výpočetní prostředky mezi několik uživatelů prostřednictvím tzv. sdílení času[7], významně omezila jejich výkon. Proto se programátoři her pro sálové počítače zaměřili žánrově na strategické, logické a puzzle hry namísto čisté akce. V průběhu 70.let tak vznikli hry jako Star Trek, Hunt the Wumpus, nebo již dříve zmíněný Empire. Za nejvíce důležitou hru lze označit Colossal Cave Adventure, nebo jak se ji později začalo zkráceně říkat, Adventure. Adventure vytvořil v roce 1976 Will Crowther. Jednalo se o textovou hru založenou na konceptech deskové hry Dungeon and Dragons[8]. Ve hře hráč ovládal postavu pomocí textového rozhraní a pomocí textových příkazů prozkoumával jeskyni, na jejímž konci měl být ukrytý poklad. Koncept hry byl následně rozvíjen několika dalšími lidmi. Jedním z těchto lidí byl například Don Woods, který Adventure zasadil do světa J.R.R Tolkiena. Hráč se tak mohl ocitnout ve Středozemi. Adventure tak dalo vzniknout zcela novému žánru textových adventur a nepřímo se zasadilo o vznik žánru her na hrdiny. (Tišnovský, 2014)

Zatímco většina her byla vytvořena na hardwaru s omezenou výpočetním výkonem, jedním počítačem schopným hostovat působivější hry byl systém PLATO vyvinutý na Illinoiské Univerzitě. Systém byl navržen jako vzdělávací počítač a dokázal připojit stovky uživatelů po celých Spojených státech prostřednictvím vzdálených terminálů, které obsahovaly vysoce kvalitní plazmové displeje a umožňovaly uživatelům vzájemně komunikovat v reálném čase. To umožnilo hostovat celou řadu graficky náročných her, a to včetně těch pro více hráčů. Převážně šlo o vůbec první hry na hrdiny (RPG), které byli deriváty Adventure, jež vycházelo z D&D pravidel. Některé z nich tak díky většímu výkonu mohli klást větší důraz na boj a vývoj postavy nežli pouze na řešení hádanek. Začali se tak objevovat hry jako The Dungeon, Moria nebo Oubliette, které umožnovali připojit se k ostatním hráčům a společně bojovat s nestvůrami a vyrážet na výpravy. Hry jako Adventure silně ovlivnili vývoj raných her na osobní počítače. (Stanton, 2015)

1.4.3 Arkádový boom

Videoherní průmysl vzkvétal a Atari jasně vévodilo celému trhu se svým širokým katalogem her pro 2600 a především díky svým arkádovým automatům. S tituly jako PONG, Asteroids

[7] Sdílení výpočetního výkonu mezi několika uživateli současně pomocí multiprogramingu a multitaskingu
[8] Dungeon and Dragons – u nás v ČR známá pod názvem Dračí Doupě

nebo Missile Commands určovalo směr. Alespoň do chvíle, než přišli Space Invaders. Společnost Taito zaměřena spíše na pinballové stroje a jukeboxy se totiž rozhodla, že vstoupí trh s arkádovými automaty a nemohla si přát lepší start než se Space Invaders. Space Invaders byla vůbec první "střílečka", která položila základní kámen pro svůj vlastní shoot'em up žánr. Cílem hry bylo porazit vlny mimozemských lodí, které postupně v několika horizontálních lajnách sestupovali dolů a útočili na hráče. Hráč ovládal pozemní stroj, se kterým musel sestřelit co nejvíce mimozemských lodí a získat tak nejvíce bodů. Ze Space Invaders se stal obrovský úspěch pro společnost Taito a také jedna z nejvíce výdělečných her všech dob. Není proto divu, že je mnohými označována jako jedna z her, která nejvíce ovlivnila herní průmysl. Space Invaders totiž roku 1978 odstartovali zlatý věk arkádových videoher. (Stanton, 2015)

Obrázek 2: Space Invaders
Zdroj: (Palazzesi, 2011)

Přiživit se na úspěchu Space Invaders se snažilo několik firem, mezi nimi byla i společnost Namco. Ta v roce 1979 vydává sofistikovanější verzi Space Invaders nazvanou Galaxian. Hra má barevnou grafiku a je o něco těžší než Space Invaders. (Glancey, 1996)

V roce 1980 přichází Atari se hrou Battlezone. Ta používá vektorovou grafiku místo rastru, což umožnilo vytvořit drátové "3D" prostředí. Ve světě plném 2D her šlo o převratnou novinku. Hráč se ve hře zhostil ovládání tanku, jenž měl za úkol zničit další nepřátelské tanky. Společnost Atari byla následně oslovena armádou Spojených států amerických, aby jim vytvořili více

“realistickou” verzi této hry. Jedná se o první propojení her a armádního výcviku v historii. (Stanton, 2015)

Ve stejném roce vychází i hra Defender za níž stojí Eugene Jarvis, jedna z nejvýznamnějších postav zlaté éry arkádových automatů. Jarvis byl mistr ve vytváření extrémně obtížných her, které však byli naprosto férové vůči hráči. Což znamenalo, že hry byli skutečná výzva, avšak pokud hráč neuspěl mohl si za to pouze sám svou chybou. Toto se několik let později stane jádrem jednoho samostatného žánru tzv. souls like her. Defender byl první 2D side-scrolling střílečka, která však byla velmi komplexní. Hráč měl za úkol chránit havarované astronauty, zatímco musel ničit jednu vlnu útočících mimozemšťanů za druhou. (Stanton, 2015)

V roce 1980 přichází hra, která se stane legendou, Pac-Man. V záplavě akčních střílečeк jako Space Invaders, Defender nebo Galaxian, působil Pac-Man jako naprosté zjevení. Designer Toru Iwatani uvádí, že chtěl vyvinout hru, kterou by si mohl zahrát každý. Ve hře samotné se tak zhostíte role Pac-Mana, který se pohybuje po 2D bludišti a snaží se sníst všechny tečky. V bludišti však není sám, pohybují se po něm čtyři duchové řízení umělou inteligencí, kteří se Pac-Mana snaží naopak sníst. Hráč se tak musí duchům vyhýbat a snažit se získat power-up, který mu umožní stát se na chvíli z kořisti lovcem a zničit tak duchy. Cílem hry bylo získat následně co nejvíce bodů. Pac-Man je první hra v historii, kde byli použity vzorce chování. To znamená, že každý z duchů měl svou vlastní “osobnost” a dle toho se choval každý z nich odlišně. Během dvou let se prodalo 400 000 kusů arkádových automatů se hrou, což Namcu přineslo zisk 2,5 miliard dolarů. To z Pac-Mana dělá nejvíce výdělečnou arkádovou hru historie. (Stanton 2015) Pac-Man je často titulován jako největší milník v historii videoher. Postava se objevuje ve více než 30 oficiálně licencovaných herních spin-offech, stejně tak jako v mnoho neautorizovaných klonech. Podle Davie-Brownova indexu[9], je Pac-Man nejvíce rozpoznovánou značkou mezi videohrami pro americké konzumenty, kde značku samotnou poznává až 94 % vzorku. Z Pac-mana se během 80. let stala ikona videoherní kultury. (BMI Gaming, nedatováno)

[9] nezávislý index pro brand marketéry a agentury, který kvantifikuje vnímání spotřebitelů

1.5 Pád Atari a vzestup Nintenda

Atari ohromena úspěchem Pac-Mana, rozhodla o koupení licence na hru pro jejich konzoli. Vývojáři z Atari se tak snažili předělat Pac-Mana pro Atari 2600. Ke konci roku 1982 se Atari podařilo vyexpedovat 7 milionů kusů, z toho však 5 milionů jim zůstávalo na skladě a drtivý zbytek na skladě obchodníků. O hru nebyl zájem. To byl jeden z hřebíčků do rakve velikánovy jako je Atari. V témže roce vedení Warner Communications rozhodlo o koupi práv pro nadcházející film Steva Spielberga, E.T. CEO Atari Ray Kassar tento nápad označil za velmi hloupý. I přes nelibost Kassara, tak Atari bylo nuceno zaplatit 25 milionů dolarů za licenci a vytvořit hru během šesti týdnu. Toto rozhodnutí vlastníka Atari, Warner Communications se tak stalo firmě, která stála u zrodu videoherního průmyslu osudným. E.T. je dodnes považována jako nejhorší hra historie. Atari vyrobilo přes 5 milionů kusu cartridgů o které neměl nikdo zájem a které nakonec skončili zakopané v poušti v Novém Mexiku. Atari tak padlo ke dnu a muselo vyklidit prostor pro nové mohykány jako Nintendo nebo Sega. (Stanton, 2015)

Japonské Nintendo bylo tou dobou číslo jedna na japonském trhu, nicméně raketového vzestupu se dočkalo i na ostatních trzích, a to díky hře Donkey Kong, která vyšla v roce 1981. Ačkoliv nešlo u první příběhovou hru, byla to první hra, kde byli vysvětleny motivace jednotlivých aktérů. Poprvé se zde objevil i slavný Mario, který měl za úkol osvobodit dámu v nesnázích ze spáru mohutné opice Donkey Konga. Jedná se o první klasickou plošinovku, kde se hráč musí dostat z nejnižšího podlaží až do toho nejvyššího, a přitom se vyhýbat nástrahám ze stran Donkey Konga. (Stanton, 2015)

V roce 1985 Nintendo vydává svou konzoli Nintendo Entertainment System (NES). S pádem Atari v roce 1983 však v USA nastává tzv. Krach videoherního průmyslu. Šlo o rok a půl trvající ekonomickou krizi, jenž otřásla videoherním průmyslem. Byla zapříčiněná zahlcením trhu enormním množstvím herních automatů, konzolí a her, které většinou byli jen kopírky těch více známých. Zároveň však nastala velká poptávka po osobních počítačích, které narozdíl od konzolí byli mnohem levnější a sloužily k více účelům. Nintendo tak čelilo velké výzvě. Obchodníci o nové konzoli nechtěli ani slyšet, protože měli sklady plné konzolí, které nedokázali prodat. Nintendo tak snažilo dělat vše proto, aby se odlišili, počínají tím, že jejich konzole již nebyla označována jako konzole, ale jako entertainment system (zábavní systém). To se Nintendu podařilo a NES byl obrovský úspěch jak na japonském, tak americkém trhu.

Konzole se celosvětově prodalo téměř 62 milionů. Započala tak japonská nadvláda nad videoherním světem. (Stanton, 2015)

Nintendo si bylo vědomo všech chyb, které se Atari dopustilo a snažilo se je neopakovat. Zatímco na konzoli od Atari mohl vyrábět hry kdokoliv, u Nintenda šli na to zcela opačně a nedovolili nikomu, kdo nebyl Nintendo vývojář, vyvíjet hry pro NES. Tím si zajistili potřebnou kvalitu her, na kterou dojelo Atari. Za obrovským úspěchem zábavního systému od Nintenda však stojí hry od Shigera Miyamota. Tvůrce, jenž stojí za Donkey Kongem, totiž v roce 1985 přichází se hrou Super Mario Bros. Z Maria se stává senzace a kníratý instalatér se tak zapisuje do dějin a stává se maskotem Nintenda. Samotné hry se prodalo přes 40 milionů kusů. Během několika let vznikají pokračování ať už jde o sequely, reimaginace nebo pouhé spin-offy. Kromě Donkey Konga a Maria, stojí Myamoto za hrou Legend of Zelda, která jako první hra umožnovala uložit rozehranou hru. (Donovan, 2010)

1.6 První 3D hry

S prvními 3D hrami rostou i požadavky na počítačem ovládané protivníky a vývoj umělé inteligence se tak stává nedílnou součástí všech videoher. V 90. letech tak vzniká mnoho žánrů jako first person střílečky, real-time strategie nebo MMO. Arkádové automaty přestávají být populární a na vzestupu jsou domácí konzole a osobní počítače. Videohry, s příchodem 3D, které začínají vypadat více realisticky začínají být spojovány s kontroverzními tématy a vznikají tak například ratingové společnosti jako ESRB.

První takovou vlaštovkou byla hra Ultima Underworld, která vyšla v roce 1992. Jednalo se o RPG, kde hráč prozkoumával temné kobky ztraceného města a bojoval s nestvůrami. Jako téměř většina her tohoto ražení vycházela ze systému Dungeon&Dragons. Každá z nestvůr měla svou sadu instrukcí, které vykonávali v závislosti na situaci. Ačkoliv nešlo o pravé 3D jako v případě několika dalších her, hra vytvářela iluzi 3D světa, ačkoliv vše bylo stále 2D.
Ve stejném roce vychází na MS-DOS Wolfenstein 3D, kde se hráč ujme role spojeneckého agenta B.J.Blazkowicze, který se musí dostat z nacisty ovládaného hradu Wolfenstein. Hra samotná pak ustanovila nový moderní žánr first person střílečeк. Ve hře byl také použit algoritmus Finite State Machine (FSM), kterému se tato práce podrobněji věnuje v kapitole Oblasti umělé inteligence využívaných ve videohrách. Algoritmus umožnoval nastavit specifické reakce umělé inteligence na různé situace. Na Wolfensteina o rok později navazuje

další hra od idSfotware, která je dodnes považována za jednu z nejslavnějších her, která ovlivnila herní průmysl. Doom. Hra, která představila krvavou akci, která neměla v té době obdoby. Hráč, coby vesmírný mariňák, se musel probojovat hordami démonů, aby mohl uniknout z vesmírné základny. Každý z démonů, měl svůj vlastní vzor chování a hráč se tak musel vypořádat z celou řadou rozdílných nepřátel. Doom byl také první hrou, která běžela na novém druhu softwaru, který později dostal název engine. Těm se tato práce podrobněji věnuje v kapitole Herní enginy. Mimo hry pro jednoho hráče, však Doom nabídl i multiplayer. Až 4 hráči proti sobě mohli bojovat v aréně a sbírat tzv. fragy. Frag byl hráči udělen za zabití jiného hráče. Hráč, který byl zabit se po smrti znovu objevil v jiné části arény. Hráči proti sobě bojovali do té doby, dokud někdo z nich nenasbíral určitý počet fragů a nestal se tak vítězem. Zrodil se tak herní mod zvaný deathmatch, který se následně stal nedílnou součástí všech multiplayerových her. (Stanton, 2015)

1.6.1 První boti

První skutečně 3D hrou byl Quake od idSoftware. Quake vyšel v roce 1996 a nabídl plné 3D prostředí. Quake byl stylem hraní velmi podobný Doomu, ale kladl větší důraz na hru pro více hráčů To můžeme vidět například u jeho třetího dílu Quake 3: Arena z roku 1999, který je ryze multiplayerový (Donovan 2010). Nabídl i možnost hrát proti botům. Boti jsou počítačem ovládané protivníci, kteří jsou naprogramováni tak, aby se chovali jako skuteční hráči. Zatímco novodobý boti se učí pomocí machine learningu, první boti si museli vystačit pouze se sadou instrukcí od vývojářů. Za umělou inteligencí botů v Quake 3 stál Jan Paul van Waveren, který podrobně popsal fungovaní botů na konferenci GDC. Každý z botů měl svou vlastní osobnost, svůj způsob hry a pohybu a také své oblíbené zbraně. Jejich schopnost pohybu, přesnosti a techniky se měnily v závislosti na nastavené obtížnosti. S každou vyšší obtížnosti boti dostali do vínku novou sadu instrukcí, mohli tak například využívat zkratek, přeskakovat složitější překážky. Boti spolu také spolupracovali za určitým cílem, například při kradení vlajky soupeři tak chránili bota nesoucí vlajku. Boti také uměli poslouchat příkazy zadané od hráčů, pokud byli patřili do stejného týmu. Mimo jiné také uměli reagovat na poznámky v ingame chatu, dost podobně jako fungují dnešní chat boti. (van Waveren, 2001)

1.6.2 Umělá inteligence v prvních real-time strategiích

První skutečnou real-time strategií je Duna 2 z roku 1992, ačkoliv kořeny “real-time” strategií můžeme vysledovat již do roku 1982, ke hře Cytron Masters, která jako první integrovala

základní umělou inteligenci. Duna 2, je tak první strategii, která běžela v reálném čase a přišla s herními mechanismy a konceptem umělé inteligence, jež jsou používány dodnes.

Pokud chceme pochopit fungování umělé inteligence v tomto žánru, musíme nejdříve určit principy, podle, kterých se AI ve hře řídí. Jedním z nich je tzv. path finding. Není to technika používaná pouze v žánru RTS, ale i ve všech hrách napříč žánry. Pathfinding určuje, kudy se umělá inteligence může pohybovat, respektive jak název napovídá, jedná se o hledání cesty z bodu A do bodu B, a to, pokud možno co nejkratší. Nejčastěji používanou metodou je tzv. A* algoritmus. Jedná se o algoritmus, který hledá cestu napříč několika uzly z počátečního uzlu až do uzlu koncového. V závislosti na nastavených parametrech tak lze hledat nejkratší cestu nebo nejrychlejší cestu, popřípadě i nejdelší cestu. Aby však tento algoritmus fungoval v samotné hře, je třeba definovat uzly v každé specifické mapě. První strategie jako Duna 2 nebo Warcraft, používali jednoduchou čtvercovou síť, po které se umělá inteligence pohybovala. Problém však nastal při přechodu z top-down pohledu na pohled izometrický. Zatímco čtvercová síť v top-down pohledu byla šalamounsky řešená přes počet pixelů, tak v izometrickém pohledu je řešena pomocí vektorů. A*algoritmus ve hrách měl ve své době i své chyby. Každý rozkaz, který hráč vydal, musel být propočítán, což se na tehdejším hardwaru neobešlo bez lehčího prodlení. Takovým problémem trpěl například StarCraft z roku 1998. Dalším častým problémem, bylo zasekávání jednotek. Pokud totiž hrač vydal rozkaz pro velkou skupinu jednotek, každý z nich se následně řídila stejným výpočtem a často skončila zaseknutá v sobě. S řešením přišel až Craig Reynolds s tzv. flocking algoritmem. Zatímco některé hry jako Age of Empires nebo Command&Conquer i v pozdějších dílech používali čtvercovou síť, novější RTS používají síť tvořenou z polygonů, které je mnohem přesnější. (Xu, 2008)

V žánru strategií snad více než v žádném jiném nejvíce záleží na designu AI. Dát však hráči zajímavý zážitek není snadné. K tomu musí být AI lehce nepředvídatelná. Prosté přidání prvku náhody tak vůbec nestačí. Řešením je metoda zvaná Goal-Oriented reasoning, která umožnuje AI řešit rozhodnutí více logickým způsobem, které klasicky řeší hráč. Kdy postavit jakou budovu, kdy zaútočit na hráče atd.
Dalším principem je podvádění. Umělá inteligence podvádí, avšak ne, aby přivedlo hráče k hranici příčetnosti, ale aby byla jeho důstojným oponentem. Příkladem je například naprostého ignorování “fog of war”. Fog of War je častým elementem RTS her. V podstatě umožňuje vidět hráči pouze to co doopravdy vidí. Takže hráč neví, co se děje na druhé straně mapy, pokud zde nemá svou jednotku nebo postavenou například hlídkovou věž. V Age of Empires například

umělá inteligence podvádí tím stylem, že všechny suroviny těží 2x nebo 4x rychleji v závislosti na obtížnosti (Xu, 2008). V Command and Conquer 3 zase AI na vyšší obtížnosti dostává dvojnásobný příjem než hráč. Nastavení té správné úrovně "podvádění" je tak nedílnou součástí tvorby umělé inteligence pro RTS. V době boomu RTS her bylo k podvádění přistupováno spíše z důvodu nedostatečné výzvy pro hráče. Umělá inteligence tak musela podvádět, aby vůbec měla šanci se vyrovnat hráčovi. To se v průběhu let mění a například druhoválečná strategie Company of Heroes z roku 2006 je toho krásným příkladem. Zde hráč i na normální obtížnost zažívá skutečnou výzvu. Umělá inteligence v tomto případě podvádí jen na vyšší obtížnosti. Avšak to, co činilo umělou inteligenci tak schopnou byla okamžitá reakce a možnost ovládat a udávat rozkazy několika jednotkám v rámci milisekund. Což je pro lidského protivníka takřka nemožné. Hráč tak musel rychle reagovat a byl neustále pod tlakem.

1.6.3 Simulace a první neuronové sítě ve videohrách

V roce 1987 zakládají Will Wright a Jeff Braun studio Maxis. O dva roky později vydávají budovatelský simulátor SimCity. Wright si při vývoji SimCity kladl jednoduchou otázku, která provází jeho hry do dnešní doby a to, proč by hry měli být jen o vítězství nebo prohře, když mohou být prostě o hraní samotném. SimCity samotné tak bylo o stavbě města, ale jakým způsobem bylo postaveno a řízeno, to bylo čistě na hráči. Na pozadí probíhaly simulace řízené umělou inteligencí, zejména v dalších dílech šlo o velmi podrobné simulace každého elementu města. SimCity tak zvedlo vlnu zájmů o tvorbu simulací všeho druhu, a to především i pro vědecké účely. (Stanton, 2015)

Jedním z těchto případu je i Creatures z roku 1996. Hra, ve které se hráč stará o vylíhnuté příšerky, které učí, co mají dělat a jak se chovat. Tato hra jako poprvé využila machine learning v interaktivní simulaci. Příšerky používají neurální síť k tomu, aby se naučili, co dělat. Hra je tak považována za průlom ve výzkumu umělého života, jehož cílem je modelovat chování bytostí interagujících s jejich prostředím. (Donovan, 2010)

V roce 2000 vychází další hra Willa Wrighta, The Sims. The Sims v sobě kombinuje všechno co byste čekali od simulace života. Hráč se ujme ovládání Simiků, virtuálních postav, které mají částečně svobodnou vůli a hledí si svého. Poprvé zde byli použity inteligentní objekty. Inteligentní objekty se používají k implementaci chování. Objekt určuje, jak s ním každá postava interaguje, což má oproti centralizované logice mnoho výhod. Každý Simík má

základní touhy, které dle kterých se řídí výběr jejich akcí, a to na základě robustního stromu chování. Mimo jiné zde mohla umělá inteligence mezi sebou vytvářet emocionální interakce a vztahy. (Stanton, 2015)

Nejdále však zašla hra Black and White z roku 2001 od Lionhead Studios. Hra kloubí simulaci umělého života a strategie. Hráč se ujme role boha, který vládne na ostrově obydleném rozličnými kmeny. Jako bůh, může hráč učit pomocí interakcí svého společníka ovládaného umělou inteligencí. Jedná se o další skvělé využití neurální sítě ve videohře. (Wexler, 2002)

1.7 Nové milénium

S příchodem nového milénia se herní průmysl začal postupně měnit. Internet se stával více dostupným a počítače začali být součástí každé domácnosti. Slavné Atari už je téměř zapomenuto a na konzolovém trhu nyní vládne Sony se svým PlayStationem a Microsoft se s Xboxem a Nintendo si zpovzdáli udržuje svou věrnou fanouškovskou základnu, ale nedosahuje takové popularity jako ostatní konzole. Je to také doba kdy se mobilní telefony stávají součástí našich životů a herní průmysl začíná experimentovat i s nimi. Avšak pravý boom mobilního hraní má teprve nastat.

1.7.1 Vývoj umělé inteligence v akčních hrách

V roce 1998 vychází hra Thief. Thief je first person stealth hra, ve kterém se ujmete role zloděje Garreta ve středověkém fiktivním světe. Hra jako první používala senzorový model, který umožnil umělé inteligenci realisticky reagovat na světlo a zvuk. Na čemž byli založeny i herní mechaniky. Ve stejném roce vychází i Half-Life, scifi first person střílečka. Hráč jakožto vědec Gordon Freeman, musí uniknout z výzkumného střediska poté co se pokazí experimenty s mimozemským zařízením. Half-Life jako první představil interaktivní cutscény[10], které byli kombinací skriptů a umělé inteligence. V průběhu hry hráče doprovázeli i umělou inteligencí řízení společníci. Mimo jiné zde byla i efektně použita skupinová umělé inteligence. (Stanton, 2015)

[10] také často označováno jako in-game cinematic je neinteraktivní sekvence ve videohře

V roce 2001 jedna vychází scifi střílečka Halo. V té se hráč zhostí role Master Chiefa, geneticky upraveného supervojáka, který bojuje proti mimozemšťanům. Umělá inteligence v Halo byla velmi inovativní, představila systém krytů, kde se protivníci ovládané umělou inteligencí dokázali schovávat do krytů, používat potlačovací palbu a granáty. Dokázali také reagovat na situaci v okolí, například pokud byl jejich vůdce zastřelen a situace se pro ně nevyvíjela přívětivě, dali se na útek. Halo také zpopularizovalo používání „stromu chování“ ve videohrách. (Stanton, 2015)

V září 2005 vychází hororová střílečka F.E.A.R od studio Monolith. F.E.A.R kombinuje prvky umělé inteligence z Half-life a Halo. AI v této hře je dodnes považována jako nejlepší z celého žánru FPS. Hra využívá skupinové umělé inteligence, kdy jeden z nepřátel použije krycí palbu a druhý se přemístí do lepšího krytu, popřípadě se snaží na hráče zaútočit z boku. To však není nic revolučního, revoluční je způsob, jakým umělí inteligence funguje. Namísto toho, aby se AI řeklo, jak se chovat v každé situaci, dává hra nepřátelům sadu cílů a sadu možných akcí, které jim umožní vyřešit každou situaci po svém. Umělá inteligence tak v podstatě reaguje dynamicky na to, co se zrovna děje okolo. Například, pokud jsou v nebezpečí, dají se na ústup, ale pouze tehdy, pokud dokáží identifikovat bezpečnou cestu. Pokud ne, schovají se a možná i budou pálit po hráči naslepo, aby zpomalili jeho postup. Umělá inteligence také dokáže skvěle fungovat jako skupina, kdy se například jeden snaží upoutat pozornost hráče, zatímco další dva se ho snaží napadnou zezadu. Dokáží využit jakéhokoliv krytu a objektů. Nic z toho není řízeno. Neexistuje žádný příkaz k obcházení hráče a vpadnutí mu do zad. Umělá inteligence analyzuje prostředí a reaguje na něj takovým způsobem, jaký nepřinesla žádná jiná hra do dnešních dnů (Horti, 2017).

1.7.2 Grand Theft Auto a nástup sandbox her

V říjnu 2001 vychází na PlayStation 2 městská akce s otevřeným světem Grand Theft Auto 3. Ačkoliv se nejedná o první díl v sérii, první a druhý díl vyšli v roce 1997 a 1999, je třetí díl první 3D hrou v sérii. Hráč se v GTA 3 ujme role kriminálníka Clauda, který se snaží pomstít těm kteří ho zradili. Hra se odehrává ve virtuálním New Yorku nazvaném Liberty City, které hráči mohli prozkoumávat, jak se jim jen zlíbilo. Hráč se v GTA 3 ujme role kriminálníka Clauda, který se snaží pomstít těm kteří ho zradili. Hra dokázala vybudovat iluzi naprosté svobody, kde hráč mohl dělat cokoliv. K tomu však bylo potřeba vytvořit umělou inteligenci, která by na hráče reagovala takovým způsobem jako reálný člověk. GTA 3 je ne nadarmo

považována za jednu z nejdůležitějších her všech dob. Hra totiž zpopularizovala žánr tzv. Sandbox[11] her. Několik dalších dílů na sebe nenechalo dlouho čekat, a tak vyšli GTA: Vice City nebo GTA: San Andreas a GTA IV. Každý další díl překonal svého předchůdce ať už v prodejích nebo v posouváním žánru a her jako takových (Stanton, 2015).

1.8 Současnost

S nástupem chytrých telefonů, raketově roste i množství her a počet jejich hráčů. Herní trh, trendy a technologie se mění rok co rok. Kreativní schopnosti vývojářů jsou dnes mnohem důležitější než schopnost technicky rozběhnout hru s omezeným hardwarem. Do popředí se dostávají různé předplatitelské služby po vzoru Netflixu a hry koncipované jako služby. V dnešní době je her nepřeberné množství a dnes víc něž kdy víc záleží na schopnosti hru prodat.

1.8.1 Procedurální generování

S růstem herního trhu, rostou i náklady na vývoj videoher. A ty největší herní blockbustery se pohybují v stovkách milionů dolarů. Například Call of Duty: Modern Warfare 2 z roku 2009 stál Activision 250 milionů dolarů nebo Grand Theft Auto V, který překonal hranici 265 milionů dolarů. Nutno však podotknout, že v mnoha případech, více jak polovinu rozpočtu tvoří výdaje na marketing a propagaci. Jenom pro upřesnění, první filmové Avengers z roku 2012 stáli 220 milionu dolarů (Bigas, 2018). Není proto divu, že se herní vývojáři snaží najít řešení, jak si ušetřit práci a peníze. Jedním z těchto řešení je i procedurální generování. Jedná se o techniku, při které je obsah automaticky generován počítačem na základně inteligentních algoritmů. Tato technika byla poprvé použita již v 80.letech minulého století, a to z důvodu hardwarové limitace ve hře Sentinel, která obsahovala 10 000 úrovní s pouhou velikostí 64 kb. S nástupem výkonnějšího hardwaru se však od této techniky pomalu ustoupilo. Znovuobjevena byla až v posledních patnácti letech. Prvním zářným příkladem využití procedurálního generování byla hra Minecraft od studia Mojang. Minecraft využívá inteligentní algoritmy k vygenerování 3D světa, který je tvořen z kostek (Alayón, 2018). V roce 2009 využívá procedurálního generování i FPS Borderlands. V tomto případě ovšem není algoritmus využit ke generování světa, nýbrž zbraní, jenž Borderlands nabízí nepřeberné množství. V nejnovějším díle, Borderlands 3 z roku 2019, autoři uvádí, že hra nabízí přes jednu miliardu zbraní. A to díky procedurálnímu generování.(Fisher, 2019). Procedurální generování je

[11] Sandbox – virtuální pískoviště, kde si hráč může dělat co se mu zlíbí

používáno ve spouště her rozličných her, avšak nejdále zašla hra No Man‘s Sky od britského studia Hello Games z roku 2016. Hra samotná je založená na průzkumu neznámého vesmíru. Všechny základní elementy: lodě, zbraně, rostliny, živočichové na každé z planet jsou generovány umělou inteligencí, jenž využívá procedurální generování k vytvoření doslova nekonečného vesmíru. (Alayón, 2018)

Obrázek 3: No Man's Sky
Zdroj: (Hello Games, 2016)

1.8.2 Pokročilá AI a první neurální síť v RTS

Jedna z populárnějších forem pokročilých A.I. ve videohrách je například umělá inteligence vetřelce ze hry Alien:Isolation od studia Creative Assembly z roku 2014. Vetřelec je ve hře řízen dvěma vzory, které řídí jeho pohyb a chování. Director AI a Alien AI. Prvních z nich je pasivní controller, který má na starosti vytváření zábavného zážitku pro hráče. Z tohoto důvodu Director AI ví, kde se jak hráč, tak vetřelec nacházejí. Tyto znalosti však s vetřelcem nesdílí. Co však s vetřelcem sdílí je tzv. Menace Gauge neboli jakási hladina hrozby a stresu, která je založena na několika faktorech jako je blízkost vetřelce od hráče, množství času, které vetřelec stráví v dosahu sledovacího zařízení hráče atd. Tato hladina informuje vetřelcovu umělou inteligenci a když dosáhne jisté úrovně, změní se priority v jeho stromě chování a on začne prohledávat okolí, aby našel hráče. Avšak nikdy neví jeho přesnou lokaci, jenom to, že by měl prohledat danou oblast. Hráče tak musí vždy najít sám. K tomu má dispozici sadu „nástrojů“. Jedním z nich je senzorový systém, který umožnuje vetřelcovi zvukové a vizuální podněty v prostředí. Hluky, jako jsou kroky, výstřely, otevírání dveří, dokonce i pípání sledovače pohybu, to vše pomáhá umělé inteligenci vystopovat hráče.

Kromě zvukových senzorů může vetřelec také spoléhat na vizuální senzory. Takže může zahlédnout hráče, jak probíhá kolem nebo si může pouze povšimnout otevřených dveří, které byly dříve zavřené. Dalším nástrojem, který pomáhá vetřelcovi ulovit hráče je Searching System[12]. Ten umožnuje vetřelcovi trošku podvádět, neboť pracuje databází možných schovek, které mu dali k dispozici vývojáři. Nicméně to samotné vetřelcovi pouze umožnuje jenom prohledávat dané úkryty v oblasti zcela náhodně, pokud ovšem něco neuslyší nebo neuvidí. Jedním z nejvíc diskutovaných témat ohledně umělé inteligence v této hře je fakt, že se vetřelec postupem času učí a adaptuje se na hráčův styl hraní. Vývojáři však vše uvádějí na pravou míru a v podstatě vyvracení, že by byl použit jakýkoliv systém neuronových sítí. Vetřelec se řídí jasně definovaným stromem chování, který mu ovšem zezačátku hry není zcela dostupný. Jsou zde větve stromu, které jsou zablokované a on tak nemá přístup k sadě různých činností a chování. Například zezačátku hru nemusí být aktivní část stromu, která reaguje na zvuk otevírání dveří v dálce. Pokud však hráč otevře dveře v zorném poli vetřelce tato větev se odemkne a on bude moci v budoucnu reagovat na zvuk otevřených dveří. Jak hráč postupuje hrou, tím více větví se vetřelcovi zpřístupňuje, a to dává iluzi, že se vetřelec postupně učí a adaptuje. (Maass, 2019)

V roce 2017 vychází real-time strategie z druhé světové války Blitzkrieg 3. Stojí za ní ruské studio Nival Interactive. Hra obsahuju první neuronovou síť v žánru RTS, pojmenovanou generál Boris. Boris byl navržen tak, aby napodobil chování lidského protivníka a je schopen hrát hru na úrovni top hráčů, aniž by používal jakékoliv skryté informace o nepříteli, jak je typické pro umělou inteligenci v tomto žánru. Boris se tak dokáže adaptovat a následně i predikovat další tahy hráče a být na ně řádně připraven, aby mohl vrátit úder. (Orlovskiyi 2017)

[12] vyhledávací systém/hledací systém

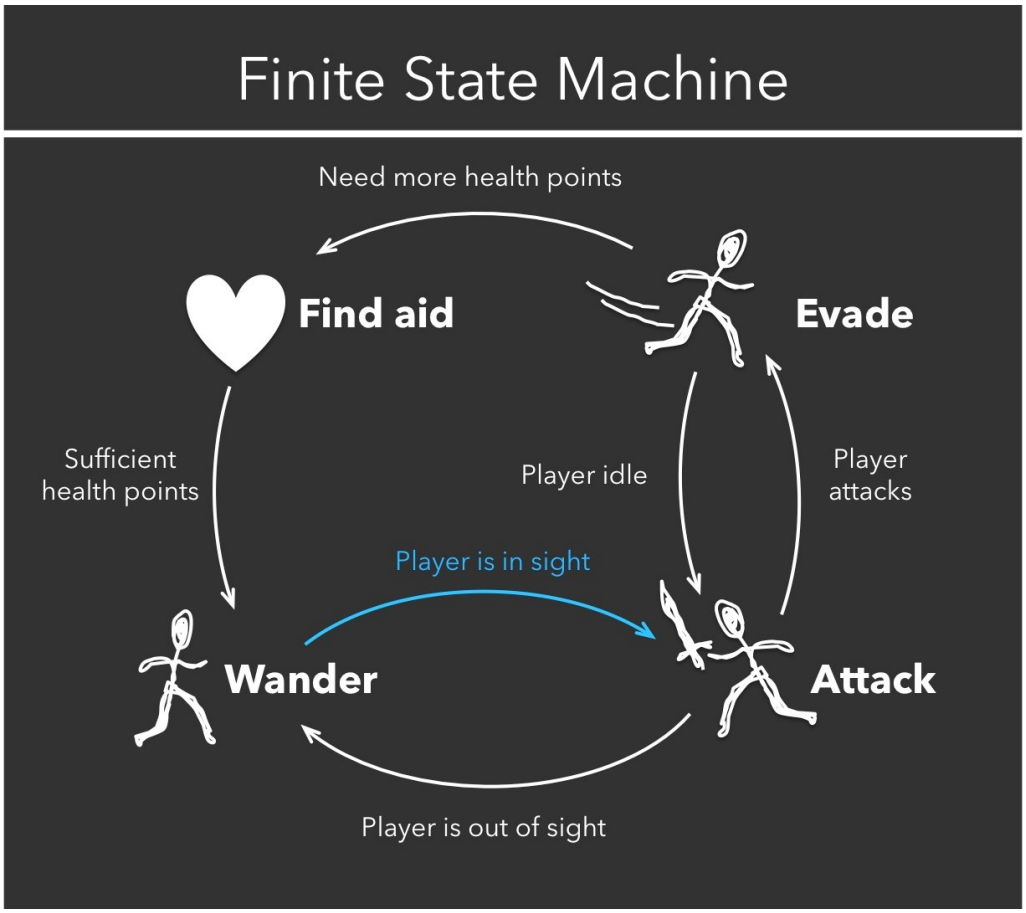
2 Technologie a oblasti umělé inteligence využívaných ve videohrách

Tato kapitola se zabývá nejčastěji používanými metodami pro tvorbu umělé inteligence ve videohrách.

2.1 Finite State Machine

Nejběžnější úlohou AI ve videohrách je ovládání nehratelných postav. Designéři často používají triky, aby tyto NPC vypadaly inteligentně. Jeden z nejstarších triků je Finity State Machine (FSM), který známe z videoher z devadesátých let dvacátého století. V FSM vývojář zobecňuje všechny možné situace, s nimiž se může umělá inteligence setkat, a poté naprogramuje konkrétní reakci pro každou situaci. V zásadě by AI FSM měla rychle reagovala na akci lidského hráče svým předem naprogramovaným chováním. Například ve střílečce by se AI chovala následovně. Pokud by se objevil hráč, pak by na něho začala střílet a pokud jí klesne zdraví, tak uteče nebo se pokusí sehnat lékárničku. Podobnou velmi zjednodušenou situaci znázorňuje Obrázek 4. Zde můžeme vidět, že daná umělou inteligencí ovládaná postava může vykonávat čtyři různé akce v závislosti na situaci. Najít lékárničku, uskočit/uhnout, toulat se anebo útočit. V praxi to funguje tak, že NPC začíná ve stavu „toulaní se“ a pokud při svém toulání narazí na hráče, tak zaútočí. Pokud se však hráč ztratí z dohledu, NPC se vrátí k toulce. Pokud však hráč zaútočí na NPC, to může uskočit a následně vrátit úder. Pokud zdraví NPC klesne na nízkou hodnotu, NPC se může dát na útěk a vyhledat například lékárničku. (Lou, 2017)

Jak lze už vidět, zjevnou nevýhodou FSM, je jeho předvídatelnost. FSM se také nedá použít v každé hře. Například v žánru strategií by se hráč ihned naučil, jak jednoduše oponenta porazit. Avšak i přesto je tento systém hojně využíván ve hrách jako Call of Duty nebo Battlefield.



Obrázek 4: Finite State Machine
Zdroj: (Lou, 2017)

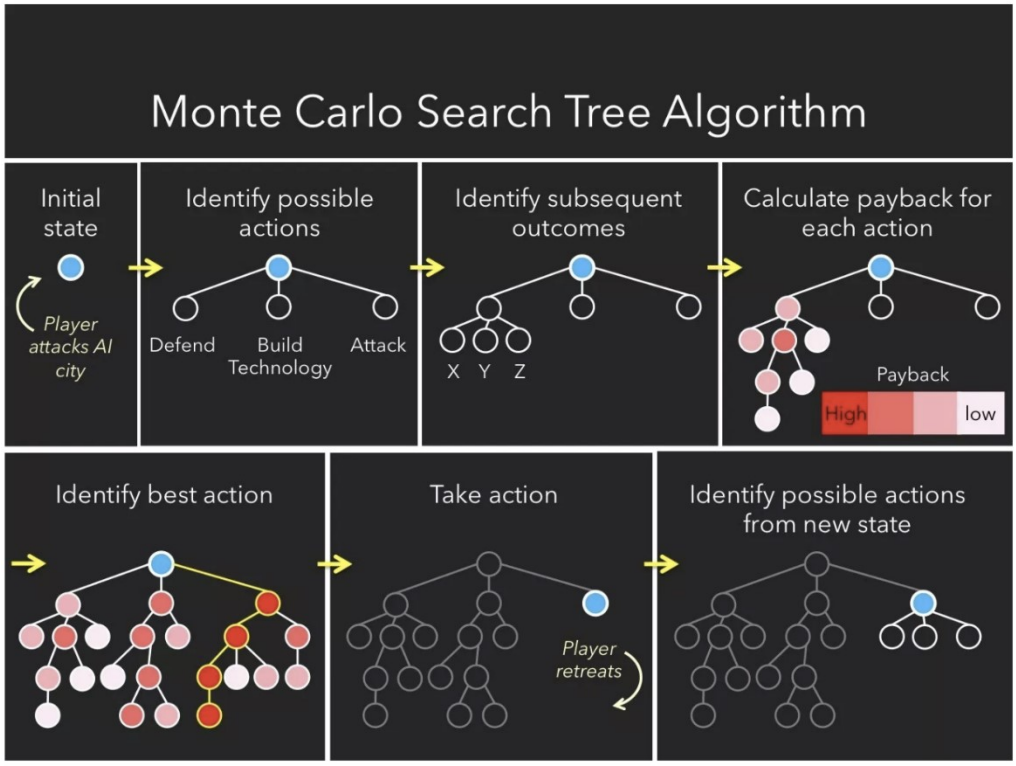
2.2 Monte Carlo Search Tree

Pokročilejší metodou, která je používána, aby se předešlo problému s opakovatelností jako u FSM, je Monte Carlo Search Tree (MCST).

MCST technika používá náhodné pokusy k vyřešení problému. Tato metoda umělé inteligence byla použita například i v Deep Blue, prvním počítačovém programu, který porazil lidského šachového šampiona v roce 1997. V každém bodě hry, Deep Blue používalo MCST ke zvážení všech možných tahů, poté umělá inteligence zvážila všechny možné odvetné tahy soupeře a pak opět všechny možné reakce na reakci atd. Všechny možné tahy se tak začali rozrůstat, jako větve u stromu, proto také tato metoda nese název „search tree“. Umělí inteligence tak vždy použije tu nejvýhodnější větev. Poté co lidský protivník udělá tah, AI opět hledá nejlepší větev v reakci na hráčův tah a z možností, které má k dispozici. Ve videohrách dokáže umělá inteligence s MCST vypočítat tisíce možných tahů a vybrat ty, které pro ni budou nejvýhodnější. (Lou, 2017)

Podobný algoritmus používá mnoho strategických her. Nicméně, vzhledem k povaze jednotlivých her, které jsou mnohem náročnější na umělou inteligenci než šachy, protože nabízejí mnohdy nepřeberné množství možností, MCST algoritmus vybere některé z možných tahů náhodně. Tím pádem je chování protivníka ovládaného umělou inteligencí mnohdy nepředvídatelné. Například ve hře Civilization, kde hráči rozvíjejí svojí civilizaci několika stovkami rozhodnutí, je téměř nemožné předpřipravit chování umělé inteligence a pokud bychom chtěli, aby počítač propočítal všechny možné tahy, trvalo by to také extrémně dlouho. Proto se zde používá MCST, kde umělá inteligence vyhodnocuje jednotlivé možné tahy, ale pouze v náhodně zvolené větvi. (Lou, 2017)

Na obrázku níže lze vidět jakým způsobem umělí inteligence přemýšlí. Nejprve se podívá na všechny možnosti, které má, v uvedeném příkladu jsou to tyto: obrana, budování nebo útok. Poté vytvoří strom, který předpovídá pravděpodobnost úspěchu pro další každý možný krok. Na obrázku 5 lze vidět, že možnost s největší pravděpodobností úspěchu je možnost útoku (tmavě červená se rovná vyšší pravděpodobnosti zisku pro umělou inteligenci). Počítač se tedy rozhodne zaútočit. Když hráč provede další krok, počítač opět opakuje proces vytváření stromu znovu.



Obrázek 5: Monte Carlo Search Tree Algorithm
Zdroj: (Lou, 2017)

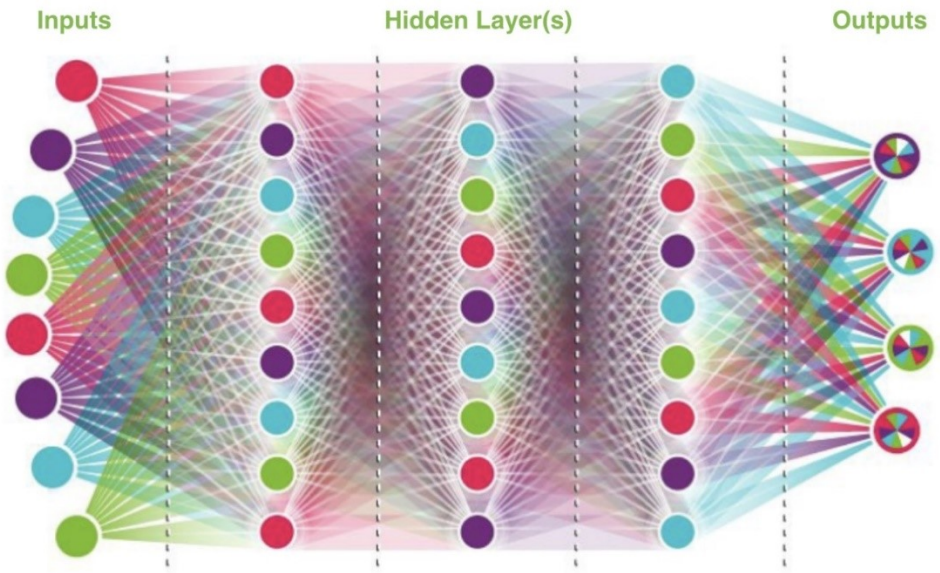
2.3 Behavior Tree

Na rozdíl od FSM nebo jiných systémů používaných pro programování AI je Behavior Tree strom hierarchických uzlů, které řídí tok rozhodování umělé inteligence. Když si představíme strom, tak každý list stromu je příkaz, kterým se umělá inteligence řídí a větve jsou uzly, které kontrolují to, jakým způsobem jsou jednotlivé příkazy vyhodnoceny. Stromy chování mohou být extrémně hluboké se spoustou pod stromů a umožnují tak vytvořit obrovské knihovny chování, které mohou být propojeny za účelem věrohodného chování umělé inteligence. Tvorba stromu chování je na rozdíl od FSM velmi přehledná a lehce modifikovatelná. Například lze začít pouze tvorbou základního chování, na které jdou navrstvit další větve, které se budou zabývat alternativními způsoby dosažení cílů. Což například umožní, aby umělá inteligence měla nouzovou taktiku, pokud by určité chování selhalo. (Simpson 2014)

To, jakým způsobem stromy chování fungují je názorně předvedeno v kapitole Tvorba základní struktury stromu chování.

2.4 Neuronové sítě

Dalším méně častěji používaným druhem umělé inteligence ve videohrách jsou neuronové sítě. Neuronové sítě jsou silně inspirovány fungováním biologické neuronové sítě. Především tím, jak jsou lidé schopni se učit novým věcem pomocí příkladů. Na takovém principu funguje i umělá neuronová síť. Generační neuronová síť je strukturována stejně jako standardní neuronová síť. Začíná určitým počtem vstupních uzlů, které se pak posílají do jedné nebo více skrytých vrstev až nakonec poskytnou nějaký výstup. (Maass 2019)



Obrázek 6: Deep learning
Zdroj: (Comi, 2020)

Příklad učení se lze předvést na příkladu hry Had. Had je 2D hra, kde hráč ovládá hada, který je tvořen z několika za sebou jdoucích kostek. Hráč má tří možnosti pohybu, doleva, doprava nebo rovně. Když se hráč dotkne zdi nebo se kousne do ocasu, hra končí. Ve hře se nachází tečky, která znázorňují jídlo, které když had sní, jeho ocas se rozroste o další kostičku. Takže čím více had sní tím delší bude. Pokud chceme, aby had řízený neuronovou sítí dosáhl co největšího skóre a vůbec ve virtuálním světe přežil, je potřeba mu poskytnout vstupní informace (Inputs). Ty mohou být například tvořeny 6 Ano/Ne otázkami: je něco přede mnou, je něco vlevo, je něco vpravo, je jídlo přede mnou, je jídlo vlevo, je jídlo vpravo. Tohle řešení nám tedy poskytne 6 uzlů s hodnotami 1 nebo 0. Další věcí, kterou musíme vnuknout umělé inteligenci je systém odměn a trestů. Za každý krok směrem k jídlu získá jeden bod, avšak abychom zabránili tomu, aby se had zacyklil a točil se dokola, je třeba mu udělit i trest, kdy mu bod bude odečten za každý krok dál od jídla. Při dosažení jídla by měl had dostat nejvíce bodů. Jak jsme již stanovili, máme 6 různých vstupních uzlů. Každý vstupní uzel se připojuje ke každému z prvních skrytých uzlů vrstvy prostřednictvím toho, co označujeme parametr. Na Obrázku 6 lze vidět tyto linky připojené ke každému z uzlů. Tyto parametry jsou tím, co náš model bude upravovat, protože se učí, jaké vstupy posílit nebo oslabit, aby poskytly co nejpřesnější výstupy. V našem případě je „nejpřesnější výstup“ definován jako „had, který shromažďuje nejvyšší počet bodů“.

Způsob, jakým se generační neuronová síť učí spočívá v tom, že se nejprve rozhodne o velikosti každé generace (řekněme, že chceme, aby každá generace obsahovala 200 hadů). Dále vytvoří mikro-variace ve váhách pro každého z 200 hadů v první generaci, poté nechá vypustit každého

z 200 hadů v první generaci a vybere nejúspěšnější hady (hady, které získaly nejvíce bodů). Řekněme, že vybereme 10 nejlepších hadů, což je 5 % nejlepších, kteří získali nejvíce bodů v naší první generaci. Těchto 10 hadů se pak stane „rodiči“ druhé generace. Parametry těchto 10 hadů se následně používají k definování výchozího bodu druhé generace. Druhá generace 200 hadů opět vytvoří mikro-variace s těmito parametry a nejlepší hadi budou vybráni jako „rodiče“ třetí generace atd. Tento proces opakujeme pro každou následující generaci, dokud se rychlost učení hadů nezačne stagnovat, tedy dokud se generační zlepšení nezpomalí nebo nezastaví. Může trvat několik generací, než vůbec první had spolkne jídlo, a tudíž potrvá i několik stovek generací, než vůbec had bude schopen hrát hru alespoň na úrovni hráče. (Binggeser 2020)

3 Herní enginy

Tato kapitola se věnuje základnímu vymezení termínů herní engine, kde je popsáno, co jsou herní enginy a k čemu slouží a jak fungují. Následně se krátce věnuje vzniku herních enginů a jejich historii. Další část této kapitoly patří jednotlivým dostupným herním enginům na trhu, kde je každý specificky rozebrán a popsán.

3.1 Základní vymezení

Herní engine jak už název napovídá je nejen hnací motor her, ale i prostředí pro jejich vývoj. Vývojáři používají nástroje, které jim poskytuje právě herní engine k vytvoření her pro konzole, počítače a mobilní zařízení. Hlavní devízou herních enginů je fakt, že je to univerzální nástroj skládající se z několika různých komponent od renderovacích, fyzikálních až po audio komponentách a umělou inteligenci. Tyto nástroje pro grafiku, renderování, animace, zvuk atd. jsou obvykle poskytovány v integrovaném vývojovém prostředí, které umožňuje zjednodušený a rychlý vývoj her. Herní enginy bývají občas spojovány s termínem „middleware“13. Stejně jako jiné typy middlewaru, herní enginy obvykle poskytují abstrakci platforem, což umožňuje, aby stejná hra mohla běžet na různých platformách od herních konzolí a osobních počítače až po mobilní zařízení, a to jen s několika, změnami zdrojového kódu hry. Jak již bylo zmíněno, tak komplexnější herní enginy jsou složeny z jednotlivých komponent. Je obvyklé, že základní komponenty enginu jsou nahrazovány nebo rozšiřovány specializovanější komponenty jiných společností. Některé herní enginy jsou navrženy jako řada volně propojených komponent, které lze selektivně kombinovat a vytvořit vlastní engine, místo běžnějšího přístupu s rozšiřováním nebo přizpůsobováním. Tato možnost úprav je pro mnoho vývojářských firem vysokou prioritou vzhledem k široké škále použití, pro které se enginy používají. Protože herní enginy neslouží jen k vytváření her, ale často se používají pro jiné druhy interaktivních aplikací. Aplikací, kde je nutné vykreslit scénu v reálném čase, jako jsou marketingová ukázky, architektonické vizualizace, různé simulace a modelovací prostředí. Herní engine je tedy ultimátní nástroj a poskytuje vše co vývojáři potřebují nejen k tvorbě her. (Ward 2008)

[13] software, který je mezi operačním systémem a aplikacemi, které jsou v něm spuštěné. Middleware v podstatě funguje jako skrytá transakční vrstva a umožňuje komunikaci a správu dat pro distribuované aplikace. (Microsoft, nedatováno)

3.2 Komponenty

Herní enginy se skládají z několik komponent, které spolu tvoří jeden ucelený celek. Nejčastěji tak jde o renderovací engine, audio a fyzikální engine a také o modul určený k tvorbě umělé inteligence pro hry.

3.2.1 Renderovací engine

Renderovací engine má na starosti 3D grafiku a její vykreslování a to pomocí několika metod od rastrování až po ray-tracing. Renderovací engine není přímo psaný pro CPU nebo GPU. Většina renderovacích enginů je postavena na jednom nebo více tzv API jako je Direct3D, OpenGL nebo Vulkan skrz které pak následně GPU může zpracovávat data. Knihovny nízké úrovně, jako jsou DirectX, Simple DirectMedia Layer (SDL) a OpenGL, se také běžně používají přímo ve hrách, protože poskytují hardwarově nezávislý přístup k jinému počítačovému hardwaru, jako jsou vstupní zařízení (myš, klávesnice a gamepad), síťové karty, a zvukové karty. Před hardwarově akcelerovanou 3D grafikou byl použit softwarový rendering. Softwarové vykreslování je však stále v některých případech používáno. (Gregory 2009)

3.2.2 Audio engine

Audio engine je komponenta, která se skládá z algoritmů souvisejících s načítáním, úpravou a výstupem zvuku prostřednictvím reproduktorového systému klienta. Engine musí být schopen načítat, dekomprimovat a přehrávat zvukové soubory. Pokročilejší audio enginy dokáží vypočítat a vyprodukovat efekty, jako jsou ozvěny, Dopplerův efekt, nastavení výšky / amplitudy, oscilace atd. Engine většinou provádí výpočty na CPU nebo na vyhrazeném ASIC[14]. Stejně jako renderovací engine i audio engine může využívat API, jako jsou OpenAL, SDL audio, XAudio 2, Web Audio atd. (Gregory 2009)

3.2.3 Fyzikální engine

Fyzikální engine je zodpovědný za emulování fyzikálních zákonů. Tyto enginy můžeme rozdělit na dva druhy, a to na real-time, ty, které pracují v reálném čase a na high-precision. High-precision enginy vyžadují více výpočetního výkonu pro výpočet velmi přesných fyzikálních zákonů a obvykle je používají vědci a počítačové animované filmy. Zatímco, fyzikální enginy, které pracují v reálném čase, se používají ve videohrách, kde používají

[14] ASIC – integrovaný obvod navržený a vyráběný pro určitou specifickou aplikaci, v tomto případě zvuk

zjednodušené výpočty se sníženou přesností, a to z důvodu náročnosti na hardware. (Gregory 2009)

V této podkapitole se však zaměříme pouze na fyzikální enginy, které jsou důležité při vývoji her. Ve většině počítačových her je rychlost zpracování a plynulá hratelnost důležitější než přesnost fyzikální simulace. Proto je většina fyzikálních enginů navržena tak aby produkovala reálnou fyzikální simulaci jen pro jednoduché výpočty a často i s různými odchylkami. V drtivé většině případu se totiž fyzikální enginy nesnaží přímo o reálnou simulaci, ale spíše jen o to se co nejvíce reálnosti přiblížit, nebo alespoň o to, aby to tak vypadalo. Nicméně některé hry využívající fyziku při soubojích nebo hádankách potřebují, aby fyzika byla co nejreálnější, a to je i případ Source enginu, který přinesl revoluční fyzikální model.(Gregory 2009)

3.2.4 Umělá inteligence

Umělá inteligence je obvykle oddělená od hlavního programu a je jí vyčleněn speciální modul ve kterém pracují softwaroví inženýři se speciálními znalostmi. Ve většině her se umělá inteligence chová zcela odlišné a to protože, neexistuje žádná globální řešení, jaké nabízejí ostatní moduly, a tak AI musí být pro každou hru vytvořena od píky. Novější enginy tak alespoň nabízejí pokročilé nástroje či defaultní presety pro tvorbu umělé inteligence.(Gregory 2009)

3.3 Historie

V dobách, kdy ještě nebyl ustálen termín herní engine, bylo naprosto obvyklé psát kód pro každou hru zcela od nuly a pro každou platformu zvlášť. Obzvláště v těchto dobách byl i každý systém zcela odlišný, narozdíl od dnešní doby, kdy většina zařízení běží na 64bitové architektuře. Proto vývoj her, bylo velké úskalí.

Změna přišla až v roce 1993 s vydáním hry Doom od studia idSoftware. John Carmack, hlavní programátor v idSoftware nejen že vytvořil nový druh softwaru, ale rovněž stvořil nový způsob, jakým se dají videohry tvořit. Doom byl navržen s jasně definovaným jádrem a komponenty jako 3D renderovací systém, kolizní systém a audio systém. Zde je potřeba však upozornit na to, že Doom, posléze Doom engine nebyl ve své podstatě tak úplně 3D, ale engine samotný dokázal vykreslit objekty, postavy a celé úrovně pomocí 2D spritů[15]. Vykreslování bylo velice

[15] jedná se o označení pro dvourozměrný obrázek či animaci

náročné na výkon a ve své době vyžadovalo tzv. 386 s podporou VGA, aby bylo vůbec možné hru rozběhnout. Skutečnost, že se jedná o zlomový moment ve vývoji videoher, si vývojáři začali uvědomovat až ve chvíli, kdy se na idSoftware začali obracet další vývojářská studia s nabídkou licence na jejich engine. Nicméně po úspěchu s Doom enginem, který následně přejmenovali na idTech se vyrojila spousta dalších společností s jejich vlastními technologiemi, které vždy přinesli něco nového a započala tak nová éra. (Lowood 2014)

Jako první dokázala reagovat společnost NovaLogic s jejich Voxel enginem, na kterém běželi hry ze série Comanch, Blade Runner a Command & Conquer. Voxel měl svůj vlastní způsob, jak vykreslit volumetrické objekty jako trojrozměrné bitmapy. Ostatní enginy také postupně začali aplikovat vektorovou grafiku, která byla náročnější na výpočetní výkon a méně detailní než 3D bitmapa. Prvním 3D enginem podporující DOS byl XnGine od vývojářů z Bethesda Softworks, který byl představen v roce 1995 spolu s hrou Terminator: Future Shock. Engine se stal populárním díky kompatibilitě s 3dfx grafickými kartami a především proto, že umožňoval použít grafické assety s vysokým rozlišením, to vše, ale až po příchodu Quake Enginu. Quake Engine byl první skutečný 3D engine, vyvinutým nikým jiným než lidmi z idSoftware. Quake engine dokázal chytrým způsobem vykreslovat 3D levely, tím, že vždycky byla zpracována jen určitá část levelu, a to pouze ta, kterou zrovna hráč viděl. Toho docílili použitím tzv. Z-bufferingu, takže částí levelu, které byli mimo dosah hráče byli vymazány z vyrovnávací paměti, a tudíž se nemuseli vykreslovat a neovlivňovali výkon. Dalším velmi populárním enginem byl Renderware z roku 1996, který po dlouhou dobu umožňoval vytvářet multiplatformní hry na platformy jako PlayStation 2, Wii, GameCube, Xbox, Xbox 360, PlayStation 3 a PSP. idSoft v roce 1997 vydal hru Quake II běžícím na modifikovaném a vylepšeném quake enginu, který nyní nesl název idTech2. Ten podporoval OpelGL, barevné světelné efekty a podporu jazyka C. Co bylo důležité, byla však podpora DLL knihoven, která umožnila modifikaci enginu samotného. Tento moment také následně vedl ke vzniku „modovací komunity “- skupiny hráčů a malých nezávislých studií, které začali tvořit nové hry úpravou stávajících her pomocí bezplatných sad nástrojů poskytovaných původními vývojáři. Pozdější hry jako Quake III Arena od idSoftu nebo Unreal od Epic Games, byli navrženy tak aby toto umožňovaly. (Paul et al. 2012). Ještě, než však přijde řeč na souboj Quake III a Unrealu, je potřeba si připomenout jeden zásadní milník a tím je GoldSRC, neboli gold source. Za enginem GoldSRC stála společnost Valve vedená Gabem Newellem. Jednalo se o silně modifikovaný Quake 2 engine, který dal vzniknout hrám jako Half-life, Day of defeat nebo Counter-Strike. Samotný Gaben v rozhovoru pro GameSpot přiznává, že vzali důležité

komponenty z Quake enginu, jehož licenci zaplatili a na nich vystavěli vlastní engine. Dále však dodává, že 75 % kódu enginu je jejich vlastní kód. (Gamespot, 2001) GoldSRC byl svým způsobem revoluční díky tomu, že jako první podporoval jak OpenGL, tak i Direct3D od Microsoftu, tedy dvě konkurenční technologie. Což ve své době nemělo obdoby.

Praxe licencování takové technologie, jako je herní engine se ukázala být následně velmi výnosná, cena za licence herního enginu se v tu dobu mohla vyšplhat až k milionu dolarů v závislosti na počtu potřebných licencí. Je paradoxní, že ačkoliv to byl právě idSoftware kdo první přišel s herním enginem, musel nakonec pokleknout před konkurenčními Epic Games a jejich Unreal Enginem, který se stal velmi populární. Unreal Engine používal svůj vlastní skriptovací jazyk UnrealScript a co je však důležité je, že Unreal Engine měl v sobě zabudovaný editor map UnrealED, díky čemuž každá hra, která běžela na tomto enginu mohla být snadno modifikována.

V roce 2004 vychází Half-Life 2, který běží na upraveném GoldSRC enginu, který nese jméno Source. Engine podporuje dynamická světla a stíny, realistické odrazy vodní hladiny, a především revoluční fyzikální model. Ve stejném roce vychází i první CryEngine, který umožňuje vytvářet dříve nevídané obří světy. V roce 2008 vychází hra Battlefield: Bad Company, která je poháněná Frostbite enginem, který umožňuje 92% zničitelnost prostředí. (Paul et al. 2012)

V průběhu několika dalších let vzniká nespočet enginů. Každá velká herní vývojářská firma si udržuje svoje vlastní enginy pro své specifické hry. Electronic Arts tak využívá právě již zmíněný Frostbite engine pro své hry ať už je to nejnovější Battlefield, FIFA nebo Need for Speed, všechny hry běží na stejném enginu. Hry od Ubisoftu tak pohání řada enginů například upravený CryEngine Dunia (série Far Cry), AnvilNext (série Assassins's Creed nebo Tom Clancy's Wildlands) nebo Snowdrop (Division). Pro ty ostatní jsou tu řešení, které nabízejí firmy jako EpicGames, Unity Technologies nebo Crytek, jenž se zaměřují na tvorbu enginu.

4 Dostupné herní enginy a jejich analýza

Tato podkapitola je zaměřena na nejvíce rozšířené herní enginy volně dostupné na trhu. Jedná se především o Unreal Engine od společnosti EpicGames, CryEngine od Cryteku a v poslední řadě o engine Unity od Unity Technologies. Dále se zde věnují stručné historii jednotlivých enginů a následně jejich aktuální verzi, jenž byli pečlivě zanalyzovány pomocí SWOT analýzy.

4.1 Unreal Engine

Unreal Engine je dílem společnosti EpicGames. Jedná se o engine převážně vytvořený pro tvorbu videoher. V nynější době je podporován pouze Unreal Engine čtvrté generace, který byl vypuštěn v roce 2014.

4.1.1 Historie

V této části se věnuji stručné historii vzniku společnosti EpicGames a jejich UnrealEnginu, kde krátce rozebírám technologickou stránku enginů a jejich přínosů pro herní průmysl.

Vznik

V roce 1991, Tim Sweeney zakládá studio Potomac Computer Systems, v Rockvillu v Marylandu. Studio bylo zaměřeno na vývoj počítačových her, a ještě v tomtéž roce vychází hra ZZT, která běžela na DOSu a stál za ní Tim Sweeney. Hra jako taková byla freeware, takže mohla být volně šiřitelná. Časem se studio rozrostlo a změnilo název na Epic MegaGames. Během několik let vydali hry jako Epic Pinball nebo Jill of the Jungle, ale nejvíce se proslavili plošinovkou Jazz Jackrabbit. Za hrou stál Cliff Blezszinski a Arjan Brussee. Inspirací jim byli převážně konzolové plošinovky jako Sonic nebo Super Mario Bross. Hra vyšla v roce 1994 na DOS a byla následně portována i na MS Windows. Hra se stala velmi populární a pokračování na sebe nenechalo dlouho čekat. Druhý díl, který vyšel v roce 1998 je dodnes opěvován jako jedna z nejlepší plošinovek na PC. Zatímco Epic MegaGames zažíváš úspěchy s JackJazz Rabbit, Tim Sweeney od roku 1995 pracuje na něčem co později nazve Unreal Engine. (Plante 2012)

Unreal Engine 1.0

Sweeney začal pracovat na enginu, na kterém by následně mohl postavit first-person střílečku. Tou střílečkou byl Unreal. Hra samotná byla velmi úspěšná a prodalo se jí přes 1,5 milionů kopií. (IGN 2000). Co však bylo mnohem úspěšnější než samotná hra, byl herní engine.

Unreal Engine měl v sobě zabudovanou detekci kolizí, barevné osvětlení a v omezené míře i texture filtering[16]. Unreal používal 16bitovou barevnou hloubku a ambientní efekty jako je volumetrická mlha, o čemž si mohli ostatní enginy nechat jen zdát (Grove 2019).Unreal tak posunul hranice dále. Samotný engine se zprvu spoléhal na softwarový rendering, takže grafické výpočty byli zpracovávány procesorem. V průběhu času však engine začal těžit z příchodu grafických karet. Tim Sweeney však uvádí, že musel přepsat několikrát samotný renderovací algoritmus (Sutorcen 2016). Docílil tím možnosti používat softwarový a hardwarový rendering dle potřeb dané hry a také možnosti mít podporované jak OpenGL tk Direct3D nebo i Glider API. Co se týče zvuku, tak Epic použil technologii Galaxy Sound System. Jednalo se o software napsaný v assembleru[17], který umožnil integraci dalších technologií jako EAX nebo Aureal. Díky čemuž bylo možné použít audio trackery, což umožnilo designerům her plynulý přechod mezi audio stopami a navodit tak atmosféru podle toho co se dělo přímo ve hře. Velkou předností Unrealu byla i umělá inteligence nepřátel. Ta dokázala uhýbat, schovávat se a přepadávat nic netušící hráče ze zálohy. (Lee 2016)

Epic začal svůj engine licencovat i třetím stranám, a tak koncem roku 1999 existovalo již několik her, které běželi na Unreal Enginu jako například Deus Ex nebo Wheel of Time. Na rozdíl od idSoftware, jehož činnost v oblasti licencování jejich enginu spočívala v pouze v dodání zdrojového kódu, Epic poskytoval technickou podporu všem držitelům licence, pořádal meetingy, kde s nimi diskutoval o dalších možnostech vylepšení Unreal Enginu a potřeb samotných vývojářů. V potaz však taky musíme brát cenu licence, která se obvykle vyšplhala ke 3 milionům dolarů (Sutorcen 2016).

Engine také nabízel integrovaný level editor UnrealED, který umožnil stavbu levelu v reálném čase. UnrealED dovolil pomocí skriptovacího jazyka UnrealScript modifikovat hry běžící na Unreal Enginu, což vyvolalo obrovské nadšení v moderské komunitě. Díky moderům tak životní cyklus některých her byl mnohem delší než by si vývojáři samotní dokázali kdy představit a některé moderské komunity okolo těchto her jsou aktivní dodnes (Lee 2016).

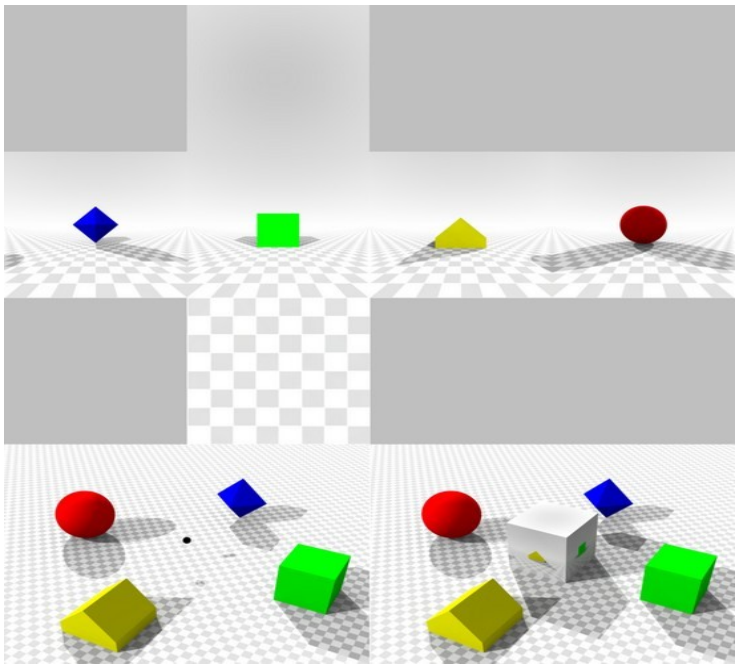
[16] metoda používaná ke stanovení barvy textury pro pixel mapovaný na texturu pomocí barev blízkých pixelů

[17] nízko úrovňový programovací jazyk, jehož základem jsou symbolické reprezentace jednotlivých strojových instrukcí a konstant potřebných pro vytvoření strojového kódu programu pro určitý procesor.

Unreal Engine 2

V roce 2002 vychází hra America's Army, zdarma dostupná multiplayerová střílečka vyvinutá americkou armádou jako rekrutovací platforma, kterou poháněl Unreal Engine druhé generace. Další hrou, která následovala byl Unreal Championship, který byl exkluzivitou pro první Xbox.

Ačkoliv byl Unreal Engine druhé generace založen na svém předchůdci, byl skoro celý přepsán pro potřeby moderních her. Kromě kompletně přepsaného renderování, které nyní dokázalo vykreslit objekty skoro stokrát detailněji než původní Unreal, byl také engine doplněn o několik dalších funkcí. Jednou z těchto nových funkcí byl i systém na animaci koster, který byl poprvé ukázán při prezentaci Unreal Tournamentu pro PlayStation 2. Dále byli přidány nástroje pro tvorbu cutscén, systém částic, který umožnil generovat efekty jako mlha nebo kouř, a dokonce i možnost propojit engine se softwarem třetích stran jako 3D Studio Max nebo Maya. Zlepšení se dočkalo také osvětlení, a to díky použití metody zvané radiozita (Grove 2019). Radiozita je metoda, která dokáže pracovat v reálném čase, ale vychází ze zákona zachování energie, a proto nedokáže pracovat s průhlednými objekty, zrcadly nebo texturami(Goral et al. 1984). V dalších buildech Unreal Enginu druhé generace přibyl i tzv. cube mapping. Jedná se o metodu zobrazování okolí, které je promítáno na stěny krychle jako 6 čtvercových textur. Scéna se tedy vyrenderuje šestkrát a poskládá se do kostky. Oproti sférickému mappingu šlo o velký krok kupředu. Další zásadní změnou byla podpora textur ve vyšším rozlišení. Zprvu tak šlo použít textury 2048x2048 a později dokonce 4096x4096. (Sutorcen 2016)



Obrázek 7: Cube mapping
Zdroj: (Horvath 2020)

Fyzikální simulace jako ragdoll[18] anebo libovolné rigid body úkony byli zprostředkovány pomocí fyzikálního enginu Karma. S Unreal Tournament 2003 přišel update jenž umožňoval simulovat fyziku vozidel skrze Karma engine. To se však příliš nezdálo skupince lidí, kteří si říkali Psyonix, jež vytvořili modifikaci na základě kódu Epicu, pomocí něho předělala způsob, jakým fungovala fyzika pro vozidla. Epic nadšen výkonem lidí z Psyonix se rozhodl implementovat modifikaci přímo do enginu hry a najal je na práci pro další díl Unreal Tournamentu. (Pánek 2015a)

S velkým rozmachem internetu také rostla obliba her pro více hráčů a Unreal Engine 2 byl stavěn i s touto myšlenkou. Umožnil tak aby se ve hře prohánělo až 32 hráčů. Vylepšení se dočkala i umělá inteligence, a to zejména spolupráce s boty a jejich reakce na hráče. (Sutorcen 2016)

Z her, které běželi na Unreal Enginu 2 můžeme zmínit například celou sérii Splinter Cell, Bioshock 1 a 2, Brothers in Arms nebo Killing Floor. Poslední hra, která vyšla a běžela na tomto enginu byl Tom Clancy's Splinter Cell: Blacklist z roku 2013. (Sutorcen 2016)

[18] Procedurální fyzikální animace, která je často používána jako náhrada za tradiční animace například smrti ve videohrách

Unreal Engine 3

V roce 2004 byl představen screenshot Unreal Enginu třetí generace. Engine byl v tu dobu již 18 měsíců ve vývoji a světlo světa spatřil roku 2006, kde byl kompletně představen spolu se hrou Gears of War, jež se stal první hrou běžícím na Unreal Enginu 3.

Vzhledem k modulární povaze technologie společnosti Epic obsahoval Unreal Engine 3 kód z první a druhé generace. Tudíž vývojáři se nemuseli učit s komplet novým systémem, ale pouze s jeho novou verzí. Nicméně některé části enginu byli opět kompletně překopány nebo razantně vylepšeny ať už se jedná o renderovací systém nebo ten fyzikální či zvukový. Narozdíl od druhé generace, která využívala vždy jen fixní počet renderovacích jednotek GPU, Unreal Engine 3 dokázal využít veškerý dostupný výkon. Engine podporoval Direct3D 8 a 9 a také OpenGL 2 a 3. V průběhu let až Direct3D 11. Dalšího vylepšení se dočkalo i osvětlení. Od možnosti využití 64bitové hloubky až po HDR. Kromě osvětlení se lidé z Epicu zaměřili na post-processing efekty jako je light bloom nebo depth of field (hloubka ostrosti) či motion blur, které umožňovalo rozmazávání scén nebo určitých objektů. Zlepšeny byli i stíny, které mohli být již plně dynamické. Mohla být tak použity tzv. měkké stíny, kde, pokud je objekt blíže zdroji světla, jeho stín je rozmazanější a nejasnější, zatímco pokud je objekt dále, jeho stín je mnohem víc ostřejší. Kromě měkkých stínů byla představena i technika self-shadowing, kde stíny nestatických postav mohou dopadat na jiné postavy nebo objekty. Další světelné efekty mohli být prováděny v reálném čase jako například sluneční svit, přímé světlo, bodové světlo anebo lesklé či neprůhledné světelné efekty. Mimo jiné byli zlepšeny volumetrické efekty jako mlha, kouř atd a také se dočkali zlepšení částicové efekty jako exploze, mraky, kapky vody, prach, déšť či sníh. Jedna z dalších věcí, které Unreal Engine třetí generace podporoval byl i parallax mapping. Jedná se o vylepšenou technologii bump mappingu, která umožňuje pomocí optického klamu vykreslit hloubku například mezi jednotlivými cihlami na cihlové zdi nebo různé zvrásnění povrchu (Unreal Technology 2008). Co se týče rozlišení textur tak původní verze podporovala rozlišení 8192x8192 a novější buildy přinesly podporu až 16384x16384. Unreal Engine 3 byl také mnohem více flexibilnější což umožnilo moderům mnohem snáze vytvářet různé mody do her běžících právě na tomto enginu. Umělá Inteligence byla mnohem více sofistikovanější a od původního zaměření na střílečky byla přizpůsobena například i pro závodní hry. Přibyla také mnohem větší podpora dalšího softwaru třetích stran a zde je třeba zmínit podpora PhysX od NVIDIE, je technologie, která umožňuje vypočítat složité fyzické výpočty pomocí grafické karty a byla pouze podporována grafickými kartami od společnosti NVIDIE (Pánek 2015b).

Vzhledem k agresivnímu licencování získala tato iterace enginu velkou podporu několika velkých firem Atari, Activision, Capcom, Disney, Konami, 2K, Midway, THQ, Ubisoft, Sega, Sony, Electronic Arts, Square Enix a dalších. Téměř každá druhá hra té doby běžela na Unreal Enginu 3. Jmenovitě můžeme vybrat například sérii Borderlands, Batmana, Dishonered, několik dílů Gears of Wars, Life is Strange, trilogie Mass Effect nebo i Medal of Honor či dokonce nové díly Mortal Kombatu. Unreal Engine byl také využíván k řadě neherních projektů ať už šlo o různé simulace nebo k návrhům 3D prostředí (Lee 2016).

4.1.2 Unreal Engine 4

Engine byl představen v roce 2012 na akci GDC. Spolu s ním bylo vydáno demonstrační video, které předvedlo Unreal Engine 4 v plné síle. Na GDC byli představeny zásadní novinky, které UE4, měl přinést. Například globální osvětlení v reálném čase nebo přidání tzv. blueprientů, vizuálně skriptovacím systému, pro rychlejší vývoj. Engine vyšel v březnu 2014.

Specifika

Unreal Engine 4 je masivním nástrojem, který nabízí spoustu novinek. Například nový systém globálního osvětlení, umožňuje vytvářet realističtější osvětlení pro interiéry tak exteriéry, a to v reálném čase. Mimo to podporuje UE4 plný ray tracing. Novinkou je i virtual texturing nebo flexibilní editor materiálu, které usnadňují práci s texturami. Mimo jiné UE4 nabízí integrovaný clothing tool od NVIDIE. Kromě látky UE4 umožnuje vykreslit stovky až tisíce fotorealistických vlasů či chlupů v reálném čase. Dalším nástrojem je Niagara, jenž je spíše zaměřená na filmovou tvorbu, umožňuje vytvoření částicových a vizuálních efektů právě pro tyto účely. Kromě VFX nabízí i samostatné integrované nástroje pro tvorbu a úpravu animací. Epic Games i přicházejí se svým novým destrukčním systémem nazvaný Chaos Destruction, který umožňuje dosáhnout působivé kvality destrukce objektů.

Největší novinkou je však vizuální skriptovací systém. Stejně jako mnoho dalších skriptovacích jazyků, se používá k definování objektově orientovaných tříd nebo objektů v rámci enginu. Mimo jiné UE4 nabízí i robustní multiplayerový framework nebo integraci pluginů, všeho druhu. Důležitou součástí je také modul umělé inteligence, který nabízí tvorbu stromu chování a AI Perception, díky němuž může umělá inteligence přijímat informace související se zrakem nebo zvukem.

Licenční model

Spolu s technickým demem Epic na GDC představil i nový licenční model na bázi předplatného. To stálo 19 amerických dolarů za měsíc, za ně dostali vývojáři přístup k plné verzi enginu, včetně zdrojového kódu. Dále však pět procent z každého komerčně prodávaného titulu šlo do kapsy Epic Games. (Orland 2014) O půl roku později Epic vypustil free verzi pro univerzity, a to včetně osobní licence pro každého studenta. O rok později Epic spustil systém grantů, které měly být poskytnuty na kreativní projekty v UE 4. V březnu 2015 Epic Games mění licenční model, ruší předplatné a uvolňují Unreal Engine 4 zcela zdarma. Výměnou za to však požadují 5 % ze zisku, pokud produkt vydělá více než 3000 dolarů za kvartál. (Sweeney 2015)

4.1.3 SWOT Analýza Unreal Engine 4

Tato SWOT analýza se zaměřuje na nejnovější verzi Unreal Enginu od společnosti Epic Games.

Silné stránky

Mezi silné stránky Unreal Engine 4 patří bezesporu grafické možnosti a technická kvalita. Samozřejmě však záleží na jednotlivých vývojářích, ale možnosti, které UE 4 nabízí po této stránce jsou úctyhodné. Nejsilnější stránkou Unreal Enginu je však nový vizuální skriptovací systém. Ten je uživatelsky přívětivý a dostupný. Klíčovou složkou jsou Blueprinty, které nabízejí vizuálně atraktivnější alternativu ke psaní kódu. Další silnou stránkou UE4 je velká komunita kolem tohoto enginu, která se částečně podílí i na jeho tvorbě. S tím souvisí i obrovské množství různé dokumentace. Mezi silné stránky také patří diverzifikace samotného enginu, kde samotný engine není stavěný pouze na tvorbu videoher, ale také je využíván filmaři, architekty a dalšími profesemi.

Slabé stránky

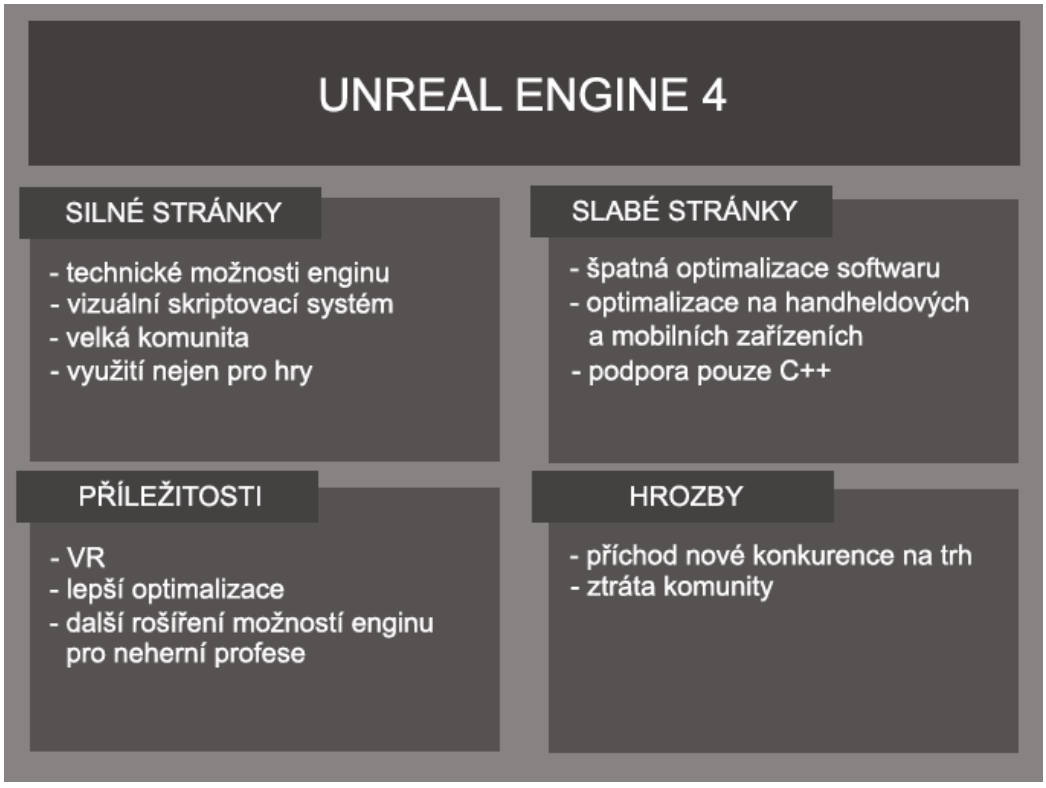
Mezi slabé stránky UE 4 patří špatná optimalizace na mobilních zařízeních. Unreal Engine 4 po vzoru Unity nabízí vývoj na více platforem, ale engine zkrátka není tolik uzpůsoben pro tento typ tvorby. S tím souvisí i ne zrovna dobrá optimalizace softwaru jako takového, kdy pády programu jsou relativně časté. Další slabou stránku je podpora pouze jednoho programovacího jazyka a to C++, což však hned neznamená, že je to nutně špatné, ale oproti konkurenci, která nabízí širší podporu, tak Unreal Engine 4 zaostává.

Příležitosti

Jednou z příležitostí je větší podpora virtuální reality. Zatímco malé projekty většinou vznikají v Unity, Unreal Engine 4 má technologické kapacity, aby umožnil vývojářům vytvořit graficky lépe vypadající VR zážitek. Unreal Engine 4 má na to, aby se stal vedoucím hráčem na poli herních enginů, k tomu ovšem musí zapracovat na lepší optimalizaci softwaru a rozšíření možností pro vývojáře. Velkým skokem bylo představení vizuálního skriptovacího systému, což umožnilo pracovat v enginu i lidem, kteří nemají programování úplně v krvi. Mimo jiné to usnadnilo i práci filmařům, animátorům nebo designerům.

Hrozby

Konkurence za Unreal Enginem tolik nezaostává, Unity již šlape Unreal Enginu na paty a CryEngine je vhodnou alternativou k enginu od Epic Games. Unreal Engine tak i nadále musí dbát na svou věrnou komunitu herních vývojářů a nedat jim důvod odejít ke konkurenci. Hrozbou může být i příchod nového hráče na trh, například firmy, které mají své vlastní interní enginy a rozhodli by se licencovat, mohlo by to velice zamíchat kartami na poli herních enginů.



Obrázek 8: Unreal Engine SWOT analýza
Zdroj: vlastní

4.2 CryEngine

CryEngine je herní engine vyvinutý německou společností Crytek. V současné době je na trhu CryEngine ve verzi pět.

4.2.1 Historie

V této části se věnuji stručné historii vzniku společnosti Crytek a jejich CryEnginu, kde krátce rozebírám technologickou stránku enginů a jejich přínosů pro herní průmysl.

Vznik

V roce 1999 v německém Frankfurtu nad Mohanem vzniká pod rukama tří bratrů Cevata, Avniho a Faruka Yerliových studio Crytek. Jejich první projekt bylo technologické demo nazvané X-Isle: Dinosaur Island, které mělo ukázat sílů jejich technologie, která v té době umožnovala vykreslit obrovský svět, což ostatní enginy té doby neuměly. Společnost NVIDIA byla ohromena z jejich tech dema a rozhodla se s Crytekem podepsat smlouvu o distribuci X-Isle jakožto benchmarkovém softwaru pro jejich grafické karty. V roce 2000 byl Crytek osloven francouzskou společností Ubisoft, aby přetvořili X-Isle do plnohodnotné hry (Crytek 2020b).

CryEngine 1

V roce 2004 tak vychází Far Cry, první hra na této technologii, kterou následně nazvou CryEngine. Engine dokázal vykreslit obrovský otevřený svět, který neměl konkurence. Pomocí pokročilého heightmap[19] systému a polygonů, dokázal CryEngine vykreslit realistické prostředí s dohledem až 2 kilometry. Engine se také pyšnil tzv. Sandbox editorem, který umožňoval editaci v reálném čase (tedy to co umožnuje konkurence až v posledních letech). CryEngine také používal speciální skriptovací systém, který umožnil kombinovat textury různými způsoby tak aby vytvořil různé speciální efekty. Engine podporoval real-time per-pixel lighting, světelné odrazy hrbolatých objektů, lom světa a volumetrické efekty záře a dokázal bez problému vykreslit průhledné displeje a lesklé povrchy. K vytvoření dynamického prostředí engine používal kombinaci předem vypočítaných stínů, a to pomocí šablon a světelných map. Engine také uměl vykreslit měkké stíny a volumetrické efekty. Co se týče fyzikálního enginu, byl relativně pokročilý, podporoval inverzní kinematiku, fyziku vozidel, rigid bodies, fyziku

[19] rastrový obrázek používaný hlavně jako diskrétní globální mřížka v sekundárním modelování výšky. Každý pixel ukládá hodnoty, jako jsou například údaje o výšce povrchu, pro zobrazení v 3D počítačové grafice. (3DMaster 2019)

tekutin, látky a soft-body efektů. Systém umělé inteligence byl také o něco pokročilejší než například konkurence s Unreal Enginem. Umělá inteligence měla přednastavené chování, které však šlo lehce editovat, aniž by se člověk dotknul samotného zdrojového kódu (Yerli 2002).

Far Cry se tak i přesto, že byl vyvinut jako plnohodnotná hra, stal svým způsobem opět grafickým benchmarkem. CryEngine mimo jiné podporoval HDR a byl neustále aktualizován, aby dokázal vytěžit z nových grafických karet maximum. Engine byl také licencován, ačkoliv asi ne úplně v tak velké míře, jak by si vývojáři samotní přáli (Crytek 2004).

CryEngine 2

Stále ještě v roce 2004 pro vydání Far Cry, se na Crytek obrací lidé z Electronic Arts s nabídkou na tvorbu nové herní série. Crytek tak ohlašuje strategické partnerství s EA a práva na Far Cry včetně doživotní licence na CryEngine 1 přecházejí francouzské společnosti Ubisoft, která následně sérii i engine vyvíjí dále. Engine v rukách Ubisoftu získává jméno Dunia Engine, který se používá ve Far Cry dodnes (Berardini 2006). Crytek vyvíjí novou sérii Crysis pro Electronic Arts a spolu se hrou vyvíjí i novou verzi CryEnginu, kterou se následně snaží licencovat. Na CryEnginu 2 vychází v roce 2007 hra Crysis, která se opět stává herním benchmarkem té doby.

CryEngine 2 přichází s technologií Polybump 2, která jim umožňuje vykreslit velmi kvalitní povrch. Technologie funguje na principu, kde každý povrch nese svou informaci. Extrahovaná informace může být pak vykreslena v nízkém rozlišení, avšak model vypadá téměř jako kdyby byl vykreslen v plném rozlišení, a to díky tomu, že data jsou uložena v přechodném souboru, kterým lze manipulovat různými způsoby, aniž by muselo znovu dojít k jeho opětovnému výpočtu. To umožňuje mnohem rychlejší a méně náročné vykreslování (Crytek 2007).

CryEngine také poprvé podporuje i konzole a to Playstation 3 a Xbox360. Podpora DirectX9 a 10 byla samozřejmostí. Další novinky, které CryEngine druhé generace uměl bylo například osvětlení v reálném čase a dynamické měkké stíny, ambient occlusion[20], motion blur nebo hloubka ostrosti. Vylepšení se dočkalo i audio, možné nyní byli například dynamické zvuky prostředí. V neposlední řadě byl vylepšen i systém umělé inteligence. Crytek přišel s tzv. modulárním systémem umělé inteligence, což umožnilo snadnou modifikovatelnost chování umělé inteligence dle potřeb. Dalším pokrokem bylo dynamického hledání cest, což umožnilo

[20] technika stínování a vykreslování použitá k výpočtu toho, jak je každý bod scény vystaven okolnímu osvětlení

umělé inteligenci hledat cestu ve 3D prostředí bez předem vytyčených bodů. Byli představeny také tzv. chytré objekty, které umožnily nastavit chování pro objekt, který následně mohl interagovat s dalšími objekty dle potřeby, například automatické dveře nebo různé pasti či další zařízení (Crytek 2007).

Ačkoliv Crysis tak CryEngine získává jedno ocenění za druhým, vývojářům z Cryteku se nedaří tak úplně prodat jejich engine ostatním vývojářům. Na jaře roku 2007 Crytek oznamuje, že chtějí rozšířit využití CryEnginu. O půl roku později se Ringling College of Art & Design v Sarasotě na Floridě stává první vysokoškolskou institucí na světě, která si licencuje herní engine pro vzdělávací účely (Crytek 2007).

CryEngine 3

Crytek představuje svůj CryEngine 3 v roce 2009 na Game Developers Conference v San Francisku. Engine je vyvíjen pro Windows, Playstation 3, Xbox 360 a dokonce i Wii U. CryEngine 3 umožnil real-time editaci simultánně na všech konzolích a PC v jedné síti. Což usnadnilo vývojářům obrovský kus práce. Engine dále generoval terén pomocí voxelových objektů, uměl generovat vegetaci nebo podporoval real-time měkké částicové efekty. Mimo jiné měl také destrukční systém, cyklus dne a noci s tím také související efekty (Crytek 2009).

CryEngine 3 se však mílovými kroky posunul v oblasti animací postav a umělé inteligence. Animace postav byla integrována do Sandbox editoru a postavy tak mohli být na animovány pomocí parametrů animace koster, která používala procedurální warping a inverzní kinematické algoritmy ke splynutí animací, tak aby vypadali mnohem více uvěřitelněji. Tento zabudovaný systém také používal i automatický lip sync postav, a to jen pomocí audio souborů. Co se týče AI systému, tak ten byl ve třetí verzi enginu také implementován přímo do Sandbox editoru a obsahoval algoritmus hledání cest ve zničitelném prostředí, avšak místo klasické bodové navigace systém používal automatizovanou navigaci v mesh vrstvě. Zvukový systém byl navržen tak, aby efektivně integroval zvuky do realistického prostředí. Toho bylo docíleno použitím tzv. hierarchical area shapes jež dokázali směrovat okolní zvuk závislý na vzdálenosti. Další novinkou v audiu byla schopnost spouštět zvukové efekty v rámci specifických animací, takže se dá říci, že zvuk byl ovládán pomocí animací (Crytek 2011).

V roce 2011 Crytek vydává CryEngine Free SDK. Jedná se o verzi pro nekomerční použití, která je zcela zdarma. V tomtéž roce Australská armáda oznamuje, že spolu s Crytekem vytváří trénovací simulátor pro své piloty (Australian Government - Department of Defence 2011).

Zatímco počet her na předchozích dvou verzích CryEnginu by se vešel do jedné desítky, třetí verze enginu měla o poznání mnohem větší úspěch. Jmenovat můžeme mimo Crysis také Enemy Front, Monster Hunter Online, State of Decay nebo Sniper:Ghost Warrior (Giant Bomb nedatováno).

CryEngine (3.6-4)

Na podzim roku 2013 se Crytek rozhodl k rebrandování celé značky CryEngine. Počínaje verzi 3.6.0. se engine nazýval jednoduše CryEngine a dále oznámili, že další verze CryEnginu nebudou dále numericky označovány. Jedním z důvodů, bylo tvrzení, že tento engine je napsán zcela od nuly a nenese tedy žádné podobnosti se svými předchůdci. CryEngine umožnil podporu Playstation 4, Xbox One, Wii U, Windows i Linuxu (Makuch 2013)

Vývojáři v Cryteku se snažili opět posunou hranice dále a přišli s několika novými nebo vylepšenými technologiemi. Jednou z nich je například Irradiance Volume, což je systém pro vytváření realistického osvětlení, jež je schopen simulovat radiozitu a přidat barvu odraženému světlu. V praxi to znamená, že pokud se bílé světlo odrazí od červeného objektu, jeho odraz bude mít narůžovělý odstín. Dále můžeme zmínit dynamické globální osvětlení v reálném čase, stínování částic (particle shading) nebo parallax occlusion mapping. S plnou podporou DirectX11 přišla také plná podpora hardwarové teselace na všech površích, a to včetně rozpohybovaných postav. Vylepšen byl i fyzikální engine, kde fyzika šla použít téměř na všechno ve hře, a to včetně budov, rekvizit, stromů a vegetace. Tyto objekty dokázali realisticky reagovat síly, jako jsou větrné proudy, výbuchy, gravitace, tření a střety s jinými objekty. Mimo jiné byla podpora i procedurální deformace, která umožnila použití realistické materiálové deformace na površích, jako je kov a plast. Co se týče AI systému, ten nabídla uživatelsky přívětivé prostředí pro tvorbu a úpravu umělé inteligence. Umožnil tak například vývojářům ovládat základní AI pomocí skriptovacího flow diagramu (Crytek 2020a).

Na CryEnginu vyšli hry jako Ryse: Son of Rome, Prey, Evolve nebo také RPG Kingdom Come: Deliverance od českých vývojářů z Warhorse.

4.2.2 CryEngine V

16.března 2016 Crytek na konferenci Game Developer Conference v americkém San Francisku oznamuje novou verzi CryEngine, kterou nazývá CryEngine V. Touto verzí chce Crytek konkurovat EpicGames a Unity Technologies a ukousnout si část trhu s enginy. V této verzi dochází k zásadní změně podmínek licencování, větší náklad k uživatelskému prostředí a

zaměření na VR, a to právě v reakci na Unreal Engine, potažmo Unity. CryEngine V má podporu pro Windows, Linux, Playstation 4, Xbox One, Oculus Rift, OSVR, PSVR a HTC Vive. Podpora mobilních platforem je stále ve vývoji.

Specifika

CryEngine V přináší mnoho zásadních novinek a jde vstříc svým potenciálním zákazníkům. V této verzi se podpory dočkal programovací jazyk C# a změněno bylo kompletně celé uživatelské rozhraní, a to podle novodobých standardů.

Podpory se dočkala i nová verze DirectX, a to konkrétně ve verzi 12. Předělán byl i renderovací engine, což snížilo zátěž hardwaru v náročných situacích. Vylepšeny byli volumetrické efekty, zejména kouře a mraků, a to speciálně pro VR. Vylepšení se dočkali i částicové efekty. Novinkou je dynamická vodní kaustika[21] která funguje v reálném čase.

Co se týče fyzikálního enginu ten nabízí komplexní podporu pro všechny základní fyzikální simulace. CryEngine V se tak může chlubit i například kompletní fyzikou vegetace a jednotlivých jejich částí, což umožňuje vytvořit naprosto realistické prostředí. Engine také umí realisticky spočítat vztlak a chování vody. Dále je podporován i destrukční systém, který umožňuje zničitelné prostředí. V neposlední řadě je nutné zmínit i pokročilá fyzika dynamických objektu jako jsou například řetězy nebo provazy. Novinkou je také tzv. fyzikální přizpůsobení postav. Jedná se v podstatě o systém flexibilní kolize, která je propočítávána v reálném čase, kde bere v potaz všechny okolnosti v závislosti na situaci, což umožňuje realistické interakce s ostatními postavami či okolím. CryEngine V dále nabízí škálovatelnou pokročilou a vylepšenou techniku animací a robustní renderovací systém do něhož například patří technika jako je screen-space subsurface scattering a to na všech podporovaných platformách Jedná se o metodu odrazu světla o různých typů materiálu. Dále například i iris parallax mapping, který obstarává realisticky vypadající oči. U animací můžeme vyzdvihnout parametrické animace koster, díky čemuž lze snadno upravovat animace. V AI systému CryEngine V přináší multi-layer navigation mesh. Jedná se o upravenou verzi dynamického navigace umělé inteligence z předchozích verzí. Umožňuje tak vývojářům jasně stanovit kde se ve 3D prostoru může AI pohybovat. Samotná umělá inteligence se dočkala také výrazného vylepšení, a to pomocí různých modulárních senzorových systémů. CryEngine se také poprvé dočkal svého vlastní zabudovaného a robustního audio editoru ACE, který umožňuje flexibilní

[21] jedná se o optický jev, který způsobuje odraz světla od zakřivené plochy (například sklenice vody skrz kterou prochází sluneční paprsky)

úpravu veškerého audia přímo v Sandbox editoru. Engine také umožňuje propojení s audio softwarem třetích stran. Dále podporuje real-time zpracování 3D zvuků za pomocí funkce nazvané HRTF. Důležitým milníkem je spuštění CryEngine Marketplace, a to po vzoru Unity a Unreal Enginu 4, kde lidé mohou prodávat a nakupovat assety do svých projektů (Crytek 2020a).

Licenční model

S příchodem nové verze CryEnginu byl oznámen i nový způsobem licencování. Dřívější tučná suma za licenci se stala minulostí. Crytek se totiž rozhodl přejít na model "pay what you want", tedy zaplať, kolik chceš. Vývojáři se tak mohli sami rozhodnout, jestli si cení CryEngine na 1000, 500 nebo 10 dolarů, popřípadě, že nezaplatí nic. CryEngine V se tak stal volně dostupným pro všechny. Crytek se rozhodl, že bude dále nabízet placené členství, které umožní přímou podporu od lidí z Cryteku pro vývojáře. Licenční ujednání však i tak nakazovala jisté nutnosti jako logo enginu při zapnutí hry, logo na propagačních materiálech atd., ale Crytek nepožadoval žádné procenta ze zisku při jakémkoliv komerčním užití (Crytek 2020a).

To se změnilo s příchodem verze 5.5, kde se Crytek rozhodl změnit licenční model. Od této verze Crytek požaduje 5 % ze zisku. Engine samotný zůstává zdarma s možností placeného členství a placené podpory. Crytek také nabízí Enteprise verzi, která je určená pro velká vývojářská studia, kde lze licence nastavit podle potřeb. Součástí Enterprise verze je například prémiová podpora ze strany Cryteku, workshopy a edukační zácvik (Crytek 2020a).

4.2.3 SWOT analýza CryEngine V

Tato SWOT analýza se zaměřuje na nejnovější verzi enginu od německé firmy Crytek a to CryEngine V.

Silné stránky

Mezi silné stránky CryEngine V patří rozhodně fyzikální engine, který je oproti konkurenci na dosti pokročilé úrovni. Kromě fyzikálního enginu je třeba také zmínit grafické možnosti tohoto enginu, které jsou taktéž na vysoké úrovni a jejich robustní renderovací systém. V těchto ohledech je CryEngine V daleko za konkurencí. Dále můžeme mluvit i o podpoře nejmodernějších technologií, a to včetně virtuální reality. Jednou z nejsilnějších stránek

CryEnginu je však možnost vytvářet obrovské a realistické světy s úchvatnou vegetací a simulací vody.

Slabé stránky

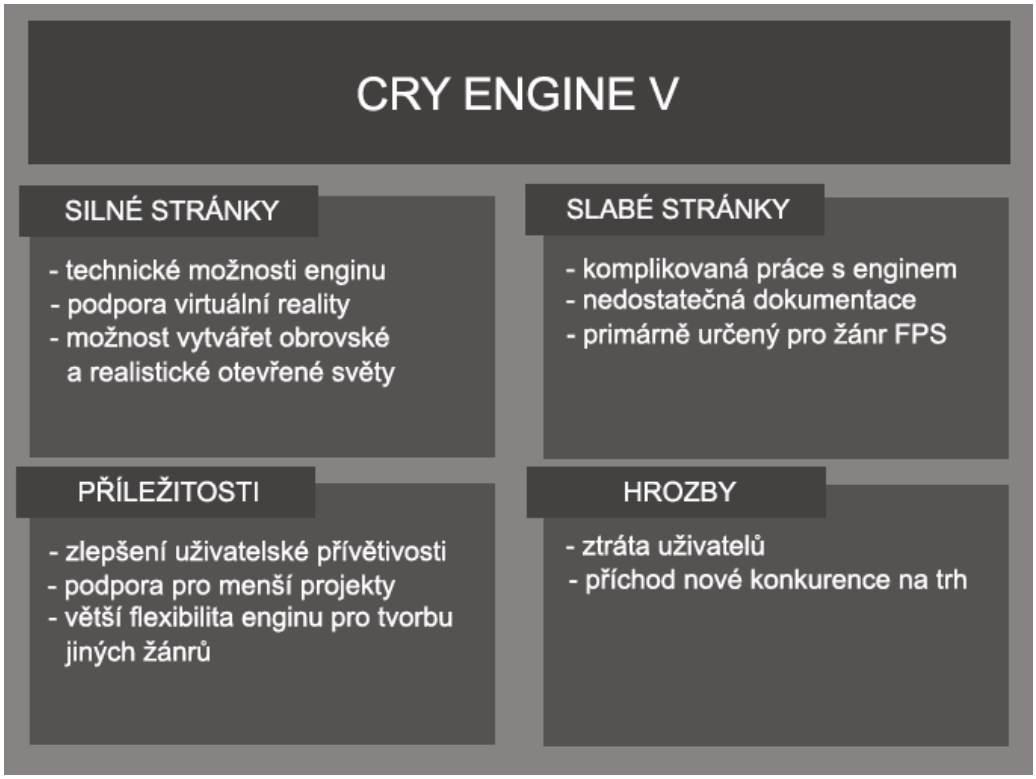
Mezi slabé stránky tohoto enginu rozhodně patří jeho komplikovanost. CryEngine V je velmi komplexní nástroj se kterým šikovní lidé dokáží vytvořit úchvatné hry, ale je to také nástroj velmi komplikovaný. Tomu nasvědčuje i počet různých školení, které Crytek nabízí. Osvojit si tak tvorbu v CryEnginu je velmi náročné a vyžaduje i velké znalosti programovacích jazyků jako C++ nebo C#. Stejně tak jako Unity nebo UnrealEngine nabízí své tržiště, kde lidé mohou obchodovat s assetama do her. Nicméně oproti konkurenci nabízí velmi malé množství assetů. CryEngine je vytvořen především pro žánr FPS a je tak velmi těžké, ne-li téměř nemožné vytvořit žánrově jinou hru. CryEngine také trpí na nedostatek dokumentace a pokročilejší dokumentace potřebná k vývoji je skryta za placenou podporou.

Příležitosti

Příležitostí pro Crytek je mnoho. Především zlepšení uživatelské přívětivosti může vést k většímu rozšíření CryEngine což následně může vést i k většímu zisku pro samotnou firmu. S tímto bodem souvisí i dostupnější a rozsáhlejší dokumentace. K tomu se váže i další příležitost, a to podpora indie vývojářů a malých studií. CryEngine je v tomto ohledu převážně zaměřen na větší studia, ale ty si ve většině případů napíší vlastní engine podle svých potřeb nebo sáhnou po konkurenci, která je v tomto mnohem více flexibilnější. Proto vidím obrovskou příležitost v podpoře malých a nezávislých vývojářů, kterým mohou pomoci růst a z této situace následně mohou profitovat oba subjekty. Podpora začínajících studií může být provedena v několika formách, po workshopy až po odbornou asistenci. Další velkou příležitostí, která může být však velmi náročná na zrealizování, je větší flexibilita enginu, alespoň co se týče diverzity žánrů. Jak již bylo zmíněno ve slabých stránkách, CryEngine se specializuje převážně na žánr first person střílečeк. Pokud by nabídli enginovou podporu pro jiné žánry, věřím, že by mnoho vývojářů po CryEnginu sáhlo.

Hrozby

Největší hrozbou pro Crytek je ztráta potenciálních uživatelů a tím pádem zákazníků. Pokud Crytek nenabídne se svým CryEnginem dostatečnou žánrovou podporu a konkurenci se podaří technologicky předčit CryEngine v žánru FPS, tak to pro Crytek může mít fatální následky. Je třeba aby Crytek se svým CryEnginem nabídl plnou podporu rozdílných žánrů.



Obrázek 9: Cry Engine V SWOT analýza
Zdroj: vlastní

4.3 Unity

Unity engine je vyvinutý dánsko-americkou společností Unity technologies. Jedná se o multiplatformní engine.

4.3.1 Historie

V roce 2004 v Kodani vzniká společnost Over the Edge Entertainment. O rok později vydávají svoji první hru GooBall, která je však komerčním propadákem. Ve studiu se nevzdávají a přemýšlejí o dalším směřováním společnosti. Padne tedy rozhodnutí zaměřit se na vývoj nástrojů pro tvorbu videoher (Haas 2014).

Unity

V červnu 2005 je Unity engine představen na Apple Inc.'s Wolrdwide Developers Conference jako Mac OS X exkluzivní herní engine. OTEE se snažili s Unity jít co nejvíce naproti svým uživatelům a samotní vývojáři enginu o něm mluvili jako o softwaru pro herní vývojáře od herních vývojářů. Ještě v témže roce je Unity dostupné pro všechny na platformě od Applu. První verze tak běžela pouze na Mac OS X a vytvořené projekty tak mohly být spuštěny pouze na tomto operačním systému. To se však změnilo s příchodem verze 1.1, která umožnila exportování projektu na Windows od Microsoftu a do webového prohlížeče. Tento krok nemohl přijít v lepší dobu, neboť jediná alternativa byl Flash od společnosti Adobe, který byl se svým neohrabaným systémem peklo pro vývojáře. Unity tak přineslo hardwarově akcelerovanou 3D grafiku to prostředí webových prohlížečů. Verze 1.1. také přinesla podporu externích C/C++ pluginů (Haas 2014).

Unity 2.0

Druhá verze Unity byla ve vývoji dva roky a vyšla během roku 2007. Šlo o velký skok, neboť verze 2.0 přichází s plnou podporou Windows a zlepšenou podporou pro webové prohlížeče napříč všemi platformami. Druhá verze enginu tak využívá Microsoft DirectX a OpenGL. Mimo jiné také přichází s web streamingem, real-time měkkými stíny, networkingem a zabudovaným editorem terénu, novou verzí katalogů assetů nebo novým zdrojovým kódem pro GUI (Peckham 2019).

S příchodem smartphonů, a především prvního iPhonu od Applu se vývojáři rozhodli zabudovat podporu pro tvorbu mobilních her pro zařízení od Applu. Engine pro tvorbu her pro iPhony byl však vydán samostatně jako Unity iPhone, a to v prosinci 2008 a byl nabízen ve verzi Basic a Pro (Haas 2014).

Na začátku roku 2009 firma mění název na Unity Technologies. Firmě také začíná být jasné, že pokud chtějí více růst, musí vydat engine i na dalších platformách. V roce 2009 tak vychází verze Unity 2.5, která plně podporuje Windows platformu. Jedná se v podstatě o úplně nový engine, neboť celý kód byl přepsán, aby mohl běžet na Windows (Haas 2014).

Unity 3.0

Unity 3.0 vychází 27. září 2010. Třetí verze enginu přináší řadu žádoucích novinek jako sjednocení editoru, lightmapping, deffered rendering, automatické UV mapování nebo audio filtry. Ve verzi 3.0 také přibyla podpora pro konzoli Wii. Touto dobou je již Unity číslo jedna na trhu s hernimy enginami pro mobilní hry. Ve verzi 3.5 přichází dlouho očekávaná podpora pro Flash. Novinkou je také Unity Asset Store, kde výtvarníci nebo designéři mohou prodávat vlastnoručně vyrobené assety (Peckham 2019).

Na této verzi enginu běží hry jako například King's Bounty: Legions, Shadowgun nebo Cities in Motion.

Unity 4.0

Na podzim 2012 vychází Unity ve své čtvrté verzi. Přibyla zde podpora DirectX11, Linuxu a plná podpora Adobe Flash. Přidány byli i nové nástroje pro tvorbu animací. V roce 2013 došlo k dohodě mezi Unity Technologies a Facebookem, kde se zavázali k blízké spolupráci. Výsledkem byla plná podpora Facebookové herní platformy v Unity (Haas 2014).

Z her můžeme jmenovat například karetní Hearthstone od Blizzardu, Plague Inc., Dead Trigger, Angry Birds nebo i Rust či Blitzkrieg 3. Drtivá většina her je stále primárně vyvíjena na mobilní platformy.

Unity 5.0

Unity Engine se v roce 2015 ve verzi 5.0 dočkal řady vylepšení. Mezi ně patří především nový systém osvícení, kde Unity konečně umožnilo globální osvícení v reálném čase a light mapping. Vylepšen byl i audio systém, přibyla také podpora cloudu nebo Nvidia PhysX. Přibyli zde i filmové efekty, aby hry z Unity nevypadaly tak genericky. Unity 5.0 nabídla podporu dalších zařízení a to Playstation 4, Xbox One, Nintendo Switch nebo dnes již ukončenou službu Google Daydream VR (Peckham 2019).

Na verzi 5.0 běží hry jako Cities: Skylines, Fallout Shelter, Angry Birds 2 nebo Pilars of Eternity a také populární Pokémon Go.

Unity 2017

S příchodem nové verze Unity Technologies oznámili, že mění systém verzování svého enginu, a to podle roku. Verze 2017 tak nabídla nový přepracovaný renderovací engine, color grading nebo nový 3D editor. V Unity Technologies se s touto verzí chtěli pustit i do neprobádaných vod mimo herní vývoj. Proto tato verze nabízí řadu nových nástrojů například pro animátory a plné integrování Autodesk 3DS Max nebo Maya do samotného enginu. S touto verzí se také mění model licencování, který do té doby fungoval na klasické placené bázi. Od verze 2017 tak Unity přešlo na předplatitelský model. Pro osobní použití byl engine vypuštěn zdarma (Peckham 2019).

S větší konkurenceschopnosti Unity enginu na poli herních enginů roste i počet vývojářů, kteří hledají alternativu k již zajetým enginům. Začíná tak růst i počet nemobilních her. Jmenovat můžeme například Life is Strange:Before the Storm, Sudden Strike 4, Cuphead nebo Escape from Tarkov.

Unity 2018

S touto verzí přichází obrovský skok v grafických možnostech enginu a Unity se tak více přibližuje konkurenci. Mimo jiné update v sobě zahrnuji i do enginu zabudovaný machine learning nástroje (Peckham 2019).

4.3.2 Unity 2019

Unity 2019 je nejnovější verze Unity enginu vyvinutý společností Unity Technologies. Engine samotný není využíván pouze k tvorbě her, ale také ve filmovém nebo automobilovém průmyslu, kde je používán například k vytvoření plnohodnotného modelu auta pro virtuální realitu. Unity je nejen multifunkční engine, ale také multiplatformní. Nabízí podporu až pro bezmála dvou desítek platforem. Od konzolí současné generace, všechny možné mobilní operační systémy až po virtuální reality nebo chytré televize (Unity Technologies 2020).

Specifika

Důležitou novinkou je Scriptable Render Pipeline, který umožňuje upravit renderovací procesy podle cílové platformy, aby tak vývojáři mohli lépe optimalizovat svůj produkt pro specifický hardware. S ní přichází i Universal Render Pipeline, který cílí na méně početná vývojářská studia a umožňuje jim tak z již předpřipravených řešení pro danou platformu. V neposlední řadě také nabízí ultimátní High Definiton Render Pipeline (HDRP), který ždímá výkon GPU na

maximum za účelem vykreslení co nejvíce realistické grafiky. Je tak mířen specificky pro PC a novou generaci konzolí (Playstation 5 a Xbox Series X). Ale také například pro tvorbu dem v automotive a dalších odvětvích. Unity v této verzi nabízí dlouho chybějící podporu HDR nebo post-processingu. V preview verzi je k dispozici také ray-tracing. Pro výtvarníky přibyli nové nástroje pro animaci. Optimalizován byl běh na mobilních zařízeních (Unity Technologies 2020).

Licenční model

Unit Technologies nabízí dva druhy licencování, individuální a byznys. V individuálním plánu lze získat licenci na Unity engine zdarma pokud je dotyčný student a získá tím přístup nejen k nejnovější verzi enginu ale také k Learn Premium. To umožňuje přístup k prémiové dokumentaci, tutorialovým videím, ale především seancím se samotnými vývojáři Unity. Zdarma lze získat licenci i pro osobní použití, kde je k dispozici samotným engine a pár základních věci jako seznamovací dokumentace a nějaké free assety. O tuto licenci si mohou zažádat i malé organizace, které nepřekračují zisk 100 000 dolarů ročně. Pro ostatní je tu byznys licence, která funguje na bázi předplatného, ty jsou dva druhy. Plus a Pro. Plus stojí 40 dolarů měsíčně, za předpokladu, že roční výnos nepřesahuje 200 000 dolarů ročně. Členové Plus získají mimo jiné přístup k Premium Learninu, Live-Ops analytiku a real-time cloudovou diagnostiku. Členství Pro stojí 150 dolarů měsíčně, a kromě všeho co nabízí Plus verze, obsahuje také licenci pro tři zařízení, prioritní zákaznickou podporu a přístup k poradcům. Pro větší firmy tu existuje i Enterprise řešení, kde je všechno řešeno individuálně (Unity Technologies 2020).

4.3.3 SWOT Analýza

Tato SWOT analýza se zaměřuje na nejnovější verzi enginu od společnosti Unity Technologies a to Unity 2019.

Silné stránky

Mezi silné stránky patří jednoznačně multiplatformita enginu. Vývojáři tak vytváří jednu verzi hry, kterou následně mohou optimalizovat skrze samotný engine na daná zařízení. Práce na optimalizaci je tak značně ulehčena. Počet podporovaných zařízení je oproti konkurenci také obrovský. Další silnou stránkou je uživatelská přívětivost samotného enginu a množství dokumentace, která nováčkům značně usnadní práci. Další velkou devízou Unity je jeho

rozsáhle tržiště s assetama, které funguje již skoro deset let. Za tu dobu se zde nashromáždilo obrovské množství obsahu od programátorů, výtvarníků nebo hudebníků, jejíž práci za nějaký obnos můžete použít ve svém projektu. Další silnou stránkou je royalty free licence, kde samotní vývojáři si neberou žádná procenta ze zisku. Pouze jim platíte měsíčně za licenci.

Slabé stránky

Jednou ze slabých stránek Unity je technická zaostalost oproti konkurenci. Nové věci, které Unity Technologies implementují už u konkurence několik let fungují zcela bezproblémově a jsou často již techniky překonané. Dalším faktem, který tomu nahrává je i zaměření Unity na menší tituly a menší vývojáře. To však nutně nemusí být špatné, ale zároveň tak Unity nenabízí tolik možností na vytvoření pokročilejších titulů, protože to, co umí Unity, umí konkurence a ještě lépe. Další slabou stránkou jsou téměř neexistující nástroje pro tvorbu v týmu. Práce ve větším kolektivu se tak stává velmi obtížnou. Za slabou stránku se dá označit i špatná optimalizace. Ačkoliv se na optimalizaci kódu v Unity Technologies zaměřili v posledních verzích, hry na některých platformách stále haprují. Jde především o náročnější hry nebo hry s otevřenějšími světy, pro které Unity není stavěné.

Příležitosti

Příležitost vidím především ve zlepšení podpory pro velké vývojáře. Pokud Unity nabídne alternativu k Unreal Enginu nebo CryEnginu, po technické stránce, může se stát za několik let jedničkou na poli herních enginů. Samotní vývojáři tomu jdou trošku naproti se svým představením nového renderovacícho systému, nicméně jak se ukázalo, není zrovna dvakrát optimalizovaný pro dnešní hardware. Ovšem s příchodem nových konzolí do konce roku 2020, se může situace změnit. Unity engine je všeobecně považován za engine pro malé vývojáře a malé experimentátorské týmy v rámci velkých vývojářských studií ho také rádi využívají. Takto vznikl například i úspěšný Hearthstone. Pokud by se však Unity Technologies spojili například s technologickým gigantem jako je NVIDIA, po vzoru EA a jejich Frostbite enginu, kde společně nabídli plnou podporu ray tracingu, mohlo by to Unity nakopnout po technologické stránce.

Hrozby

Unity těží především z možnosti importovat svůj projekt na širokou škálu platforem, a to především těch mobilních. Avšak konkurence například v podobě Unreal Enginu, Unity

pomalu, ale jistě dotahuje. Pokud se Unity nebude neustále vyvíjet, tak může nastat situace, že přijde někdo další.

Obrázek 10: Unity 2019 SWOT analýza
Zdroj: vlastní

5 Návrh a aplikace umělé inteligence pro 3D hru

Tato část diplomové práce se zprvu věnuje základnímu stanovení požadavků pro návrh a následnou aplikaci umělé inteligence. Dále se věnuje vhodného herního enginu, a to na základě analýzy a daných požadavků. Poté samotnému návrhu stromu chování pro umělou inteligenci a jeho následnou implementaci do virtuálního 3D prostředí.

5.1 Stanovení požadavků a výběr vhodného herního enginu

Výstupní částí této práce je návrh a následná implementace umělé inteligence pro 3D stealth hru. Žánrově jde o hru, kde se hráč musí dostat z bodu A například do bodu B, aniž by si ho všimli nepřátelé řízení umělou inteligencí. K vytvoření takového dema, je však potřeba sada nástrojů, engine. V této podkapitole definuji požadavky na umělou inteligenci a také rozebírám návrh stromu chování. V neposlední řade se zabývám i výběrem vhodného herního enginu, pro tvorbu tohoto projektu.

5.1.1 Definování požadavků a návrh stromu chování

Ze všeho nejdříve je nutné definovat co by umělá inteligence měla umět. Pro potřeby dema potřebujeme, aby se umělá inteligence co nejvíce přiblížila chování strážného. To znamená, aby hlídkovala v určitém svém teritoriu, a to nejlépe podle vytyčené trasy. Dále je nutné, aby umělá inteligence dokázala vnímat své okolí. To znamená, že strážní musí vidět a slyšet. Aby mohl strážný vidět, bude třeba mu nastavit zorné pole a reakce na určitě objekty, které se v jeho zorném poli ocitnou. U sluchu bude nutné nastavit rádius, ve kterém strážný bude mít šanci zaslechnout podezřelé zvuky jako například běh hráče nebo spadnutí objektu. Pokud strážný uvidí hráče ve svém zorném poli, měl by se dát do pronásledování. Pokud se mu při pronásledování hráč ztratí, měl by strážný prohledat blízké okolí a pokud nic nenajde, měl by se vrátit zpět ke svému hlídkování. Pokud však strážný hráče chytne, měl by ho začít mlátit například obuškem, proto bude třeba implementovat i model poškození.

5.1.2 Výběr vhodného enginu

Na základě podrobné analýzy a funkcí jednotlivých enginů jsem zvolil Unreal Engine 4 od společnosti Epic Games, a to z několika důvodů. Nejzásadnějším z nich je možnost vizuálního skriptování pomocí blueprintů, jenž mi usnadní práci na tomto projektu. Dalším důvodem je senzorový systém umělé inteligence v Unreal Enginu 4, díky čemuž bude snadnější vyřešit část

se sluchovým vjemem. Mimo jiné UE umožnuje snadnější práci se stromem chování, který je napojen na vizuální skriptovací systém a je součástí licence zdarma, oproti například Unity, kde se plugin pro tvorbu stromu chování stojí 80 dolarů.

5.2 Implementace umělé inteligence do virtuálního 3D prostředí

Tato podkapitola je zaměřená na tvorbu základní struktury stromu chování v Unreal Enginu 4, kde je popisován postup a základní funkce enginu potřebné pro tvorbu stromu. Následně se práce věnuje postupu implementace vzoru chování "strážného".

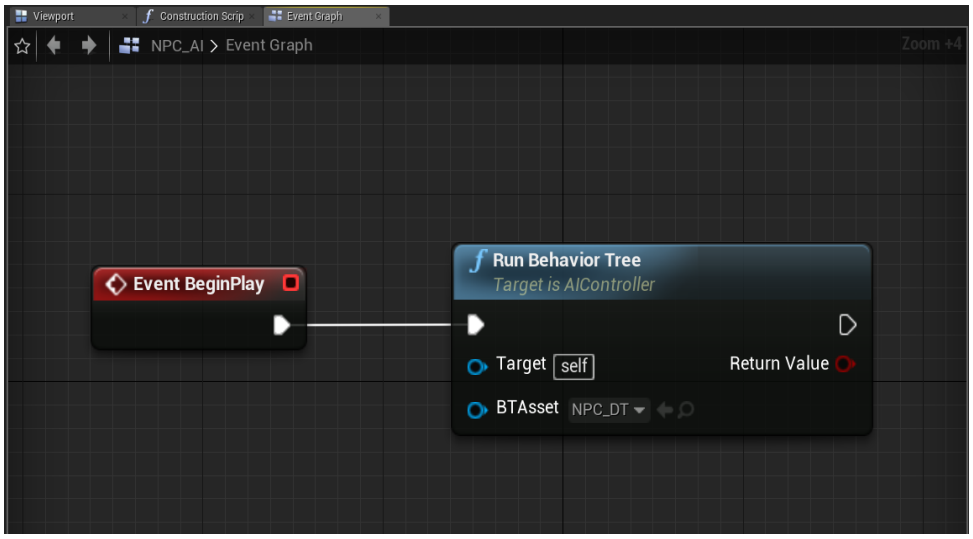
5.2.1 Tvorba základní struktury stromu chování v enginu

V prvé řadě je třeba vytvořit novou třídu blueprintu. Tento blueprint bude třídy charakter[22], který pojmenujeme NPC[23]. To v praxi znamená, že daný objekt bude mít mesh a dále i zapnuté kolize s okolním virtuálním světem, což je důležité. Poté se jen jeho kostra propojí s modelem v tomto případě s tzv. manekýnem, který slouží jako základní model, který bude využíván pro potřeby této práce.

Dále je potřeba vytvořit další blueprint, který umožní propojit strom chování s daným NPC. Pojmenujeme ho NPC_AI a přiřadíme jako atribut k NPC. Nyní se lze pustit do samotné tvorby stromu chování. Unreal Engine umožňuje vytvořit, exportovat a částečně i importovat různé flowcharty a stromy rozhodování. Vytvoříme tedy nový strom chování a pojmenujeme ho NPC_DT (non playable character _ decision tree). Kromě stromu chování je třeba ještě vytvořit tvz. BlackBoard. BlackBoard slouží jako místo pro zápis dat se kterými lze pak pracovat v rámci stromu chování.

V blueprintu NPC_AI vytvoříme EventGraph, který slouží pro tvorbu a správu událostí a k následném volání funkcí a k provádění akcí v reakci na události ve virtuálním světě spojené s daným blueprintem. Zde, jak lze vidět na obrázku níže se po spuštění události vyvolá funkce Run Behavior Tree, ke které byla přiřazen strom chování NPC_DT.

[22] postava
[23] non playable character (postava řízená umělou inteligencí)



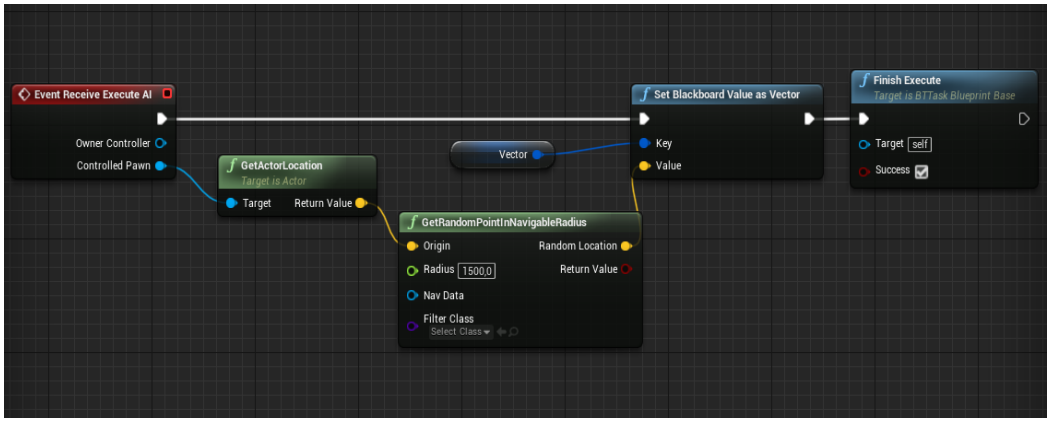
Obrázek 11: Ukázka Event Graphu
Zdroj: vlastní

Nyní je čas na vytvoření samotného stromu chování. Strom začíná rootem, což je počátek, ze kterého budeme vycházet. Zde také musíme určit několik tzv. composite nodes neboli česky uzlů. Každý z uzlů má své specifické schopnosti, které jsou klíčové při sestavování stromu rozhodování. Selector vyhodnocuje uzly z leva doprava a zastaví se v momentě, pokud jeden z jeho pod uzlů uspěje, pak se Selector vyhodnotí jako úspěšný. Pokud všechny jeho pod uzly neuspějí, pak se vyhodnotí jako neúspěšný. Sekvence spouští uzly stejně tak jako Selector zleva doprava, ovšem přestane v momentě, kdy jeden z jeho pod uzlů se vyhodnotí jako neúspěšný, pak se celá Sekvence vyhodnotí jako neúspěšná. K tomu, aby Sekvence mohla být úspěšná, musí být všechny její pod uzly vyhodnoceny jako úspěšné.

Dalším krokem je tedy zvolení vhodných uzlů. Zde se jako vhodný kandidát jeví Selector na který se naváže Sekvencí. Sekvence bude vyhodnocovat task[24], který bude dávat instrukce umělé inteligenci. Tento task bude mít jediný cíl a to, říct umělé inteligence kam by se měla posunout ve virtuálním světě. Task se tvoří pomocí již výše zmíněného EventGraphu a to pomocí sledu událostí. Zprvu se tedy vytvoří startovací událost, která se spustí v momentě, kdy se začne vyhodnocovat uzel Sekvence. V EventGraphu tedy vytvoříme sled několika událostí začínající zjištěním polohy NPC v rámci virtuálního světa. To lze zjistit pomocí funkce GetActorLocation, která vypíše souřadnice X,Y,Z se kterými pak následně může pracovat funkce GetRandomPointNavigableRadius. V tuto chvíli je třeba vytvořit nový Blackboard klíč, v tomto případě vektor. Zpět v EventGraphu je nutné vytvořit novou proměnnou vektor u níž

[24] Úkol či zadání

je třeba nastavit typ na Blackboard klíč. Proměnná vektor musí být public, aby se s ní mohlo i nadále pracovat i mimo tento task.



Obrázek 12: Task – FindRandomLocation (vizuální zobrazení)
Zdroj: vlastní

Hotový task, pojmenovaný FindRandomLocation, se následně přidá do stromu chování. Další task, který bude vyhodnocen bude MoveTo a Wait, kde je nutné nastavit sekundový interval během, kterého se nic dít nebude. Následně celý koloběh započne znovu. Takto vystavěný strom chování ovšem jenom způsobí to, že NPC se bude pohybovat po světě naprosto náhodně.

5.2.2 Implementace zraku

Aby NPC postava mohla vnímat například vidět a slyšet hráčskou postavu, je nutné ji přiřadit smysly. Ve stromu rozhodování je nutné vytvořit další sekvenci, kterou lze pojmenovat například ChasePlayer, protože to je přesně to, co bude v této fázi od umělé inteligence požadováno, aby začala pronásledovat hráčskou postavu. K tomu je třeba vytvořit další nový task, kde použijeme funkci GetPlayerCharacter. Pokud funkce vrátí hodnotu null, pak hráčova postava neexistuje nebo není typu character. Pokud však vrátí správnou hodnotu spustí se funkce GetActorLocation, která zjistí souřadnice hráčovy postavy ve virtuálním světě a následně se postupuje stejným způsobem jako v případě předešlého tasku. Dokončený task, FindPlayerLocation přiřadíme do sekvence ChasePlayer v NPC_DT. Následně je nutné ještě přiřadit task MoveTo s BlackBoard klíčem, který určuje cílové souřadnice, kam se má umělá inteligence vydat v rámci virtuálního světa.

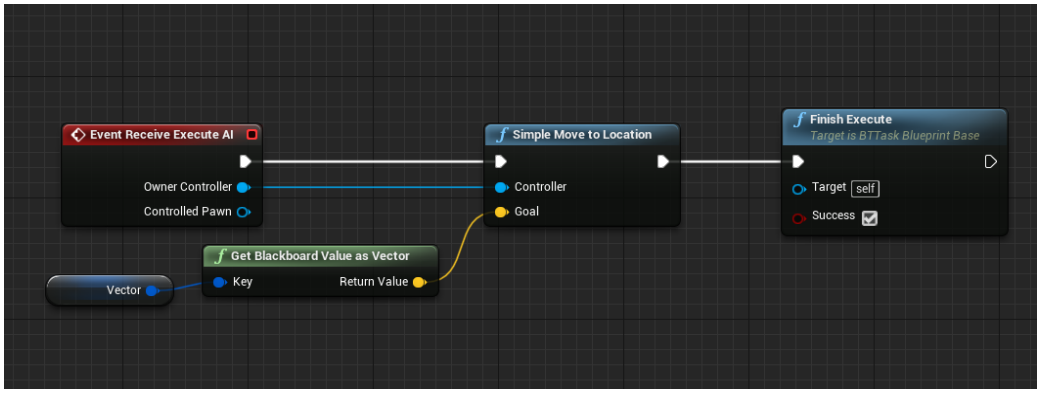
V této fázi existují dvě větve u stromu rozhodování, které jsou defaultně vyhodnocovány zleva doprava. Nicméně, aby umělá inteligence dělala to, co je potřeba, je nutné ji pomoci tím, že určíme podmínky, kdy se jaká větev má vyhodnocovat. V tomto případě, je potřeba vytvořit podmínku, že když NPC uvidí hráče, pustí se do pronásledování, tj. spustí se sekvence ChasePlayer. Pokud tedy hráč nebude v zorném poli daného NPC, bude se věnovat své rutině. K vytvoření podmínky je potřeba vytvořit nový Blackboard klíč. Klíč bude typu boolean, neboť v tomto případě mohou nastat pouze dvě situace a to, že NPC hráčskou postavu vidí anebo nevidí, tudíž klíč bude mít hodnotu jedna nebo nula.

Vytvořený klíč je následně použit k vytvoření Decoratoru, který v Unreal Enginu zastupuje funkci podmínky. Je nutné nastavit podmínku na IsNotSet, tedy aby se podmínka splnila, pokud hodnota klíče bude nepravda. Tedy pokud NPC nevidí hráče, věnuje se tomu, co dělat má. Stejně tak se vytvoří druhý Decorator u Sekvence ChasePlayer, ovšem s podmínkou, že pokud klíč vrátí hodnotu 1 tj. hráč je viděn NPC, tak se spustí daná sekvence pronásledování.

K tomu, aby umělá inteligence mohla vůbec hráčovu postavu vidět je nutné ji přiřadit schopnost vnímat. To umožňuje knihovna AIPerception, která v sobě obsahuje několik funkcí jako zrak, sluch či hmat. Zde je nutné povolit Auto Register as Source.Vzhledem k tomu, že NPC potřebuje zrak zvolíme funkci AISight, kde lze nastavit radius, periferní vidění atd. V praxi to funguje tak jako lidské oko, tak jako člověk i NPC ovládané umělou inteligencí má zorné pole. Ovšem umělá inteligence narozdíl od člověka může podvádět tím, že se lehce poupraví parametry jejího zorného pole a tím se samotné pole zvětší. K tomu, aby byla hráčova postava vůbec viditelná a NPC jí mohla zaregistrovat ve svém zorném poli je třeba ke hráčově postavě přidat funkci AIPerceptionStimuliSource a následně povolit AISense_Sight. V EventGraphu se dále vytvoří nová počáteční událost OnTargetPerceptionUpdated, která se spustí v momentě kdy NPC uvidí hráčovu postavu. K tomu, aby umělá inteligence dále věděla, co má vůbec vidět je potřeba použít CastTo. Jedná se v podstate o kontrolu, jestli výchozí objekt (NPC) je stejného typu jako cílový objekt (hráčova postava) a pokud ne, je ho třeba definovat.

Pro splnění stanovených požadavků je však třeba, aby NPC začalo hráčskou postavu pronásledovat. V nynějším stavu pouze dojde na místo, kde se hráč nachází, pokud je hráč viděn. Je nutné tedy vytvořit další task ChasePlayer. V EventGraphu tohoto tasku opět vytvoříme počáteční událost, tentokrát EventReceiveExecuteAI, která se spustí v momentě, kdy

se task začne vyhodnocovat ve stromě rozhodování. Následně se použije funkce SimpleMoveToLocation, kde jako proměnou využijeme Blackboard klíč jako vektor.



Obrázek 13: Task – Chase Player (vizuální zobrazení)
Zdroj: vlastní

V sekvenci ChasePlayer vložíme již vytvořený task ChasePlayer. Nyní již NPC ovládané umělou inteligencí může pronásledovat hráče, nicméně z hlediska animace se model glitchuje a to z důvodu neustálé změny polohy vůči virtuálnímu světu. Proto je nutné použít Decorathor zvaný Cooldown, který umožní vykonávat danou sekvenci v určitém časovém rozmezí. V tomto případě je vhodné nastavit pauzu zlomek sekundy např. na 0,2 sekundy. Může se však lišit v závislosti na druhu animace.

5.2.3 Implementace hlídkové trasy

Aby se umělá inteligence mohla chovat jako strážný, musí umět hlídkovat, tj. musí se umět pohybovat z bodu A do bodu B, popřípadě z bodu B do C atd. K tomu je nutné v blueprintu NPC vytvořit novou proměnnou patrol_path, která musí být public, protože se s ní dále bude pracovat i mimo tento blueprint. Následně je potřeba vytvořit nový blueprint třídy Actor, který pojmenujeme taktéž PatrolPath. PatrolPath v sobě bude uchovávat informace o souřadnicích hlídkových bodů. Tedy těch bodů kudy by se měla umělá inteligence procházet. V PatrolPath se posléze vytvoří nová proměnná s názvem patrol_points a protože bude udávat souřadnice hlídkovacích bodů v prostoru bude proměnná typu vektor a následně místo jedné proměnné zvolíme array a to z důvodu, že budeme potřebovat více hlídkovacích bodů, které budou ukládány do pole. Proměnná musí být opět public a pro vizualizaci a následnou editaci v 3D prostoru lze zaškrtnout i políčko show 3D widget.

Nyní lze v 3D prostředí pomocí blueprintu PatrolPath rozmísťovat po virtuálním světe hlídkovací body. Toho lze dosáhnout jednoduchým přetažením blueprintu přímo do okna 3D prostředí. Při kliknutí na 3D objekt hlídkovacího bodu se otevře nabídka, kde lze přidávat další hlídkovací body, a to buď pomocí zadání souřadnic X, Y, Z nebo pomocí vizuálního umisťování přímo v 3D editoru, ovšem pouze za předpokladu, že tato možnost byla povolena v nastavení proměnné patrol_points.

Následně je nutné v blueprintu NPC změnit typ proměnné patrol_path na objekt PatrolPath. Poté ve 3D prostředí se konkrétnímu modelu přiřadí proměnná patrol_path, tím je k danému NPC přiřazena vytvořená hlídkovací trasa.

V tuto chvíli však umělá inteligence neví, co by měla dělat a proto, je potřeba vytvořit další větev ve stromě rozhodování. Také je potřeba vytvořit další task s názvem FindPathPoint a v neposlední řadě udělat další Blackboard klíče, se kterými bychom mohli pracovat. První z nich ponese název PatrolPathVector a jak už název napovídá, bude se jednat o proměnou typu vektor. Druhý klíč bude typu integer a ponese název PatrolPathIndex. Ten budeme potřebovat, aby umělá inteligence věděla, ke kterému bodu na své hlídkovací trase má zrovna jít. V tasku FindPathPoint jako počáteční událost zvolíme EventReceiveExecuteAI, aby se task spustil v momentě, kdy bude vyhodnocován. Následně je nutné pro potřeby tasku vytvořit další proměnné, path_vector a path_index,kde obě budou typu Blackboard klíč a samozřejmě opět public. Nyní použijeme CastTo, které zjistí, jestli se jedná o NPC a když ano přiřadí NPC hlídkovací trasu, respektive souřadnice pomocí funkce Get, která čerpá z informací uložených v poli patrol_points. Zde přichází i jistý neduh Unreal Enginu, kdy NPC actor používá X,Y,Z souřadnice, které jsou stanoveny světem. Tudíž pokud se počáteční bod nachází v bodě 0,0,0 a NPC dojde na počáteční bod, jeho souřadnice jsou taktéž 0,0,0. Ovšem tohle neplatí u hlídkovacích bodů. Ty neberou počáteční bod 0,0,0 z bodu daného 3D prostorem ale z bodu daného prvním hlídkovacím bodem. Tudíž pokud se umístí počáteční hlídkovací bod na souřadnice 25,30,0, bude pozice hlídkovacího bodu 0,0,0 a souřadnice dalších hlídkovacích bodů se odvíjejí od počátku, který je daným prvním bodem. Aby NPC mohla hlídkovat a správně využívat dané hlídkovací body, je potřeba souřadnice transformovat. V tasku je tedy nutné zvolit funkci GetActorTransform a následně TransformLocation, což se pomocí další funkce SetBlackboardValueAsVector přetransformuje v jednotné souřadnice, se kterými lze následně lépe pracovat viz. příloha A Task-FindPathPoint.

Nově vytvořený task nyní stačí vložit do nové větve stromu chování. Nesmí se však zapomenout na to, že uzly jsou vyhodnocovány zleva doprava a proto, pokud chceme, aby NPC strážný vykonával hlídkovací rutinu a pokud by při hlídce zahlédl hráče a následně se za ním pustil do pronásledování, musí být nová větev umístěna první zleva, viz. obrázek 15. Jestliže je již task FindPathPoint vložen, musí se také vložit task MoveTo. Zde je potřeba změnit Blackboard klíč na PatrolPathVector. Následně je nutné vytvořit ještě další task PathIndex+, který bude fungovat jako jednoduché počítadlo a při každém projetí celé větve přičte k indexu +1. Umělá inteligence tak při dalším vyhodnocování větve a tasku s hlídkovací trasou použije souřadnice z pole o jedna větší. Tudíž bude vědět, že z bodu A má jít do bodu B a z bodu B do bodu C atd. Pro ošetřeni možných chyb je potřeba vytvořit kontrolu, která bude hlídat to, aby přírůstek indexu nebyl nikdy větší než velikost pole. Zde následně vytvoříme podmínku. Pokud bude přírůstek menší nebo roven velikosti pole, task se vyhodnotí jako úspěšný. Pokud ovšem bude přírůstek větší, task se vyhodnotí jako neúspěšný. PathIndex+ task tak nebude napojen přímo na sekvenci, ale je zde nutné vytvořit selektor mezi sekvencí a taskem a to z důvodu, že pokud task bude neúspěšný, začne se následně vyhodnocovat další task v pořadí. Po vložení je dále nutné nastavit ve vlastnostech tasku path index na PatrolPathIndex. Celý EventGraph tohoto tasku lze vidět v příloze B.

Další task v pořadí, který potřebujeme, aby byl vyhodnocen je task, jež je nutné opět vytvořit. Bude se jednat o task, který bude mít na starost to, aby se umělá inteligence na konci své hlídkové trasy nezastavila, ale místo toho pokračovala zase opět znovu stejnou trasu. Vytvoříme tedy další task LoopPath. V EventGraphu opět vytvoříme jako počáteční událost EventReceiveExecuteAI a spolu s událostí vytvoříme i proměnou path_index typu Blackboard klíč a pomocí funkce SetBlackboardValueAs nastavíme na integer. V blueprintu PatrolPath vytvoříme novou public proměnou loop a potřeba bude také nový Blackboard klíč typu boolean, PathLoop. V tasku LoopPath je také potřeba vytvořit další novou proměnou vázanou na Blackboard klíč PathLoop, proto ji nazveme například BB_loop. Pomocí funkce GetBlackboardValueAsBool zajistíme, že výsledek bude typu boolean, díky čemuž můžeme vytvořit novou podmínku. Když se podmínka splní path_index se vynuluje což zapříčiní to, že umělá inteligence při dalším hledání trasy vybere počáteční bod své trasy, tj. souřadnice, které jsou na první pozici v poli patrol_points. U hotového tasku je nutné ve vlastnostech nastavit PatrolPathIndex a PathLoop. Pro úplnou funkčnost musíme otevřít blueprint NPC_AI a přidat několik věcí do EventGraphu. Použijeme funkci GetControlledPawn a ověříme, zda se jedná o NPC, a to pomocí funkce CastToNPC, díky čemuž docílíme toho, že budeme pracovat s

konkrétním NPC actorem. Dále potřebujeme z funkce CastToNPC získat výstup ve formě proměnné. V eventovém okně CastTo zvolíme As NPC a následně protáhneme eventovou linii a vybereme promote to variable. Proměnou pojmenujeme my_pawn. Nyní můžeme použít PatrolPath s proměnou loop a také získat data z Blackboard pomocí funkce Get. Dále použijeme i funkci SetValueAsBool, abychom nejenom zajistili výstup jako boolean ale především, proto, že v tomto momentě potřebujeme zapojit Blackboard klíč PathLoop. Z již zmíněné funkce použijeme tedy Key Name a funkční MakeLiteralName, kde hodnota bude přesný název klíče, tedy PathLoop. Poté stačí ve 3D otevřít vlastnosti PatrolPath a zaškrtnout políčko Loop. Umělá inteligence nyní hlídkuje z bodu A do bodu B, z bodu B do bodu C a z bodu C jde opět do bodu A a celý cyklus se opakuje.

Ne každý člověk však postupuje přímo tímto způsobem. Pro větší nahodilost a diverzitu chování bude třeba vytvořit jiný vzorec chování. Druhým vzorce tedy bude, že se umělá inteligence bude vracet zpátky po bodech po kterých přišla. Tudíž, půjde-li z bodu A do bodu B odtud do bodu C, nepůjde z bodu C rovnou do bodu A, ale vrátí se přes bod B. K tomu bude potřeba nový task, velmi podobný PathIndex+, který přičítal přírůstky. Co, ale je tedy potřeba je, aby se na konci path_index nevynuloval, ale aby měl záporný přírůstek a v momentě, kdy dojde na nulu se opět začal přičítat kladné přírůstky. Pro usnadnění lze vzít základ PathIndex+. Task tedy z duplikujeme a přejmenujeme na PathIndex-. V EventGraphu je důležité změnit přičítání +1 na -1. Dále je vhodné smazat CastTo a vše spojené s polem path_points, protože tato část kontrolovala konce pole path_points, což je teď naprosto nepotřebné. Další změna se týká části, kde se porovnávali hodnoty přírůstku a pole. Zde je třeba změnit porovnávání z menší nebo rovno na větší nebo rovno. V dalším kroku se vytvoří nový Blackboard klíč Direction typu integer. V tuto chvíli by totiž umělá inteligence chodila pouze mezi dvěma body, proto je nutné jí udat směr, aby věděla, že má nejdřív dojít na konec a pak až se vrátit. Otevřeme tedy znovu task PathIndex+, kde vytvoříme proměnou direction, kterou propojíme s Blackboard klíčem. Následně použijeme funkci GetBlackboardValueAsInt a to porovnáme zda se rovná hodnotě 1, což bude tomto případě znamenat cesta dopředu. Přidáme podmínku, že pokud se hodnota rovná 1, tak NPC postupuje dopředu, a tudíž přičte přírůstek. Pokud ne, task se vyhodnotí jako neúspěšný. Ovšem pokud chceme, aby umělá inteligence pokračovala dál, je nutné při druhé podmínce před koncem tasku PathIndex+ znovu použít proměnou direction napojenou na funkci SetBlackboardValueAsInt, kdy v momentě, kdy je druhá podmínka vyhodnocena jako true hodnota je 1, pokud však false hodnota proměnné je -1. Což v praxi znamená, že jakmile NPC dorazí na konec cesty, změní se hodnota proměnné direction a ona bude vědět, že má jít

opačným směrem. Totožným způsobem se bude postupovat u PathIndex-, akorát s tím, rozdílem, že je nutné nastavit hodnoty opačně. Tudíž hodnota direction se bude porovnávat s -1. Dále je důležité stanovit počáteční hodnotu proměnné direction. Bez ní by NPC jen stálo v prostoru a nevědělo co má dělat. Hodnotu direction nastavíme v blueprintu NPC_AI, pomocí funkce GetBlackboard a následně SetValueAsInt. Jako Key Name použijeme funkci MakeLiteralName, která se bude odkazovat na název proměnné direction a v IntValue nastavíme hodnotu na 1. takže umělá inteligence bude procházet trasu od počátku.

Pro větší uvěřitelnost je vhodné přidat pauzu, mezi hlídkováním. Respektive, aby se NPC zastavilo v bodě a chvíli pozorovalo okolí. K tomu slouží task Wait Blackboard Time, který slouží jako pauza, než se vyhodnotí další task nebo větev. Čas se bude řídit podle Blackboard klíče, proto je nutné jeden vytvořit. Klíč musí být typu float. Stejně tak jako bylo potřeba definovat proměnou direction, je nutné definovat i námi vytvořený klíč WaitTime. Opět tedy v blueprintu NPC_AI postupujeme totožně akorát s rozdílem, že musíme dávat pozor na to, aby byla použita funkce SetValueAsFloat, protože pracujeme s datový typem float. Následně v blueprintu PatrolPath vytvoříme proměnou WaitTime, opět typu float a zde nastavíme hodnotu na 0,5. Proměnou nastavíme public, aby mohla být snadno editovatelná přímo z rozhraní 3D prostředí. Zpět v NPC_AI využijeme transformovanou proměnnou my_pawn a pomocí GetPatrolPath a GetWaitTime navážeme na funkci SetValueAsFloat.

5.2.4 Hledání hráče a návrat k hlídkovací trase

Nyní již umělá inteligence zvládne hlídkovat po trase a v případě, že uvidí postavu ovládanou hráčem tak ji okamžitě začne pronásledovat. Pro dodržení daných požadavků potřebujeme, aby umělá inteligence, pokud se jí hráč ztratí z dohledu, začala prohledávat blízké okolí, kde viděla hráče naposledy a když ho nenajde, vrátí se zpět ke své hlídkovací trase.

Začneme tím, že ve stromě chování NPC_DT, přidáme Decorator k sekvenci PatrolPath. Tedy k sekvenci, která má na starost hlídkování. Decorator bude založený na podmínce vycházející z Blackboard klíče CanSeePlayer. U Key Querry nastavíme IsNotSet. Tudíž bude totožný jako již jednou vytvořený Decorator na počátku části této práce. Účelem této podmínky je, aby se umělá inteligence věnovala hlídkování, za předpokladu, že neuvidí hráče. Pokud hráče uvidí, je nutné nastavit, aby umělá inteligence ihned zanechala hlídkování a dala se do pronásledování. Toho lze docílit tím, že se ve vlastnostech vytvořeného Decoratoru nastaví hodnota Observer

aborts na both. To bude mít za následek to, že se veškeré vyhodnocení všech uzlů v této větvi pozastaví a vyhodnocování se ve stromě chování vrátí na počátek kde Selector vybere další větev, kde se již nachází sekvence ChasePlayer.

Dalším krokem bude upravení rychlosti pohybu NPC. Pro dodržení zadání, je potřeba, aby se chování umělé inteligence přiblížilo co nejvíce chování reálného člověka. Ten totiž narozdíl od počítačem ovládané postavy neběhá z bodu A do bodu B a podobně, ale svou trasu absolvuje v drtivé většině chůzí a až posléze, pokud se by zahlédl narušitele tak se dá do jeho pronásledování během. Ve stromě chování je tedy nutné vytvořit tzv. službu. Služba je v podstatě skript vázaný na Selector, Sekvenci nebo Task, který se spouští spolu s nimi. Službu pojmenujeme ChangeNPCSpeed, protože skrz ni budeme schopni upravit rychlost pohybu NPC. V EventGraphu služby vytvoříme jako počáteční událost EventReceiveActivation, což znamená, že se spustí, jakmile začne být vyhodnocována sekvence. Odtud budeme muset použít CastTo a zvolíme NPC_AI. Následně pomocí funkce Get získáme proměnou my_pawn, která odkazuje na objekt NPC strážného a odtud budeme potřebovat zjistit hodnoty komponenty[25] CharacterMovement, opět pomocí funkce Get. Odtud jíž můžeme použít funkci SetMaxWalkSpeed. Dále je nutné vytvořit novou public proměnnou float s názvem speed[26], kterou napojíme na již vytvořenou funkci, a to z důvodu snadnější editace rychlosti přímo ve stromě chování. Zpátky ve stromě chování vložíme k sekvenci Patrol Path službu ChangeNPCSpeed. Ve vlastnostech služby již vidíme možnost editace rychlosti. Zde nastavíme požadovanou rychlost pohybu pro hlídkování. Následně je nutné přidat službu i u sekvence ChasePlayer, kde ovšem rychlost pohybu bude mnohem větší, protože by se mělo jednat o běh. Umělá inteligence nyní, když zahlédne hráče při své obchůzce, tak se dá do pronásledování během. Jakmile ztratí hráče z dohledu, zase zvolní a chůzí pokračuje tam kde skončila.

Nyní potřebujeme, aby umělá inteligence prozkoumala místo, kde viděla hráče naposled a následně ho hledala v blízkém okolí a k hlídkování by se vrátila za předpokladu, že by byla v hledání neúspěšná. K tomu bude potřeba zaznamenat stav ve kterém se umělá inteligence právě nachází. Do stromu chování proto přidáme další Selector do levé větve nad Patrol Path a rovněž přesuneme i Blackboard Decarator. Zde využijeme větev se sekvencí Find Random Location,

[25] jedná se speciální typ objektu, který na sebe mohou navázat Actoři. Komponenty jsou užitečné pro sdílení běžného chování, jako je například schopnost zobrazit vizuální reprezentaci nebo přehrávat zvuky. Mohou také představovat koncepty specifické pro projekt, například způsob, jakým vozidlo interpretuje vstup a mění svou vlastní rychlost a orientaci. (Epic Games 2015)
[26] rychlost

která poslouží jako základ. Otevřeme tedy task FindRandomLocation, kde nastavíme radius ve funkci GetRandomPointInNavigableRadius. Čím větší rádius, tím větší okolí bude umělá inteligence schopná prozkoumat. Nyní potřebujeme podmínku, aby AI věděla, kdy má hledat a kdy ne. K tomu dopomůže nově vytvořený blueprint typu enumeration[27], který se následně pojmenuje ChaseStatus. Ten otevřeme a doplníme tři stavy ve kterých se AI může nacházet, stav hlídky, stav hledání a stav pronásledování. Dále se vytvoří nový Blackboard klíč typu enum. Ve vlastnostech klíče je třeba přiřadit klíči blueprint ChaseStatus. Ve stromě chování je nutné vytvořit další sekvenci nad sekvencí FindRandomLocation, kde poté vytvoříme nový Blackboard Decorator, který propojíme s klíčem typu enum, viz. příloha A. V podmínce je dále nutné nastavit key value na pronásledování, a to z důvodu vyhodnocování jednotlivých větví, protože chceme, aby se hned po pronásledování dala umělá inteligence do pátrání po okolí. Sekvenci FindRandomLocation můžeme nyní vhodněji přejmenovat na Search.

V tasku ChasePlayer vytvoříme novou public proměnnou chase_status typu Blackboard klíč. Dále použijeme funkci get, abychom získali proměnnou a funkci SetBlackboardValueAsEnum kde nastavíme value na 2. Hodnotu 2 byla zvolena z důvodu umístění činnosti pronásledování v listu ChaseStatus, kde hodnotu nula má stav hlídky, hodnota 1 stav hledání a stav pronásledování 2. Tuto funkci v EventGraphu umístíme mezi SimpleMove a Finish Execute. Následně je nutné zkontrolovat, zda ve stromě chování je u tasku ChasePlayer skutečně nastavena Blackboard hodnota na ChaseStatus.

Vytvořený klíč je následně použit k vytvoření Decoratoru, který v Unreal Enginu zastupuje funkci podmínky. Je nutné nastavit podmínku na IsNotSet, tedy aby se podmínka splnila, pokud hodnota klíče bude nepravda. Tedy pokud NPC nevidí hráče, věnuje se tomu, co dělat má. Stejně tak se vytvoří druhý Decorator u Sekvence ChasePlayer, ovšem s podmínkou, že pokud klíč vrátí hodnotu 1 tj. hráč je viděn NPC, tak se spustí daná sekvence pronásledování.

Pro splnění stanovených požadavků je však třeba, aby NPC začalo hráčskou postavu pronásledovat. V nynějším stavu pouze dojde na místo, kde se hráč nachází, pokud je hráč viděn. Je nutné tedy vytvořit další task ChasePlayer. V EventGraphu tohoto tasku opět vytvoříme počáteční událost, tentokrát EventReceiveExecuteAI, která se spustí v momentě, kdy

[27] datový typ s omezenou množinou pojmenovaných hodnot

se task začne vyhodnocovat ve stromě rozhodování. Následně se použije funkce SimpleMoveToLocation, kde jako proměnou využijeme Blackboard klíč jako vektor.
Další Decorator, který je třeba použít je Loop. Ten přiřadíme k sekvenci Search. Loop umožní nastavit počet opakování dané sekvence. Tudíž v tomto případě Loop bude určovat kolikrát bude umělá inteligence pátrat po okolí, než vzdá hledání a vrátí se k hlídkování. Abychom zabránili zacyklení musí být vytvořena nová větev s novým taskem ChangeStatus, jehož cílem bude změnit hodnotu z hledání na hlídkování. Opět jako počáteční událost vytvoříme EventReceiveExecuteAI. Dále budeme potřebovat dvě proměnné. První z nich bude odkazovat na Blackboard klíč, a proto ji pojmenujeme BB_chase_status a pomocí funkce SetBlackboardValueAsEnum zajistíme, že výstup bude typu enum. Druhou proměnnou, kterou vytvoříme bude status a bude odkazovat na Chase Status typu enum. Tím docílíme toho, že stejný task budeme moci použít i v dalších větvích. Nesmíme dále zapomenout proměnou status napojit na již zmíněnou funkci SetBlackboardValueAsEnum, protože proměnná status bude ta proměnná, s níž bude funkce pracovat. U hotového tasku nastavíme BB Chase Status na ChaseStatus a status zvolíme hlídkování. Nyní se tedy vyhodnotí několikrát sekvence Search a to v závislosti na hodnotě zadané u Decoratoru Loop a následně se spustí task ChangeStatus, který změní status z hledání na hlídkování. Task ChaseStatus tedy můžeme použít i v sekvenci Chase Player, kde ale nesmíme zapomenout nastavit status na pronásledování.

Nyní umělá inteligence začne prozkoumávat blízké okolí, pokud se jí při pronásledování hráč ztratí z dohledu. Co ale potřebujeme je, aby umělá inteligence došla na místo, kde hráče viděla naposledy a až tam začala prohledávat okolí. Toho docílíme jednoduše tím, že před sekvencí Search vytvoříme jednu novou větev s taskem MoveTo, kde klíčem bude TargetLocation, jenž byl definován na začátku této kapitoly. Dále bude nutné vytvořit v sekvenci Search nový Blackboard dekorator, který bude mít za úkol přerušit sekvenci, pokud umělá inteligence během pátrání zahlédne hráče. V decoratoru je nutné jako Blackboard klíč nastavit CanSeePlayer, aby podmínka věděla, že má pracovat s proměnou, která má na starost to, jestli je hráč viděn či nikoliv a key query na IsNotSet. Dále potřebujeme nastavit, aby se sekvence přerušila, když bude podmínka splněna. To lze nastavit ve vlastnostech decoratoru v sekci Flow Control, kde se observer aborts nastaví na self.

5.2.5 Útok strážného

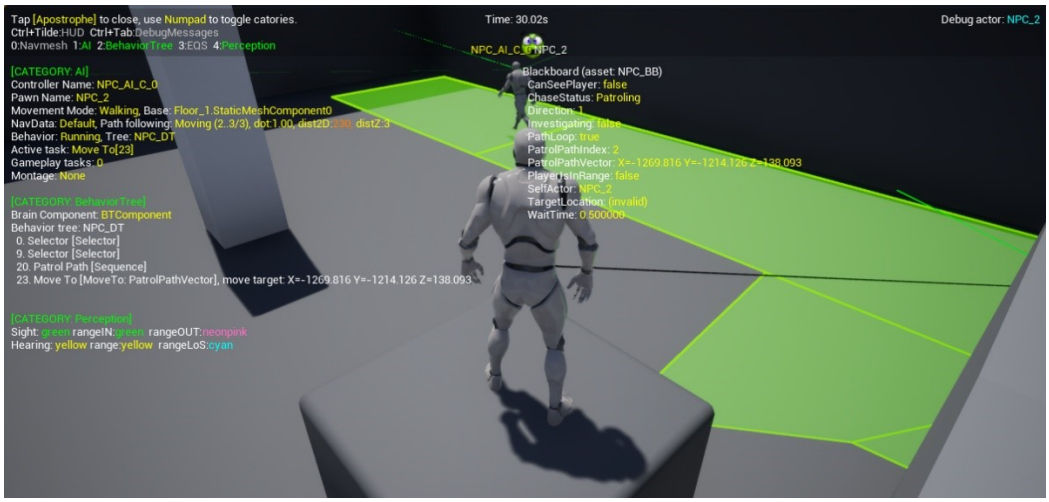
Umělá inteligence nyní funguje tak jak potřebujeme. Avšak, pro potřeby herního dema je vhodné implementovat i útok strážného na hráčovu postavu. Vytvoříme Blueprint Interface,

což je druh blueprintu, který můžeme implementovat do jiného blueprintu. A zde vytvoříme funkci Melee Attack. Vytvořený Blueprint interface, nazvaný Combat Interface, přidáme v Class Settings NPC. Zde také vytvoříme nový event pojmenovaný Melee Attack, který následně zapne animaci útoku.

Aby umělá inteligence věděla, kdy má zaútočit je potřeba jí nastavit vzdálenost od hráče, při které začne útočit. Začneme tím, že vytvoříme nový blackboard klíče typu boolean a pojmenujeme ho například PlayerInRange, který nám bude značit, jestli je hráč v dosahu či nikoliv. Dále vytvoříme novou službu, kterou pojmenujeme PlayerInRange, která nám bude kontrolovat dosah. Zde vytvoříme funkci, která bude porovnávat vzdálenost hráčovi postavy a postavy strážného. K tomu využijeme funkci Get Distance To, jejíž výstup bude porovnáván s nastavenou hodnotou vzdálenosti. Abychom mohli nastavit vzdálenost, musíme nejdříve vytvořit proměnnou Melee Range, která bude typu float a jejíž hodnotu definujeme. Službu následně přidáme ve stromu chování k sekvenci Chase Player. To znamená, že pokaždé, když se začne vyhodnocovat tato sekvence, tak se nejdřív zkontroluje, zda není hráč v dosahu. Následně ještě nastavíme klíč, kde zvolíme PlayerInRange.

Dále budeme potřebovat, aby strážný útok provedl. Toho docílíme vytvořením nového tasku Attack. Jako počáteční událost zvolíme Event Receive Execute AI. U controlled pawn zvolíme funkci Does Implement Interface, kde vybereme dříve vytvořený Combat Interface. Nyní nám to umožní přidat funkci Melee Attack. Vzhledem k tomu, že přehrání animace chvilku trvá, je nutné zde přidat funkci Delay, která nám umožní počkat námi stanovené množství času, než se vykoná další funkce. Délka prodlevy se může lišit v závislosti na délce animace.

Ve stromě chování vytvoříme něco, čemu se říká Simple Parallel, jenž nám umožní, aby běželi dva tasky naráz. To je nutné vzhledem k tomu, že chceme, aby se strážný hnal za hráčem a zároveň se ho snažil udeřit například svým obuškem. Simple Paralle běží ve dvou režimech. První z nich znamená, že pokud je první task dokončen, v našem případě task ChasePlayer, tak se druhý task nepustí. Druhý režim, který zvolíme je režim delayed, který umožní dokončit druhý task, pokud je první task splněn. U vložené tasku Attack, vytvoříme Blackboard decorator, který bude dohlížet na to, aby se útok spustil pouze ve chvíli, pokud bude hráč v dosahu. Jako klíč zvolíme PlayerInRange



Obrázek 14: Ukázka debbug modu
Zdroj: vlastní

5.2.6 Implementace sluchu

Posledním krokem je implementace sluchu. Cílem je, aby strážný mohl reagovat na podněty ve svém slyšitelném rádiusu. Ze všeho nejdříve je nutné v nastavení projektu přidat vstup pro akci, kterou nazveme například distract a následně namapujeme na libovolné tlačítko myši či klávesu. V blueprintu ThirdPersonCharacter následně vytvoříme nový event InputActionDistract a přidáme funkci PlaySounAtLocation, která se spustí při zmáčknutí klávesy. U funkce ještě nastavíme lokaci pomocá funkce GetActorLocation, což znamená, že zvuk bude pocházet z místa na kterém se nachází hráč. Aby zvuk mohl zaregistrovat strážný použijeme funkci ReportNoiseEvent. Lokace bude stejná jako předešlém případě. U funkce dále nastavíme Instigator[28] na self a u Tagu přidáme Noise.

Do teď umělá inteligence měla pouze jeden smysl a to zrak. To nyní změníme přidáním sluchu. V NPC_AI v kartě AI Percpetion přidáme další element, a to AI Hearing. Zde následně nastavíme, aby AI vnímalo zvuk ze všech zdrojů, to znamená enemies, neutrals a friendlies. Nyní se NPC ihned dozví přesnou lokaci hráče a začne ho pronásledovat, to však úplně nechceme, a proto bude nutné silně modifikovat event, který má toto na starosti, nejlépe celým smazáním. Následně vytvoříme nový spouštěcí event OnPerceptionUpdated(AIPerception). S novým eventem však chceme, aby na zvuk mohl reagovat více než jeden strážný, a proto zvolíme ForEachLoop, kde jako element pole zvolíme funkci GetActorsPerception na níž použijeme další smyčku viz. příloha pro lepší pochopení. Pro druhou smyčku jako element pole

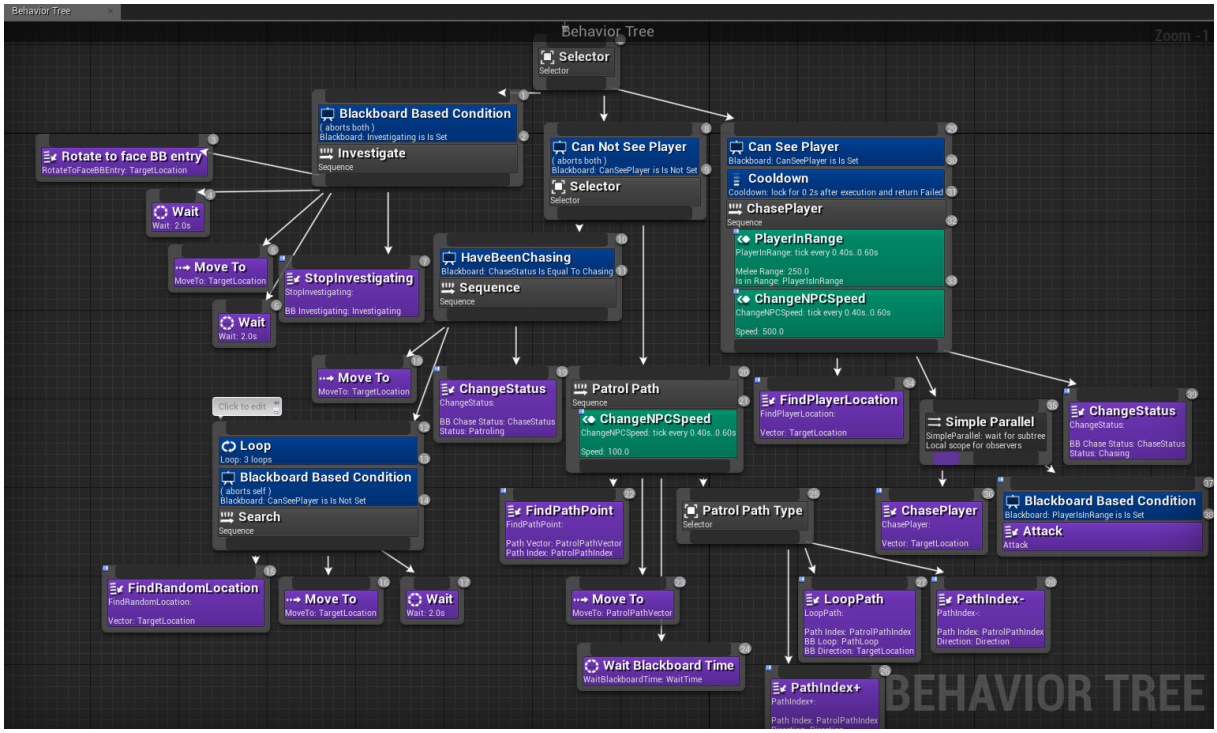
[28] podněcovatel, v tomto případě actor, který spustil hluk

zvolíme BreakAIStimulus. U položky Tag nastavíme podmínku, že pokud se tag name rovná noise, spustí se task, kde NPC půjde prozkoumat zvuk. Tedy pokud umělá inteligence uslyší podezřelý zvuk, vydá se ho prošetřit. K tomu však budeme muset vytvořit zcela novou větev ve stromě rozhodování.

Vytvoříme tedy novou sekvenci a spolu s ní i nový blackboard klíč typu boolean, který pojmenujeme Investigating, jenž přiřadíme k novému decaratoru u dané sekvence a u něho nastavíme observer aborts na both. V sekvenci následně připojíme task Rotate to face BB entry, který je defaultně k dispozici. A následně chceme, aby po natočení se směrem ke zdroji zvuku chvíli počkal, respektive abychom co nejvíce napodobili chování člověka, který chvíli přemýšlí, jestli se mu to nezdálo atd., takže zvolíme task Wait. Následně MoveTo, aby se vydal na inkriminované místo a opět použijeme i Wait. Následně budeme muset vytvořit nový task, který bude mít na starost ukončení prošetřování místa. Task pojmenujeme StopInvestigating. Vytvoříme tedy počáteční event spolu s novou blackboard proměnou. Pomocí funkce Get získáme hodnotu proměnné a tu dále pomocí funkce SetBlackboardValueAsBool nastavíme jako boolean viz. task StopInvestigating v příloze A. Task následně přidáme do stromu rozhodování, kde k tasku přiřadíme klíč Investigating.

Nyní se můžeme vrátit do blueprintu NPC_AI, kde pomocí funkce GetBlackboard získáme klíč, který pomocí funkce SetValueAsBool nastavíme na typ boolean. To proto, že mohou nastat pouze dva stavy. Vyšetřuje či nevyšetřuje. U této funkce následně key name nastavíme na Investigating. Tato funkce musí být napojená na podmínku, kterou jsme již dříve stanovili viz. příloha A. Dále tedy potřebujeme, aby se NPC vydalo na místo odkud vycházel zvuk. Toho docílíme opět přes blackboard klíč, který získáme pomocí funkce Get. Klíč následně nastavíme na vektor pomocí funkce SetValueAsVector. Zde key name nastavíme na TargetLocation a hodnota vektoru bude pocházet z funkce BreakAIStimulus, kterou napojíme na StimulusLocation. Vše stačí následně zkompilovat.

Finální strom chování, podle kterého se bude řídit umělá inteligence lze vidět na obrázku 15.



Obrázek 15: Strom chování
Zdroj: vlastní

6 Zhodnocení

Na základě vytyčených požadavků jsem úspěšně implementoval zcela funkční strom chování pro umělou inteligenci ve 3D hře. Chování „strážného“ tak obsahuje několik definovaných činností od hlídkovaní, pronásledování hráče nebo průzkum okolí, pokud umělá inteligence zaregistruje něco podezřelého. S tím souvisí i úspěšná implementace vjemů jako je zrak či sluch. NPC ovládané umělou inteligencí tak má k dispozici zrak a pomocí zorného pole, které bylo parametrově nastaveno, aby se co nejblíže přiblížilo člověku, může identifikovat hrozby či podezřelé činnosti. Mimo jiné také disponuje sluchem. Může tak reagovat na zvukové podněty v daném rádiusu. Například dokáže slyšet dopadnuti hráče z větší výšky nebo také zapískání či jakýkoliv jiný hluk způsobený hráčem. Při tvorbě jsem bral v potaz modifikovatelnost tohoto stromu pro více účelů. Kupříkladu řešení přes blackboard proměnné mi umožnilo upravovat důležité prvky přímo ve stromě chování, aniž bych musel pokaždé zasahovat do samotného kódu. Strom chování tak lze použít s lehkými úpravami i pro další herní žánry a neslouží tedy pouze účelově k jednomu demu. Mimo jiné takto navržený strom chování usnadní práci pro tvorbu dalších úrovní a dá se snadno rozšiřovat o další prvky. Například lze snadno nastavit různé rysy jednotlivým strážným, kde kupříkladu strážný s nadváhou běhá pomaleji nebo rozespalý strážný, který reaguje na věci ve svém okolí o poznání méně. Osobně zde vidím i další prostor pro zlepšení, které umožní větší uvěřitelnost chování AI. Například implementace komunikace mezi jednotlivými strážnými by dozajista přidala na věrohodnosti jejich chování.

Volba enginu byla pro tento typ projektu zcela adekvátní. Unreal Engine 4 mi nabídl vše co jsem potřeboval. Nepřeberné množství dokumentace, skvělý a intuitivní vizuální skriptovací systém, který je za mě opravdu jednou z nejlepších věcí, co tento engine může nabídnout a přívětivé uživatelské prostředí, dělají z Unreal Engine 4 jednoznačně výbornou volbu. Ovšem engine samotný taky není zcela bez problémů. Během několika desítek hodin práce jsem se musel potýkat s pády aplikace. Naštěstí, zde funguje automatické ukládání, které je poměrně časté.

Ohledně konkurenčních enginů, například CryEngine V nabízí velmi pokročilé možnosti pro tvorbu umělé inteligence, ne-li mnohem pokročilejší než samotný Unreal Engine. Ovšem malé množství potřebné dokumentace, která je většinou skryta za placenou podporou jednoznačně odradí spoustu nováčků. Což je očividný problém celého CryEnginu. Jedná se o velmi

komplexní nástroj, který toho ve výsledku umí více než konkurence, ale dojíždí právě na již zmíněné problémy s nedostatkem dokumentace a další důvody, kterými se zabývám podrobněji v kapitole tomu zaměřené.

Co se týče Unity, tak engine samotným je mnohem více přístupný, než CryEngine nebo Unreal. Nabízí nepřeberné množství různých tutoriálů od vývojářů nebo komunity. Dokonce začínají experimentovat s vizuálním skriptováním po vzoru Epic Games. Nicméně Unity toho moc nenabízí po stránce umělé inteligence, což při mé volbě bylo zcela zásadní. Unity je stále engine pro menší a experimentálnější hry, ačkoliv se v posledních letech snaží dohánět konkurenci a být alternativou, ještě stále v mnoha ohledech zaostávají.

Abych to shrnul, tak nejlepším enginem pro tento typ projektu se jednoznačně jeví Unreal Engine 4 od Epic Games. Engine samotný totiž není stavěn jenom pouze pro tvorbu her a taky tak přistupuje k jistým věcem, které jsou díky tomu snadněji přístupnější. Například vizuální skriptovací systém umožňuje tvorbu komplexních projektů, aniž by člověk musel sáhnout na zdrojový kód, což samo o sobě usnadňuje samotnou práci.

7 Závěr

Práce se první řade věnuje stručné historii videoher a vývoji umělé inteligence ve hrách napříč několika desetiletími. Zabývá se počátky herního průmyslu a vznikem prvních her. Nástupem herních automatů, prvních konzolí a domácích počítačů. Dále příchodem prvních 3D her, prvních botů ovládaných umělou inteligencí, neuronovými sítěmi a procedurálním generováním v oblasti videoher.

V další kapitole se práce věnuje nejčastěji používaným metodám pro tvorbu umělé inteligence ve videohrách. Jmenovitě tak jde například o metody Finite State Machine, která se používá především v žánru FPS, Monte Carlo Search Tree, jenž se používá u strategických her. Mimo jiné se zabývá i stromy chování a neuronovými sítěmi.

V další části se práce věnuje tomu, co jsou herní enginy a k čemu slouží. Jsou zde rozebírány jednotlivé části enginu a také krátká historie. Poté se práce zaměřuje na dostupné herní enginy na trhu. Tato kapitola se zpočátku zabývá Unreal Enginem od společnosti Epic Game, následuje část o CryEngine od německých vývojářů z Crytek a v neposlední řádě se věnuji enginu Unity od Unity Technologies. U každého enginu krátce rozebírám jejich vývoj a následně se podrobněji zabývám posledními verzemi těchto enginům jenž jsem podrobil detailní analýze.

V následující části práce se zabývám návrhem a následnou aplikací umělé inteligence pro 3D hru. V prvé řadě jsem stanovil požadavky a poté jsem dle analýz vybral vhodný engine pro tvorbu tohoto projektu. Následně se zabývám návrhem a implementací stromu chování pro umělou inteligence ve 3D hře jenž se skládá z několika kroků. V prvním z nich se věnuji tvorbě základní struktury stromu chování a několika základním činnostem, které by měla umělá inteligence vykonávat. Dalším krokem je implementace zraku, kde podrobně popisují způsob, jakým jsem tento problém řešil. Posléze se zaměřuji na řešení hlídkovací trasy pro NPC ve 3D virtuálním prostředí. V následujícím kroku se zabývám dalšími činnostmi jako je hledání hráče a průzkum okolí nebo útok strážného. V neposlední řadě se tak věnuji implementaci sluchu, díky čemuž může umělá inteligence reagovat na zvukové podněty ve svém okolí.

Hranice umělé inteligence ve videohrách je stále kam posouvat. Ať už je to inteligence nepřátelských bojovníků, doprovodných společníků řízených umělou inteligencí, s nimiž může hráč interagovat až po protivníky řízenými neuronovou sítí, kteří by měli znamenat pro hráče

skutečnou výzvu. Není to však jenom o tom, jak umělá inteligence dokáže reagovat na akce hráče, ale v mnoha případech jde vývojářům o vytvoření živoucího světa. Hry jako Last of Us nebo Red Dead Redemption 2 jsou založeny především na budování pouta mezi hráčem a postavami ovládané umělou inteligencí. Dospěli jsme do doby, kdy je toto všechno v počítačových hrách možné a naprosto uvěřitelné. Samozřejmě, že se z velké části stále jedná spíše o iluzi a umělá inteligence ve hrách je v mnoha případech jen soubor scénářů pro každou možnou situaci. Avšak s nástupem neuronových sítí a samoučících se algoritmů je jen otázka času, než přijde ta skutečná umělá inteligence.

Seznam použité literatury

3DMASTER, 2019. Guide for beginners: What is a Heightmap? *VisCircle 3D configurator* [online]. [vid. 2020-04-29]. Dostupné z: https://viscircle.de/guide-for-beginners-what-is-a-heightmap/?lang=en

ALAYÓN, David, 2018. Video Games, AI, and Procedurally Generated Content. *Medium* [online] [vid. 2020-04-09]. Dostupné z: https://medium.com/future-today/video-games-ai-and-procedurally-generated-content-cb8b45299513

AUSTRALIAN GOVERNMENT - DEPARTMENT OF DEFENCE, 2011. *Avatars train on Navy's future ship* [online]. Dostupné z: http://www.defence.gov.au/minister/Claretpl.cfm?CurrentId=11969

BARTÁK, Roman. 2017. *Co je nového v umělé inteligenci*. Praha: Nová beseda. Co je nového. ISBN 978-80-906751-2-4.

BERARDINI, César, 2006. *Ubisoft Acquires Far Cry Intellectual Property, Perpetual License of CryEngine - Xbox* [online] [vid. 2020-04-20]. Dostupné z: https://web.archive.org/web/20090107142531/http://news.teamxbox.com/xbox/10596/Ubisoft-Acquires-Far-Cry-Intellectual-Property-Perpetual-License-of-CryEngine/

BIGAS, Jiří, 2018. *Nejdražší hry všech dob. Kolik z nich jste hráli? » Vortex* [online] [vid. 2020-04-09]. Dostupné z: https://www.vortex.cz/nejdrazsi-hry-vsech-dob-kolik-z-nich-jste-hrali/

BINGGESER, Peter, 2020. Designing AI: Solving Snake with Evolution. *Medium* [online] [vid. 2020-04-19]. Dostupné z: https://becominghuman.ai/designing-ai-solving-snake-with-evolution-f3dd6a9da867

BMI GAMING, nedatováno. *The History Of Video Arcade Games : Visual History Of Video Games and the Story of Video Arcade Games Were Invented | From BMI Gaming* [online] [vid. 2020-04-20]. Dostupné z: https://www.bmigaming.com/videogamehistory.htm

CRYTEK, 2004. *CryENGINE® 1 | Crytek Video Game Developer Company* [online] [vid. 2020-04-20]. Dostupné z: https://web.archive.org/web/20110417065611/http:/crytek.com/cryengine/cryengine1/overview

CRYTEK, 2007. *CryENGINE® 2 | Crytek Video Game Developer Company* [online] [vid. 2020-04-20]. Dostupné z: https://web.archive.org/web/20110519001209/http:/crytek.com/cryengine/cryengine2/overview

CRYTEK, 2009. *Crytek GmbH: Crytek Announces CryENGINE® 3* [online] [vid. 2020-04-20]. Dostupné z: https://web.archive.org/web/20090323125304/http://www.crytek.com/news/news/?tx_ttnews%5Btt_news%5D=143

CRYTEK, 2011. *CryEngine 3 Overview | Crytek* [online] [vid. 2020-04-20]. Dostupné z: https://web.archive.org/web/20110508001441/http:/crytek.com/cryengine/cryengine3/overview

CRYTEK, 2020a. CRYENGINE | Features. *CRYENGINE* [online] [vid. 2020-04-20]. Dostupné z: https://www.cryengine.com/features

CRYTEK, 2020b. *Crytek - video game developer, makers of CRYENGINE* [online] [vid. 2020-04-26]. Dostupné z: https://www.crytek.com

DILLON, Roberto. 2011. *The golden age of video games: the birth of a multi-billion dollar industry*. Boca Raton, FL: A K Peters/CRC Press. ISBN 978-143-9873-236.

DONOVAN, Tristan, 2010. *Replay: the history of video games*. East Sussex, England: Yellow Ant. ISBN 978-0-9565072-0-4.

EPIC GAMES, 2015. *Components* [online] [vid. 2020-04-29]. Dostupné z: https://docs.unrealengine.com/en-US/Engine/Components/index.html

FENG, Jie a Peter L. NEWTORN. 2016. *Unreal Engine 4 AI Programming Essentials: Jie Feng, Peter L. Newton*. Birmingham, Velká Británie: Packt Publishing Limited. ISBN 9781784393120.

FISHER, Tyler, 2019. Borderlands 3 Doesn't Actually Have 1 Billion Guns, It Has Way More. *GAMING* [online] [vid. 2020-04-09]. Dostupné z: https://comicbook.com/gaming/news/borderlands-3-ps4-xbox-pc-billion-guns/

GAMESPOT, 2001. *Half Life: Interview With Gabe Newell* [online] [vid. 2020-04-19]. Dostupné z: https://web.archive.org/web/20010723160349/http://extra.gamespot.co.uk/pc.gamespot/features/halflife_uk/02.html

GIANT BOMB, nedatováno. CryEngine 3 (Concept). *Giant Bomb* [online] [vid. 2020-04-20]. Dostupné z: https://www.giantbomb.com/cryengine-3/3015-2917/

GLANCEY, Paul, 1996. *The Complete History of Computer and Video Games*. B.m.: EMAP Images.

GORAL, Cindy M, Kenneth E TORRANCE, Donald P GREENBERG a Bennett BATTAILE, 1984. Computer Graphics Volume18, Number3 July 1984. 10.

GREGORY, Jason, 2009. *Game engine architecture*. Wellesley, Mass: A K Peters. ISBN 978-1-56881-413-1.

GROVE, Ricky, 2019. *Focus: Unreal Engine - A Brief History of Unreal* [online] [vid. 2020-04-19]. Dostupné z: https://magazine.renderosity.com/article/5330/focus-unreal-engine-a-brief-history-of-unreal

HAAS, John, 2014. *A History of the Unity Game Engine* [online]. Worcester, Massachusetts. Worcester Polytechnic Institute. Dostupné z: https://web.wpi.edu/Pubs/E-project/Available/E-project-030614-143124/unrestricted/Haas_IQP_Final.pdf

HORTI, Samuel, 2017. Why F.E.A.R.'s AI is still the best in first-person shooters. *Rock, Paper, Shotgun* [online]. [vid. 2020-04-19]. Dostupné z: https://www.rockpapershotgun.com/2017/04/03/why-fears-ai-is-still-the-best-in-first-person-shooters/

HORVATH, Michael, 2020. *mjhorvath/Mike-Wikipedia-Illustrations* [online]. POV-Ray SDL [vid. 2020-04-28]. Dostupné z: https://github.com/mjhorvath/Mike-Wikipedia-Illustrations

IGN, 2000. *http://pc.ign.com: The Year's Best-Selling Games* [online] [vid. 2020-04-19]. Dostupné z: https://web.archive.org/web/20000613095857/http://pc.ign.com/news/6602.html

INVENTIONS, Mary Bellis Inventions Expert Mary Bellis covered, inventors for ThoughtCo for 18 years She is known for her independent FILMS, DOCUMENTARIES, including one about ALEX a er Graham Bell our editorial process Mary BELLIS, 2019. Spacewar: The First Computer Game. *ThoughtCo* [online] [vid. 2020-04-03]. Dostupné z: https://www.thoughtco.com/history-of-spacewar-1992412

LEE, Joanna, 2016. *The history of Unreal Engine - Learning Unreal Engine Game Development* [online] [vid. 2020-04-19]. ISBN 978-1-78439-815-6. Dostupné z: https://subscription.packtpub.com/book/game_development/9781784398156/1/ch01lvl1sec10/the-history-of-unreal-engine

LOU, Harbing, 2017. AI in Video Games: Toward a More Intelligent Game. *Science in the News* [online]. [vid. 2020-04-03]. Dostupné z: http://sitn.hms.harvard.edu/flash/2017/ai-video-games-toward-intelligent-game/

LOWOOD, Henry, 2014. *Game Engines and Game History | Kinephanos* [online]. [vid. 2020-04-19]. Dostupné z: https://www.kinephanos.ca/2014/game-engines-and-game-history/

MAASS, Laura E. Shummon, 2019. Artificial Intelligence in Video Games. *Medium* [online] [vid. 2020-04-19]. Dostupné z: https://towardsdatascience.com/artificial-intelligence-in-video-games-3e2566d59c22

MAKUCH, Eddie, 2013. New CryEngine revealed. *GameSpot* [online]. [vid. 2020-04-20]. Dostupné z: https://www.gamespot.com/articles/new-cryengine-revealed/1100-6413371/

MICROSOFT, nedatováno. *Co je middleware – definice a příklady | Microsoft Azure* [online] [vid. 2020-04-19]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-middleware/

ORLAND, Kyle, 2014. Unreal Engine 4 now available as $19/month subscription with 5% royalty. *Ars Technica* [online] [vid. 2020-04-28]. Dostupné z: https://arstechnica.com/gaming/2014/03/unreal-engine-4-now-available-as-19month-subscription-with-5-royalty/

ORLOVSKIYI, Sergej, 2017. *Neural AI* [online] [vid. 2020-04-02]. Dostupné z: http://en.blitzkrieg.com/news/aiboris

OXFORD DICTIONARY, 2020a. *artificial-intelligence noun - Definition, pictures, pronunciation and usage notes | Oxford Advanced Learner's Dictionary at OxfordLearnersDictionaries.com* [online] [vid. 2020-04-30]. Dostupné z: https://www.oxfordlearnersdictionaries.com/definition/english/artificial-intelligence?q=artificial+intelligence

OXFORD DICTIONARY, 2020b. *video-game noun - Definition, pictures, pronunciation and usage notes | Oxford Advanced American Dictionary at OxfordLearnersDictionaries.com*

[online] [vid. 2020-04-23]. Dostupné
z: https://www.oxfordlearnersdictionaries.com/definition/american_english/video-game

PÁNEK, Jan, 2015a. Unreal Engine a postupný vývoj - Unreal Engine 2. *Svethardware.cz* [online] [vid. 2020-04-19]. Dostupné z: https://www.svethardware.cz/unreal-engine-a-postupny-vyvoj/41366-3

PÁNEK, Jan, 2015b. Unreal Engine a postupný vývoj - Unreal Engine 3. *Svethardware.cz* [online] [vid. 2020-04-20]. Dostupné z: https://www.svethardware.cz/unreal-engine-a-postupny-vyvoj/41366-4

PAUL, Partha Sarathi, Surajit GOON a Abhishek BHATTACHARYA, 2012. HISTORY AND COMPARATIVE STUDY OF MODERN GAME ENGINES. 6.

PECKHAM, Eric, 2019. How Unity built the world's most popular game engine. *TechCrunch* [online]. [vid. 2020-04-26]. Dostupné z: https://social.techcrunch.com/2019/10/17/how-unity-built-the-worlds-most-popular-game-engine/

PLANTE, Chris, 2012. Better with age: A history of Epic Games. *Polygon* [online] [vid. 2020-04-19]. Dostupné z: https://www.polygon.com/2012/10/1/3438196/better-with-age-a-history-of-epic-games

SIMPSON, Chris, 2014. *Behavior trees for AI: How they work* [online] [vid. 2020-04-19]. Dostupné
z: https://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php

STANTON, Richard, 2015. *A Brief History of Videogames*. Philadelphia, PA: Running Press Book Publishers, a member of the Perseus Books Group. ISBN 978-0-7624-5615-4.

SUTORCEN, 2016. This is Unreal - The story of Unreal Engine and the games it powers. *Hotgates* [online]. [vid. 2020-04-19]. Dostupné z: https://hotgates.eu/this-is-unreal/

SWEENEY, Tim, 2015. *If You Love Something, Set It Free* [online] [vid. 2020-04-28]. Dostupné z: https://www.unrealengine.com/en-US/blog/ue4-is-free

ŠVÁRA, Ondřej, 2013. *Videohry - historie virtuální zábavy*. ISBN 978-80-87749-06-7.

TIŠNOVSKÝ, Pavel, 2014. *Prehistorie počítačových her (1958–1988)*. B.m.: Internet Info. ISBN 999-00-015-0154-9.

UNITY TECHNOLOGIES, 2020. *Powerful 2D, 3D, VR, & AR software for cross-platform development of games and mobile apps.* [online] [vid. 2020-04-20]. Dostupné
z: https://store.unity.com/

UNREAL TECHNOLOGY, 2008. *Unreal Technology* [online] [vid. 2020-04-20]. Dostupné
z: https://web.archive.org/web/20081217133451/http:/www.unrealtechnology.com/features.php?ref=technology-overview

VAN WAVEREN, J.M.P, 2001. Quake III Arena Bot. In: [online]. Delft University of Technology. Dostupné
z: http://mrelusive.com/publications/presentations/2001_thesis/Q3ABotAI_presentation.pdf

WARD, Jeff, 2008. *What is a Game Engine?- GameCareerGuide.com* [online] [vid. 2020-04-19]. Dostupné z: https://www.gamecareerguide.com/features/529/what_is_a_game_.php

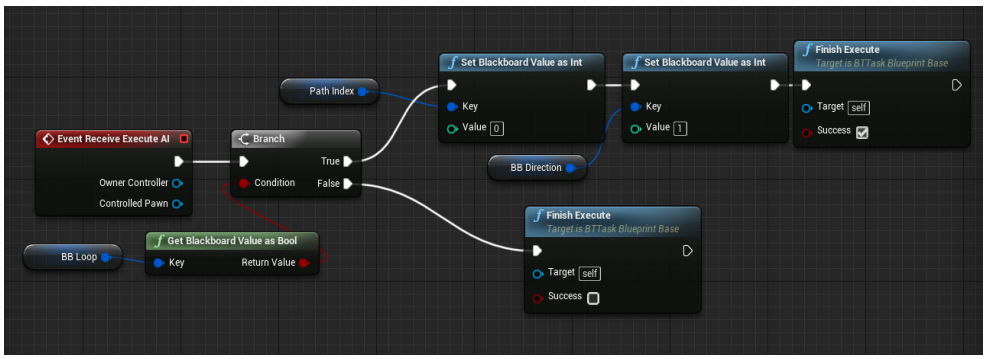
WEXLER, James, 2002. *Artificial Intelligence in Games* [online]. Rochester, NY 14627. University of Rochester. Dostupné z: https://www.cs.rochester.edu/~brown/242/assts/termprojs/games.pdf

WILLAERT, Kate, 2019. The Sumerian Game: The Most Important Video Game You've Never Heard Of. *A Critical Hit!* [online]. [vid. 2020-04-20]. Dostupné z: https://www.acriticalhit.com/sumerian-game-most-important-video-game-youve-never-heard/

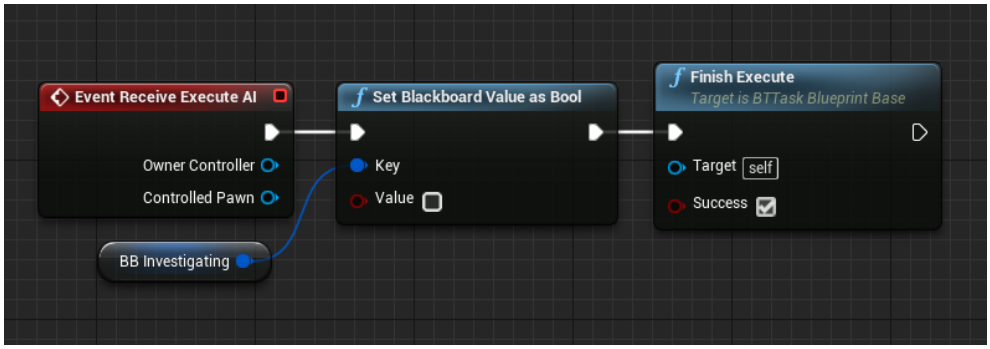
XU, Siyuan, 2008. *History of AI design in video games and its development in RTS games - Final Step* [online] [vid. 2020-04-27]. Dostupné z: https://sites.google.com/site/myangelcafe/articles/history_ai

YERLI, Faruk, 2002. *Crytek GmbH: Crytek announces its Game Engine CryENGINE* [online] [vid. 2020-04-20]. Dostupné z: https://web.archive.org/web/20081115062536/http://www.crytek.com/news/news/?tx_ttnews%5Bpointer%5D=17&tx_ttnews%5Btt_news%5D=76&tx_ttnews%5BbackPid%5D=9&cHash=e4dea47a80

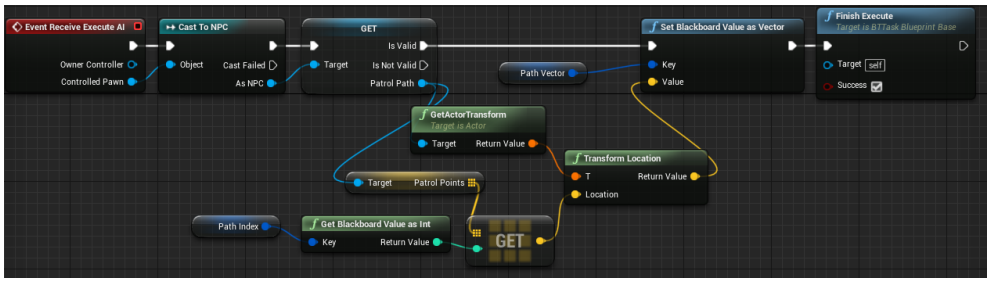
Příloha A



Task – LoopPath

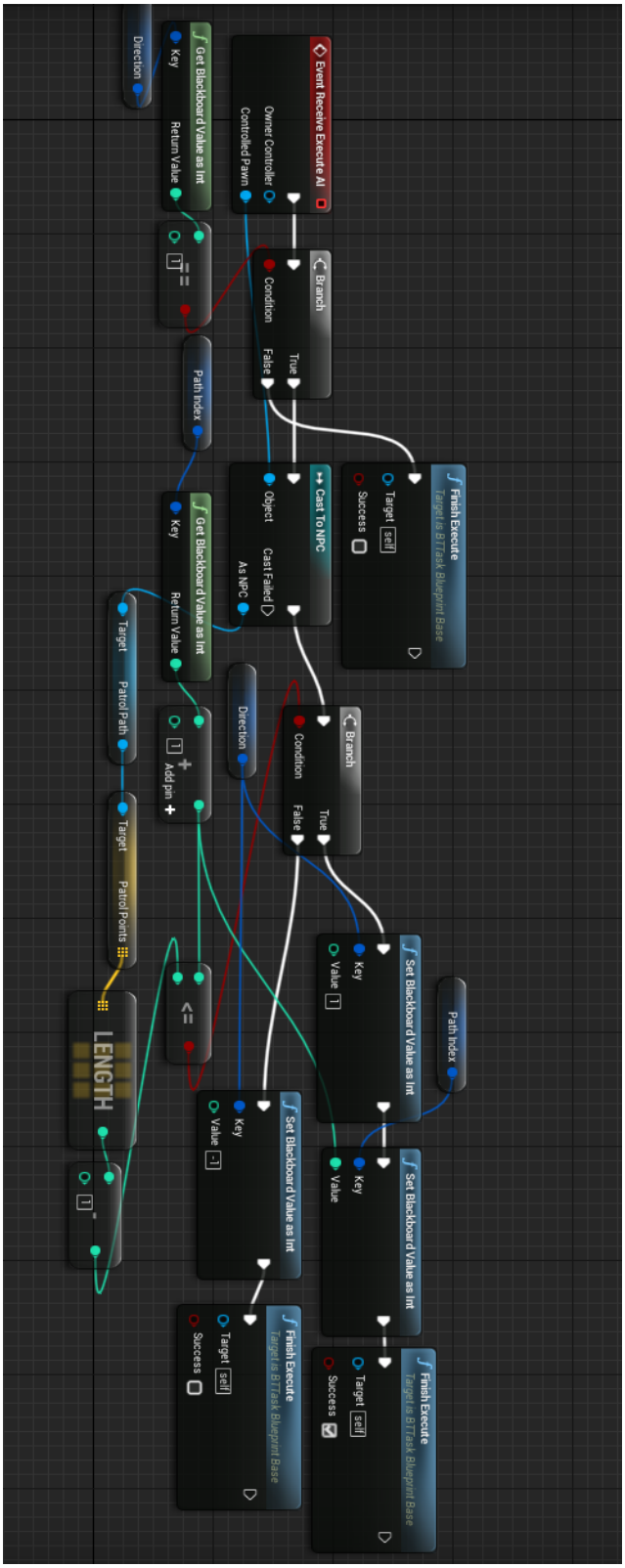


Task – StopInvestigating



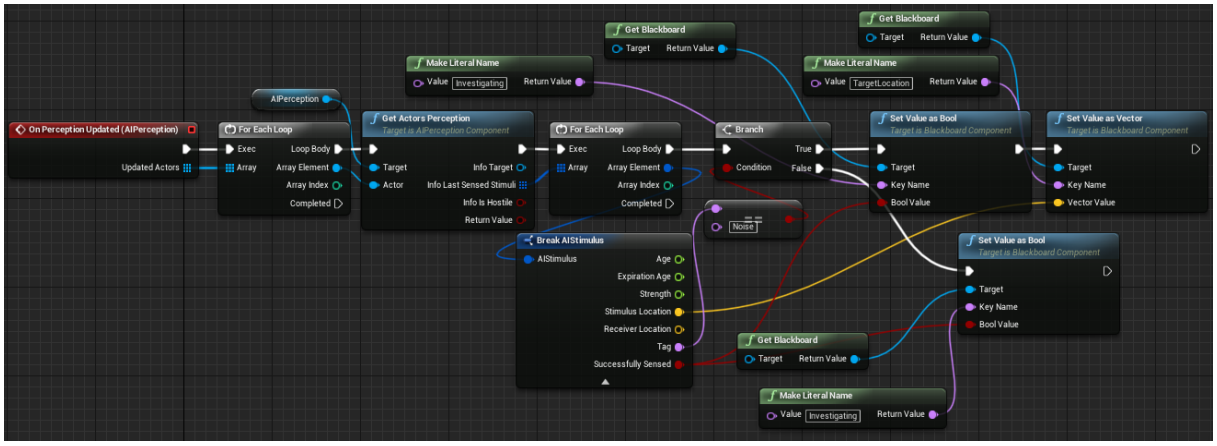
Task – FindPathPoint

Příloha B

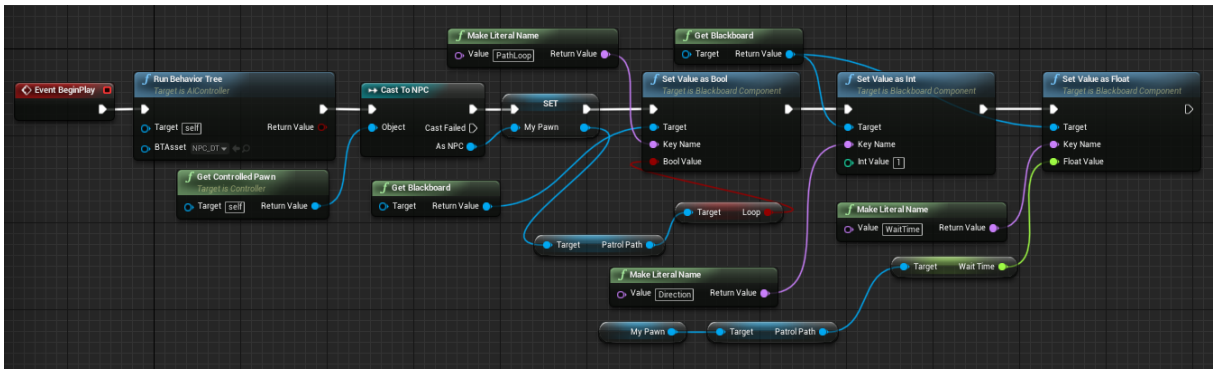


Task – PatrolPathIndex+

Příloha C



NPC_AI 1/2



NPC_AI 2/2