VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HERNÍ MECHANIKY PALNÝCH ZBRANÍ A INTERAKCE S PROSTŘEDÍM

FIREARMS AND ENVIRONMENT INTERACTION GAME MECHANICS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ZBYNĚK LAMAČKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ CHLUBNA**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

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

Zadání bakalářské práce

Student: **Lamačka Zbyněk**
Program: Informační technologie
Název: **Herní mechaniky palných zbraní a interakce s prostředím**
Firearms and Environment Interaction Game Mechanics
Kategorie: Počítačová grafika

Zadání:

1. Seznamte se s vývojovým prostředím herních enginů (Unity, Unreal Engine...).
2. Vyberte a nastudujte netriviální a inovativní herní mechaniky zaměřené na použití palných zbraní a na interakci s prostředím herního světa.
3. Navrhněte aplikaci demonstrující vybrané mechaniky.
4. Demo aplikaci implementujte.
5. Proveďte měření, případně uživatelskou studii hodnotící implementované výsledky.
6. Vytvořte video reprezentující výsledky vaší práce.

Literatura:

- Gregory, Jason. *Game engine architecture*. crc Press, 2018. ISBN 1351974289, 9781351974288
- Bishop, Lars, et al. "Designing a PC game engine." IEEE Computer Graphics and Applications 18.1 (1998): 46-53.
- Adams, Ernest, and Joris Dormans. Game mechanics: advanced game design. New Riders, 2012. ISBN 0321820274, 9780321820273

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

Vedoucí práce: **Chlubna Tomáš, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Tato bakalářská práce se zabývá herními mechanikami palných zbraní a interakcí s prostředím. Cílem této práce je tvorba herního dema implementujícího tyto mechaniky v prostředí Unreal Engine 4. Práce popisuje teoretické základy palných zbraní i způsob řešení implementačních problému při vývoji videoher. Dále se práce zaměřuje na implementaci herních mechanik jako je například zpětný ráz, modifikace zbraní a různý záblesk při střelbě na základě konfigurace zbraně.

Abstract

This bachelor thesis deals with the game mechanics of firearms and interaction with the environment. The aim of this thesis is to create a game demo implementing these mechanics in the Unreal Engine 4. The thesis describes the theoretical foundations of firearms as well as the way of solving implementation problems in video game development. Furthermore, the thesis focuses on the implementation of game mechanics such as recoil, weapon modification, and different muzzle flash based on weapon configuration.

Klíčová slova

Unreal Engine, UE4, herní engine, herní demo, hra, palná zbraň, Blueprint

Keywords

Unreal Engine, UE4, game engine, game demo, game, firearms, Blueprint

Citace

LAMAČKA, Zbyněk. *Herní mechaniky palných zbraní a interakce s prostředím.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Chlubna

Herní mechaniky palných zbraní a interakce s prostředím

Prohlášení

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

.....
Zbyněk Lamačka
9. května 2022

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Tomáši Chlubnovi za jeho rady a za čas, který mi věnoval.

Obsah

Kapitola 1

Úvod

Od vytvoření prvních videoher v polovině minulého století, kdy hry byly omezené hardwarem a také faktem, že počítač byl považován za nástroj pro práci a ne zábavu, se změnilo hodně. V roce 2019 měl trh s videohrami hodnotu přibližně 150 miliard dolarů. V roce 2025 by se tato hodnota měla zvednout na 250 miliard a s průběhem času by měla růst i nadále [5].

V dnešní době existuje mnoho žánrů videoher a stále vznikají nové, příkladem nového a úspěšného žánru je Battle royale. Ke zpopularizování tohoto žánru došlo v roce 2017 pomocí titulů PUBG a Fortnite. Na počítačích a herních konzolích patří střílečky mezi nejoblíbenější žánr videoher [3].

Cílem této práce je návrh a implementace herního dema, které se zaměřuje na mechaniky palných zbraní a interakci s okolím. Mezi implementované mechaniky patří modifikace zbraní pomocí doplňků jako je tlumič, ručka nebo taktická svítilna a laser. Další mechaniky jsou změna tvaru záblesku při výstřelu v závislosti na doplňcích a různý zpětný ráz podle nasazených doplňků. Mezi interakci s prostředím patří vyhazování nábojnic při střelbě a odhazování zásobníku při nabíjení zbraně a zanechávání stop po střelách na zasažených objektech. Práce se snaží dosáhnout nejvyšší možné míry realističnosti při zachování zábavnosti pro běžné hráče.

Kapitola 2 se zaměřuje na vysvětlení herního enginu a popisu metod využívaných při vývoji videoher. Kapitola dále popisuje palné zbraně v reálnem světě. Kapitola 3 se věnuje popisu návrhu herních mechanik a zdůvodnění návrhových rozhodnutí. Kapitola dále obsahuje popis zbrani a doplňků, které jsou v práci implementovány. Kapitola 4 se zabývá popisem některých částí implementace práce. Kapitola obsahuje i zjednodušené kódy některých implementovaných mechanik. V kapitole 5 je vysvětlen způsob testování herního dema a výsledky, kterých bylo dosaženo. Kapitola 6 obsahuje shrnutí dosažených výsledků.

Kapitola 2

Teorie

Tato kapitola slouží k vysvětlení a definici pojmů, které se v textu této práce vyskytují. V sekci 2.1 je vysvětlem pojem herní engine. Dále jsou v této sekci uvedeny některé v současnosti používané herní enginy. Sekce 2.2 slouží k popisu palných zbraním v reálném světě. V sekci 2.3 jsou popsány některé obecně používané metody při vývoji videoher.

2.1 Herní engine

Herní engine je softwarový framework primárně určen k vývoji videoher. Může být použit i k tvorbě filmů jako například seriál The Mandalorian a film Ford v Ferrari. Engine umožňuje rozdělit tvorbu videohry na vývoj enginu a na tvorbu samotné hry, což vede ke znovupoužitelnosti, která má za následek snížení nákladů. Analogie by mohla být linka na výrobu bot. Po vytvoření jedné výrobní linky může být linka využívaná pro výrobu více druhů bot.

Herní engine obsahuje části řešící například vykreslování, umělou inteligenci, skriptování, správu paměti, paralelizaci, fyzikální engine, zvukový engine a komunikaci přes síť.

Hranice mezi enginem a hrou není jasně definována. Engine by měl být všestranný a výkonný, což jsou vlastnosti, které jdou proti sobě. Při optimalizacích, které mají za úkol zvýšení výkonu dochází ke snížení všestrannosti a naopak [6, str. 12-13].

2.1.1 CryEngine

CryEngine je multiplatformní herní engine vyvinutý společností Crytek. Engine byl původně vytvořen pro technologické demo X-Isle: Dinosaur Island, ze kterého následně vznikla First-person shooter (FPS) videohra Far Cry. CryEngine byl využíván hlavně pro svou hezkou grafiku [6, str. 28-29]. Mezi nejvýznamnější tituly vytvořené na Cryenginu patří série Crysis a první díl Far Cry. Vývoj Star Citizen začal v CryEngine 3, ale po soudních sporech byl vývoj titulu přesunut na Lumberyard. V roce 2006 Ubisoft koupil práva na celou sérii Far Cry včetně licence na CryEngine, na základě které vznikl Dunia Engine, který Ubisoft využívá pro sérii Far Cry dodnes.

2.1.2 Open 3D Engine (Lumberyard)

Lumberyard je multiplatformní source-available engine vyvíjený společností Amazon na základě CryEngine. První verze byla zveřejněna v roce 2016. Videohry využívající tento engine: New World a Star Citizen. V roce 2021 oznámil Amazon spolupráci s Linux Foun-

dation na vytvoření nového open-source enginu. Výsledkem této spolupráce je Open 3D Engine (O3DE) [12].

2.1.3 Unity

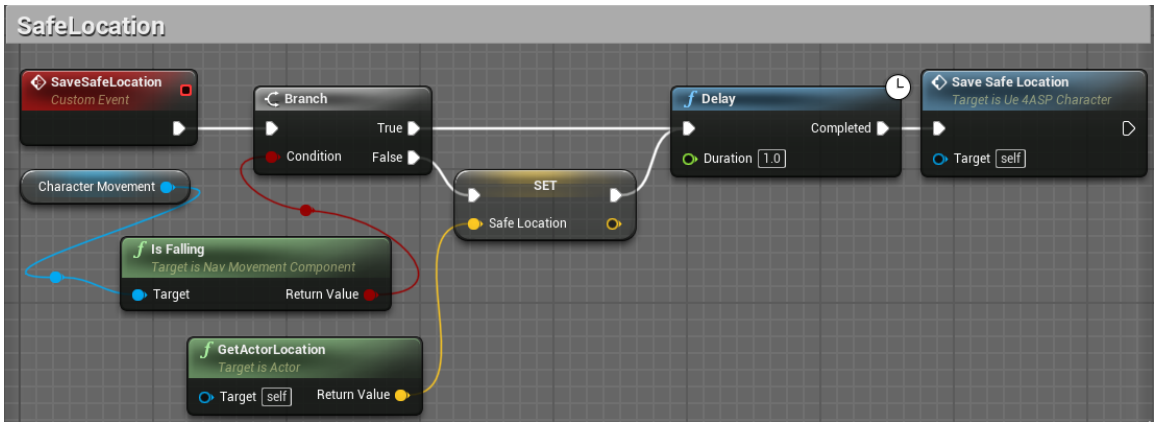
Unity je multiplatformní herní engine vytvořený firmou Unity Technologies v roce 2005. V Unity se programuje v jazyce C#, což je rozdíl oproti jiným enginům, ve kterých se programuje ve výkonnějších jazycích jako například C++. Unity je používáno převážně pro mobilní hry a hry s nižším rozpočtem, protože co se týká kvality po grafické stránce a stránce výkonu, Unity nedosahuje úrovně Unreal Enginu, který je na poli enginů hlavním konkurentem. Mezi počítačové hry využívající tento engine patří Cities: Skylines a Ori And The Blind Forest. Titul Hearthstone byl původně pouze počítačová hra, ale v průběhu života hry se dostal i na mobilní zařízení.

2.1.4 Unreal Engine 4

Unreal Engine (UE) je multiplatformní open-source herní engine vyvíjený společností Epic Games. První verze byla vydána v roce 1998. Aktuální verze 4 byla uvolněna v roce 2014. Kromě programování v C++ engine podporuje visual scripting pomocí blueprintů. Engine se používá jak pro tvorbu her, tak pro tvorbu filmů a seriálů a v neposlední řadě pro architektonické návrhy. Typickým žánrem, který se na tomto enginu vyvíjí jsou střílečky. Tituly běžící na UE4 jsou například Darksiders III, Hellblade: Senua's Sacrifice a Gears 5.

Blueprint

Blueprint je název vizuálního skriptování v UE4 a UE5, skládá se z uzlů (node) a propojů mezi nimi. Uzel reprezentuje funkci, nebo event (událost). Propojení mezi uzly je barevně rozděleno podle druhu spoje. Spoj realizující řídicí strukturu (control flow) je bílé barvy, zatímco ostatní barvy realizují propoje různých datových typů. Například světle zelená barva reprezentuje číslo s plovoucí řádovou čárkou a modrá reprezentuje objekt. Funkci nebo event vytvořený pomocí blueprintu je možné volat v rámci jiné funkce nebo eventu.



Obrázek 2.1: Obrázek znázorňující funkci implementovanou pomocí blueprintu

Na obrázku 2.1 lze vidět implementaci funkce `SaveSafeLocation`, která je podrobně popsána v sekci 4.1. Obrázek ukazuje přehlednost barevného rozdělení datových typů. Modrá

barva reprezentuje objekt, červená reprezentuje datový typ boolean a žlutá reprezentuje vektor neboli trojici hodnot.

Mezi výhody blueprintů patří jednodušší implementace a úprava vytvářené komponenty. Pomocí blueprintů mohou programovat i členové vývojové týmu, kteří nejsou programátoři v tradičním významu. Další výhodou je lepší přehlednost určitých struktur jako jsou třeba rozhodovací stromy.

Oproti tomu výhodou kódu psaného v jazyce C++ je vyšší výkon, což platí obzvláště pro matematické operace. Další výhoda je snazší porovnávání změn v různých verzích implementace, jako je například program Diff, nebo Git Merge.

Problém s výkonem implementace pomocí blueprintů oproti implementaci v jazyce C++ nicméně nemusí být markantní. Kromě už zmíněného problému s výkonem v matematických výpočtech je problémové volání jednotlivých uzlů. Jakmile se vykonávání dostane dovnitř uzlu, je výkon srovnatelný s voláním z kódu v jazyce C++. Přepisem blueprint třídy, která obsahuje malý počet výpočetně náročných uzlů do jazyka C++ se výkon významně nezlepší. Přepis třídy iterující přes velký počet uzlů bude mít na výkon velký vliv.

Velkým problémem jsou funkce `Tick`, které jsou volány každý snímek (Při 60 snímcích za sekundu je funkce volána každých 16,7 milisekund). Volání těchto funkcí z tříd implementovaných pomocí blueprintů je mnohem pomalejší než volání z tříd implementovaných v jazyce C++. Tento problém je umocněn častým voláním těchto funkcí. Obecně využívání těchto funkcí není doporučováno, pokud to není zcela nutné, toto platí obzvláště pro třídy, které mají velký počet instancí [1].

S ohledem na výše zmíněné výhody a nevýhody je blueprint vhodný zejména pro prototypování, kde rychlost implementace je vítanou výhodou a nižší výkon není problematický.

2.1.5 Unreal Engine 5

Unreal Engine 5 (UE5) byl uveden 5. dubna 2022. UE5 je zpětně kompatibilní s UE4.26 a 4.27. Engine obsahuje nové technologie, které se zaměřují hlavně na vizuální stránku. Mezi oznámené tituly pro tento engine patří zatím bezejmenné díly série The Witcher a Tomb Raider.

Nanite

Nanite je mikropolygonní geometrický systém, který umožňuje vytvářet hry s velmi velkým množstvím geometrických detailů bez nutnosti vkládat detaily do normálových map nebo manuálního vytváření LOD (Level of Detail). Díky technologii Nanite je možné do scény vložit model v nejvyšší kvalitě (LOD), který může mít miliony trojúhelníků při zachování hratelného počtu snímků za vteřinu (fps) a bez velkého snížení kvality.

Nanite funguje na principu cluster culling, kde je model rozdělen do mnoha clusterů. Velikost clusteru se dynamicky mění podle vzdálenosti modelu od kamery [9]. Každý cluster si může měnit LOD, nebo být úplně ořezán, pokud není vidět. Díky tomuto dochází ke zjemnění kontroly nad kvalitou, ve které je model renderován.

Lumen

Lumen je systém pro dynamické odrazy a Global Illumination, který běží v reálném čase (real-time). Umožňuje dynamický rozptyl nepřímého osvětlení, čímž způsobuje Color Bleeding. Color Bleeding je jev, kdy světlo, které se odráží od povrchu převezme barvu tohoto povrchu a bude osvětlovat okolní povrchy touto barvou. Lumen dokáže pracovat s neo-

mezeným počtem odrazů, což je vhodné pro scény s jasným světlem a velkým množstvím difuzních materiálů.

V Unreal Engine 4 je dynamické osvětlení řešeno pomocí přímých zdrojů, protože nepřímé osvětlení je velmi výpočetně náročné. Z tohoto důvodu je v UE4 všechno nepřímé nasvícení řešeno pomocí aproximace a jiných metod, jako je například výpočet osvícení předem a následné vložení do scény. Toto řešení má za následek, že nepřímé osvětlení není spolehlivé a velmi složitě se dynamicky mění.

Lumen využívá softwarovou implementaci Ray-Tracingu, která se dá hardwarově akcelerovat. Díky tomuto řešení Lumen funguje na nových i starých grafických architekturách. V režimu s HW akcelerací jsou efekty aplikované ve vyšší kvalitě. HW akcelerace se používá pro architektury Turing, Ampere, RDNA2 a Xe-HPG[1].

2.1.6 Proprietární engine

Pro volbu enginu existují dvě možnosti. První možnost je použít existující engine třetí strany (Unreal Engine, Unity a další), nebo si vytvořit vlastní, zpravidla proprietární, engine. Vývoj moderního enginu je časově (a tedy i finančně) velmi náročný proces, který si malá nezávislá studia zpravidla nemohou dovolit. Oproti tomu velcí vydavatelé sdílí vlastní engine mezi svými studii, pokud to povaha vytvářených her dovoluje. Běžné je využívání stejného, nebo novějšího enginu pro nový díl herní série. Příkladem je série The Witcher, kde druhý díl je postaven na REDengine a třetí díl používá REDengine 3.

Ke sdílení enginu může docházet i mezi různými tituly a někdy i mezi tituly různých žánrů. Příkladem může být studio BioWare s titulem Mass Effect: Andromeda, kde vývojáři použili Frostbite engine vytvořen studiem DICE původně pro Battlefield: Bad Company. Obě zmíněná studia patří pod vydavatele Electronic Arts (EA).

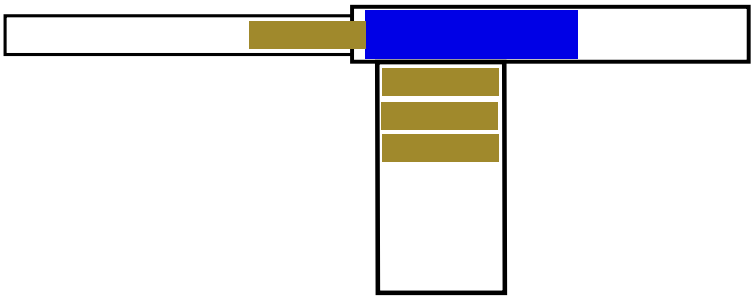
Tento konkrétní případ sdílení enginu skončil neúspěchem mimo jiné kvůli tomu, že Frostbite je engine vytvořen pro střílečky z první osoby, ale série Mass Effect je role-playing game (RPG) ze třetí osoby. Vývojář pracující na tomto titulu řekl, že Unreal Engine je univerzální engine, který dokáže dělat téměř všechno dobře, ale v ničem neexceluje. Oproti tomu Frostbite dokáže dělat mnohem méně věcí, ale dělá je velmi dobře [14]. Věci, které Frostbite dělá velmi dobře se týkají FPS her. Vývojáři proto museli některé specifické funkcionality pro žánr her ze třetí osoby vytvořit úplně od začátku.

2.2 Palná zbraň

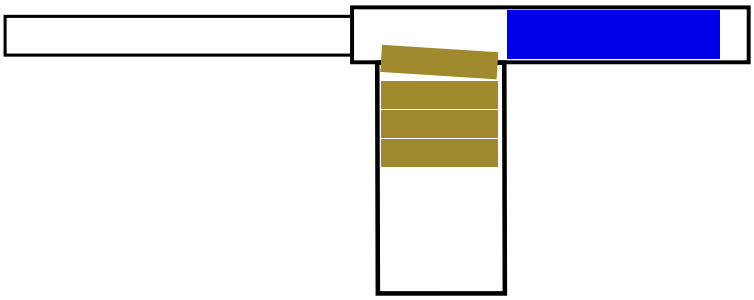
Při střelbě z uzavřeného závěru se náboj před výstřelem nachází v nábojové komoře a závěr je nachystán za ním k zapáleni náboje. Při zmáčknutí spouště se nepohybuje směrem dopředu. Díky nachystanému náboji v nábojové komoře může mít zbraň střílející z uzavřeného závěru nabito o jeden náboj víc, než je kapacita zásobníku. Pozice prvního náboje je zobrazena na obrázku 2.2.

Oproti tomu zbraň střílející z otevřeného závěru má náboj ke střelbě připravený na vrcholu zásobníku. Po stisknutí spoustě se závěr začne pohybovat směrem dopředu, nabere po cestě náboj ze zásobníku a zasune ho do nábojové komory, kde ho odpálí. Zbraň střílející z otevřeného závěru má všechny náboje uloženy v zásobníku a její kapacita nábojů je rovna kapacitě zásobníku. Pozice prvního náboje je zobrazena na obrázku 2.3.

[1] https://www.intel.com/content/www/us/en/architecture-and-technology/visual-technology/arc-discrete-graphics/xe-hpg-microarchitecture.html



Obrázek 2.2: Pozice náboje před výstřelem u zbraně s uzavřeným závěrem



Obrázek 2.3: Pozice náboje před výstřelem u zbraně s otevřeným závěrem

Na obrázcích 2.2 a 2.3 je zobrazena rozdílná poloha náboje k výstřelu. Oba obrázky znázorňují zbraň připravenou k výstřelu. Hnědý obdélník symbolizuje náboj a modrý obdélník závěr.

AK-47

AK-47 (obrázek 2.4), hovorově kalašnikov, je nejpoužívanější a nejznámější útočnou puškou na světě. Zbraň byla vyráběna v sovětském svazu a v zemích pod jeho vlivem. Zbraň je známa díky své vysoké spolehlivosti v těžkých podmínkách a levnou výrobou. V současnosti se používá zejména v zemích třetího světa a zločineckých organizacích.

Obrázek 2.4: Útočná puška AK-47, převzato[2]

[2] https://upload.wikimedia.org/wikipedia/commons/6/65/AK-47_type_II_noBG.png

M4

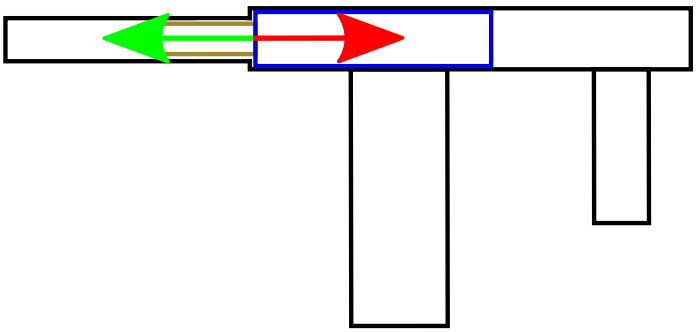
Karabina M4 (obrázek 2.5) je zkrácená verze útočné pušky M16A2. Zbraň se vyrábí od roku 1994 a od té doby se stala jednou z nejrozšířenějších Západních zbraní. Je ve výzbroji více než 60 armád světa a byla označena jako jedna z nejvýznamnějších zbraní 21. století [11].

Obrázek 2.5: M4A1 vybavena ručkou a zaměřovačem ACOG, převzato[3]

2.2.1 Zpětný ráz

Zpětný ráz (Recoil), hovorově kopání, je síla působící na palnou zbraň při výstřelu. Jedná se o důsledek Newtonova třetího pohybového zákona (zákon akce a reakce). Vektory hybností projektilu a zbraně jsou zobrazeny na obrázku 2.6. m_1 značí hmotnost a v_1 rychlost projektilu, m_2 a v_2 značí hmotnost a rychlost zbraně. Pak bude platit zjednodušená rovnice zpětného rázu [7, 13]:

$$m_1.v_1 = m_2.v_2 \tag{2.1}$$



Obrázek 2.6: Obrázek znázorňující vektory hybnosti projektilu (zelená šipka) a zbraně (červená šipka) při výstřelu

Zpětný ráz se u velkých zbraňových systémů jako jsou tanky a dělostřelectvo dá kompenzovat pomocí pružin a hydraulických systémů pracujících na stejném principu jako tlumiče pérování u automobilů. Z rovnice (2.1) plyne, že velikost zpětného rázu je dána projektilem. Kompenzace pouze spočívá v odvedení této hybnosti mimo soustavu. Existují i zbraně, které nemají zpětný ráz (bezzákluzové dělo). Zpravidla se jedná o protitankové zbraně jako

[3] https://upload.wikimedia.org/wikipedia/commons/a/a7/M4A1_ACOG.jpg

je AT4 a M40. Tato práce se zabývá pouze běžnými osobními palnými zbraněmi jako je AK-47 a M4.

2.2.2 Záblesk při střelbě

Záblesk při výstřelu, anglicky muzzle flash, je označení pro světelný efekt, který vzniká při expanzi plynu, který má vysokou teplotu a tlak při výstřelu z palné zbraně. Jde o důsledek spalovaní střelného prachu a reakce nespáleného střelného prachu s okolním vzduchem. Tvar a velikost závisí od energie vzniklé při výstřelu, druhu použitého střelného prachu a v neposlední řadě od doplňku nainstalovaném na hlavni. Na obrázcích 2.7 a 2.8 je vidět rozdílná velikost záblesku u tanku Merkava vybaveného kanonem ráže 120 milimetrů a pistole FN Five-seveN ráže 5,7 milimetrů.

Obrázek 2.7: Záblesk při výstřelu z tankového kanonu ráže 120mm, převzato a upraveno[4]

Obrázek 2.8: Záblesk při výstřelu z pistole ráže 5,7mm, převzato[5]

[4] https://upload.wikimedia.org/wikipedia/commons/6/69/Merkava_Muzzle_Flash.jpg
[5] https://upload.wikimedia.org/wikipedia/commons/2/2d/FNFLASH01.jpg

2.3 Herní mechaniky

Tato sekce slouží k vysvětlení některých pojmů týkajících se herních mechanik, které se v této práci vyskytují. Dále sekce obsahuje popis některých problémů, ke kterým může dojít při vývoji videoher a jsou zde i popsány některé postupy řešení daných problémů včetně jejich výhod a nevýhod.

2.3.1 NPC

Non-player character je postava ve hře, která není kontrolována hráčem, ale její chování je naprogramované a většinou ho spouští nějaká hráčova akce. Objevují se ve hrách různých žánrů, ale jejich výskyt je nejtypičtější pro hry žánru RPG.

Ve hře plní dvě role. První rolí je interakce s hráčem a druhá role je dekorativní. Dekorativní role má za úkol rozšířit herní svět o více postav, které ale nejsou pro hráče nijak důležité. Obvykle se vyskytují například ve městech nebo jiných oblastech, kde by hráč čekal velké množství lidí, kde se snaží vytvořit dojem, že město nebo oblast „žije“.

Druhým typem jsou NPC, které nějakým způsobem interagují s hráčem. Jedná se například o obchodníky, u kterých hráč může nakupovat nebo prodávat vybavení a jiné věci, dále jde o postavy, které hráči zadávají úkoly.

2.3.2 Osy v 3D prostoru

Těleso v trojrozměrném prostoru může rotovat kolem tří navzájem kolmých rovin. Pro označení osy, kolem které se provádí rotace se používá terminologie z letectví. Na obrázku 2.9 jsou vidět všechny tři osy, kolem kterých může objekt rotovat.

1. Pitch - Rotace kolem osy jdoucí zleva doprava. Způsobuje pohyb nosu letadla nahoru, nebo dolů.
2. Yaw - Rotace kolem vertikální osy. Nos se pohybuje doleva, nebo doprava.
3. Roll - Rotace kolem osy jdoucí zepředu letadla dozadu. Nos se otáčí podle směru, nebo proti směru hodinových ručiček.

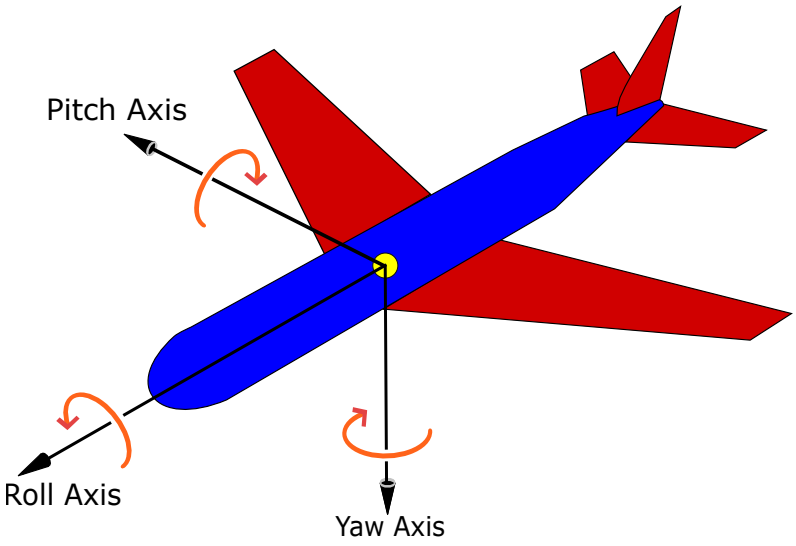
2.3.3 Částicový systém

Částicový systém, anglicky particle system je tvořen relativně velkým množstvím relativně jednoduchých geometrických entit. Tyto entity jsou často orientovány směrem do kamery a jsou částečně průhledné. Částice jsou průběžně vytvářeny a ničeny, jejich životnost je zpravidla velmi krátká. Systém se používá pro tvorbu efektů jako je kouř a oheň.

2.3.4 Inverzní Kinematika

Inverzní kinematika je matematický proces využívaný v oblasti robotiky a počítačových animací. Jedná se o proces výpočtu natočení kloubů, pokud je známa koncová pozice konce končetiny [10]. Jako příklad se uvádí dotek robotické ruky skleničky. Díky inverzní kinematice lze vzít prst ruky a položit ho na skleničku a inverzní kinematika dopočítá natočení všech kloubů v dané končetině.

Opakem je dopředná kinematika, která dopočítá koncový bod ramena za předpokladu, že je známé natočení kloubů. Tento proces je jednodušší než inverzní kinematika.



Obrázek 2.9: Obrázek znázorňující všechny tři osy v prostoru na příkladu letadla, převzato[7]

2.3.5 Pohled kamery

Existují tři základní způsoby, jak řešit integraci kamery do modelu hráče u střílečeк z první osoby. V současnosti se používá pouze jeden způsob.

Nejjednodušší přístup je místo použití celého modelu pro hráče použít model, který obsahuje pouze ruce. Výhoda tohoto přístupu je jednodušší implementace. Nevýhoda je velmi nízká realističnost. Například při pohledu směrem pod sebe hráč zjistí, že nemá nohy a jenom levituje v prostoru. Dalším problémem je vrhání stínu. Protože model obsahuje pouze ruce (a zbraň), pouze tyto komponenty vrhají stín, což vede opět ke snížení realistického dojmu ze hry. Toto řešení se využívalo ve starších hrách, které zpravidla nepoužívaly stíny.

Druhým přístupem je použití celého modelu postavy a využití inverzní kinematiky k synchronizaci pohybu postavy, zbraně a kamery. Tato metoda při dobré implementaci vypadá velmi realisticky. Nevýhodou je časově náročnější implementace. Jedná se o jediné řešení, které se využívá v moderních hrách.

Posledním přístupem je použití pouze rukou a následně přidání další postavy hráče, která obsahuje všechny komponenty. Tento model není renderován, ale pouze vrhá stín. Ruce oproti tomu jsou viditelné, ale nevrhají stín. Výhodou tohoto přístupu jsou realističtější stíny, ale chybějící nohy stále přetrvávají. Při této metodě je důležitá kvalita modelu pro vrhání stínu.

Při dost nízké kvalitě modelu je možno vidět nesoulad mezi tím, co dělá postava hráče a co dělá stín. Naopak při použití vysoké kvality je časová náročnost vytvoření modelu srovnatelná s řešením pomocí inverzní kinematiky a tedy není důvod pro volbu tohoto hybridního přístupu.

[7] https://en.wikipedia.org/wiki/Aircraft_principal_axes#/media/File:Yaw_Axis_Corrected.svg

2.3.6 Střelba

Pro implementaci střelby se používají dvě různé metody. Obě metody mají výhody a nevýhody a neexistuje jediný správný přístup. Výběr vhodného režimu střelby závisí na požadavcích konkrétní hry.

Line Trace

Metoda Line Trace využívá pro detekci zásahů průsečíky s úsečkou. Pro realizaci výstřelu je nutné znát počáteční a koncový bod úsečky. Jako počáteční bod se zpravidla volí ústí hlavně a jako konec se volí bod ve směru vektroru hráčovy kamery. Podle vzdálenosti tohoto bodu od hráče se určuje dostřel zbraně. Při výstřelu se vytvoří neviditelná čára a najdou se průsečíky s objekty. Podle druhu zasaženého objektu se může chování lišit. Například betonová zeď „projektil“ zastaví a další průsečíky se ignorují. Při zásahu nepřátelské postavy může „projektil“ pokračovat dále. Toto chování zcela záleží na vývojáři.

Výhodou tohoto přístupu je jednoduchá implementace a menší nároky na výpočetní zdroje. Mezi nevýhody patří nerealističnost. Nerealističnost má dva důvody. Prvním důvodem je instantní zásah, kdy „projektil“ má nekonečnou rychlost. Druhým důvodem je absence gravitace. Omezení jako absence gravitace a nutnost používat pouze úsečky je možno obejit, ale vede to na výrazně složitější implementaci. Nerealističnost nemusí nutně vadit. Například u FPS hry s rychlým tempem a malou mapou si hráč pravděpodobně nerealističnosti nevšimne.

Projektil

Tato metoda pro střelbu využívá kolize fyzických projektilů. Při výstřelu se vytvoří objekt reprezentující projektil. Projektil má určitý impulz, který reprezentuje rychlost a směr projektilu. Projektil se vytvoří před ústím hlavně, impulz směřuje směrem, kam zbraň míří. Pomocí nastavení kolizí se dá nastavit, jak se projektil chová při zásahu určitého objektu. Při zásahu může dojít například ke zničení projektilu, projektil může pokračovat beze změny pokračovat dál nebo může dojít ke snížení rychlosti.

Výhody tohoto přístupu je vyšší realističnost, protože projektil má určitou rychlost a je možno pro projektil simulovat gravitaci. Nevýhodou jsou vyšší nároky na paměťovou náročnost. Toto platí obzvláště pro malé mapy s rychlým tempem a velkým počtem hráčů, kde jsou souboje mezi hráči velmi časté. Například při sto hráčích, kteří mají rychlopalné zbraně může být naraz přítomno několik tisíc projektilů, což může mít velký vliv na výkon hry.

Oba přístupy je možno kombinovat a dosáhnout tak vysoké míry realističnosti při zachovaní udržitelné paměťové náročnosti. Příkladem může být použiti fyzických projektilu pro zbraně, u kterých se dá předpokládat boj na dlouhou vzdálenost jako například odstřelovací pušky nebo zbraně vysoké ráže, jako například kanony v tanku. Pro běžné zbraně na relativně blízkou vzdálenost se může používat střelba pomocí metody line trace, protože v této situaci si hráč nevšimne nerealističnosti.

2.3.7 Nabíjení zbraní

Ve střílečkách je nabíjení řešeno dvěma způsoby. V obou případech je někde uložen aktuální počet nábojů v zásobníku a počet náhradních nábojů, které hráč nese.

V prvním případě, který je v dnešních střílečkách používanější, dochází při nabíjení k doplnění pouze chybějícího počtu nábojů do zásobníku. Tento způsob není realistický, protože při tomto způsobu nabíjení by střelec musel z aktuálního zásobníku vysypat zbývající náboje a následně je někdy použít k naplnění prázdného zásobníku. Místo nabíjení, které trvá maximálně pár sekund by se doba nabíjení zvýšila minimálně na řád minut. Výhodou je lepší přívětivost pro hráče, který není trestán ztrátou nábojů za časté částečné nabíjení. Příkladem jsou hry Battlefield 4, Call of Duty Modern Warfare a Rainbow Six Siege.

V druhém případě se aktuální zásobník zahodí včetně všech nábojů, které se v něm nachází a nabije se nový zásobník, který je plný. Při nabíjeni se vždy nabijí celá kapacita zásobníku. Tento způsob je realistický, protože dochází k výměně starého zásobníku za nový a žádná další manipulace není potřebná a je tedy časově optimální. Nevýhodou může být, že hráči dojde munice, pokud často nabíjí při vystřílení male části nábojů ze zásobníku. Příkladem je hra Mafia. Hra Ghost Recon Breakpoint umožňuje hráčovi zapnout obtížnější herní mód, kde dochází k zahazování zásobníků. Hra Insurgency používá model, kde hráč má k dispozici určitý počet zásobníku. Při nabíjení se částečně plný zásobník schová do vesty pro případné další použití. Pokud hráč vystřílí všechny plné zásobníky, dojde k nabití schovaného zásobníku, který má stejný počet nábojů jako při jeho schování.

2.3.8 Střílečka

Střílečky jsou žánrem akčních her a jedná se o nejprodávanější žánr videoher [3]. Jejich cílem je poražení nepřátel, ke kterému dochází zpravidla jejich zničením či zabitím. Hráč využívá palné zbraně, granáty nebo nože pro boj zblízka.

Střílečky se dělí podle různých kritérií na subžánry. Podle pohledu kamery se dělí na střílečky z první osoby a na střílečky z pohledu třetí osoby. Obrázek 2.10 znázorňuje rozdíl v umístění kamery u těchto subžánrů.

First-person shooter (FPS)

Střílečka z první osoby, nebo anglicky First-person shooter (FPS), je subžánr střílečeк. Hlavním rysem FPS her je pohled kamery, kdy kamera je umístěna v hlavě hráčovy postavy, obvykle v místě mezi očima. Díky tomu hráč vidí to stejné, co „vidí“ jeho postava ve hře. Mezi nejvýznamnější videohry tohoto žánru patří Wolfenstein, Half-Life, Doom, Counter-Strike a série Call of Duty a Battlefield. Hraní videoher tohoto žánru má pozitivní vliv na mentální flexibilitu. Hráči těchto her měli kratší reakční dobu při změně typu úkolů [4].

Hry tohoto žánru jsou typicky technologicky nejsložitější na tvorbu. Je to dáno tím, že tyto hry se snaží hráče vtáhnout do hry. K tomu je potřeba například velmi realistické prostředí, animace, fyzika a audio. [6, str. 14-15]

Third-person shooter (TPS)

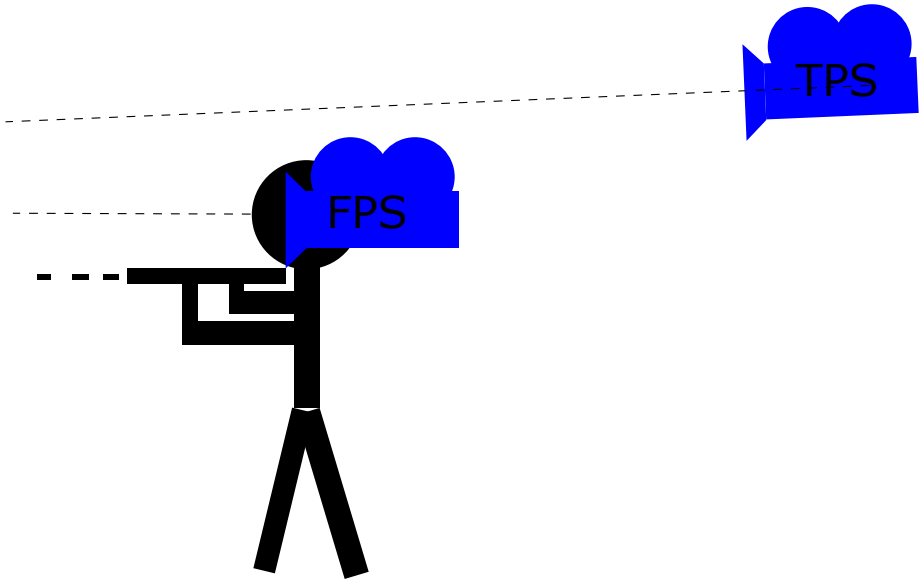
Střílečka ze třetí osoby, nebo anglicky Third-person shooter (TPS), je subžánr střílečeк. Žánr je charakteristický kamerou, která sleduje postavu hráče „přes rameno“. Pohled třetí osoby může hráči pomáhat lepším zorným polem a nahlížením za roh. Na druhé straně pokud pohyb kamery není implementován správně, může se kamera zaměřovat do míst, kde se „nic neděje“ a kazit tím hráči požitek ze hry. Některé hry tohoto žánru umožňují míření pomocí pohledu první osoby, což má za následek lepší přesnost při střelbě. Existují i střílečky umožňující hráči volně přepínat mezi pohledem první a třetí osoby. Příkladem

tohoto přístupu je hra Grand Theft Auto V. Tituly tohoto žánru jsou série Tomb Raider, Mass Effect a Gears of War.

Narozdíl od her z první osoby, kde stačí mít kvalitní model a animace pouze pro viditelné prvky (hlavně ruce) je u tohoto žánru důležitá kvalita modelu a animací celého hráče, protože kamera zabírá celou postavu.

Hry tohoto žánru zpravidla mají střelbu jako doprovodnou herní mechaniku. Příkladem jsou série Tomb Raider, kde hlavní mechanika je průzkum herního světa a řešeni logických úloh pro další postup. Dalším příkladem je série Mass Effect, kde i během boje je kladen důraz na super schopnosti hráče a jeho spolupráci s týmem. Posledním příkladem je série The Division, která přestože obsahuje střelbu se řádí spíše mezi RPG hry.

Oproti tomu u FPS her zpravidla hraje hlavní roli samotné střílení. Příkladem jsou série Call of Duty a Battlefield.



Obrázek 2.10: Obrázek znázorňující rozdíl v umístění kamery u FPS a TPS her.

Kapitola 3

Návrh

V první části této kapitoly se nachází zdůvodnění návrhových rozhodnutí, ke kterým došlo. Další sekce obsahují popis zbraní a doplňků, které se v práci vyskytují. Sekce 3.8 slouží k popisu návrhu herních mechanik, které jsou v práci implementovány. Na konci této kapitoly se nachází sekce 3.9, která popisuje potenciální rozšíření této práce. Cílem je vytvoření herního dema využívajícího mechaniky palných zbraní s možností úpravy zbraní pomocí doplňků a interakce s prostředím ve formě stop po střelách. Některé implementované mechaniky:

- Stopy po střelách
- Zahazování zásobníku při nabíjení
- Vyhazování nábojnic při střelbě
- Modifikace zpětného rázu podle nasazených doplňků na zbrani
- Modifikace záblesku při výstřelu podle nasazených doplňků na zbrani

3.1 Volba enginu

Pro volbu enginu byly zváženy dvě varianty. První byla Unity a druhá Unreal Engine. Jak už bylo zmíněno v sekci 2.1.4, Unreal Engine je hlavně zaměřen na vývoj střílečeck, což je jedním z důvodu výběru Unreal Enginu pro implementaci této práce. Dalším důvodem je programování pomocí blueprintů, které je časově výhodnější než implementace v běžném programovacím jazyce, jak bylo vzpomenuto v sekci 2.1.4.

3.2 Volba žánru

S ohledem na zadání připadaly při výběru žánru v úvahu pouze dva žánry. První možnost byla střílečka z první osoby a druhou moznosti byla střílečka ze třetí osoby. Jak už bylo popsáno v sekci 2.3.8, ve střílečkách ze třetí osoby bývá střelba pouze jako doplňkový prostředek. Vzhledem k tomu, že práce je zaměřená na mechaniky palných zbraní, střelba bude hrát dominantní roli. Z tohoto důvodu je jako žánr zvolena střílečka z první osoby, kde bývá mnohem větší důraz na střelbu.

3.3 Pohled kamery

Jak už bylo zmíněno v sekci 2.3.5, existují tři způsoby řešení pohledu kamery, které se liší mírou realističnosti a složitostí implementace. Protože se v současnosti používá pouze řešení pomocí plného modelu a inverzní kinematiky, není důvod pro tuto práci volit jiné řešení. To, že tato metoda dosahuje nejlepší realističnosti je další důvod pro volbu tohoto způsobu.

3.4 Volba způsobu střelby

Jak bylo popsáno v sekci 2.3.6, existují dvě metody implementace střelby a oba mohou být v této práci použity. V práci je implementována realizace pomocí fyzických projektilů, protože je tato metoda realističtější a nevýhoda v podobě vyššího nároku na paměť je zanedbatelná, protože v práci existuje pouze jedna postava, která může střílet.

Režim střelby

Zbraně v této práci obsahují tři režimy střelby:

- Single Shot - Střelba jednotlivými ranami
- Burst - Dávka dvou, nebo tří ran
- Full Auto - Plně automatický režim střelby

Režim střelby může být uložen v rámci objektu konkrétní zbraně, nebo v rámci hráče. Výhodou uložení režimu do zbraně je vyšší realističnost, protože každá zbraň má svůj režim pevně nastavený a lze ho změnit jenom pokud hráč drží danou zbraň v ruce. Oproti tomu uložení režimu do hráče má za následek změnu režimu všech zbraní naráz. Výhodou tohoto přístupu je lepší požitek ze hry.

Protože se tato práce zaměřuje na realističnost, bylo zvoleno uložení do zbraně. Dalším důvodem k tomuto rozhodnutí byl průzkum mezi testery, jak je uvedeno v sekci 5.3.

3.5 Zbraně

V práci jsou implementovány dvě zbraně. První zbraní je útočná puška AK-47 a druhou zbraní je karabina M4. Práce se snaží držet reálných vlastností a parametrů těchto zbraní v co největší míře. S ohledem na různorodost však některé parametry byly změněny. Tyto změny se týkají pouze režimu střelby. U karabiny M4 je udávaná kadence v rozmezí 700–950 ran za minutu. V této práci je kadence nastavena na hodnotu 900 ran za minutu.

Pro potřeby této práce jsou důležité tyto parametry: Váha zbraně, váha projektilu, úsťová rychlost a kadence. Tyto parametry jsou uvedeny v tabulce 3.1 [8, 13, 2].

Návrh režimů střelby se nedrží reality z několika důvodu. Prvním důvodem byla snaha implementovat více rozdílných věcí. Druhým problémem byla karabina M4, která má různé režimy střelby pro různé verze zbraně. AK-47 v reálu nemá střelbu dávkou, střílí pouze jednotlivými ranami, nebo plně automaticky. M4 střílí pouze jednotlivými ranami, nebo tříranými dávkami. M4A1 oproti tomu střílí jednotlivými ranami, nebo plně automaticky.

Obě zbraně v práci střílí jednotlivými ranami, dávkou a plně automaticky. Rozdílem je délka dávky, která u AK-47 je tři střely a u M4 dvě střely.

Tabulka 3.1: Fyzické vlastnosti zbraní

Parametr	AK-47	M4
Váha [kg]	4,3	3,4
Projektil [kg]	0,008	0,004
Úsťová rychlost [m/s]	710	884
Kadence [ran/min]	600	900

Upevňovaní doplňků je do jisté míry problematické, protože model ani jedné ze zbraní neobsahuje montážní lišty po instalaci doplňků (s výjimkou M4, která má pouze lištu pro kolimátor). Doplňky se proto uchycují přímo do těla zbraně na vhodné místo.

3.6 Doplňky

Doplňky, které jsou v práci implementovány jsou rozděleny do tří kategorii. Každá kategorie je určena pomocí místa, kde se doplňky uchycují ke zbrani. Každá z těchto kategorií obsahuje kromě různých doplňků i možnost nemít osazený na dané pozici žádný doplněk. Tyto skupiny jsou: hlaveň, rukojeť a mířidla.

Hlaveň

Tato skupina doplňků se upevňuje na hlaveň. Tyto doplňky mají vliv na zpětný ráz a na tvar záblesku při výstřelu. Tlumič snižuje zpětný ráz o 10 %. Druhým efektem tlumiče je eliminace záblesku při střelbě, při výstřelu se z tlumiče pouze kouří. Flash Hider snižuje zpětný ráz o 25 %. Muzzle Break na rozdíl od obou předchozích doplňků ovlivňuje pouze krátké dávky. Konkrétně dochází k úplné eliminaci zpětného rázu prvního náboje v dávce.

Rukojeť

Navzdory názvu do této kategorie patří i jiné doplňky než ručky. Tyto doplňky se připojují do místa pod hlavní. Vertical Grip snižuje zpětný ráz, modifikátor má hodnotu 0,5. Angled Grip snižuje dobu potřebnou pro zaměření na polovinu. Konkrétně snižuje dobu z jedné sekundu na půl sekundy. Svítilna slouží k osvětlení relativně úzkého prostoru před postavou. Laser vyzařuje velmi úzký proud světla, který simuluje chování reálného laseru.

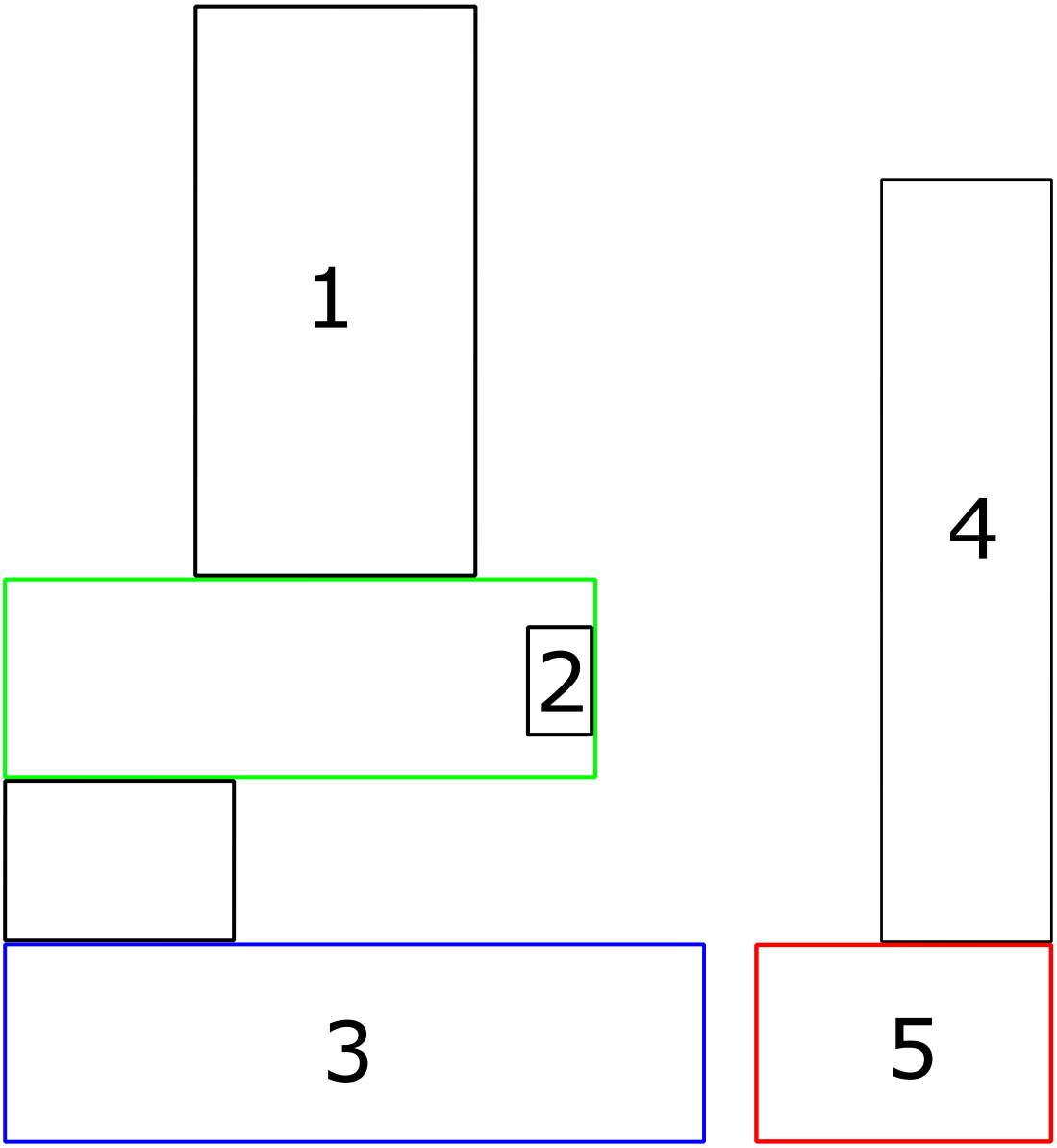
Mířidla

Do této skupiny patří zaměřovací prostředky. Modifikace této kategorie nemají přímý vliv na zpětný ráz. Tyto doplňky mění vzhled zaměřovače a zorný úhel, anglicky field of view (FoV). Tento úhel přímo ovlivňuje přiblížení zaměřovače. Čím je zorný úhel menší, tím je větší přiblížení a naopak. K tomu dochází díky tomu, že na obrazovku se vleze určité množství informací. Pokud se zorný úhel zmenší, pak se musí zvětšit všechny objekty, které se v daném zorném úhlu stále nachází, aby pokryly celou plochu obrazovky.

3.7 Mapa

Mapa je rozdělena na dvě navzájem oddělené části. První část je místo, kde se hráč objeví po zapnutí hry a druhá část mapy je místo, kde se vyskytují roboti. Nákres mapy je na obrázku 3.1.

Oblast číslo 1 je pro střelbu na větší vzdálenost. Modrý obdélník (3) značí zakrytou střelnici, kde je minimum světla, je vhodná pro demonstraci záblesků při střelbě a doplňků jako laser a taktická svítilna. Z místa číslo 2 je vidět na oblast 4, kde se pohybují roboti, na které lze střílet. Místo 5 označuje „továrnu“ na roboty. Roboti z ní vybíhají a v případě zničení se v ní opět objeví.



Obrázek 3.1: Nákres mapy

3.8 Herní mechaniky

Tato sekce obsahuje popis návrhu herních mechanik, které jsou práci implementovány.

3.8.1 Zpětný ráz

Zpětný ráz je simulován pomocí úpravy parametrů hráčovy kamery. Konkrétně se jedná o parametry Pitch a Yaw. Hodnota, o kterou se tyto parametry změní, je součin základní hodnoty, která je jako atribut každé zbraně, a modifikátorů.

Výpočet zpětného rázu

Pro výpočet zpětného rázu bylo provedeno několik zjednodušení. Prvním zjednodušením je uvažovaní konstantní váhy zbraně. Váha zbraně je brána jako součet váhy prázdné zbraně a plného zásobníku.

Pro výpočet zpětného rázu dle rovnice (2.1) se používá jako hmotnost náboje pouze hmotnost samotného projektilu místo hmotnosti celého náboje. K tomuto rozhodnutí došlo, protože samotná nábojnice se bezprostředně po výstřelu nehýbe, proto je její hybnost nulová. Projektil oproti tomu je vystřelen určitou rychlostí.

Dalším zjednodušením je rychlost výstřelu projektilu. Tato práce pro výpočet zpětného rázu využívá jako rychlost projektilu úsťovou rychlost střely. Úsťová rychlost střely je rychlost, kterou má střela při opuštěni hlavně, tedy po tom, co během letu celou délkou hlavně zrychlovala. Reálná rychlost při výstřelu je nižší. Toto řešení je kvůli zjednodušení výpočtu.

Dalším zjednodušením je uvažovaní, že zpětný ráz působí pouze v době výstřelu, konkrétně při zapálení nábojnice.

Pomocí rovnice (2.1) se dopočítá rychlost zbraně. Tato rychlost má opačný směr, než je směr rychlosti projektilu, zbraň se tedy pohybuje směrem do střelcova ramene.

Tato rychlost se následně použije jako základní (base) zpětný ráz ve vertikálním směru. Pro výpočet v horizontálním směru se rychlost vydělí dvěma a každá polovina se použije jako základní zpětný ráz ve pro svou stranu.

K tomuto zjednodušení došlo, protože cílem práce není simulátor zpětného rázu, ale herní demo. Pro vyšší realističnost by bylo vhodné počítat vždy aktuální váhu zbraně plus váhu prázdného zásobníku plus váhu aktuálního počtu nábojů, které se v zásobníku aktuálně nacházejí. Dále by bylo nutné počítat rychlost projektilu v každém bodě hlavně až do okamžiku opuštění hlavně. Pomocí této rychlosti by se v každém okamžiku počítala hybnost projektilu, ze které by se spočítala rychlost samotné zbraně.

Efektivní zpětný ráz

Efektivní zpětný ráz je konečná hodnota ve stupních, o kterou zbraň při výstřelu bude rotovat. Efektivní zpětný ráz se počítá jako součin základního (base) zpětného rázu a tři modifikátorů.

Prvním modifikátorem, označen jako *length*, je délka dávky.

Druhý modifikátor, označen jako *aim*, určuje, zda hráč míří (ADS, Aim down Sight), nebo střílí od boku (Hip Fire).

Třetí modifikátor, označen jako *attachments*, je výsledek doplňků používaných na zbrani.

Hodnota *base* označuje náhodnou hodnotu z intervalu $[0, VerticalRecoil]$ pro výpočet zpětného rázu ve vertikálním směru. Rovnice vyjadřující výpočet efektivního zpětného rázu

vypadá následovně:

$$effective = base \cdot length \cdot aim \cdot attachments \tag{3.1}$$

Celkový zpětný ráz je kombinací zpětného rázu v horizontálním a vertikálním směru.

Vertikální zpětný ráz se počítá dle rovnice (3.1) a tato hodnota se přičte jako Pitch input pro ovládání kamery.

Horizontální zpětný ráz na rozdíl od vertikálního může mířit na obě strany (doprava a doleva). Pro zohlednění tohoto faktu každá zbraň obsahuje pro horizontální recoil dvě hodnoty. Jednu, která určuje recoil doprava a druhá doleva. Každá zbraň používá symetrický horizontální zpětný ráz. Hodnota doleva má tedy stejnou hodnotu, jako hodnota doprava, pouze je negativní.

Horizontální zpětný ráz se počítá dle rovnice (3.1), ale jako *base* se bere náhodná hodnota z intervalu $[RecoilDoleva, RecoilDoprava]$ a tato hodnota se přičte jako Yaw input pro ovládání kamery.

3.8.2 Záblesk při střelbě

Záblesk při výstřelu je řešen dvěma separátními částmi. První část je tvorba charakteristického tvaru při výstřelu. Tohoto je dosazeno pomoci Particle Systemu. Podle použitého doplňku na hlavni se tento tvar mění a při použití tlumiče je nahrazen vypuštěním kouře z hlavně. Jako doplňující efekt se pro všechny konfigurace hlavně po krátkou dobu vypouští malé množství kouře.

Druhou částí je vytvoření krátkého světelného záblesku, který se chová jako všesměrový bodový zdroj světla, který při výstřelu osvítí scénu. Světelný impulz je přítomný pro všechny konfigurace hlavně s výjimkou tlumiče, kdy chybí.

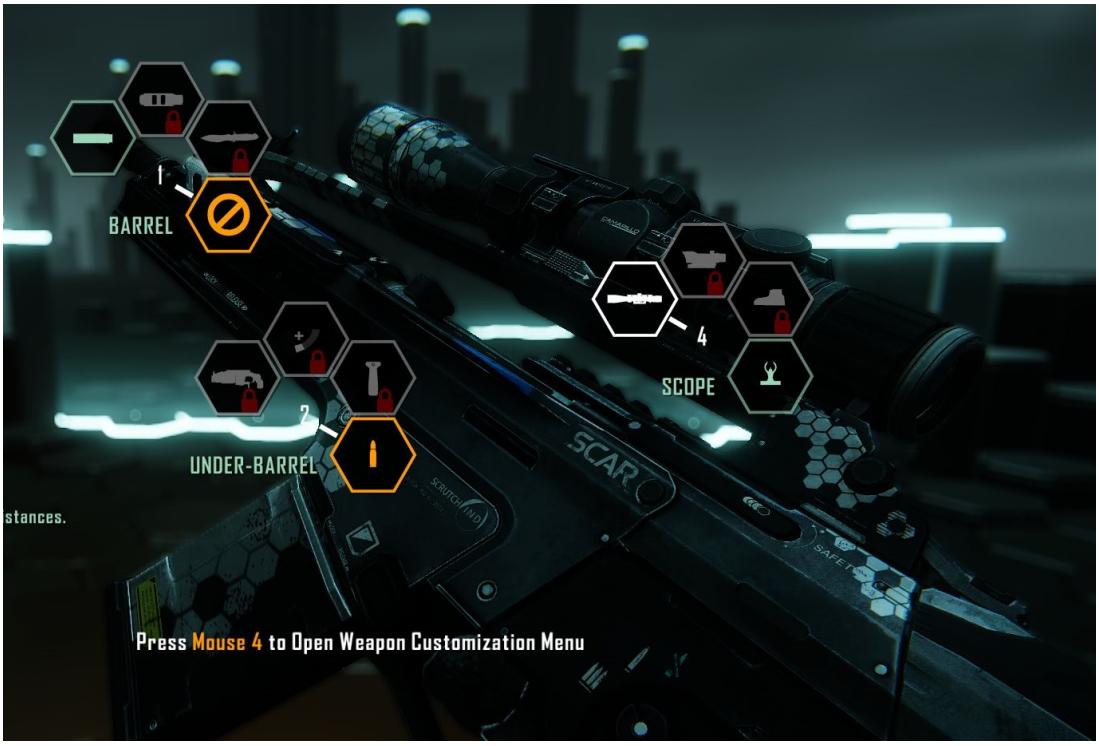
3.8.3 Nabíjení zbraní

Volba mezi oběma způsoby implementace nabíjení ve hrách, jak je zmíněno v sekci 2.3.7, je otázka, zda dojde k upřednostnění realističnosti, nebo komfortu hráče. Vzhledem k tomu, že tato práce klade důraz na realističnost, nabíjení je realizováno pomocí nabíjení celého zásobníku.

Všechny zbraně, které jsou v práci implementovány, střílí z uzavřeného závěru, což znamená, že náboj je před výstřelem umístěn v nábojové komoře. Při nabíjení je třeba kontrolovat, zda se před nabíjením nacházel náboj v nábojové komoře. Tato kontrola je triviální, protože pokud měla zbraň k dispozici alespoň jeden náboj, pak byl jeden náboj připraven v nábojové komoře. Pokud se náboj nacházel v nábojové komoře, pak při nabití nového zásobníku bude mít zbraň kapacitu o jeden náboj vyšší, než je kapacita zásobníku.

3.8.4 Menu pro úpravu zbraní

Menu pro výběr modifikace zbraně je inspirováno videohrou Crysis 3 (obrázek 3.2). Při zmáčknutí klávesy pro modifikaci zbraně (klávesa X) se kamera posune vpravo do pozice, ze které vidí kolmo na zbraň. Posun na pravou stranu je volen, protože levá strana zbraně je z velké části zakryta levou rukou hráče. Na obrazovce dojde k vykreslení tří skupin tlačítek, pomocí kterých se volí doplněk. Tlačítko pro aktuálně vybraný doplněk v dané skupině je zvýrazněno pomocí tmavého pozadí.



Obrázek 3.2: Menu pro úpravu zbraní ve hře Crysis 3, převzato a upraveno[1]

3.8.5 Interakce s prostředím

V úrovni jsou přítomni 4 NPC, kteří se postupně vygenerují ve dvou sekundových intervalech. Tito NPC se pohybují po rovnoběžných trasách mezi body A a B, každý jinou rychlostí. Hráč může NPC zničit střelbou. Po zničení NPC dojde k jeho oživení (respawn).

Práce obsahuje mechaniku vyhazování nábojnic při střelbě a odhazování zásobníku při přebíjení. Pro potřebu další munice se v úrovni nachází několik beden s municí, kde si hráč může doplnit náboje.

Stopy po střelách

Při zásahu jakéhokoliv objektu projektilem se na něm vytvoří „díra“. Jedná se o jednoduchou texturu, která svým vzhledem připomíná díru po střele. Tato textura se při nárazu projektilu do objektu aplikuje na objekt v místě kolize. Vzhledem k velmi malým rozměrům projektilu, a tedy i této textury, bylo nutné zvýšit vzdálenost, ze které se tato textura renderuje.

Pro zlepšení realističnosti se v místě kolize vytvoří pomocí částicového systému malé množství částic, které simuluje úlomky vzniklé při nárazu projektilu do objektu.

3.9 Potenciální rozšíření

Pro animaci nabíjení se používá pouze jedna animace hráče, konkrétně jde o animaci výměny zásobníku. Pro vyšší realističnost by bylo vhodné rozlišit animaci, kdy se nachází náboj v nábojové komoře, kde pro úspěšné nabití stačí výměna zásobníku (aktuálně použí-

[1] https://www.notebookcheck.net/Crysis-3-Benchmarked.89410.0.html#1120222-4

vaná animace) a stav, kdy se před nabíjením nenacházel náboj v nábojové komoře. V tomto případě je nutno kromě výměny zásobníku provést i natažení závěru. Toto vyžaduje vytvoření nové animace pro postavu a vytvoření animace pro zbraň.

Jako potenciální rozšíření práce je braní v potaz, zda nabíjená zbraň střílí z otevřeného, nebo uzavřeného závěru. Díky tomuto by bylo dosaženého správného chování pro oba typy zbrani. Dále by nabíjení mohlo být realizováno pomocí pevného počtu zásobníku, kde při částečném vystřílení zásobníku by se tento zásobník uložil pro další použití. Po vystřílení všech plných zásobníku by se začaly nabíjet tyto částečné zásobníky. Tento přístup je realizován ve hře Insurgency.

Dalším možným rozšířením je implementace realistické destrukce prostředí. Příkladem může být střelba na dřevěný povrch, kdy se v povrchu vytvoří reálná díla (oproti pouze aplikaci textury imitující průstřel) a povrch bude rozštěpen na třísky, které se rozletí do okolí. V práci bylo experimentováno s destrukcí pomocí systému APEX, který je součástí Unreal Enginu 4. Problémem tohoto systému je nastavení parametrů pro objekty, které se dokážou „rozpadnout“. Při experimentech se objekt buď rozpadl při sebemenší síle, která na něj působila, nebo vůbec. Unreal Engine obsahuje nově i další systém destrukce – Chaos. Chaos vyžaduje instalaci speciální verze enginu (4.26-Chaos a 4.27-Chaos) a je pouze v experimentálním stavu. Není tedy aktuálně vhodný pro použití v této práci. Hlavní využití tohoto systému je v Unreal Enginu 5.

Kapitola 4

Implementace

Tato kapitola slouží k popisu implementace herních mechanik. V kapitole se nacházejí i zjednodušené popisy blueprintů implementující popisované chování. Sekce 4.1 a 4.2 popisují implementaci některých prvků hráčovy postavy, respektive zbraně. Sekce 4.3 popisuje implementaci nabíjení včetně animace. Sekce 4.4 obsahuje popis hodnot modifikátorů zpětného rázu. Na konci kapitoly se nachází sekce 4.6, která vysvětluje implementaci robotů, kteří slouží jako cvičné cíle.

4.1 Třída hráče

Třída hráče je implementována pomoci třídy `Ue4ASP_Character`. Třida má dvě zodpovědnosti. První zodpovědností je ukládaní dat týkajících se postavy hráče a manipulace s nimi. Druhá zodpovědnost je obsluha vstupů od hráče. Tato zodpovědnost vychází přímo z chovaní Unreal Enginu, který za normálních okolností předává vstupy aktuálně kontrolovanému objektu.

Zaměřovaní

Zaměřovaní je řešeno pomocé třídy `Character_CameraBP`, která modifikuje chovaní třídy `PlayerCameraManager`. Tato třída vrací lokaci, rotaci a zorný uhel, který se nastaví kameře. Při zaměřovaní dochází ke změně pozice kamery. Při střelbě od boku se kamera nachází ve výchozí pozici, což je v hlavě postavy mezi očima. Každá zbraň má definovaný socket, kam se přesune kamera při míření. Nová pozice kamery se nachází v místě socketu a rotace kamery je shodná s rotací původní kamery, aby nedošlo ke změně směru, kterým postava míří. Během procesu zaměřování se pomocí lineární interpolace dopočítává aktuální lokace kamery. Tato interpolace navozuje pocit, ze postava postupně zvedá zbraň od boku k ramenu a zaměřuje.

Oživení hráče

Ke „smrti“ hráče může dojít pouze v případě, kdy postava hráče vypadne z mapy. V tom případě dojde k teleportaci hráče na místo, kde se nacházel před pádem. Kvůli tomuto je nutné ukládat poslední bezpečnou pozici hráče, na které se nacházel, než vypadl z mapy.

Ukládání poslední validní pozice je řešeno pomocí eventu `SaveSafeLocation`, který je vidět na obrázku 2.1. Po zavolání eventu dojde ke kontrole, jestli postava hráče padá. Pokud ne, do proměnné `SafeLocation` se uloží aktuální pozice hráče. Pokud postava padá,

hodnota se nepřepisuje. V obou případech se následně pomocí funkce `Delay` provede čekání na dobu jedné sekundy a poté se znovu provede zavolání eventu `SaveSafeLocation`.

4.2 Třída zbraně

Zbraně jsou realizovány pomoci dědičnosti. Abstraktní třida `BP_Gun` implementuje funkce jako je střelba a zpětný ráz a řeší manipulaci s proměnnými. Odvozené třídy `BP_AK47` a `BP_M4` mají na starost práci s modely doplňků a zbraně samotné.

Výměna doplňků

Zbraň obsahuje proměnnou datového typu enum (v rámci třídy BP_Gun), ve které je uložen aktuálně vybraný doplněk dané kategorie. Třídy `BP_AK47` a `BP_M4` obsahují pro každý doplněk socket, kam se vybraný static mesh nebo skeletal mesh umístí.

Protože pro jednu kategorii doplňků je k dispozici více socketů, může dojít k situaci, kdy zbraň bude mít nasazeno více modifikací přes sebe. Je pouze na vývojáři, aby zabránil možnosti výskytu této situace. Manipulace s doplňky je zajištěna pomoci tří funkci: `SetMuzzleAttachment`, `SetGripAttachment` a `SetScopeAttachment`, které řeší doplněk na hlavni, pod hlavní a zaměřovač. Všechny tři funkce fungují na stejném principu, pouze pracují s jinými daty.

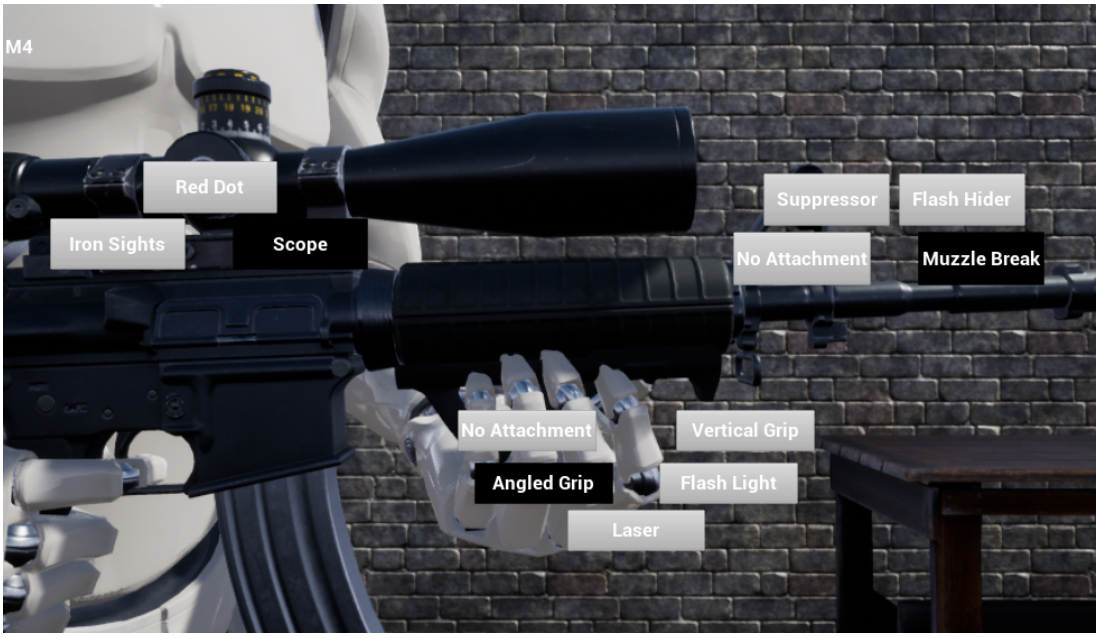
Funkce nastavující doplněk

Funkce `SetMuzzleAttachment` je implementována v třidě `BP_Gun`, kde se provádí nastavení aktuálně vybraného doplňků zbraně. Třídy `BP_AK47` a `BP_M4` funkci překrývají a řeší fyzickou změnu doplňků (static mesh a skeletal mesh).

Funkce dostane od uživatelského rozhraní pro modifikaci doplňků jako parametr doplněk, který má nastavit. Tento parametr je typu Enum. Nejprve dojde k zavolání rodičovské funkce, kde se provede nastavení proměnné reprezentující aktuálně vybraný doplněk. Následně se zavolá funkce `RemoveMuzzleAttachment`, která odstraní static mesh nebo skeletal mesh všech doplňků této kategorie. Díky tomuto odstranění nemůže dojít k existenci více doplňků stejné kategorie naráz. Následně se nastaví static nebo skeletal mesh podle vstupního parametru funkce.

UI pro volbu doplňků

UI pro volbu doplňků je implementováno pomocí widget třídy `Attachments`. Tato widget třída po stisknutí klávesy pro modifikaci (klávesa X) vytvoří tři skupiny tlačítek, pomocí kterých se vybírá doplněk a kamera se posune kolmo na pravou stranu zbraně. Tento widget se vykresluje přes aktuální scénu, uprostřed obrazovky je tedy vidět aktuálně modifikovaná zbraň. UI pro modifikaci je zobrazeno na obrázku 4.1. Každá skupina tlačítek nastavuje doplněk pro určitou kategorii doplňků. Každá skupina tlačítek provádí volání příslušné funkce pro nastavení doplňků (viz 4.2). Každé tlačítko ze stejné skupiny při stisknutí zavolá stejnou funkci ale s jiným parametrem.



Obrázek 4.1: Menu pro úpravu zbraní implementované v této práci

4.3 Nabíjení zbraně

Nabíjení se, stejně jako všechny vstupy od hráče, zpracovává v třídě `Ue4ASP_Character`. Po stisknutí klávesy „R“ proběhne kontrola, zda hráč drží nějakou zbraň. Pokud ano, dojde k zavolaní eventu `Reload` třídy `BP_Gun`. Event jako první krok provede kontrolu, zda má zbraň dostatek nábojů pro nabití (stačí alespoň jeden náboj). Pokud podmínka platí, následuje funkce `Do Once`, která zajistí, že část kódu, která za ní následuje se provede pouze jednou, dokud nedojde k jejímu resetovaní. Toto znemožní hráči znovu nabíjet v rámci už probíhajícího nabíjení. Následně se zakáže střelba. Dále se kontroluje, zda má zbraň dostatek munice pro nabití celého zásobníku, nebo jenom části při nedostatku munice. Počet nabíjených nábojů se odečte od kapacity nábojů. Následně se spustí animace nabíjení a pomocí funkce `Delay` se počká na její dokončení (přibližně dvě sekundy). Tato animace je důvod, proč je nutné zakázat střelbu a opakované nabíjení. Po skončení animace se provede reset funkce `Do Once` a povolí se střelba. Následně se provede kontrola, zda před nabíjením byl náboj v nábojové komoře, pokud ano, přičte se k počtu nabíjených nábojů číslo jedna a toto číslo se následně nastaví jako aktuální počet nábojů.

Animace

Animace nabíjení zbraně pouze budí dojem výměny starého zásobníku za nový, reálně se však zásobník ve zbrani nemění. Animace výměny zásobníku probíhá v rámci čtyř fází.

První fáze nastává, když se ruka postavy hráče přiblíží k zásobníku ve zbrani. Poté, co postava v rámci animace uchopí zásobník, dojde k nastavení viditelnosti zásobníku pomocí funkce `HideBonebyName` na hodnotu False a pomocí funkce `SpawnActorfromClass` dojde k vytvoření nového zásobníku, který se pomocí funkce `AttachActorToComponent` připojí do ruky postavy. Díky tomu, že se tyto akce provedou naráz, působí to dojmem, jako kdyby postava vzala zásobník ze zbraně do ruky. Nově vytvořený zásobník má vypnuté kolize i fyziku, aby se mohl připojit do ruky postavy.

Druhá fáze nastává, když se ruka se zásobníkem dostane do nejnižšího bodu animace (kdy je ruka nejblíž zemi). V tomto okamžiku dojde k zapnutí kolizí a simulace fyziky pro zásobník v ruce. Následkem toho zásobník z ruky vypadne na zem.

Třetí fáze začíná, když postava v rámci animace sahá pro nový zásobník (který se nachází v oblasti zad). V rámci této fáze se vytvoří nový zásobník, který se připojí do ruky. Zásobník má opět vypnuté kolize i fyziku.

Poslední fáze nastává při „vkládání“ nového zásobníku do zbraně. V okamžiku vložení dojde pomocí funkce `DestroyActor` ke zničení zásobníku v ruce a viditelnost původního zásobníku ve zbrani se pomocí funkce `UnHideBonebyName` nastaví na True.

Původní zásobník ve zbrani zbraň nikdy neopustil. Zásobník, který je odhozen na zem je nově vytvořený objekt a zásobník, který se do zbraně „vkládá“ je zničen. Díky dobrému načasování všech funkcí v rámci animace však dochází k iluzi výměny starého zásobníku za nový.

4.4 Zpětný ráz

Jak už bylo zmíněno v sekci 3.8.1, zpětný ráz je imitován pomocí změny parametrů hráčovy kamery. K výpočtu hodnoty, o kterou se parametr změní, se používá vzorec (3.1).

Hodnoty modifikátorů

Hodnoty modifikátorů jsou zvoleny tak, aby byl v co největší míře vidět rozdíl mezi jednotlivými doplňky a režimy střelby. Hodnoty byly hledány experimentálně a nemají žádnou oporu v reálnem světě. V tabulkách 4.1 a 4.2 jsou uvedeny hodnoty modifikátorů na základě vstupních parametrů. Vliv doplňků na modifikátor *attachments* je popsán v sekci 3.6.

Tabulka 4.1: Vliv délky dávky na modifikátor *delka*.

Délka dávky	Modifikátor
1-2	0,1
3-7	0,25
8-11	0,5
12+	1

Tabulka 4.2: Vliv způsobu míření na modifikátor *aim*.

Režim míření	Modifikátor
ADS	0,1
Hip-Fire	1

4.5 Záblesk při výstřelu

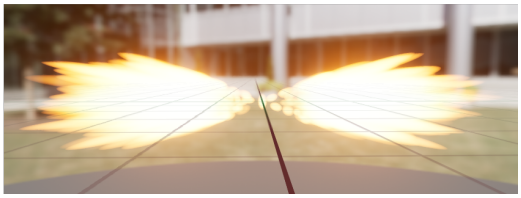
Záblesk při výstřelu (anglicky Muzzle Flash) je řešen pomocí částicového systému Niagara. Niagara používá dva druhy objektů. Prvním druhem je `Niagara Emitter`, který definuje chování jednoho zdroje částic. Druhým typem je `Niagara System`, který kombinuje několik

emitorů do výsledného efektu. V rámci `Niagara System` se dají měnit parametry jednotlivých emitorů, které ho tvoří.

Práce obsahuje tři objekty `Niagara Emitter`. Jedná se o emitor, který vypouští částice pouze jedním směrem, druhý vypouští částice do celého okolí a poslední generuje částice, které se chovají jako kouř. Tyto emitory jsou následně zkombinovány do čtyř objektů `Niagara System`. Každý doplněk na hlavni má tedy svůj vlastní systém.

- Bez doplňku na zbrani – Jeden všesměrový emitor 4.5
- Muzzle Break – Dva směrové emitory, které míří doprava a doleva 4.2
- Flash Hider – Osm směrových emitorů, které jsou rozděleny v pravidelných rozestupech v kruhu 4.3
- Tlumič – Jeden emitor kouře 4.4

Každý systém (včetně tlumiče) navíc obsahuje i jeden emitor kouře, který generuje malé množství kouře po krátkou doby po výstřelu.



Obrázek 4.2: Muzzle Break

Obrázek 4.3: Flash Hider

Obrázek 4.4: Tlumič

Obrázek 4.5: Bez doplňku

4.5.1 Stopy po střelách

Stopy po střelách jsou realizovány pomocí jednoduché textury, která se nanáší na zasažený objekt v místě zásahu. Při zásahu hrany objektu se textura zatočí na za hranu. Na obrázku 4.6 je zobrazen zásah rovné plochy a na obrázku 4.7 je zobrazen zásah hrany.

Obrázek 4.6: Stopa po střele na rovném povrchu

Obrázek 4.7: Stopa po střele na hraně objektu

4.6 NPC

Non-player charactery (NPCs) v práci plní roli mobilních terčů. V levelu se vyskytují čtyři roboti, kteří se pohybují po paralelních linkách různou rychlostí. Roboti jsou po zásahu zničení a do tímto způsoben uvolněné linky se vytvoří nový robot.

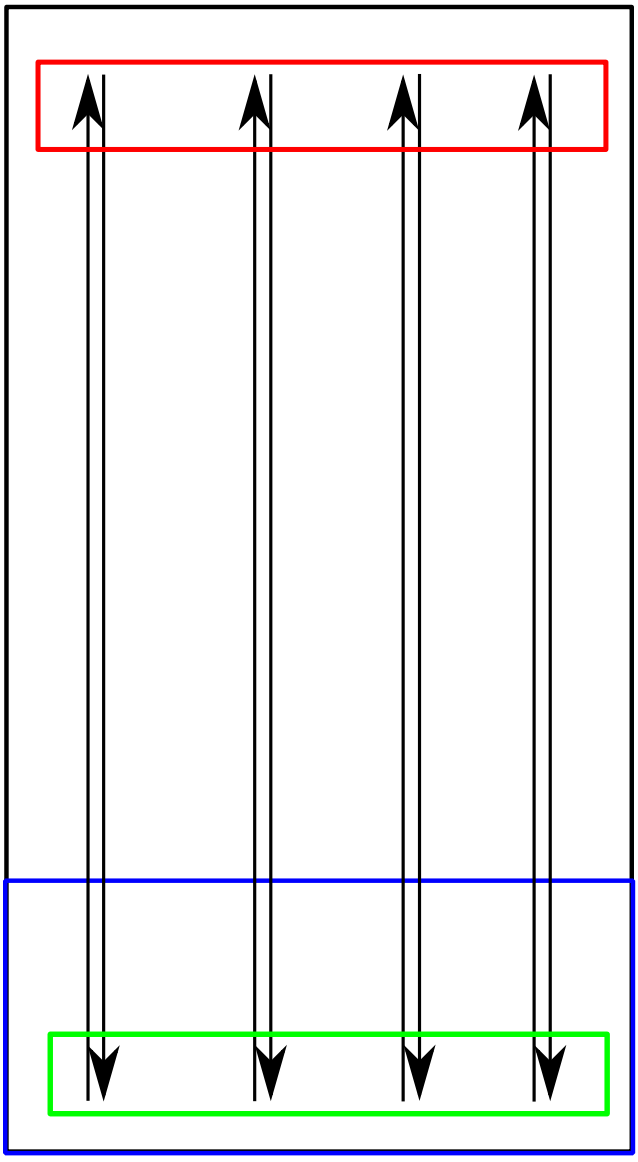
Generování (spawnování) NPC je řešeno pomocí třídy `AI_Spawner`, která při spuštění levelu vytvoří čtyři roboty třídy `AI` a pro každého zavolá funkci `Patrol`. Pro generování se využívá objektů tříd `AI_SpawnPoint` a `AI_PatrolPoint`, kde každý obsahuje pouze jednu proměnnou typu vector, která reprezentuje souřadnice ve světě. První objekt reprezentuje místo, kde se robot objeví a druhá místo, kam robot poběží.

Generování funguje jako cyklus, kde se v každé iteraci vytvoří jeden robot. Robot se vytvoří na pozici dané hodnotou v objektu `AI_SpawnPoint` a přičte se k němu offset ve směru „Y“ vynásobený indexem robota. Tato hodnota se následně uloží do vytvořeného robota jako proměnná `SpawnPoint`. Stejným způsobem se vypočítá hodnota proměnné `PatrolPoint`. Následně se pomocí indexu vypočítá rychlost, která se použije pro funkci `SetMaxWalkSpeed`. Nakonec se pro vytvořeného robota zavolá funkce `Patrol`.

Patrol

Pohyb NPC je řešen pomocí vzájemného volání dvou funkcí `AI MoveTo`. Nejprve se NPC pohybuje k pozici dané parametrem `PatrolPoint`, po dosažení tohoto bodu se zavolá funkce `AI MoveTo` na pozici danou parametrem `SpawnPoint`. Po dosažení tohoto bodu se cyklus

opakuje. Na náčrtku 4.8 jsou naznačeny trasy robotů. Zeleně je naznačen `SpawnPoint` a červeně `PatrolPoint`. Oblast v modrém obdélníku značí místo, kam se nedá střílet.



Obrázek 4.8: Naznačení tras, po kterých se roboti pohybují

Ničení

Po zásahu robota nedochází k jeho zničení, ale je pouze přesunut pomocí funkce `Teleport` na `SpawnPoint`. Následně se mu funkcí `Set Actor Hidden In Game` nastaví viditelnost na hodnotu False. Funkce `Stop Active Movement` mu zruší veškerý pohyb. Následuje funkce `Delay`, která čeká dvě sekundy (simuluje tím oživování) a následně nastaví robota jako viditelného. Nakonec se zavolá funkce `Patrol`, která znovu zahájí pohyb.

Kapitola 5

Testování

V této kapitole je popsáno testování vytvořeného herního dema. Testování je rozděleno do dvou částí. V první části je testován dojem ze hry pomocí skupiny testerů. Druhá část je zaměřuje na testovaní výkonu. Z důvodu relativně malého počtu testerů byla zpětná vazba od testerů získána pomocí hovoru na platformě Discord.

5.1 Testeři

Testování probíhalo pomocí skupiny pěti testerů. Všichni testeři měli zkušenosti s FPS hrami. Každý z testerů měl zkušenost s alespoň jedním dílem série Battlefield (BF) a Call of Duty (CoD). Tři testeři mají zkušenost s titulem Rainbow Six Siege (R6) a dva se hrou ARMA III. Přehled testerů a her, které hrají, je zobrazen v tabulce 5.1.

Tabulka 5.1: Přehled testerů a her, které hrávají

Tester	BF	CoD	R6	ARMA
Tester 1	✓	✓	✓	✓
Tester 2	✓	✓	✓	×
Tester 3	✓	✓	×	×
Tester 4	✓	✓	✓	×
Tester 5	✓	✓	×	✓

5.2 Průběh testovaní

Testování bylo řešeno iterativním způsobem. Díky tomuto mohly být oprávněné připomínky testerů analyzovány a chování hry upraveno. Mezi tímto způsobem změněné chování patří změna rychlosti projektilu. Konsensus mezi testery na otázku rychlosti projektilu byl: „nedá se s tím trefit“. Po zvýšení rychlosti projektilu tyto stížnosti ustoupily.

Otázky pro testery

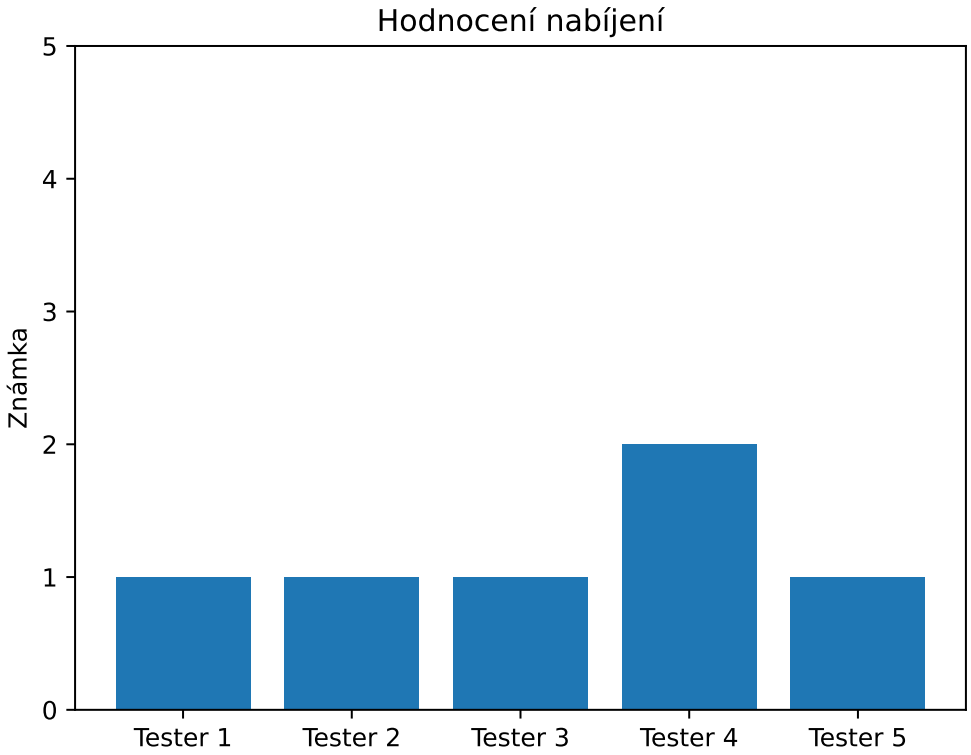
Protože byla data od testerů získávána pomocí hovoru, nebylo nutno vytvářet dotazník. V první fázi testování bylo řešeno, zda se testeři setkali s nějakým neočekávaným chováním a zda by nějakou mechaniku implementovali jinak. V druhé části byly pokládány otázky na hodnocení některých mechanik. Příklady otázek druhé části jsou:

- Jaká část hry se vám líbí nejvíce? Oznámkujte.
- Jaká část hry se vám líbí nejméně? Oznámkujte.

5.3 Výsledky testovaní

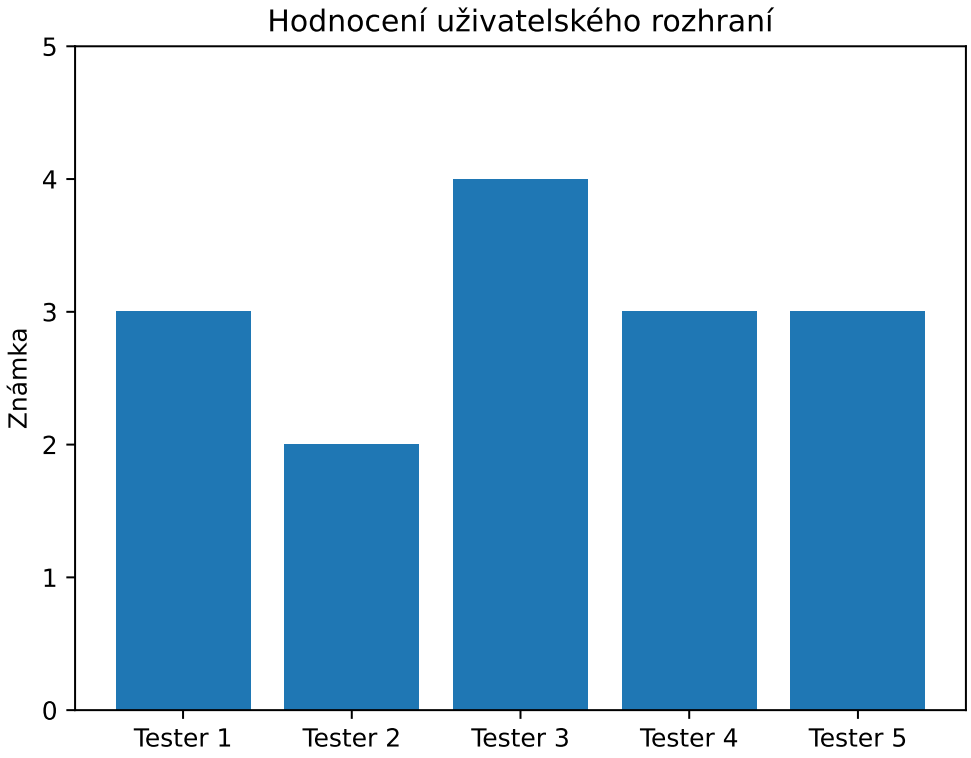
Další kritizovanou mechanikou bylo navázání režimu střelby na zbraň. Ohledně této kritiky mezi testery nebyl konsensus. Stížnost byla pouze od testerů, kteří hrají jenom Battlefield a Call of Duty. Tester hrající Rainbow měl k této problematice neutrální postoj. Zbylí dva testeři hrající i Armu naopak navázání režimu střelby na zbraň chválili, protože to je reálné chování.

Na otázku ohledně nejlepší a nejhorší části práce panovala mezi testery shoda. Jako nejlepší hodnotili animaci nabíjení a odhazování zásobníku a jako nejhorší hodnotili uživatelské rozhraní pro modifikaci zbraně. Na obě tyto části bylo vytvořeno hodnocení pomocí známek jako ve škole.



Obrázek 5.1: Graf znázorňuje hodnocení nabíjení jednotlivými testery. Známka 1 je považována ze nejlepší a 5 za nejhorší. Průměrná známka: 1,2

Z hodnocení 5.1 vyplývá, ze mechanika nabíjení je podařená a od testera, který jako jediný tuto mechaniku hodnotil známkou „2“, si vysloužila kritiku, že při nabíjení prázdné zbraně nedochází k natažení závěru. Tato mechanika je popsána v sekci 3.9.



Obrázek 5.2: Graf znázorňuje hodnocení uživatelského rozhraní pro modifikaci zbraní jednotlivými testery. Známka 1 je považována ze nejlepší a 5 za nejhorší. Průměrná známka: 3,0

Oproti tomu hodnocení 5.2 pro uživatelské rozhraní pro modifikaci zbraní dopadlo hůř. Testeři kritizovali vzhled rozhraní, ale funkčnost tlačítek jako metody pro volbu doplňků hodnotili pozitivně.

5.4 Výkon

Po stránce výkonu byl testován vliv počtu nábojnic na výkon hry (měřen ve snímcích za sekundu (FPS)). Testování bylo provedeno na dvou systémech. Prvním je desktop a druhým je notebook. Specifikace obou systémů je uvedena v tabulce 5.2.

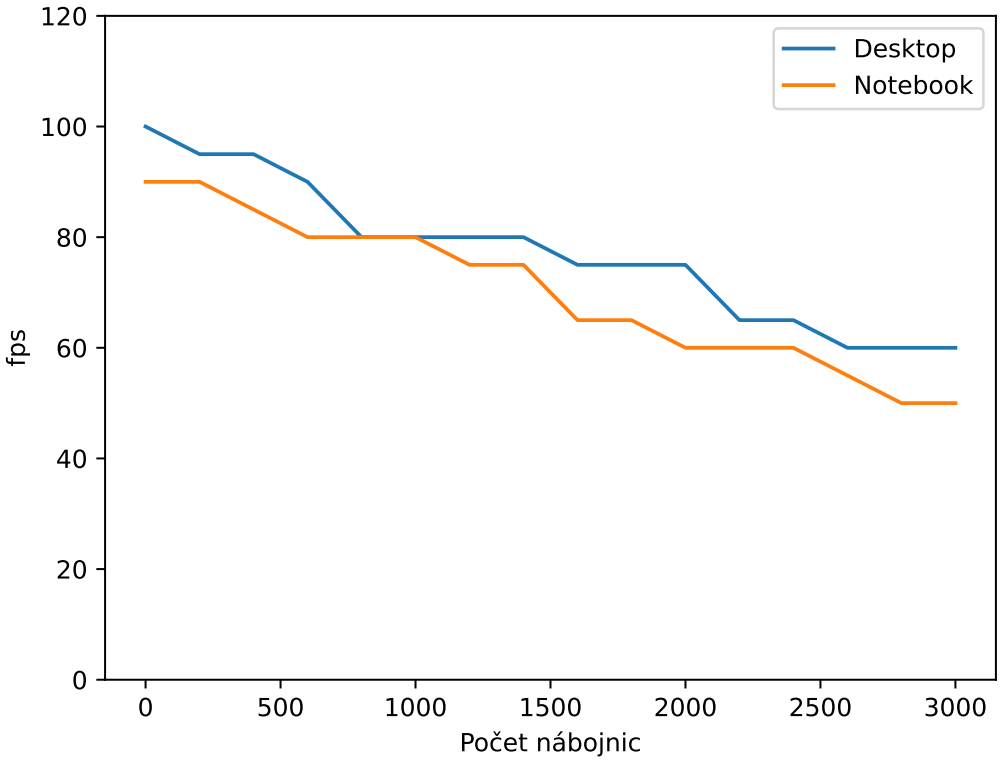
Tabulka 5.2: Hardware testovacích systémů

Komponenta	Desktop	Notebook
CPU	R5 1600	i5 8300H
GPU	RX 480 8GB	GTX 1060 6GB
RAM	16GB	16GB
SSD	SATA	SATA

Pro testování byla z důvodu úspory času a lepšího pohodlí při testování použita mírně upravená verze hry. Při výstřelu došlo k vytvoření stovky nábojnic a kapacita zásobníku byla zvednuta na tisíc nábojů. Celkem je tedy možno vygenerovat sto tisíc nábojnic s jedním zásobníkem. Měření výkonu bylo velmi komplikované z několika důvodů. Prvním důvodem byla velká nestabilita výkonu během výstřelu. Při výstřelu došlo k vygenerování sta nábojnic, pro které se simuluje fyzika. Po několik sekund po výstřelu se výkon držel kolem hranice 20 FPS. Následně se výkon vrátil k původní hodnotě. Pro účely testování byla použita až tato ustálená hodnota. Druhým problémem byl velmi velký rozdíl výkonu mezi jednotlivými testovacími běhy. Kvůli tomu byl použit aritmetický průměr z deseti běhu.

Testovaní bylo provedeno pouze do hodnoty 3 000 nábojnic. U vyšších hodnot docházelo k pádu enginu. Pouze v jednom z deseti běhů se podařilo změřit hodnotu 3 500 a vzhledem k velké odchylce mezi jednotlivými běhy je tato hodnota nespolehlivá. K vizualizaci výsledků byla použita Python knihovna Matplotlib. Na grafu 5.3 je znázorněna závislost výkonu na počtu nábojnic.

Na grafu je vidět, že výkon obou systémů je podobný. I škálování výkonu s počtem nábojnic je pro oba systémy srovnatelné. Problém nastává u hranice 3 000 nábojnic, kde jsou velmi časté pády enginu. Pravděpodobnou příčinou tohoto chování je simulace fyziky pro nábojnice. Simulace fyziky je potřebná, aby nábojnice padala. V rámci optimalizace by bylo možné vypnout simulaci fyziky pro nábojnice, které se nachází na zemi.



Obrázek 5.3: Graf znázorňuje závislost výkonu hry [fps] na počtu nábojnic ve scéně.

Kapitola 6

Závěr

Cílem práce bylo vytvoření herního dema, které se zaměřuje na herní mechaniky palných zbraní a na interakci s prostředím. V první řadě bylo nutné nastudovat fungovaní Unreal Enginu a seznámit se s vývojovým prostředím. Dále bylo pro tvorbu dema nutno nastudovat animace a inverzní kinematiku. Kromě věcí týkajících se IT bylo nutno nastudovat i věci mimo IT obor, jako jsou palné zbraně, konkrétně zpětný ráz. Na základě těchto teoretických poznatků byly navrženy herní mechaniky, které byly poté implementovány.

Výsledkem práce je herní demo, které obsahuje jednu úroveň. V úrovni jsou přítomny všechny navržené a implementované mechaniky jako je střelba, různý záblesk při výstřelu a různý zpětný ráz při různých doplňcích, vyhazování nábojnic při střelbě a odhazování zásobníku při nabíjení. Do práce byla integrována zpětná vazba od testerů a došlo díky tomu k eliminaci několika chyb a také k úpravě několika herních mechanik.

Práce by se dala vylepšit vytvořením systému pro destrukci okolí, kdy by například při střelbě na dřevěný povrch docházelo k odlamovaní třísek, které by se rozletěly do okolí. Dalším potenciálním rozšířením by mohlo být vytvoření více animací pro nabíjení zbraní. Toto se týká hlavně vytvoření dvou verzí animace nabíjení, kde jedna animace pouze vyměňuje zásobník a druhá animace kromě výměny zásobníku provede i natažení závěru.

Literatura

[1] *Balancing blueprint and c++* [online]. 2022 [cit. 2022-05-03]. Dostupné z: https://docs.unrealengine.com/4.27/en-US/Resources/SampleGames/ARPG/BalancingBlueprintAndCPP/.

[2] *M4 Carbine* [online]. 2022 [cit. 2022-05-03]. Dostupné z: https://www.https://www.military.com/equipment/m4-carbine.

[3] Clement, J. *U.S. most popular video game genres 2018* [online]. 2019 [cit. 2022-04-03]. Dostupné z: https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/.

[4] Colzato, L., Van Leeuwen, P., Van Den Wildenberg, W. a Hommel, B. DOOM'd to switch: superior cognitive flexibility in players of first person shooter games. *Frontiers in Psychology*. 2010, sv. 1. DOI: 10.3389/fpsyg.2010.00008. ISSN 1664-1078. Dostupné z: https://www.frontiersin.org/article/10.3389/fpsyg.2010.00008.

[5] Dobrilova, T. *How Much Is the Gaming Industry Worth in 2022?* [online]. 2022 [cit. 2022-05-05]. Dostupné z: https://techjury.net/blog/gaming-industry-worth/.

[6] Gregory, J. *Game engine architecture*. 2. vyd. CRC Press, 2014. ISBN 978-1-4665-6006-2.

[7] Hall, M. J. Measuring felt recoil of sporting arms. *International Journal of Impact Engineering*. 2008, sv. 35, č. 6, s. 540–548. DOI: https://doi.org/10.1016/j.ijimpeng.2007.03.007. ISSN 0734-743X. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0734743X07000619.

[8] Hartink, A. E. *Complete Encyclopedia of Automatic Army Rifles*. 1. vyd. Hackberry, 2001. ISBN 978-1-9310-4004-4.

[9] Karis, B. *Nanite SIGGRAPH* [online]. 2021 [cit. 2022-04-03]. Dostupné z: https://advances.realtimerendering.com/s2021/Karis_Nanite_SIGGRAPH_Advances_2021_final.pdf.

[10] Kofinas, N., Orfanoudakis, E. a Lagoudakis, M. G. Complete Analytical Forward and Inverse Kinematics for the NAO Humanoid Robot. *Journal of Intelligent & Robotic Systems*. 2014, sv. 77, č. 2, s. 251–264. DOI: 10.1007/s10846-013-0015-4.

[11] McNab, C. *The M4 Carbine*. 1. vyd. Bloomsbury Publishing, 2021. ISBN 978-1-4728-4228-2.

[12] Mehta, A. a O'Brien, R. *Built for Builders: The Story of AWS and Open 3D Engine – Developer Preview | Amazon Web Services* [online]. 2021 [cit. 2022-01-16]. Dostupné z: https://aws.amazon.com/blogs/gametech/open-3d-engine/.

[13] Prasanta Kumar Das, D. P. D. *Science and Engineering of Small Arms*. 1. vyd. CRC Press, 2021. ISBN 978-1-0320-5824-5.

[14] Schreier, J. *The Story Behind Mass Effect: Andromeda's Troubled Five-Year Development* [online]. 2017 [cit. 2022-04-03]. Dostupné z: https://kotaku.com/the-story-behind-mass-effect-andromedas-troubled-five-1795886428.

Příloha A

Assety

Tato kapitola obsahuje seznam všech assetů, které byly v této práci použity.

1. https://www.pngwing.com/en/free-png-beqwu
2. https://www.kindpng.com/imgv/oxhmi_realistic-bullet-hole-png-transparent-png/
3. „SciFi Light 02“ (https://skfb.ly/6nBOJ) by inuhitman
4. „Tactical Laser Gun Sight“ (https://skfb.ly/otWtH) by trolosqlfod
5. „Tactical flashlight w/ rail mount“ (https://skfb.ly/RVA9) by HELL
6. „Wooden Table“ (https://skfb.ly/6QXAJ) by shedmon
7. „Old Ammo Crate“ (https://skfb.ly/6xzJS) by SuperMopsek
8. „Flash Hider1“ (https://skfb.ly/6X6Jw) by FFeller
9. „Muzzle break“ (https://skfb.ly/6X8Ct) by FFeller
10. „SR-47“ (https://skfb.ly/6QWwE) by frontliner36
11. „FPS Weapon Bundle“ by Deadghost Interactive (https://www.unrealengine.com/marketplace/en-US/product/fps-weapon-bundle)
12. „Animation Starter Pack“ by Epic Games (https://www.unrealengine.com/marketplace/en-US/product/animation-starter-pack)