Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vývoj 3D herní aplikace v Unity s zaměřením na externí balistiku projektilu

Matěj Klouda

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Matěj Klouda

Informatika

Název práce

Vývoj 3D herní aplikace v Unity s zaměřením na externí balistiku projektilu

Název anglicky

3D development of videogame in Unity focused on external ballistics of the projectile

---

Cíle práce

Hlavním cílem této bakalářské práce je stanovení, jaké fyzikálních jevy se implementují do žánrů populárních FPS her a jaké se záměrně vynechávají.

Prvním vedlejší cíl je popis, proč tomu tak je a jak jednotlivé vlivy ovlivňují samotnou hratelnost a plynulost hry.

Druhým vedlejším cílem této bakalářské práce je vyvinout prototyp 3D herní aplikace v Unity v němž budou použity mechaniky simulující externí balistiku vystřeleného projektilu reálného světa a zahrnout do práce popis postupu tohoto procesu.

Posledním dílčím cílem bude také zahrnutí reakce prostředí na dopad střely a popis postupu vytváření řešení.

Metodika

V teoretické části bude popsán videoherní žánr FPS her a jejich subkategorie. Následně bude představeno několik her z toho žánru, které svými principy a mechanikami žánr významně ovlivnily.

Dále bude popsána externí balistika vystřeleného projektilu a budou popsány způsoby, jakými se jednotlivé fyzikální jevy, jenž let projektilu ovlivňují, modelují v herních aplikací.

Pomocní informací z teoretické části bude navrhnut a implementován algoritmus jenž bude řešení externí balistiky vystřeleného projektilu v prostředí Unity.

Součástí praktické části bude i navrhnutí mechaniky nárazu střely pomocí kolizních systémů a jeho působení na různé objekty.

Také bude vytvořena herní scéna ve které, bude možné všechny implementované věci otestovat.

Výsledky budou porovnány s několika videohrami a bude určeno, jaký typ modelování a jaké vlivy se v jednotlivých žánrech používají.

Zhodnocení úspěšnosti práce, navrhnutí dalších možností pro vylepšení modelu balistiky projektilů v herních aplikacích a stanovení vzorců, kterými se jednotlivé žánry řídí.

Doporučený rozsah práce

35-40 stran

Klíčová slova

Unity, Projektil, 3D FPS, trajektorie střely, působení nárazu.

Doporučené zdroje informací

PENG, XIA, 2014. 3D Game Development with Unity A Case Study: A First-Person Shooter (FPS) Game. Helsinky. Thesis. Helsinki Metropolia University of Applied Sciences. Vedoucí práce Markku Karhu.

WARLOW, T. A., c2005. Firearms, the law, and forensic ballistics. 2nd ed. Boca Raton, Fla.: CRC Press. ISBN 0415316014.

WOLF, Mark J. P. a Bernard PERRON, 2014. The Routledge companion to video game studies. New York, NY: Routledge. ISBN 9781138657052.

ZHANG, Y. F., WANG, C. L., & ZHU, D. W. (2017). FPS Game Design and Implementation Based on Unity3D. Destech Transactions on Engineering and Technology Research. ISBN: 978-1-60595-345-8

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

Ing. Tomáš Benda

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 4. 9. 2023

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 3. 11. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 15. 03. 2024

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci " Vývoj 3D herní aplikace v Unity s zaměřením na externí balistiku projektilu" jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2024 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Tomáši Bendovi za jeho odborné vedení, cenné rady a neustálou podporu během práce na této bakalářské práci. Dále bych chtěl poděkovat své přítelkyni a rodině za jejich nekonečnou podporu, trpělivost a povzbuzení, které mi poskytovali během celého mého studia. Vaše láska, pochopení a motivace byly klíčovými pilíři mého úsilí a úspěchu. Další poděkování patří hráčům a moderské komunitě, jejichž poznatky a zkoumání mi v mém výzkumu značně pomohli. Speciální poděkování patří také paní Zdeně, která mi byla velkou inspirací a ukázala mi, že se nikdy nemám vzdávat, bez ohledu na výzvy, kterým čelím.

Vývoj 3D herní aplikace v Unity s zaměřením na externí balistiku projektilu

Abstrakt

Tato bakalářská práce se zaměřuje na vývoj 3D herní aplikace v Unity, s důrazem na realistickou simulaci externí balistiky projektilů. K dosažení těchto cílů bylo nezbytné provést důkladnou analýzu fyzikálních jevů, jako jsou gravitace, odpor vzduchu a počáteční rychlost výstřelu, které mají klíčový vliv na trajektorii projektilů ve virtuálním prostředí. Praktická část práce zahrnovala vývoj prototypu hry, kde byly tyto principy implementovány a následně testovány. Zpětná vazba byla klíčová pro iterativní proces zdokonalování a finální optimalizaci aplikace. Výsledky této práce nejen ukázaly, že integrace realistických balistických modelů může značně obohatit herní zážitek, ale také poskytly cenné informace o preferencích hráčů a technických výzvách spojených s vývojem 3D her, které mohou sloužit jako výchozí bod pro další výzkum v této oblasti. Práce také stanovila pravidla pro použití jednotlivých faktorů ovlivňujících externí balistiku v závislosti na žánru hry a celkovém dalším vývojářském postupu. Navíc práce odhalila, že přesné modelování balistiky může mít významný dopad na strategické rozhodování hráčů, což dodává další rozměr strategické hloubce hry.

Klíčová slova: Unity, FPS, 3D, Externí balistika, Arma 3, Battlefield V, Call of Duty, simulace, projektil

3D development of videogame in Unity focused on external ballistics of the projectile

Abstract

This bachelor thesis focuses on the development of a 3D game application in Unity with an emphasis on realistic simulation of external projectile ballistics. In order to achieve these goals, it was necessary to perform a thorough analysis of physical factors such as gravity, air resistance and initial shot velocity, which have a key influence on the trajectory of projectiles in a virtual environment. The practical part of the work involved the development of a game prototype where these principles were implemented and subsequently tested. Feedback was crucial to the iterative refinement process and final optimization of the application. The results of this work not only showed that the integration of realistic ballistic models can significantly enrich the gameplay experience, but also provided valuable insights into player preferences and technical challenges associated with 3D game development that can serve as a starting point for further research in this area. The work also established guidelines for the use of different factors affecting external ballistics depending on the game genre and the overall further development process. In addition, the thesis revealed that accurate ballistics modelling can have a significant impact on players' strategic decision making, adding another dimension of strategic depth to the game.

Keywords: Unity, FPS, 3D, external ballistics, Arma 3, Battlefield V, Call of Duty, simulation, projectile

Obsah

Úvod

V současné době, kdy videoherní průmysl prochází dynamickým vývojem, se externí balistika projektilů ve videohrách stává klíčovým prvkem, který ovlivňuje jak realismus, tak hratelnost. Simulace střelby byla od počátku herního průmyslu velkým tématem a snaha o co nejrealističtější chování letících projektilů byla na počátku herního průmyslu jeden z hlavních cílů vývojářů. S rychlým technickým pokrokem v hardwarové technologii a exponenciálním nárůstem výpočetní kapacity dochází k tomu, že simulace ve hrách mohou být stále realističtější a v některých ohledech velmi přesně kopírovat fyziku reálného světa. Tato práce se zaměřuje na zkoumání, jak hyperrealistické chování ve hrách ovlivňuje vnímání a přijetí hráči, a také odpovídá na otázku, proč se přístup největších herních studií, které prodají tisíce kopií her každý rok, velmi liší. Proč některá studia do svých projektů implementují skoro všechny faktory reálného světa a proč jiní vývojáři řeší pouze některé z vlivů, nebo na externí balistiku nekladou v průběhu absolutně žádnou pozornost.

Ve světě, kde je střelba mimo počítačové simulace velmi komplexní a vyžaduje rozsáhlou praxi, se práce ptá, jak může být tento aspekt realisticky implementován do videoher, aniž by to negativně ovlivnilo hratelnost a zda tyto změny jsou žádané. Práce taktéž přináší důkladnou analýzu jednotlivých přístupů k implementování střelby do vlastního projektu, včetně jejich obtížnosti a zachycování potřebných dat zásahů. Stanovuje vliv fyzikálních principů ovlivňujících trajektorii projektilů a jejich aplikaci v herním designu. Kombinuje teoretický výzkum s praktickým vývojem prototypu hry v Unity, kde jsou tyto principy testovány a hodnoceny na základě naměřených výsledků.

Tato práce si klade za cíl nejen přispět k teoretickým poznatkům v oblasti herního designu, ale také poskytnout praktická doporučení pro vývojáře her. Zaměřuje se na unikátní překlenutí mezi informatikou, fyzikou a designem her, aby prozkoumala, jak mohou být teoretické poznatky o externí balistice efektivně využity k obohacení herního světa.

Hlavním cílem této práce je tedy zodpovědět otázky o preferencích hráčů ohledně realismu ve hrách a o technologických možnostech současného hardwaru, které by mohly tyto požadavky splnit. Zahrnuje také testování a sběr dat, včetně měření hratelnosti ve speciálně vytvořeném prototypu, aby se získal ucelený pohled na to, jak vývojáři a hráči vnímají realistickou simulaci střelby a jaké aspekty hry jsou pro ně nejdůležitější.

Cíl práce a metodika

1.1 Cíl práce

Hlavním cílem této bakalářské práce je stanovení, jaké fyzikální jevy se implementují do žánrů populárních FPS her a jaké se záměrně vynechávají.
Prvním vedlejším cílem je popis, proč se vůbec některé vlivy vynechávají a jak jednotlivé vlivy ovlivňují samotnou hratelnost a plynulost hry.
Druhým vedlejším cílem této bakalářské práce je vyvinout prototyp 3D herní aplikace v Unity, v němž budou použity mechaniky simulující externí balistiku vystřeleného projektilu reálného světa a zahrnout do práce popis postupu tohoto procesu.
Posledním dílčím cílem bude také zahrnutí reakce prostředí na dopad střely a popis postupu vytváření řešení.

1.2 Metodika

Dále bude popsána externí balistika vystřeleného projektilu a budou popsány způsoby, jakými se jednotlivé fyzikální jevy, jež let projektilu ovlivňují, modelují v herních aplikací.

Pomocí informací z teoretické části bude navrhnut a implementován algoritmus, jenž bude řešení externí balistiky vystřeleného projektilu v prostředí Unity.

Součástí praktické části bude i navrhnutí mechaniky nárazu střely pomocí kolizních systémů a jeho působení na různé objekty.

Také bude vytvořena herní scéna, ve které bude možné všechny implementované věci otestovat.

Výsledky budou porovnány s několika videohrami a bude určeno, jaký typ modelování a jaké vlivy se v jednotlivých žánrech používají.

Zhodnocení úspěšnosti práce, navrhnutí dalších možností pro vylepšení modelu balistiky projektilů v herních aplikacích a stanovení vzorců, kterými se jednotlivé žánry řídí.

2 Teoretická východiska

2.1 Definice žánru First-Person Shooter

Žánr FPS je v Oxfordském slovníku popsán (Oxford Learner's Dictionary FPS, 2000), jako typ videohry, ve které hráč sleduje akci pomocí očí postavy a musí útočit na nepřátele. Tato definice v podstatě vyplývá ze samotného názvu. V češtině by bylo možné tento název přeložit jako střílečka z první osoby. Základní prvek tohoto žánru je, že z ovládané postavy lze vidět pouze ruce a v některých případech trup a nohy. I když má tento žánr nespočet subžánrů, ve většině je hlavní cíl stejný, zlikvidovat nepřátelé a přežít. Jak se postupem času a vývoje her ukázalo, je tento pohled pro střelbu a dosažení výše uvedeného cíle ideální.

Tento žánr lze dále rozdělit na hry pro jednoho hráče, tzv. singleplayer, a hry pro více hráčů, tzv. multiplayer. Drtivá většina AAA her nabízí při zakoupení titulu obě varianty. Singleplayerová kampaň se zápletkou, akčními scénami a ikonickými charaktery je například to, co drželo herní sérii Call of Duty v minulých letech na vrcholu. Tato kampaň má však čas dohrání v řádku několika hodin. Multiplayerová část toto omezení nemá a spousta hráčů jí hraje až do vydání dalšího titulu své oblíbené značky.

Jelikož se až na pár výjimek chování zbraní v multiplayeru nijak zásadně neliší od chování zbraní v singleplayeru, nebude u jednotlivých titulů nikterak rozlišeno.

2.1.1 Popularita žánru

Tento žánr si velmi rychle získal popularitu, která ho neopouští do dnešních dnů, a proto se řadí mezi světově nejhranější. Dne 26.09.2023 je FPS žánr na prvních 50 příčkách statistikách hratelnosti jednotlivých her nabízených službou Steam zastoupen hned 16 krát a to se v této službě nenachází zdaleka všechny nejpopulárnější FPS tituly, protože nějaké jsou k nalezení na své samostatné platformě. (STEAM, 2023)

2.2 Subžánry FPS

Mezi jednotlivými subžánry je markantní rozdíl ve většině herních principů, na kterých hry fungují, a proto v této kapitole budou stanoveny znaky a rozdíly mezi základními subžánry žánry, ve kterých je střelba a tím pádem i externí balistika důležitá.

2.2.1 Tactical FPS

Tento subžánr videoher, také nazývaný jako military simulátor se zaměřuje na vojenskou tématiku a simulaci vojenských operací. Z tohoto důvodu s klade velký důraz nejen na chování zbraní a externí balistiku, ale i na celkovou fyziku ve hře. V této části budou popsány mechaniky a příklady gameplaye primárně z her Hell let Loose, Battlefield a nejvýraznějšího zástupce tohoto žánru, jímž je Arma 3. (TactiGamer, 2005)

Pro tento žánr jsou typické rozlehlé mapy celých krajů, které dosahují až několik kilometrů čtverečních, na kterých se utkává zpravidla vyšší počet hráčů. Základní herní módy jsou obrana/útok na území nebo zabírání bodů na mapě. Celkové tempo hry je pomalejší, avšak v každém utkání lze zažít i velmi akční momenty. Tento žánr je velmi komplexní a nemilosrdný zejména k nováčkům, kterým není odpuštěna sebemenší chyba a začátky se pro ně můžou stát až frustrujícími. (Payne, 2008)

Obtížnost hraní se také odvíjí od řešení střelby a externí balistiky. V naprosté většině her tohoto žánru je používaná metoda projektilu jako objektu, právě kvůli co nejpřesnější simulaci reality. V hrách tohoto typu nedisponuje hráč žádnými informacemi, které se normálně vyskytují na HUD. Mezi tyto informace lze například zařadit minimapu, počet zbývajících nábojů, ale i pro nás zásadní crosshair. Pomocný křížek při míření od boku poskytuje neuvěřitelnou pomoc, jelikož bez něj musíme jen odhadovat, kde přibližně je střed obrazovky a kam tedy střela poletí. Další faktor, kterým tento žánr ztěžuje samotnou hratelnost, je nepřesná střelba od boku a fakt, že kulky díky letovým vlastnostem nelítají přímo do středu obrazovky. Tu tedy lze použít jen na krátkou vzdálenost. Aby bylo zjištěno, kam hlaveň zbraně míří, je nutné podívat přes kovová mířidla zbraně nebo zaměřovač či hledí. (TactiGamer, 2005) (Payne, 2008)

V tomto herním žánru je skoro jisté, že na střelu budou mít vliv alespoň dvě základní síly vnějšího světa. Zmiňovanými vlivy jsou gravitace a odpor vzduchu. Průběžný pád kulky tzv. bulletdrop je pro žánr ikonický a jelikož se většinou hraje na velkých mapách, jde při hraní velmi dobře vidět. V propracovanějších hrách pak může střelu ovlivnit i vítr a tvar krajiny, kterou kulka letí. Těmto vlivům je dán prostor v dalších částech práce. (TactiGamer, 2005)

Samotná střelba ze zbraní je také velmi detailně modelovaná, aby byla co nejrealističtější. S některými zbraněmi je přesná střelba ve vzpřímené pozici téměř nemožná, protože zpětný ráz zbraně je obrovský. Jakmile je však tato zbraň použita v okamžiku, co

námi ovládaná postava leží, stává se ze zbraně velmi účinný nástroj na likvidaci nepřátel, či potlačení útoku. V tomto žánru je skoro vždy pro vývojáře nejvhodnějším řešením použít metodu fyzického projektilu. Na vystřelený projektil ve hře působí skoro všechny vlivy reálného světa. (Payne, 2008)

Mechanika přebíjení zbraně u pistole a u lehkého kulometu zabere úplně jiný čas. Zatím co u pistole stačí jen vyměnit zásobník a zmáčknout pojistku, u lehkého kulometu musí dojít k otevření krytu, vyndání nábojového pásu, odpojení prázdného zásobníku, připojení nového zásobníku, natažení závěru a až pak je možné pokračovat ve střelbě. Rozdíl v délce animací u těchto dvou extrému může dosáhnout i 7 sekund, s čímž musí hráč, který si zbraň zvolí, počítat. Animace na rozdíl od jiných her nejde stopnout, ani přeskočit, takže si hráč musí dobře rozmyslet v jakém momentu přebít zbraň nebo sáhnout po příruční pistoli. (Payne, 2008) (TactiGamer, 2005)

2.2.2 **Action FPS**

Subžánr Action FPS už názvem napovídá, že oproti žánru tactical je o dost rychlejší a intenzivnější. Často se zaměřuje stejně jako předešlý žánr na známé konflikty minulosti i současnosti. Také se však často děj hry odehrává uprostřed fiktivního konfliktu mezi státy nebo v průběhu útoku teroristických organizací.

Cíl tohoto subžánru není co nejvěrohodněji zkopírovat realitu, ale podobně jako u akčních filmů vytvořit pro hráče jedinečný až filmový zážitek. Mapy jsou většinou menší a tématicky lazené do tématu, v kterém je daná hra zasazena. Velikostně připomínají arény a jsou podstatně menší než u žánru tactical. (Wolf, 2014)

Řešení střelby je v tomto žánru oproti tactical naprogramováno o dost jednodušeji. Jelikož je tento žánr zaměřen primárně na akci, a ne až tak na co nejpřesnější simulaci, je většina faktorů fyziky reálného světa úplně vypuštěná. Při střelbě hráč pociťuje primárně, jaké specifikace má zbraň, se kterou se rozhodl střílet. Mezi tyto parametry, které definují chování zbraně, jsou primárně tyto Damage, Recoil, Penetration, Fire Rate. Všechny tyto atributy ovlivňují chování zbraně při střelbě. Damage jednoduše říká, kolik bodů zdraví při zásahu protivníkovi vezmete. Položka recoil určuje zpětný ráz. Tato položka se ve většině her ještě dělí na dvě další. První z nich je vertikal recoil a druhá horizontal recoil. Už podle názvu je možné poznat, že jedna posouvá zbraň po horizontální ose nahoru, zatím co vertical

odklání zbraň po ose vertikální. Penetration nám pro změnu říká, jestli je zbraň schopná střílet i skrze zdi a další překážky a jak moc je tím olivněna damage, kterou zasažený hráč utrží. Poslední základní vlastnost je Fire Rate neboli rychlost střelby určuje kolik nábojnic zbraň vystřelí za určitou časovou jednotku. Tyto parametry se jako v reálném světě navzájem ovlivňují. (Wolf, 2014)

Tyto parametry se navzájem ovlivňují, avšak vývojáři je používají spíše pro vyvážení gameplaye než k napodobení reality.

Toto lze nejlépe demonstrovat na chování brokovnice. Tato zbraň má v hrách žánru action obrovský efekt na blízkou vzdálenost. Jedna jediná rána, mířená na trup či hlavu, dokáže protivníka zabít, což není daleko od chování této zbraně v realitě. Problém však nastává při střelbě na vzdálenost střední a delší. Na střední vzdálenost má zbraň poloviční účinnost než při vzdálenosti krátké a při velké vzdálenosti, dělá minimální poškození. V realitě velmi záleží na použité munici a také na délce hlavně či modelu zbraně, ale průměrná efektivní vzdálenost se pohybuje kolem 50 metrů při použití typu munice určeného na střelbu ptáků, což je typ munice s absolutně nejkratší efektivní vzdáleností. Jakmile je do porovnání zakomponován i typ použité munice při střelbě brokovnicí, je reálná hodnota účinné vzdálenosti zbraně několika násobně vyšší čemuž herní mechaniky rozhodně neodpovídají. (Britannica, 2023)

2.3 Současné hry a jejich mechaniky

2.3.1 Battlefield

Hra, která svým působení nastavila nové standardy, jak v rámci gameplaye, tak v rámci střeleckých mechanik. Herní styl Battlefieldu je postaven na větších mapách a taktičtějším postupu, než je to třeba u jeho přímého konkurenta Call of Duty a hru lze zařadit spíše do žánru tactical i když k věrné simulaci reality má velmi daleko.

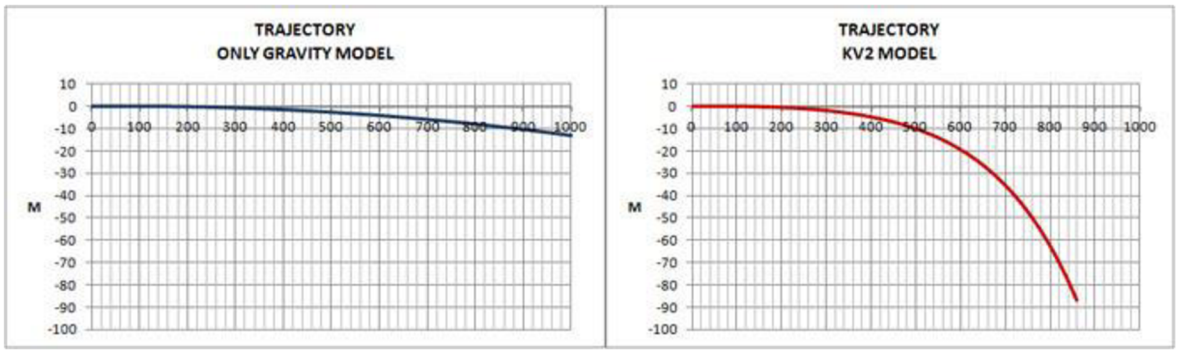
Chování zbraní v této značce je u většiny her stejné. Ve hře si můžete vybrat ze čtyř typů jednotek jejíchž názvy jsou assault, medic, support, recon. Každá jednotka má jiné vybavení a je zaměřená na jiný styl hraní.

2.3.1.1 Externí balistika

Externí balistika v této hře může být popsána, jako dobrý balanc mezi obtížností a zachování akčního gameplaye. V celé této hře jsou všechny střely dělány technikou fyzického projektilu, což jde poznat primárně na rozlehlejších mapách, kterých je ve hře většina. Každá zbraň má své parametry, které zásadně ovlivňují chování zbraně. Prvním realistickým detailem je fakt, že každá zbraň má svou úsťovou rychlost. Tato rychlost se zásadně neliší od skutečných zbraní, podle kterých je herní arzenál modelován. Jako příklad lze uvést M1 Garand, americkou poloautomatickou pušku, slavnou zejména pro svou roli ve druhé světové válce. Tato zbraň má ve hře úsťovou rychlost 780 m/s, což se oproti reálným 850 m/s liší o pouhých 70 m/s (SYM.GG, 2018). Tento rozdíl lze považovat za vyvážení zbraně pro herní účely.

Kulky v této hře mají i činitel odporu a kulka tedy zpomaluje v letu. Významná změna proběhla v této oblasti dne 21. října 2016, kdy vyšla hra Battlefieldu 1, což je první hra, kde je odpor vzduchu implementován. Komunita na internetových diskuzích se ihned po vydání nad touto změnou pozastavila a uživatelé začali pomocí měření popisovat nově implementovanou mechaniku (Consequences of BF1 air drag for players, 2019). Kompenzace pádu kulky se stává klíčovým prvkem při snaze dosáhnout přesné střelby na větší vzdálenosti. Zatímco v Battlefieldu 4 byla kompenzace lineární, v Battlefieldu 1 je mnohem více nelineární, což velmi ztěžuje intuitivní výpočty. To může mít vliv na schopnost hráčů přizpůsobit se rychle měnícím se podmínkám v boji.

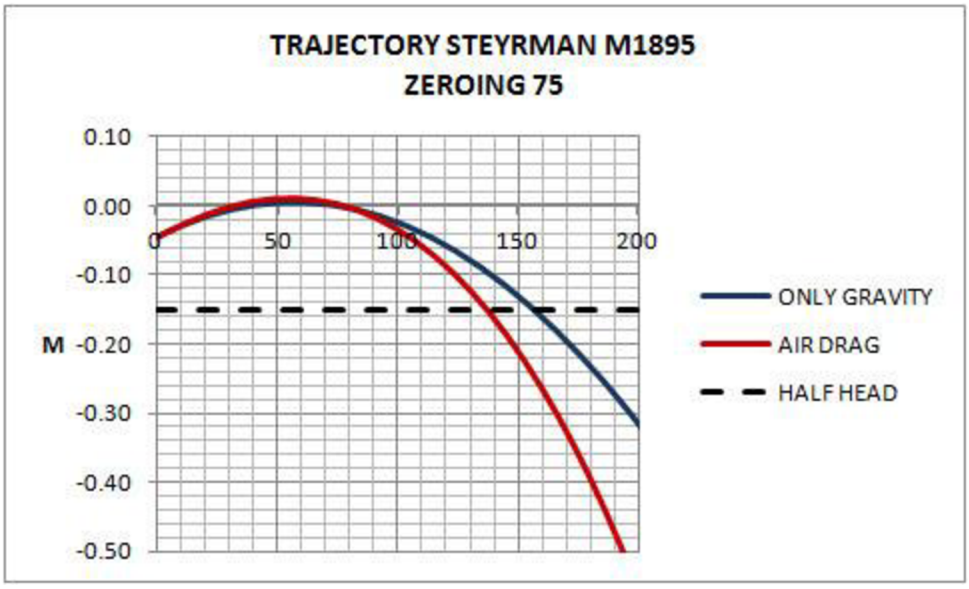
Obrázek 1 - trajektorie výstřelu STEYRMAN M1895



Zdroj: Consequences of BF1 air drag for players (2019)

V představených modelech byly zkoumány trajektorie střelby se zaměřením na STEYRMAN M1895, německou druhoválečnou pušku a také jednu z nejpomalejších jednokrokových pušek v obou hrách. Tato puška byla vybrána záměrně proto, aby efekt odporu vzduchu byl co nejlépe pozorovatelný. Porovnání trajektorií střelby, kterou ovlivňuje pouze gravitace v Battlefieldu 4, a kterou ovlivňuje také odpor vzduchu v Battlefieldu 1, ukazuje výrazný rozdíl v uražené vzdálenosti ale i pádu kulky. Zatímco v BF4 dochází k standartnímu pádu kulky, což je způsobeno pouze gravitací, v BF1 dochází k diferenciálnímu efektu, přičemž vzdušný odpor způsobuje výraznější zakřivení trajektorie a kratší dosah.

Obrázek 2 - Trajektorie výstřelu STEYRMAN M1895, blízká vzdálenost, nastavení hledí na 75 metrů



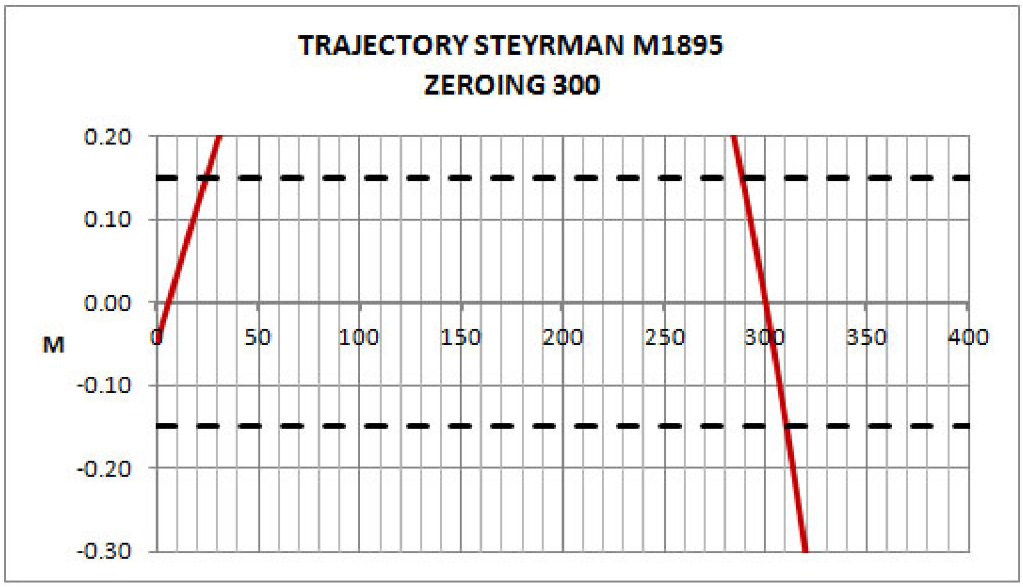
Zdroj: Consequences of BF1 air drag for players (2019)

Článek si jako srovnávací prostředek vybral model hlavy, což je část těla, jež se hráči snaží zasáhnout, kvůli velkému poškození. Tento model na výšku měří 30 cm. Při měření byla zbraň namířena přesně na střed obličeje, což je na grafu znázorněno hodnotou 0 na ose Y. Pokud tedy kulka spadne pod -0,15 metru, kulka nezasáhne hlavu. Hledí bylo při měření nastaveno na 75 metru, což jde velmi názorně na grafu vidět.

Do 100 metrů se křivky nijak zásadně neliší. Hlava protihráče by byla stále trefena až do 135 metrů kulkou, na kterou působí diferenciální odpor vzduchu. Kulka, ovlivněná pouze působením gravitace by cíl zasáhla až do vzdálenosti 155 metrů. Od těchto bodů se rozdíl kvadraticky zvyšuje a v 190 metrech je už rozdíl 0,20 metru.

Dále se článek věnuje nastavení hledí a jak moc pomáhá nastavit dostřel na hledí při střelbě na velkou vzdálenost. Největší možné nastavení je v Battlefieldech 1 a V je 300 metrů.

Obrázek 3 - Trajektorie výstřelu STEYRMAN M1895, velká vzdálenost, nastavení hledí na 300 metrů

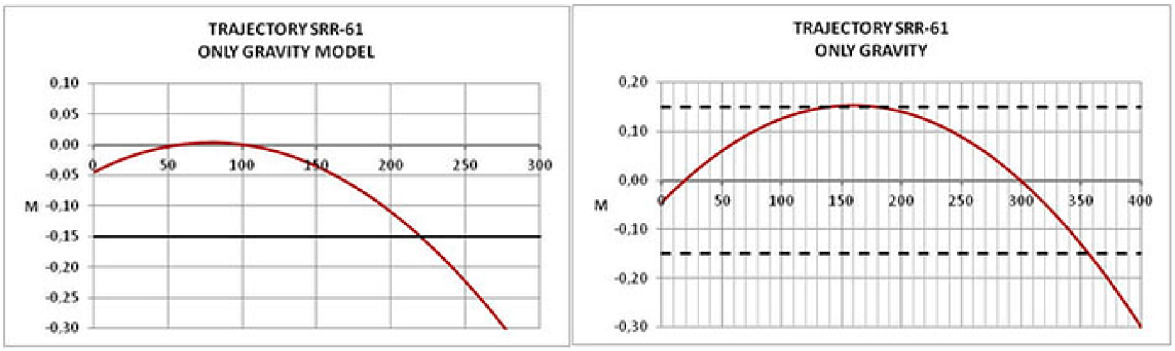


Zdroj: Consequences of BF1 air drag for players (2019)

Tento graf ukazuje překvapivé výsledky. Jelikož máme hledí nastaveno na 300 metrů, kulka v momentu dosažení námi požadované vzdálenosti v celku rychle padá a to přibližně 10 centimetrů za 10 metrů uražené vzdálenosti. Okno, při kterém kulka zasáhne oblast hlavy, je od 290 až do 310 metrů. Jakmile by hráč byl jen o metr dál nebo blíž ke střelci, kulka hlavu mine. Pro porovnání autor do článku dal grafy zbraně SRR-91 z Battlefieldu 4, na jejíž výstřely tedy odpor vzduchu nepůsobil. Důležitý je však v tomto

porovnání fakt, že SRR-91 byla i v Battlefieldu IV velmi účinnou zbraní a lze to tedy brát jako porovnání dvou extrémů (Consequences of BF1 air drag for players, 2019)

Obrázek 4 - Trajektorie výstřelu SSR-61, velká vzdálenost, nastavení hledí na 100 a 300 metrů



Zdroj: Consequences of BF1 air drag for players (2019)

Z grafů vidíme, že rozdíl je značný. Tato zbraň při nastavení 100 metrů svůj cíl trefila až do přibližně 220 metrů. Při nastavení hledí na vzdálenost 300 metrů však kulka přistála na cílové ploše až do neuvěřitelných 355 metrů bez jediné mezery v průběhu letu. Skutečnost, že největší výkyv na 355 metrů byl +- 15 centimetrů a že pád kulky je díky zanedbání odporu vzduchu lineární, dovolovala hráčům střílet velmi přesně i na obrovské vzdálenosti.

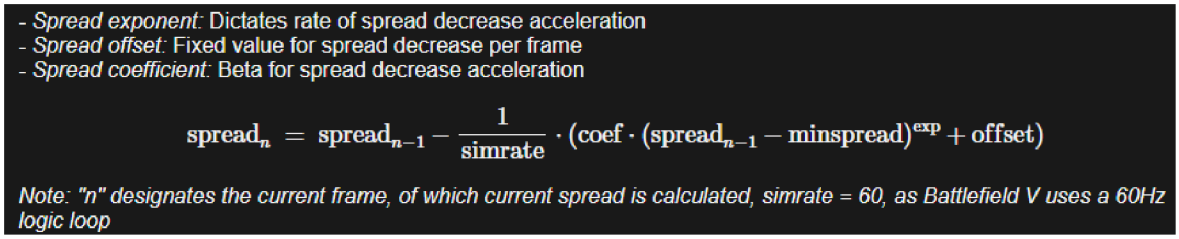
Přidání efektu odporu vzduchu lze brát jako snahu vývojářů se o něco více přiblížit realitě, ale také jako vyvážení hry a donucení odstřelovačů přiblížit se oblastem boje.

Toto jsou však všechny funkce, které simulují reálné faktory ovlivňující externí balistiku. Kulka letí přímo rovně, odpor vzduchu se nemění na základně nadmořské výšky a teploty, ale je pořád stejný. Vítr taky v tomto procesu nehraje žádnou roli a kulka letí pořád po své původní trase.

Další věc, která střelbu v těchto hrách velmi ovlivňuje je rozptyl. Tato hodnota určuje, jak moc se dráha kulky bude odchylovat od středu obrazovky nebo dráhy, kterou lze vidět při pohledu skrze mířidla. Zajímavé je, že zvýšení rozptylu na výstřel je nastaveno na 0 při střelbě mířené skrze mířidla pro lehké kulomety, těžké kulomety, samonabíjecí pušky, samonabíjecí dlouhé pušky, krátké zbraně a odstřelovací pušky. Všechny tyto zbraně patří třídám support a recon, což jen dokazuje fakt, že jsou tyto třídy zaměřeny na boj na delší vzdálenost. Rozptyl se snižuje i když zbraň není v klidu, tedy mezi výstřely při střelbě maximální možnou kadencí, a to s akcelerující rychlostí na snímek. Tento mechanismus

umožňuje hráčům lépe kontrolovat střelbu i při rychlé palbě díky mechanice snižování rozptylu. Tato mechanika velmi přesné střelby při střelbě dávkou byla vývojáři zvolena, kvůli snížení obtížnosti gameplaye a tím pádem i zvětšení okruhu lidí, která hra bude bavit. Lze to také chápat jako simulaci reality, ve smyslu, že střelec vidí, kam střely dopadají, a přizpůsobuje tomu svou střelbu, tudíž je o něco přesnější. Tento systém má název SIPS neboli spread increase per shot. Funkčnost této mechaniky je možné demonstrovat následující rovnicí (Sym.gg, 2018).

Obrázek 5 - Rovnice rozptylu v Battlefield V



Zdroj: Sym.gg (2018)

Výsledkem této rovnice snižování rozptylu je křivka rozptylu, která je nejasně logaritmická. Rovnice nám velmi názorně ukazuje, jak postih přesnosti s trvalou palbou ve hře klesá. Při použití mířidel hra používá rovnoměrné rozdělení přes poloměr, což znamená, že má vnitřní zkreslení směrem do středu. Střely tak mají tendenci být více soustředěné blíže ke středu zaměřovaného oblasti. Tento mechanismus je výhodný pro přesnou střelbu na delší vzdálenosti, kde je důležité, aby střely padaly blíže k bodu zaměření.

Systém snížení rozptylu vytváří křivku rozptylu, která je zhruba logaritmická, s klesajícími penalizacemi za další udržovanou palbu. To znamená, že čím déle střílíte, tím menší je penalizace za rozptyl, což umožňuje určitou formu kontroly přesnosti i při delším střílení.

Při střelbě od boku se používá rovnoměrné rozdělení přes oblast, což znamená, že střely jsou rovnoměrně rozloženy po celé ploše kužele rozptylu. To může vést k tomu, že střely budou více rozptýlené a méně předvídatelné ve srovnání se střelbou při míření skrze mířidla. Tento mechanismus je typicky výhodný pro střelbu na krátké vzdálenosti, kde přesnost není tak kritická, a umožňuje střelci pokrýt větší oblast.

Vertikální zpětný ráz v Battlefieldu je pevná hodnota. Vertikální zpětný ráz se sice může zvětšovat, ale jeho nárůst a pokles probíhá v diskrétních, známých intervalech.

Horizontální zpětný ráz se skládá z číselné řady například od -0,3 až do 0,3 , kde vzniká náhodně generovaná hodnota s rovnoměrným rozložením. Za zmínku také stojí, že SAR a SLR nemají horizontální zpětný ráz, avšak se u těchto typů zbraně používá zpětný ráz odvozený od rozptylu, který efektivně přidává další rozptyl, který zpětný ráz nahrazuje. (Sym.gg, 2018)

2.3.2 Call of duty

Tato herní značka byla v minulých desetiletí nejznámější a nejhranější FPS hrou. Patří do ní 23 her vydaných od roku 2003 až po současnost. Mezi tématy jednotlivých dílů najdeme jak mezinárodní konflikty, tak robotické bitvy z budoucnosti. Tato značka se experimentování nebála a zařadila do svých her i takové pohybové funkce jako běhání po zdech nebo dvojitý skok, které podle mého názoru udělali už z tak rychlých her hry s tempem hraní na hranici hratelnosti. Tyto funkce nebyly komunitou přijaty dobře a v dnešních dnech se značka opět vrátila na nižší tempo svých původních představitelů. Tato část bude věnovat zástupce jednomu z největších zástupců Action žánru. Z popisu v této kapitole je vynechán battle royal titulu se jménem Call of Duty: Warzone, kvůli jeho vybočení ze série a zařazení nových mechanik, které v původních hrách nebyly zastoupeny. (Activision, 2023)

Mapy v této hře jsou oproti mapám ostatních her popsaných v této práci malé. Jejich rozloha je v řádech stovkách metrů. To ale napomáhá hře dosáhnout a vývojáři požadovanou neustálou konfrontaci a velmi dynamický gameplay. Na některých mapách je možné přehodit celou mapu například házecím nožem, který má oproti granátům velmi dobré letové vlastnosti. Damage v této hře je názorná ukázka žánru action a v některých případech absolutně neodpovídá realitě. Při definici žánru action bylo jako příklad použito poškození brokovnicí. Tohoto a dalšího absurdního chování zbraní si ve hrách Call of Duty můžeme všimnout velmi často. (Activision, 2023)

2.3.2.1 Externí balistika

Ve hrách Call of Duty je implementace externí balistiky zjednodušena ve prospěch tempa a přístupnosti hry. Všechny zbraně jsou dělány pomocí HitScan, což znamená, že střely putují v přímce od ústí zbraně až k nepřátelskému hráči a kulku ve hře neovlivňuje absolutně nic, kromě fyzických objektů v cestě. Díky tomuto přístupu mohou hráči očekávat, že jejich střely zasáhnou cíl bez potřeby počítat s časem letu střely nebo s jejím odklonem

způsobeným větrem. Tento systém také eliminuje potřebu zohledňovat vertikální a horizontální odchylky, což by mohlo hru zbytečně komplikovat. Avšak, i přes tuto zjednodušenou balistiku, některé tituly série přidávají další balistické prvky, jako jsou různé druhy munice a jejich specifické dopady na průchodnost skrz materiály, což hráčům umožňuje strategicky využívat prostředí k dosažení vítězství. (Activision, 2023)

Granáty a celkově všechny věci určené k házení hráčem jsou dělané metodou fyzického objektu. Jejich vlastnosti, jako například váha nebo aerodynamika, jsou však upraveny pro lepší zasazení do celkového gameplaye. (Activision, 2023)

Navzdory těmto zjednodušením se Call of Duty snaží o zachování určité míry realismu a variabilitu ve zbraních a jejich používání. Například, rozptyl střel z automatických zbraní se zvyšuje s délkou střelby, což nutí hráče k používání krátkých dávek pro zvýšení přesnosti. Tato dynamika dodává hře hloubku a vyžaduje od hráčů, aby se nejen rychle rozhodovali, ale také strategicky plánovali své pohyby a využívali různé zbraně v závislosti na situaci. (Activision, 2023)

Konečně, i když systém HitScan může některým hráčům připadat příliš zjednodušený, je důležité si uvědomit, že hlavním cílem série Call of Duty je poskytnout poutavý a plynulý zážitek z hraní a že značka chce hráči dodat akční a velmi dynamický filmový zážitek. Tento přístup k balistice umožňuje hráčům soustředit se na taktiku, pohyb a rozhodování v rychlém tempu, což jsou aspekty, které přispívají k celkovému úspěchu a oblíbenosti této série videoher. (Activision, 2023)

2.3.3 **Arma 3**

Arma 3, vyvinutá studiem Bohemia Interactive, představuje vrcholné dílo v oblasti vojenské simulace, které se zaměřuje na autentičnost a rozsáhlé možnosti pro hráče. Tato hra, vydaná v roce 2013, posouvá hranice realistického zobrazení moderního bojového prostředí díky pokročilým technologiím a detailně propracovanému fyzikálnímu modelu. Tato pečlivá pozornost k detailům činí z Arma 3 nejen zábavnou hru, ale také cenný vzdělávací nástroj pro pochopení komplexních principů vojenské taktiky a balistiky.

Autentické modelování balistiky zůstává odlišujícím aspektem hry Arma. I pro hráče, kteří do hry investovali stovky hodin, může být obtížné identifikovat všechny aspekty, které systém simuluje. V oblasti externí balistiky Arma 3 nabízí bezprecedentní úroveň realismu; od simulace trajektorie střel, přes vliv atmosférických podmínek, až po specifické charakteristiky různých typů munice.

2.3.3.1 **Externí Balistika**

2.3.3.1.1 CfgAmmo

V hře Arma 3 je simulace střelby zbraní postavena na dynamickém modelu, který místo kinematického přístupu (založeného na trajektorii) využívá dynamický (založený na síle). Při simulaci letu střely jsou zohledněny dvě základní síly – odpor vzduchu a gravitace.

```
Vector3 accel=_speed*(_speed.Size()*_airFriction);
// add gravity
accel[1]-= _coefGravity * G_CONST;
```

(BOHEMIA INTERACTIVE, 2013)

Ve této hře má i zvolená munice zásadní vliv na let kulky a celkovou externí balistiku. V souboru cfgAmmo lze najít všechny parametry, které se v Armě používají na výpočet externí balistiky. Položky jsou vypsány níže (BOHEMIA INTERACTIVE, 2013).

- typicalSpeed=900;

  Rychlost při 100 % zásahu (m/s)
- hit = 20;

  Poškození způsobené při zásahu při typické rychlosti, modifikované skutečnou rychlostí v závislosti na koeficientu exploze.
- indirectHit = 18;

  HIT pro objekt, který se nacházel v dosahu (a nebyl zasažen přímo), upravený podle vzdálenosti (0 HIT je při 4x dosahu) a prostého zakrytí jinými objekty.
- indirectHitRange = 7;

  Dosah v metrech při tomto dosahu je indirectHit 100 % a lineárně se snižuje na 4x vzdálenost.
- Caliber = 1;

  1 se rovná průraznosti = 7,62 kulky, pokud má střela projít penetračním materiálem na 2x větší vzdálenost, nastavuje se na 2.
- airFriction = -0,0005;

Upravuje snížení rychlosti, čím větší záporná hodnota, tím více se zpomaluje. Lze vypočítat jako _airFriction = accel / speed^2 s použitím reálného zrychlení a rychlosti střely z balistických tabulek.

- timeToLive = 3;

Jak dlouho žije vystřelený objekt, než se zničí (lze použít pro definici maximálního dostřelu), pokud není nadefinováno, munice se zničí, pokud brzy nic nezasáhne.

- explosive = 0;

Proměnná typu float v rozsahu 0 až 1. explosive=true se překládá jako 1. Tento faktor říká, jaká část poškození je způsobena výbuchem a jaká kinetickou energií. Kinetická část je tlumena v závislosti na rychlosti střely, zatímco explozivní část nikoli. Při vychýlení střely se v místě vychýlení uplatní pouze kinetická část poškození, systém předpokládá, že střela při vychýlení nevybuchne. Koeficient se používá také pro vizualizaci účinku výbuchu.

- deflecting = 5;

Maximální úhel, pod kterým se střela vychýlí - 0 znamená, že se střela nevychýlí.

- fuseDistance = 15;

Některá munice potřebuje urazit určitou vzdálenost, než se aktivuje jako například rakety a granáty. Tento parametr lze použít i k zabránění sebevraždě AI pomocí tohoto vybavení.

V seznamu jsou dále parametry ovlivňující externí balistiku raket. Těmito projektily se však práce primárně nezabývá, proto jsou z výčtu vynechány

2.3.3.1.2 CfgWeapons

Druhý konfigurační soubor nám ukazuje jednotlivé parametry zbraní (BOHEMIA INTERACTIVE, 2013)

- initSpeed = 890;

  Úsťová rychlost zbraně

- burst = 12;

  Kolik simulovaných výstřelů je vystřeleno dávkou

- multiplier = 6;

  Parametr určuje, kolik výstřelů je za každý snímek je odebráno ze zásobníku, nemá vliv na parametr HIT Rychlost střelby je ve hře omezena. Maximální počet vystřelených nábojů za snímek je 1, takže při snímkové rychlosti 20 snímků za vteřinu je výsledek 1200 nábojů za minutu. U rychlopalných zbraní lze kompenzovat sílu zásahu a nepřímý dopad zásahu.

- modes[] = {this};

  Režimy zásobníku (ústí)

- rozptyl = 0,002;

  Jedná se o náhodný rozptyl střely z vektoru komory zbraně. Ovlivňuje hráče i umělou inteligenci. Ačkoli možná není logické, aby se rozptyl lišil mezi jednotlivými režimy střelby, protože fyzikálně jde o vlastnost ústí hlavně a použité munice, hra jej implementuje na úrovni režimu střelby a je možné mít různý rozptyl pro různé režimy.

- dexterity = 2;

  Parametr určuje, jak rychle mění komora zbraně směr, zobrazuje kurzor. Čím větší hodnota, tím rychleji.

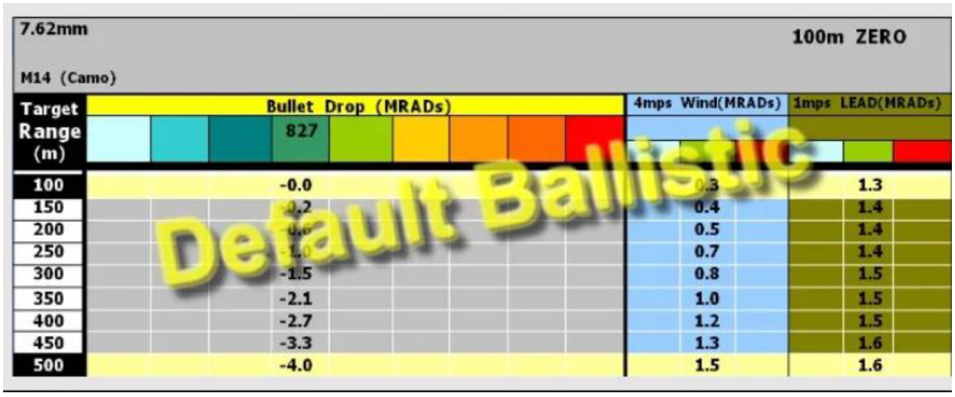
Rychlost střely se s vzdáleností snižuje, což má vliv na hodnotu zásahu (HIT value), a herní engine simuluje i průnik a odchýlení střely mezi jednotlivými objekty, přičemž střela způsobuje poškození každému objektu, kterým pronikne nebo od kterého se odchýlí. To má mimo jiné za následek ztrátu energie. Nepřímé zásahy způsobené explozemi jsou blokovány stěnami a podlahami.

Už jenom z výčtu parametrů je možné si představit, jak je tato hra komplexní a co všechno má na střelbu vliv.

Mechaniky externí balistiky této hry lze shrnout následovně. Po tom, co kulka opustí zbraň, jí ovlivňuje odpor vzduchu. Ten je však konstantní a na let nemá vliv ani teplota vzduchu ani nadmořská výška. Dále na kulku působí gravitační zrychlení o stejné síle jako na Zemi. Střela může být odkloněná větrem, který má vždy stejnou sílu a jeho efekt lineárně stoupá s uraženou vzdáleností. (BOHEMIA INTERACTIVE, 2013),

Arma 3 je velmi populární i díky své moderské komunitě. Pro zkoumání externí balistiky je však zajímavý primárně jeden mod, se jménem ACE3. Tento mód ještě zvyšuje úroveň simulace reality a posouvá hru na velmi obstojný vojenský simulátor se všemi vlivy, kterým čelí jednotky v reálném světě. Team ACE3 ve verzi 3.0.0, přidal do svého módu i pokročilou balistiku střel. S tím na jejich stránky přibyla i tabulka balistiky základní verze hry. (ACE3 Team, 2015)

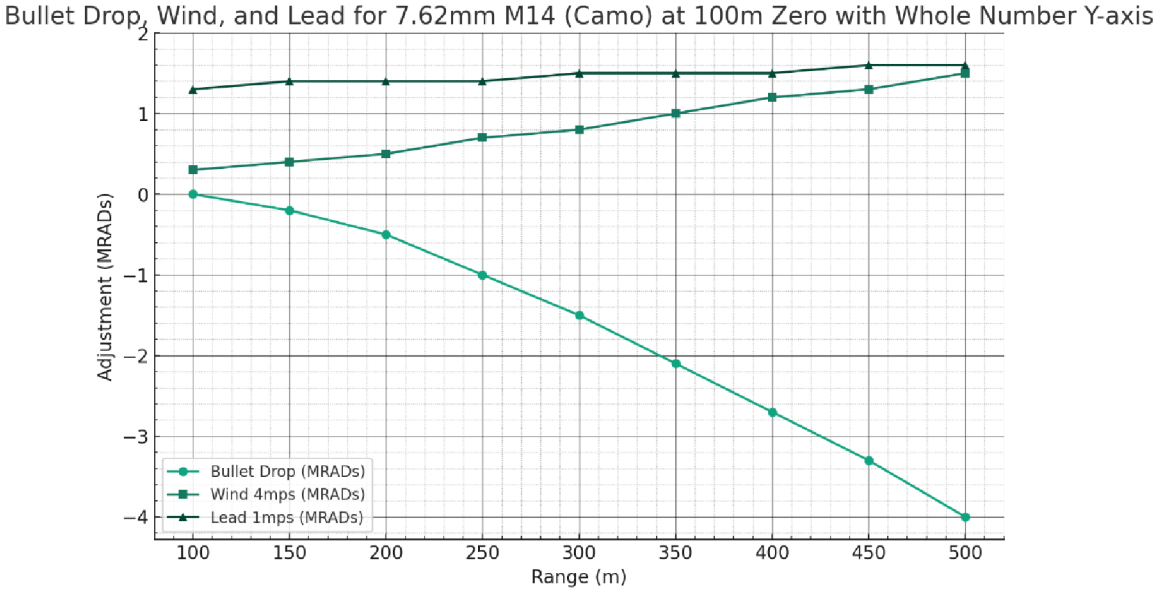
Obrázek 6 - Balistická tabulka základní hry



Zdroj: ACE3 Team (2015)

Z této tabulky lze vyčíst, že podobně, jak v předchozí hře Battlefield je zde klesání diferenciálního charakteru. Pro lepší porozumění zkoumaných faktorů je zde graf jednotlivých hodnot.

Obrázek 7 - Graf střelecké kompenzace pro M14 7,62mm v Armě



Zdroj: Vlastní zpracování

Jako první si je možné všimnout diferenciálního zrychlení pádu kulky, podobnému mechanice použité v Battlefield 1. Mířidla zbraně jsou nastavena na 100 metrů, kde začínají i námi zkoumané hodnoty. První rozdíl mezi grafy je ten, že zde se nachází na ose Y místo výškového rozdílu, potřebné nastavení mířidel v miliradiánech. Toto číslo určuje, o kolik

miliradiánu musíme posunou mířidlo dolů, aby nám správně ukazovalo místo, kam projektil doletí. Dále je v datech zanesen vliv větru, který v této základní verzi má také svůj vliv. Poslední zanesená hodnota je LEAD. Tato hodnota je také nesmírně důležitá pro střelce na dlouhou vzdálenost, protože ukazuje, o kolik MRAD musí střelec mířit před cíl pohybující se 1 metr za sekundu.

Střelbu však v této hře ovlivňuje více faktorů. Jay Crowe, spoluvýkonný ředitel tvůrčího týmu, uvedl v roce 2013 při releasu Army 3 (PC Gamer, 2013): " Vlastně můžeme začít ještě před vystřelením střely. Díky enginu Real Virtuality se často nejedná jen o příběh samotného výstřelu, ale o vaší cestě ke stisknutí spouště. Jak moc jste běhali a jakou hmotnost vybavení máte na sobě se promítne do hodnoty únavy. Ta – spolu s vaším postojem, dýcháním a zdravotním stavem – ovlivňuje kývání zbraně, a tím i přesnost. Dále je třeba vzít v úvahu vzdálenost k cíli – nastavení mířidel zbraně nebo nastavení pro kompenzaci balistické křivky – a také míření před pohybující cíl tak, aby střela skončila tam, kde jste chtěli." Z toho vyplývá, že už před samotným výstřelem může být přesnost vaší střely značně ovlivněná.

2.4 Metody externí balistiky ve videohrách

Tato kapitola se bude věnovat metodám řešení externí balistiky v současný i starších videohrách. Jsou zde uvedeny faktory jako výsledek, porovnání s realitou, náročnost na programování a náročnost na výkon počítače či konzole.

2.4.1 RayCasting

Jedna z hlavních metod, kterou se ve videohrách řeší externí balistika a detekce zásahu výstřelů je RayCasting. Jeho použití má své výhody zejména v jednoduchosti implementace a nenáročnosti na operace, které počítač musí při výstřelu zpracovat. Tato sekce je věnována principu, jakým tato metoda funguje. Podrobněji je v této kapitole popsán i jeho vliv na přesnost v ohledu simulace reality.

2.4.1.1 Princip

Princip RayCastingu je v zásadě velmi jednoduchý. Spočívá v sledování paprsků od startovního bodu, jeden na pixel, a nalezení nejbližšího objektu, blokujícího cestu tohoto paprsku. V reálném světě to lze asi nejblíže prezentovat na laserovém ukazovátku, které z praktického hlediska funguje stejně.

Jakmile se však podíváme na to, jak tato metoda funguje v prostředí Unity, zjistíme, že je přece jen o něco složitější. Jako první je důležité uvést v jaké třídě je možné jí najít.

Funkci RayCast lze najít v třídě Physics. Třída Physics je klíčová pro simulaci fyziky ve vašem projektu, aby se objekty správně zrychlovaly a reagovaly na srážky, gravitaci a různé další síly. Unity poskytuje různé implementace fyzikálního enginu, které je možné použít podle potřeb vašeho projektu. Lze dosáhnout některých základních fyzikálních cílů pomocí uživatelského rozhraní, ale pro přesnější požadavky na vaší simulaci je dobré mít nějakou znalost C# (UNITY, 2023).

Deklarace funkce je následující:

```
public bool Raycast(Vector3 origin, Vector3 direction, float maxDistance = Mathf.Infinity, int layerMask = DefaultRaycastLayers, QueryTriggerInteraction queryTriggerInteraction = QueryTriggerInteraction.UseGlobal)
```
(UNITY, 2023)

Jak je hned na první pohled zřejmé, celá funkce je logického datového typu, tedy boolean. To znamená, že tato funkce může vrátit hodnoty pouze dvě. Těmito hodnotami jsou `true` a `false`. Výsledek funkce nám tedy řekne, zda paprsek narazil na objekt či nikoliv. Aby byla deklarace funkce správně vysvětlena je zde, u každého parametru uvedeno, jaké hodnoty by měl obsahovat a jaký smysl má ve výsledku funkce. První parametr funkce se jmenuje *`origin`* a jeho datový typ je třída Vector3 (UNITY, 2023).

Třída Vector3 v Unity je součástí knihovny UnityEngine a reprezentuje trojrozměrný vektor nebo bod v prostoru. Tato třída obsahuje funkce pro provádění běžných operací s vektory, jako je například sčítání, odčítání, násobení a dělení vektorů. Třída má také několik statických vlastností, jako jsou back, down, forward, left, negativeInfinity, one a positiveInfinity, které slouží k rychlému zápisu vektorů s určitými hodnotami. V tomto případě však pomocí tohoto parametru funkci předáváme počáteční místo, ze kterého paprsek vystřelí. Stejnou třídu má i druhý parametr funkce, který má název `direction`. Tento parametr, jak už jeho název napovídá, určuje směr, jakým bude tento paprsek směřovat. (UNITY, 2023)

Další parametr, na který je možné při volání funkce narazit, je maxDistance s datovým typem float. Toto desetinné číslo určuje vzdálenost, po jakou má paprsek kontrolovat kolize s objekty. Výše v této kapitole byla parametru přiřazena funkce Math.Infinity, která vrací nekonečno, což znamená, že zásahy budou kontrolovány až po hranice prostoru hry. (UNITY, 2023)

Následující parametr je layerMask. LayerMask je parametr, který umožňuje specifikovat, na které vrstvy (layers) se má paprsek aplikovat.

Parametr je typu LayerMask, což je speciální datový typ v Unity, který umožňuje specifikovat, na které vrstvy se má paprsek aplikovat. Vrstvy jsou v Unity reprezentovány čísly od 0 do 31. Pokud je zapotřebí specifikovat více vrstev, lze použít bitový posun (bit shift). Například, pokud je cílem specifikovat vrstvy 0 a 1, lze použít následující kód: int layerMask = (1 << 0) | (1 << 1); 1.

V případě, že layerMask není specifikován, paprsek se aplikuje na všechny vrstvy 1. Pokud je cílem specifikovat pouze jednu vrstvu, lze použít následující kód: int layerMask = 1 << LayerMask.NameToLayer("Vrstva"); 2 (UNITY, 2023).

Při použití layerMask je důležité si uvědomit, že layerMask určuje, na které vrstvy se má paprsek aplikovat, nikoli na které vrstvy se má paprsek neaplikovat. Pokud vývojář chce specifikovat, na které vrstvy se má paprsek neaplikovat, může použít operátor ~ 1.

Například, pokud chce specifikovat, že paprsek se má aplikovat na všechny vrstvy kromě vrstvy 0, může použít následující kód: int layerMask = ~ (1 << 0). (UNITY, 2023)

Poslední parametr, který je při volání funkce nutné funkci předat, je typu QueryTriggerInteraction. Tato enumerace určuje, jak bude zpracována interakce raycastu s triggery.

Triggery jsou v Unity speciální typy koliderů, které nereagují na standardní fyzikální kolize, ale spouštějí při průchodu hráče, objektu nebo například raycastu události. QueryTriggerInteraction určuje, zda má být raycast ovlivněn těmito triggery nebo nikoli. Výchozí hodnota je QueryTriggerInteraction.UseGlobal, což znamená, že se bude hledat kolizní body pouze mezi objekty s aktivními triggerovými interakcemi. Pokud je vyžadováno ignorovat triggerové body a pouze hledat kolizní body mezi objekty bez triggerových interakcí, lze do kódu přidat QueryTriggerInteraction.IgnoreGlobal. (UNITY, 2023)

Jakmile jsou všechny parametry na správných pozicích, je metoda připravená hlásit jakýkoliv zásah, který podle předaných parametrů vyhodnotí. Také je důležité zmínit, že Unity používá u RayCastingu method overload, což v praxi znamená, že tím, že jsou funkci předány jiné parametry, lze zavolat jinou funkci, která však má stejné jméno a bude fungovat velmi podobně jako ta, která je v této kapitole blíže popsána. (UNITY, 2023)

Z výše popsaných informací je tedy zřejmé jakým způsobem, funkce v Unity funguje. Stále však nebyla zodpovězena jednu zásadní otázku, která zní následovně: „Jakým způsobem zjistíme, co bylo zasaženo?“ Jak jsem výše zmínil funkce je typu bool, tudíž vrací pouze hodnoty true nebo false. Z těchto dvou logických hodnot není možné zjistit nic víc, něž jestli paprsek narazil na jakýkoliv objekt odpovídající předaným parametrům. Struktura RayCastHit nám však tento problém řeší. Do této instance lze uložit, všechny potřebné informace, které o zásahu je potřeba vědět. (UNITY, 2023)

2.4.1.2 **Detekce zásahů několika objektů**

Jak je výše zmíněno, metoda Raycast je v principu velmi jednoduchá. V praxi však může nastat u některých situací případ, kdy bude tato metoda nedostatečná. Jako jeden z příkladů je možno uvést případ, kdy je vyžadováno, aby kulka, která prošla jedním protihráčem zranila i hráče za ním. V tomto případě ani žádné nastavení nepomůže, protože metoda Raycast znamenává jen jeden zásah, tudíž by vždy jeden z hráčů zůstal v bezpečí.

Z těchto důvodů je potřeba použít jinou funkci, která umí zaznamenat zásahů více. Funkce, jež toto všechno dokáže má název RaycastAll. Tato funkce funguje velmi podobně výše popsané funkci Raycast, její návratový typ je však pole struktury RaycastHit, do které ho jsou vloženy reference jednotlivých zásahů. (UNITY, 2023)

```
Physics.RaycastAll(transform.position, transform.forward, 100.0F);
```

I když na první pohled to vypadá, že tím je problém vyřešen je několik věcí, na které je nutné při použití RaycastAll si dát velký pozor. Ačkoliv se může zdát, že zásahy v poli budou seřazeny podle dráhy letu paprsku, tedy že první zásah bude na místě [0], není tomu vždy tak. Z technické dokumentace funkce je zřejmé, že zásahy se do pole vloží náhodně, tudíž je nám známo, jaké všechny objekty byly zasaženy, nevíme však v jakém pořadí. To může být problém například u zmíněného případu zásahu dvou hráčů za sebou. Při prvním zásahu hráče bude mít kulka daleko větší rychlost pohybu něž při zásahu hráče druhého. Průchod tělem kulku výrazně zbrzdí což má za efekt i výrazné snížení poškození druhého hráče. V tomto případě je tedy důležité pořadí zásahů vědět. Tento problém lze vyřešit pomocí použití funkce Array.Sort, která může zásahy seřadit například podle vzdálenosti, v jaké byly hráči zasaženy od startovního bodu paprsku. (UNITY, 2023)

2.4.2 Řešení pomocí vytvoření projektilu jako samostatného objektu

Metoda řešení externí balistiky pomocí vytvoření kulky jako samostatného objektu je celkově složitější než druhá zkoumaná metoda Raycast. Náročnější je, jak na správnou implementaci do simulace, tak na výkon počítače, na kterém je simulace spuštěná. Tento postup má však jednu zásadní výhodu. Jelikož je nábojnice samostatný objekt, lze na ní působit fyzikálními silami jako na, kterýkoliv objekt jiný. Nábojnice může mít svou váhu, hustotu, bude se pohybovat nějakou rychlostí a jelikož na ni lze působit silami jako na všechny objekty, kterým přidáme komponent Rigidbody, je simulace reality o něco jednodušší a intuitivnější

V následující části jsou popsány jednotlivé komponenty, které jsou potřeba při postupu touto metodou.

2.4.2.1 Mesh Renderer

Mesh Renderer je zásadní komponenta v Unity, která se využívá pro vizuální reprezentaci objektů, včetně potřebného projektilů, v herním prostředí. Tato komponenta umožňuje objektům zobrazovat textury a materiály, díky čemuž získávají svůj finální vzhled. Při tvorbě projektilu je Mesh Renderer nezbytný pro to, aby hráči mohli projektil vidět, což může být klíčové pro interaktivitu a vizuální zpětnou vazbu ve hře například u šípů nebo raket. Je však také možné simulovat realitu a udělat projektily velmi malé, tak že je kromě střelce uvidí ostatní hráči velmi obtížně. To nám však může znepříjemnit detekci zásahů, však tomu je věnována část níže. (UNITY, 2023)

Použití Mesh Rendereru pro projektil začíná výběrem nebo vytvořením 3D modelu, který bude sloužit jako jeho vizuální reprezentace. Tento model může být jednoduchý geometrický tvar, jako je koule nebo válec, nebo složitější model vytvořený ve specializovaném 3D modelovacím nástroji. Po importu modelu do Unity a jeho přiřazení k objektu projektilu se v komponentě Mesh Renderer na něj aplikují materiály a textury, které definují barvu, lesk a další vizuální vlastnosti projektilu. (UNITY, 2023)

Mesh Renderer také pracuje v těsné symbióze s osvětlením ve scéně, což znamená, že vizuální vzhled projektilu je ovlivněn nejen jeho materiály a texturami, ale také osvětlením, se kterým interaguje. Toto umožňuje vytvořit realistické efekty, jako jsou stíny a odrazy, které přidávají na imerzi a realističnosti herního zážitku.

2.4.2.2 **RigidBody**

Rigidbody je základní komponenta v herním enginu Unity, která umožňuje objektům využívat fyziku Newtonovy mechaniky. Tato komponenta přidává fyzikální vlastnosti, jako jsou hmotnost, tření a odpor vzduchu, do objektů, čímž umožňuje, aby na ně působily síly a aby mohly interagovat s ostatními objekty v prostředí hry. V kontextu střelby v Unity, Rigidbody umožňuje střelám pohybovat se realistickým způsobem, reagovat na gravitaci, a odrazit se od povrchů nebo objektů, s kterými přijdou do kontaktu. Použitím komponenty Rigidbody na střelu může vývojář jednoduše implementovat různé typy střelby, od jednoduchých přímých střel až po složité trajektorie závislé na fyzikálních vlastnostech.

Komponenta Rigidbody v Unity umožňuje nastavit řadu parametrů, které mají přímý vliv na pohyb a chování střely nebo jakéhokoli jiného objektu ve hře. Zde jsou některé klíčové parametry a jejich vliv na let střely:

Hmotnost (Mass): Ovlivňuje, jak moc je objekt ovlivněn gravitačními silami a kolizemi. Těžší střely mohou mít menší zrychlení, ale zároveň mohou lépe pronikat překážkami. (UNITY, 2023)

Tření (Drag): Určuje odpor vzduchu, který působí proti pohybu střely. Vyšší hodnota drag zpomalí střelu rychleji, což může být užitečné pro simulaci realistického chování projektilů na dlouhé vzdálenosti. (UNITY, 2023)

Odpor při otáčení (Angular Drag): Podobně jako tření, ale pro rotaci objektu. Ovlivňuje, jak rychle může střela ztratit svou rotační energii. (UNITY, 2023)

Použití gravitace (Use Gravity): Určuje, zda na střelu působí gravitace. Vypnutí gravitace je užitečné pro projektily, které by měly letět přímočaře bez ovlivnění padáním, nebo jestliže si chce vývojář nadefinovat tuto sílu sám. (UNITY, 2023)

Kinematický (Is Kinematic): Když je tento parametr aktivní, Rigidbody nebude ovlivněn fyzikou a pohybem, ale může být stále animován nebo pohybován skriptem. To je užitečné pro střely řízené přesně definovanými trajektoriemi. (UNITY, 2023)

Omezení (Constraints): Umožňuje omezit pohyb nebo rotaci objektu v určitých osách. To může být užitečné pro střely, které by se měly pohybovat pouze ve specifických směrech. (UNITY, 2023)

Při správném nastavení těchto parametrů může vývojář dosáhnout různých efektů a chování střel, od pomalých a těžkých dělostřeleckých granátů po rychlé a lehké šípy.

Nastavení těchto parametrů je klíčové pro dosažení požadovaného realismu a hratelnosti ve střeleckých mechanismech hry.

2.4.2.3 **Collider**

Komponenta Collider v Unity představuje základní stavební prvek pro detekci a řízení kolizí ve virtuálním prostředí. Díky své integraci s fyzikálním modulem Unity, Collider umožňuje objektům nejen detekovat přítomnost jiných objektů ve svém okolí, ale také adekvátně reagovat na tyto detekce.

Jednou ze základních vlastností Collideru je jeho schopnost připojit se k různým fyzikálním objektům, jako jsou Rigidbody nebo ArticulationBody. To, jak bylo zmíněno výše, umožňuje objektům s Colliderem fyzicky interagovat s ostatními objekty ve scéně. (UNITY, 2023)

Collider poskytuje několik důležitých vlastností, které umožňují vývojářům detailně konfigurovat chování kolizí. Vlastnost bounds například definuje ohraničující objem Collideru ve světovém prostoru, což je kritické pro určení, zda došlo ke kolizi. Vlastnost enabled určuje, zda je Collider aktivní a může zaznamenat dotek s ostatními Collidery, což umožňuje vývojářům dynamicky zapínat a vypínat detekci kolizí podle potřeby hry. (UNITY, 2023)

Dalším klíčovým aspektem je možnost konfigurovat Collider jako isTrigger. Když je Collider nastaven jako trigger, místo fyzického blokování objektů umožňuje vyvolání specifických skriptů nebo událostí, když s ním jiný objekt vstoupí do kontaktu. Toto chování je zásadní pro vytváření interaktivních herních mechanik, jako jsou pasti, cíle nebo zóny, které volají události při vstupu nebo výstupu hráče. (UNITY, 2023)

Vývojáři mohou také využívat specifické metody pro práci s Colliderem. Jednou z těchto metod je ClosestPoint, která vrací bod na Collideru nejbližší určité lokaci. Tato funkce je užitečná pro výpočet dynamických interakcí nebo pro určení, jak blízko se objekt nachází k potenciálnímu bodu kolize.

2.5 Fyzikální jevy ovlivňující vystřelenou kulku v letu

V této kapitole budou více představeny podmínky a fyzikální jevy reálného světa, které ovlivňují externí balistiku vystřelené nábojnice ze zbraně. Popis jednotlivých faktorů bude popsán jak z pohledu fyziky a okolního světa, tak z pohledu střelce. V této kapitole se popisují pouze vlivy, které byli ve zkoumaných hrách a jsou tedy této práci přínosné. Z toho důvodu zde není popis rotace kulky ani Coriolisovy síly, která vzniká rotací země.

2.5.1 Odpor vzduchu

Odpor vzduchu skutečně hraje klíčovou roli při letu kulky, zvláště při střelbě na delší vzdálenosti. Během letu se odpor vzduchu projevuje jako brzdná síla, která zpomaluje pohyb projektilu. Tato síla je přímo úměrná rychlosti kulky, což znamená, že čím vyšší je rychlost letu, tím větší je odpor a tím rychleji kulka ztrácí na rychlosti a schopnosti udržet stabilní trajektorii. Výsledkem je menší dosah kulky ve srovnání s tím, co by bylo dosaženo ve vakuu bez přítomnosti odporu vzduchu. Aerodynamický tvar a hladký povrch kulky mohou snížit odpor vzduchu, což umožňuje delší dolet. Střelci proto musí při střelbě na větší vzdálenosti zohlednit odpor vzduchu a provést korekce, jako jsou úpravy elevace a azimutu, aby kompenzovali ztrátu rychlosti a změnu trajektorie kulky způsobené odporovými silami vzduchu. (Urone, 2012)

Studie "Effect of air drag on particles ejected during explosive cratering" od A. Sherwood zkoumá vliv odporu vzduchu na balistickou trajektorii tělesa zpomalovaného Newtonovým zákonem odporu. V práci je odvozena přibližná analytická řešení pro bod dopadu částice a provedeno srovnání s přesnou integrací balistických rovnic Euler-Otto na digitálním počítači. Bylo zjištěno, že odpor vzduchu může významně snížit dosah částice, přičemž snížení dosahu je nejvýraznější u částic malého průměru s vysokou počáteční rychlostí. Dosah 1 cm částice s počáteční rychlostí 40 m/s ve vzduchu je přibližně 20 % dosahu ve vakuu. (Sherwood, 1967)

2.5.1.1 Teplota a vlhkost

Teplota vzduchu má významný vliv na jeho hustotu, což následně ovlivňuje trajektorii vystřelené kulky. S rostoucí teplotou se hustota vzduchu snižuje, což vede k menšímu vzdušnému odporu a potenciálně delšímu dosahu střely. Studie zkoumající vliv vlhkosti na trajektorii malokaliberních zbraní ukázala, že přesnost předpovědi trajektorie je

výrazně závislá na teplotě vzduchu spolu s vlhkostí, přičemž modely často zanedbávají efekt relativní vlhkosti na hustotu vzduchu. Tento efekt je zvláště významný při střelbě na dlouhé vzdálenosti, kde se proměnlivé teplotní podmínky můžou výrazně ovlivnit finální místo dopadu střely. Dále, teplota vzduchu ovlivňuje nejen fyzikální vlastnosti samotného střelného prachu a tím výkon zbraně, ale také aerodynamické vlastnosti střely během letu. Výsledky studie ukazují, že pro přesné balistické výpočty je nutné zohlednit nejen základní fyzikální parametry, ale i aktuální meteorologické podmínky, včetně teploty vzduchu. Tyto poznatky jsou klíčové pro zlepšení přesnosti střelby a balistických simulací. (OBERLIN, 2017)

2.5.1.2 **Nadmořská výška**

Nadmořská výška má také nemalý vliv na trajektorii vystřelené kulky. Výzkumy ukazují, že s rostoucí nadmořskou výškou se zvyšuje dosah kulky. Například studie zaměřující se na simulaci trajektorie kulky z útočné pušky AK-47 ukázala, že dosah kulky ve výšce 4 km je o 16 % vyšší než na úrovni moře, což je způsobuje snížení hustoty vzduchu spojené s rostoucí nadmořskou výškou. Tento efekt je způsoben nižší hustotou vzduchu ve vyšších nadmořských výškách, což vede k nižšímu odporu vzduchu a umožňuje kulkám letět dále. Výzkumy také zdůrazňují význam porozumění vlivu nadmořské výšky na balistické výkony, neboť to může mít praktický dopad na předpovídání trajektorií střel ve vojenských a forenzních aplikacích. (Krishna Reddy, 2019)

2.5.1.3 **Výpočet**

Vzorec pro výpočet odporu je tedy se započítáním základních důležitých aspektů je v knize College Physics (Urone, 2012) uveden následovně:

$$F_D = 1/2 * C * \rho * A * v^2$$

Kde C je koeficient odporu, A je plocha objektu směřující do kapaliny a ρ je hustota kapaliny. Koeficient odporu se u kulek vystřelených ze zbraně pohybuje kolem od 0,1 do 0,3 (NASA, 2006). Tato hodnota vypovídá o tom, že odpor vzduchu bude mít efekt na větší vzdálenosti, avšak tento vzorec vychází z předpokladu, že odpor vzduchu je nepřímo úměrný kvadrátu rychlosti objektu, což znamená, že s rostoucí rychlostí se odpor exponenciálně zvyšuje, což má za následek zpomalení pohybu objektu. V kontextu simulace letu střely ve videohře je použití tohoto vzorce klíčové pro realistické modelování vlivů, které na střelu

působí, a odráží složitost fyzikálních interakcí mezi střelou a jejím prostředím. (Further Applications of Newton's Laws: Friction, Drag, and Elasticity, 2013)

2.5.2 Gravitace

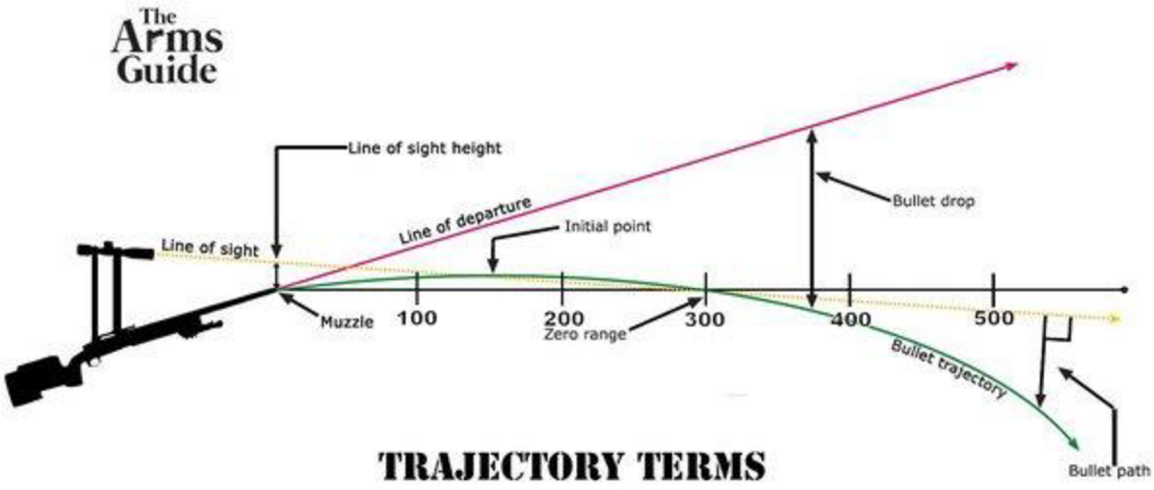
Síla, která značně ovlivňuje všechno na planetě Zemi, má na letovou dráhu vystřelené nábojnice i na střelce značný vliv. Podle definice lze gravitaci chápat jako zákon, který říká, že dva hmotné body se přitahují vzájemně stejně velikými, ale opačnými silami. Velikost gravitačních sil je přímo úměrná součinu hmotností těchto bodů a nepřímo úměrná mocnině jejich vzdáleností. (Brown B.Sc., Ph.D., F.Inst.P, 1969)

Předměty padají vlivem gravitace s gravitačním zrychlením 9,8 m/s^2. To má za výsledek, že kulka po sekundě pádu padat rychlostí 9,8 m/s. Poté bude její rychlost pádu narůstat o dalších 9,8 m/s každou další sekundu pádu. Střela tedy nejenže padá dál od střelce, čím déle k cíli letí, ale každou další sekundu padá rychleji. (Brown B.Sc., Ph.D., F.Inst.P, 1969)

Velký efekt na přesnost střelby má také rychlost letu kulky. Zásadní fakt, který střelu ovlivňuje, je ten, že na kulku působí úplně stejné gravitační zrychlení v letu jako v klidu. Za předpokladu, že by kulka letěla jen rovně, by se při experimentu, kdyby byl projektil spuštěn na zem z ruky ve stejné výšce jako je hlaveň zbraně přesně ve chvíli, kdy by kulka opustila hlaveň, dopadnou obě nábojnice na zem v úplně stejnou chvíli. Rozdíl dělá jen uražená vzdálenost, avšak jak je napsáno v předchozím odstavci, čím déle padá kulka k zemi, tím padá rychleji. To by znamenalo, že kdyby byla kulka poslána rovnoběžně k povrchu země, prakticky by ihned po opuštění hlavně neletěla na místo, které bylo cílem.

V realitě se však s efektem gravitace počítá už při konstrukci zbraně. Kulka proto po vystřelení létá kromě směrem od hlavně i mírně vzhůru. To ukazuje následující obrázek

Obrázek 8 - Terminologie střelby na dlouhou vzdálenost



Zdroj: The Arms Guide (2020)

Ve skutečnosti kulka v letu protne přímku vyslanou z hledí na cíl dvakrát. Jelikož kulka letí kvůli kompenzování gravitačních sil mírně vzhůru, protne tuto pomyslnou přímku poprvé ještě před námi požadovaným místem zásahu. Toto první protnutí je označeno na grafu jako initial point. Kulka následně urazí několik desítek metrů nad námi požadovanou výškou. V bodě, kdy přímku zaměřovače protne podruhé se říká zero range. Zero range tato vzdálenost je vzdálenost cíle, na který střelec střílí. Na tuto hodnotu je nastaveno hledí. Dále je na obrázku velmi přehledně znázorněn bullet drop neboli pád kulky. (Bullet Trajectory, 2020)

Střelci musí brát v úvahu tento nepřetržitý vliv gravitace a provádět korekce své střelby, zejména při střelbě na větší vzdálenosti, aby svůj cíl zasáhly.

2.5.3 Vítr

Efekt větru na trajektorii vystřelené kulky je zásadní faktor, který ovlivňuje jak dosah, tak přesnost střelby. Větrné podmínky mohou způsobit, že kulka během letu odchýlí od zamýšlené trajektorie, což vyžaduje od střelců, aby prováděli korekce zaměření, zvláště při střelbě na dlouhé vzdálenosti. Studie se zaměřují na různé aspekty, včetně vlivu různých rychlostí větru a jeho směru na trajektorii střely. I když je tento vliv v realitě nekonstantní a jeho síla se tedy může v čase rapidně měnit, ve hrách však často bývá vítr konstantní, proto není potřeba více tuto problematiku zkoumat. (Sierra Bullets, 2020)

3 Vlastní práce

3.1.1 Vytvoření skriptu FPS Controller

Při tvorbě samotného scriptu byl použit komponent CharacterController. Tato komponenta poskytuje vývojářům sadu nástrojů a funkcí specificky navržených pro simulaci fyzikálně věrohodného pohybu postav, aniž by bylo nutné se hlouběji zabývat složitostmi fyzikálního enginu Unity. Použitím CharacterControlleru lze dosáhnout výrazné optimalizace výkonu, jelikož je navržen pro efektivní zpracování pohybu postav bez zbytečného využití výpočetního výkonu na složité fyzikální výpočty. Funkce tohoto komponentu řeší v projektu problémy jako například plynulost pohybu, přesnou kontrolu nad postavou a celkově snížila složitost části projektu, na kterou není práce zaměřena.

Hned na začátku scriptu byly definovány základní proměnné jako rychlost pohybu a běhu, síla skoku gravitace, rychlost otáčení a limit otočení hráče nahoru. Ve funkci Start se následujícím kódem definoval komponent CharacterController. Také se zde uzamkl a zneviditelnil kurzor. Nezapomenout na kurzor je v žánru FPS velmi důležité, protože myší ovládáte míření a kameru hráče. Při vynechání těchto dvou řádků kódu by se velmi snadno mohlo hráči stát, že by kurzorem najel například na druhý monitor a při stisknutí levého tlačítka by se místo výstřelu okno se hrou minimalizovalo.

```
void Start()
    {
        characterController = GetComponent<CharacterController>();
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
    }
```

Ve funkci update je pomocí předem definovaných definování základních pohybových funkcí pro pohyb hráče v FPS hře. Jsou zde funkce, které zajišťují pohyb hráče, zjištění momentální rychlosti, skok a pohyb kamery. (Příloha 1)

Jelikož se tato práce zabývá primárně externí balistikou, tak jediné, co stojí za zmínku a je pro zkoumané mechaniky zásadní, je poslední region, jenž se zabývá ovládáním kamery. Funkce pro otáčení kamery je aktivní pouze v případě, že proměnná `canMove` má hodnotu `true`. Tato podmínka slouží k omezení pohybu hráče nebo kamery na základě specifických herních scénářů nebo událostí, jako jsou interakce v menu nebo během cutscén. Proměnná `rotationX` je upravena o hodnotu získanou z vertikálního pohybu myši (osa "Mouse Y"), která je násobena rychlostí pohledu (`lookSpeed`), tímto způsobem se docílí vertikální

rotace kamery. Aby se předešlo nadměrnému otočení kamery, které by mohlo vést k nežádoucím vizuálním efektům nebo dezorientaci hráče, je hodnota `rotationX` omezena funkcí `Mathf.Clamp` na interval mezi `-lookXLimit` a `lookXLimit`, což definuje maximální a minimální úhel pohledu nahoru a dolů. Po aplikaci těchto omezení je kamera otočena do nového vertikálního úhlu pomocí kvaternionu, který je vytvořen z hodnoty `rotationX`. Pro horizontální rotaci celého hráče je použita podobná logika, avšak bez omezení úhlu. Tato rotace je založena na horizontálním pohybu myši (osa "`Mouse X`") a aplikuje se přímo na transformaci objektu pomocí násobení jeho současné rotace kvaternionem vytvořeným z horizontálního pohybu myši a rychlosti pohledu. Tento mechanismus rotace poskytuje plynulou a intuitivní kontrolu nad pohledem hráče v herním prostředí.

3.2 Výběr vývojového prostředí

Pro provedení experimentu bylo vybráno vývojové prostředí Unity. Volba Unity jako platformy pro implementaci tohoto projektu byla motivována několika klíčovými faktory, které jsou podrobně rozebrány v následujících odstavcích.

Prvním a zásadním důvodem je vysoká míra flexibility a adaptability, kterou Unity nabízí. Díky široké škále nástrojů a komponent, které jsou ve vývojovém prostředí integrovány, bylo možné přizpůsobit a optimalizovat modelování externí balistiky tak, aby co nejvíce odpovídalo reálným fyzikálním zákonům a jevům. Unity umožňuje s vysokou přesností simulovat trajektorie letu projektilů, včetně zohlednění odporu vzduchu, gravitace a dalších faktorů ovlivňujících pohyb v reálném světě.

Unity také nabízí rozsáhlou dokumentaci a aktivní komunitu vývojářů, která je ochotná poskytnout podporu a sdílet znalosti. Tento aspekt je klíčový pro řešení potenciálních problémů a výzev, které se během vývoje projektu mohou objevit. Dostupnost rozsáhlých zdrojů a příkladů z praxe značně usnadňuje proces vývoje a umožňuje rychle nalézt řešení pro specifické požadavky projektu.

Unity poskytuje pokročilé grafické a vizuální nástroje, které umožňují realistickou vizualizaci simulací. Díky tomu je možné nejen přesně modelovat fyzikální jevy, ale také je atraktivně prezentovat, což zvyšuje srozumitelnost a edukační hodnotu projektu.

Z těchto důvodů bylo prostředí Unity zvoleno jako ideální platforma pro realizaci projektu modelování externí balistiky.

3.2.1 Vytvoření projektu

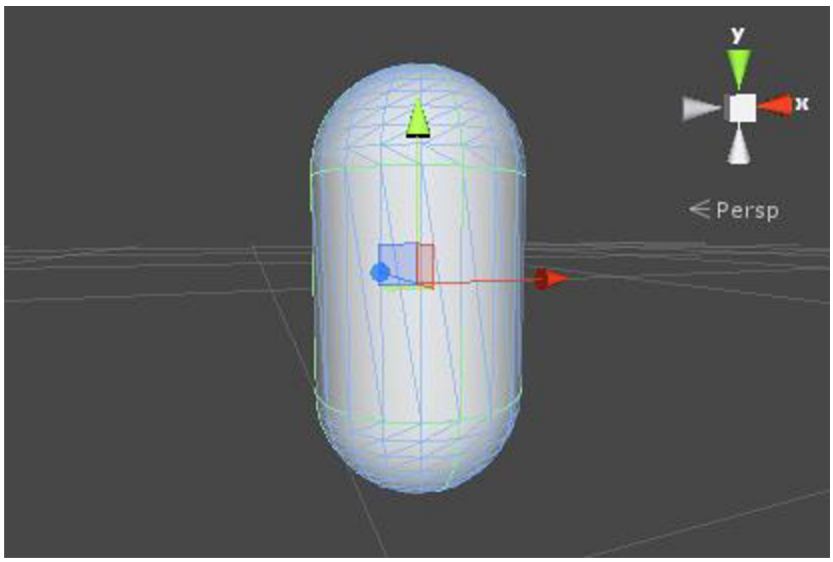
Celé měření začalo vytvořením projektu, který představuje základní kámen pro další postupy. Po pečlivém zvážení dostupných alternativ bylo rozhodnuto upřednostnit implementaci projektu s využitím šablony nastavení známé jako 3D HDRP. Výběr trojrozměrného prostředí je opodstatněn absencí adekvátní reprezentace žánru FPS v rámci dvourozměrného prostředí, což podtrhuje potřebu vysoké úrovně detailů a realističnosti, které mohou být dosaženy pouze v 3D. High Definition Render Pipeline není pro experiment nijak zásadně důležité. HDRP se vyznačuje svou schopností poskytnout výrazně detailnější nastavení v oblastech světelných efektů, shaderů, simulace mlhy, a tvorby mraků jako fyzických objektů. Tyto funkce byly považovány za klíčové pro správné zobrazení všech vybraných modelů, zejména s ohledem na vybraný model zbraně. Tento model byl

specificky navržen s využitím textur a technik HDRP, což vyžadovalo, aby byl celý projekt nastaven tak, aby odpovídal těmto vysokým grafickým standardům. Díky tomuto nastavení bylo možné zbraně v simulaci zobrazit s maximální možnou úrovní detailů a vizuální věrohodnosti.

3.3 Modely

Z důvodu úspory času bylo v průběhu řešení projektu rozhodnuto, že budou použity jednoduché modely z 3D objects knihovny ve vývojovém prostředí Unity společně s modely volně dostupnými na Unity Asset storu. Hned po vytvoření a nastavení projektu, bylo jedním z prvních kroků vytvoření a nastavení modelu hráče a jeho ovládání. Jako model byl vybrán standartní primitivní 3D objekt v Unity s názvem Capsule.

Obrázek 9 - Primitivní objekt Capsule. (Unity, 2013)



Zdroj: Unity (2023)

Tento objekt má tvar Je zde také možnost vytvoření modelu hráče pouze pomocí objektu zbraně a kamery, protože, jak bylo v teoretické části zmíněno, u singleplayerové hry vidí hráč pouze zbraň a ruce. Postup použitý v práci však nabízí daleko pro lepší podmínky vizuální kontrolu pozice hráče a v případě použití HDRP i vylepšení celkového vizuálního efektu. Jestliže by rozhodnutí nepoužít žádný model bylo implementováno, stín by vrhala pouze držená zbraň, což není nejlepší řešení. Tento model byl použit kvůli preferencím a úspoře času, jelikož hráčův model není při měření nijak zvlášť podstatný.

Jako model kulky byl kvůli problému, který je popsán v další části, zvolen primitvní model sphere, což je 3D model, který se opět nachází v knihovně Unity. Tento model má ve

hře průměr 20 cm. To bylo rozhodnuto z důvodu lepší viditelnosti při střelbě, ale také kvůli výše zmíněnému problému. (Unity, 2013)

Další dva modely jsou modely, které označují zasažené místo. Dva jsou z důvodu testování dvou zkoumaných metod. Jediné, v čem se tyto objekty liší, je barva. To pozorovatelům umožňuje zásahy daleko lépe rozeznat a ihned vidět efekt jednotlivých parametrů.

Další model, který je pro projekt zásadní, je model zbraně. I když se zpočátku zdálo sehnání tohoto modelu jako největší problém, po krátké době hledání byl objeven model zbraně AK-74.

Obrázek 10 - Použitý model zbraně AK-74

Zdroj: Creation Wasteland (2020)

Tento model vytvořil uživatel se jménem CREATION WASTELAND. Tento uživatel na svém profilu v Unity Asset Storu nabízí balíček zbraní pojmenovaný Cold War Weapon Collection. Tento balíček je zpoplatněný, avšak také nabízí jeden model zdarma na vyzkoušení a použití. Tím modelem je právě model zbraně AK-74 použitý v této práci.

Model je velmi detailně zpracován. Jelikož má každá část, jako například, spoušť, závěr, hlaveň nebo zásobník, svůj vlastní model, dá se tento asset použít na animování všech stavů, které mohou v realitě nastat. Dále, jak již bylo v úvodu praktické části zmíněno, má tento model textury dělané na HDRP. Zbraň tedy i v simulaci vypadá velmi hezky a působí opravdově. Po revizi modelu bylo také zjištěno, že mířidla zbraně jsou v souladu s realitou, a tedy plní svůj účel. Model proto splňuje všechny předpoklady, které od takto zásadního modelu očekáváme. (Creation Wasteland, 2020)

Pro model terče byl vybrán model Military target od uživatele NikiYani, který je také volně ke stažení.

Obrázek 11 - použitý model terče Military target

Zdroj: Vlastní zpracování, NikiYani (2019)

Tento model je také přesně vymodelován, avšak jde spíše u vizuální interpretaci výsledků měření. Více o tomto tématu je popsáno v dalších kapitolách.

Dále byl použit balíček textur Grid Prototype Materials (MLAgent, 2024) od uživatele MLAGENT, pro lépe vypadající textury podlahy a také Modular self-stand fence od uživatele Aleksey Kozhemyakin (Kozhemyakin, 2019), který do projektu přidal betonové ploty. Ty jsou v projektu použity na označení místa výstřelu, aby se poloha střelce při jednotlivých měřeních nijak zvlášť nelišila.

Důležitý model je také model dummy figuríny, na kterou se v projektu testuje střelba na model člověka, ke které všeobecně v FPS hrách dochází nejčastěji. Konkrétně byl použit Model 3D Character Dummy od uživatele Kevin Iglesias, který je taktéž volně ke stažení. (Iglesias, 2020)

3.4 Implementace fyzického projektilu

Jako první byl do projektu implementován script pro střelbu pomocí projektilu jako objektu ve hře. Tento postup je v projektu rozdělen do dvou scriptů. Prvním z nich má název WeaponPhysical, druhý má název BulletPhysics.

3.4.1 Script WeaponPhysical.cs

Tento script je přidán pod model zbraně z důvodu, že se parametry k němu přímo vztahují. V tomto projektu je možné najít ubraň pouze jednu, ale filozofie tohoto scriptu by implementaci dalších zbraní dovolovala bez větších zásahů do už stávajícího kódu.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
```

V tomto scriptu používáme i knihovny jako UnityEngine, UnityEngine.UI a TMPro. První je z důvodu správné implementace do Unity projektu. Další dvě jsou zde kvůli zobrazení informací do UI. Ukazatelům v UI je věnován prostor dále v této kapitole.

Další krok zahrnuje implementaci proměnných.

```
    public GameObject bulletPrefab; // Objekt, který bude vystřelen
    public GameObject ammoPrefab;
    public Animator gunAnimator;
    public Transform firePoint; // Bod odkud letí kulka, v projektu jako
empty Game object
    public int maxAmmo = 30; // Max počet nábojů
    private int currentAmmo; // Současný počet nábojů
    public float reloadTime = 1f; // Čas přebíjení
    private bool isReloading = false; // Logická proměnná pro zjištění stavu
nabíjení
    public TMP_Text ammoText;  // Proměnná ukazetele počtu nábojů v UI
    private Vector3 originalPosition = new Vector3(0.13f, -0.149f, 0.11f) ;
// Idle pozice zbraně
    private Vector3 adsPosition =   new Vector3(0, -0.136f, -0.1f); //
Pozice zbraně při míření mířidly
    public float moveSpeed = 5f; // Rychlost pohybu zbraně do pozice ADS
```

Jelikož jsou je u každé proměnné vysvětlující komentář, nebude zde znovu každá proměnná vysvětlena. Bude jim však věnováno více času u jednotlivých funkcí.

Při startu programu nastavíme počet kulek v zásobníku na maximální hodnotu a posuneme model zbraně na požadované místo. Tato poloha je poloha zbraně při střelbě od boku.

```
void Start()
    {
        currentAmmo = maxAmmo;
        transform.localPosition = originalPosition; // nastavení zbraně na
idle pozici
    }
```

V metodě Update se řeší několik klíčových aspektů ovládání a chování zbraně. Prvním krokem je kontrola, zda probíhá proces přebíjení; pokud ano, další akce se neprovádějí, což zamezuje střelbě nebo jiným akcím během přebíjení. Dalším krokem je kontrola množství zbývající munice. V případě, že hráč nemá žádnou munici, je zahájen proces přebíjení.

Další část kódu se věnuje reakci na vstupy od hráče. Pokud hráč stiskne tlačítko určené pro střelbu a má dostatečnou munici, dojde k výstřelu. Zároveň je zde implementována logika pro zaměřování, kde při držení příslušného tlačítka dochází k plynulému přesunu zbraně do pozice pro zaměření a při uvolnění tlačítka se zbraň vrací do výchozí pozice. Tento pohyb je realizován s využitím lineární interpolace mezi aktuální a cílovou pozicí zbraně, přičemž rychlost pohybu je modifikována proměnnou určující rychlost pohybu a uplynulým časem, což zajistí hladký přechod mezi pozicemi

```
void Update()
    {
        if (isReloading)
            return;

        if (currentAmmo <= 0)
        {
            StartCoroutine(Reload());
            return;
        }

        if (Input.GetButtonDown("Fire1") && currentAmmo > 0)
        {
            Shoot();
        }
        if (Input.GetMouseButton(1)) // Zaměřování
        {
```

```
            // Smooth pohyb zbraně při zamíření
            transform.localPosition = Vector3.Lerp(transform.localPosition,
adsPosition, moveSpeed * Time.deltaTime);
        }
        else
        {
            // Vrácení zbraně do pozice smootly :)
            transform.localPosition = Vector3.Lerp(transform.localPosition,
originalPosition, moveSpeed * Time.deltaTime);
        }
    }
```

Metoda je definována jako IEnumerator z důvodu, že využívá Coroutines v Unity. Coroutines jsou mocný nástroj v Unity, umožňující provádění asynchronních operací a časově závislých úloh bez nutnosti použití komplexních vláken nebo přerušení hlavního výpočetního cyklu hry. Coroutine umožňuje rozložit úlohy do několika snímků. V Unity je Coroutine ta metoda, která může pozastavit provádění a vrátit control Unity, ale v následujícím snímku může pokračovat tam, kde skončila. Specificky v kontextu této metody Reload(), použití IEnumerator umožňuje přerušení běhu metody na určitou dobu, což je v tomto případě realizováno pomocí yield return new WaitForSeconds(reloadTime);. Tato část kódu způsobí, že Coroutine počká na uplynutí definované doby (reloadTime) před pokračováním ve zbytku metody. Využití IEnumerator a Coroutines v Unity je ideální pro situace, jako je čekání na uplynutí času (např. přebíjení zbraně), provádění postupných animací, nebo asynchronní načítání zdrojů, protože to umožňuje, aby hra zůstala reaktivní a neblokovala hlavní výpočetní vlákno během těchto operací.

```
IEnumerator Reload() //přebíjení
    {
        isReloading = true;
        Debug.Log("Reloading...");
        yield return new WaitForSeconds(reloadTime);
        currentAmmo = maxAmmo;
        isReloading = false;
        UpdateAmmoDisplay();
    }
```

V metodě pro přebíjení, implementované jako Coroutine, se adresuje proces obnovy munice zbraně v kontextu videohry. Na počátku metody je nastaven příznak přebíjení na hodnotu pravda, čímž metoda signalizuje, že přebíjení probíhá. Tento příznak zabraňuje vykonávání dalších akcí, jako je střelba, během procesu přebíjení. Následně se vypíše do

konzole zpráva o probíhajícím přebíjení. Pro simulaci reálného času potřebného k přebíjení je využito čekání na uplynutí určité doby, definované proměnnou reloadTime. Po uplynutí této doby se hodnota aktuální munice nastaví na maximální hodnotu munice, čímž se dokončí proces přebíjení. Na závěr je příznak přebíjení nastaven zpět na hodnotu nepravda, což umožňuje opětovné vykonávání akcí zbraní. Aktualizace zobrazení munice je posledním krokem, který zajistí, že HUD zobrazuje nový stav munice po dokončení přebíjení.

Implementace samotné střelby je v metodě Shoot(), která vypadá následovně.

```
void Shoot()
    {
        Debug.Log("Shooting...");
        currentAmmo--;
        UpdateAmmoDisplay();

        GameObject bullet = Instantiate(bulletPrefab, firePoint.position,
firePoint.rotation);
        BulletPhysics bulletPhysics = bullet.GetComponent<BulletPhysics>();

        if (bulletPhysics != null)
        {
            // Set původní rychlosti
            bulletPhysics.initialVelocity = firePoint.forward;
            gunAnimator.Play("AK74_Fire"); // Use Play if you want to
directly play the animation
        }
        else
        {
            Debug.LogError("Bullet prefab does not have a BulletPhysics
component.");
        }
    }
```

Hned na začátku metody je v logu zobrazena hláška o tom, že střelba probíhá, a následně je odebrána z počtu aktuální munice jedna jednotka, která bude vystřelena. Hned na dalším řádku se nachází volání funkce na aktualizaci počtu munice v HUD.

Následně je demonstrováno vytváření instance objektu střely s názvem bullet pomocí metody Instantiate, která jako parametry přijímá prefab střely, který je v kódu nazván jako bulletPrefab, pozici (firePoint.position) a rotaci (firePoint.rotation) místa výstřelu. Tato instance je poté typově svázána s komponentou BulletPhysics za účelem manipulace s

fyzikálními vlastnostmi střely. Metoda GetComponent<BulletPhysics>() je využita k získání přístupu k této komponentě přiřazené k objektu střely, což umožňuje následné nastavení počáteční rychlosti střely na základě vektoru směru místa výstřelu firePoint.forward. V případě, že komponenta BulletPhysics je úspěšně nalezena, je také spuštěna animace výstřelu zbraňového modelu pomocí metody Play animatoru zbraně, který je identifikován řetězcem "AK74_Fire".

V situaci, kdy komponenta BulletPhysics není nalezena na prefabu střely, je vygenerována chybová zpráva pomocí Debug.LogError. Tato zpráva informuje o chybějící komponentě, což je kritické pro odhalení a opravu chyb v konfiguraci prefabů a přiřazení správných věcí v inspektoru. Význam tohoto přístupu spočívá ve zvýšení robustnosti kódu tím, že se explicitně ověřuje přítomnost nezbytných komponent a zajišťuje, že objekty ve scéně budou fungovat, jak je očekáváno. Tímto způsobem je zajištěna konzistence a spolehlivost interakcí ve hře, což je nezbytné pro vytváření plynulého a imerzního herního zážitku.

Poslední část tohoto scriptu je výše použitá metoda `UpdateAmmoDisplay();`

```
public void UpdateAmmoDisplay()
    {
        if (ammoText != null) //Ammo text checking
        {
            ammoText.text = $"Ammo: {currentAmmo}/{maxAmmo}";
        }
        else
        {
            Debug.LogError("ammoText is not assigned in the inspector");
        }
    }
```

Tato funkce je velmi jednoduchá. V případě, že je přiřazen ammoText ke komponentě, dojde k updatu počtu nábojů na obrazovce hráče. Jestliže text není přiřazen, je vývojáři ukázána chybová hláška.

3.4.2 **Script BulletPhysics.cs**

V tomto scriptu jsou stanoveny a vypočítávány všechny vlivy, které na gameObject kulky po vystřelení působí. V úvodní části skriptu, jsou začleněny direktivy pro použití standardních knihoven a rozhraní. Direktivy using System.Collections a using System.Collections.Generic jsou zásadní pro manipulaci s kolekcemi dat, jako jsou seznamy a slovníky, které umožňují efektivní správu skupin objektů v rámci kódu. using Unity.VisualScripting umožňuje integraci vizuálního skriptování do projektu, což rozšiřuje možnosti vývoje při tvorbě logiky her bez nutnosti detailního kódování. using UnityEngine; je vysvětleno v minulé kapitole.

```
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;
```

```
Dále je podle standartního postupu definovaná třída a na jejím počátku jsou
inicializované důležité proměnné, které jsou ve výpočtech a funkcích scriptu
nepostradatelné.
```

```
public class BulletPhysics : MonoBehaviour
{
    public GameObject bulletHolePrefab; // Model označení zásahu
    public Vector3 initialVelocity;
    public float muzzleVelocity = 100f; // Úsťová rychlost
    public float gravity = -9.81f;   // Síla gravitace
    public float dragCoefficient = 2.25f; // Koeficient odporu pro kulku
    public float bulletDiameter = 0.00762f; // Diametr kukly v metrech pro
7.62 mm kulku
    public Vector3 windEffect = new Vector3(0, 0, 0); // Efekt větru

    public Vector3 currentVelocity;  // současná rychlost
    private Rigidbody rb;
    private float airDensity = 1.225f; //hustota vzduchu na úrovni moře v
kg/m^3
```

V kódu jsou opět komentáře, tudíž je další popis nepotřebný. Metoda Start v rámci skriptu BulletPhysics je klíčová pro inicializaci střely ihned po jejím vytvoření ve scéně. Při prvním volání této metody dochází k přiřazení komponenty Rigidbody, která je nezbytná pro simulaci fyziky, k proměnné rb. Toto přiřazení umožňuje dále manipulovat s fyzikálními vlastnostmi střely, jako je její rychlost a gravitace. Specificky, metoda nastavuje počáteční rychlost střely currentVelocity na základě normalizovaného směru initialVelocity a definované úsťové rychlosti muzzleVelocity, zároveň vypíná vliv gravitace na

komponentu Rigidbody, aby se simulovalo realističtější chování střely. Normalizace vektoru znamená jeho škálování tak, aby měl jednotkovou délku (normu), což znamená, že jeho velikost bude rovna 1. Dále metoda definuje automatické zničení objektu střely po uplynutí 30 sekund, čímž se předchází přetížení herního prostředí zbytečně existujícími objekty, které již nejsou v akci. Při tvorbě standartní hry by tato funkce měla být spuštěna při zásahu neprostupného objektu, nebo jestliže objekt ztratil veškerou rychlost. Pro účely testu se však odstranění objektu až po časovém úseku hodí více.
Další následuje funkce definice metody FixedUpdate. Metoda FixedUpdate je v Unity speciálně navržena pro aktualizaci fyzikálních operací a je volána v pevných časových intervalech, což zajišťuje stabilní provádění fyzikálních výpočtů nezávisle na snímkové frekvenci hry. Tento přístup je klíčový pro zachování konzistence simulace fyziky, jako je pohyb objektů, aplikace sil a detekce kolizí, což je zásadní pro realistické chování a předvídatelné interakce v herním prostředí.

V kontextu skriptu BulletPhysics, FixedUpdate se používá pro aplikaci gravitačních sil, odporu vzduchu a vlivu větru na střelu. Tyto síly vyžadují pravidelnou aktualizaci, aby se přesně simuloval trajektorie letu střely a její interakce s okolním prostředím. Použitím FixedUpdate pro tyto výpočty se zajišťuje, že vlivy jsou aplikovány konzistentně na každou střelu, nezávisle na výkonnosti systému nebo variabilitě snímkové frekvence, což je zásadní pro udržení fyzikální integrity a realismu ve hře.

```
void FixedUpdate()
    {
        // Použití mých vlivů
        ApplyGravity();
        ApplyAirResistance();
        ApplyWindEffect();
        UpdatePosition();
    }
```

Níže v kódu jsou metody jednotlivých faktorů ovlivňující kulku. Jako první je v kódu definovaná funkce ApplyGravity();

```
void ApplyGravity()
    {
        //metoda gravitace
        currentVelocity += Vector3.up * gravity * Time.fixedDeltaTime;
    }
```

Funkce ApplyGravity v rámci skriptu BulletPhysics je navržena k simulaci působení gravitace na střelu. V této metodě se aktuální vektor currentVelocity inkrementálně upravuje přidáním vektoru gravitace, který je směrován dolů. V kódu je reprezentován jako Vector3.up s negativní hodnotou gravity. Tento vektor je vynásobeným časovým krokem od posledního volání FixedUpdate díky Time.fixedDeltaTime. Tento přístup zajišťuje, že gravitační síla je aplikována kontinuálně a konzistentně na střelu, simulujíc parabolickou trajektorii v důsledku gravitačního zrychlení. Použití Time.fixedDeltaTime zaručuje, že aplikace gravitační síly je nezávislá na snímkové frekvenci, což přispívá k předvídatelnosti a realismu pohybu střely v proměnlivých výkonnostních podmínkách.

Další faktor, který kulku v simulaci ovlivňuje je odpor větru. Tento faktor je simulován funkcí ApplyAirResistance(); Tato metoda vychází ze vzorce v teoretické části práce.

```
void ApplyAirResistance()
    {
        // spočítání plochy kulky
        float area = Mathf.PI * Mathf.Pow(bulletDiameter / 2f, 2);

        // Počítání rychlosti kulky vůči vzduchu
        Vector3 velocity = currentVelocity - windEffect;
        float speed = velocity.z;

        // spočítání odporu
        float dragForceMagnitude = 0.5f * airDensity * speed * speed *
dragCoefficient * area;
        Vector3 dragForce = -dragForceMagnitude * velocity.normalized;

        // použití odporu vzduchu
        currentVelocity += dragForce* Time.fixedDeltaTime;
    }
```

Nejprve vypočítá průřezovou plochu kulky využitím vzorce pro plochu kruhu, což umožňuje přesnější aproximaci vzdušného odporu, který kulka za letu podléhá. Následně se určí relativní rychlost kulky vzhledem k okolnímu vzduchu odečtením efektu větru od její aktuální rychlosti, což zohledňuje jak přímočarý pohyb, tak i působení externích sil. Na základě této rychlosti a vypočtené plochy se určí velikost síly vzdušného odporu pomocí empirického vzorce, který zahrnuje hustotu vzduchu, koeficient odporu, rychlost pohybu a průřezovou plochu objektu. Výsledná síla odporu se aplikuje proti směru pohybu kulky,

čímž se snižuje její rychlost v souladu s fyzikálními zákony. Použití Time.fixedDeltaTime opět zajišťuje, že aktualizace rychlosti je provedena s konzistentním časovým krokem.

Efekt větru je simulován velmi jednoduchou funkcí `void ApplyWindEffect()`. V simulaci je uvažováno, že je síla větru konstantní, a proto je možné pouze přičíst k pohybu objektu působení větru.

```
void ApplyWindEffect()
    {
        // Jednoduchý efekt větru
        currentVelocity += windEffect * Time.fixedDeltaTime;
    }
```

Další funkce pouze aktualizuje pozici vystřelené kulky na pozici novou, pod podmínkou, že je objekt přiřazený.

```
void UpdatePosition()
    {
        // Aktualizace polohy střely na základě aktuální rychlosti
        if(rb != null)
        {
            rb.MovePosition(rb.position + currentVelocity *
Time.fixedDeltaTime);
        }
    }
```

Následující funkce zajišťuje všechny události, které musí proběhnout po zasáhnutí cílového objektu kulkou a při této události je i vyvolána.

```
void OnCollisionEnter(Collision collision)
    {
        // Zkontrolujte, zda jsme zasáhli cílovou vrstvu
        if (collision.gameObject.layer == LayerMask.NameToLayer("Target"))
        {
            // Instancujte díru po kulce v místě kolize.
            GameObject bulletHole = Instantiate(bulletHolePrefab,
collision.contacts[0].point,
Quaternion.LookRotation(collision.contacts[0].normal));
            Debug.Log(collision.contacts[0].point + " Physical model");

            // možnost přiřadit otvor po kulce k zasaženému objektu pro
správný pohyb
            bulletHole.transform.SetParent(collision.transform);

            // Vypnutí fyzikálních prvků střely a její zneviditelnění.
```

```
            DisableBullet();
            Destroy(gameObject);
        }
    }
```

Když střela zasáhne objekt definovaný jako cíl, což kontroluje na základě přiřazené vrstvy, funkce nejprve ověří, že kolize skutečně nastala s objektem patřícím do specifikované vrstvy. Tato kontrola je zajištěna porovnáním vrstvy objektu, s nímž došlo ke kolizi, s vrstvou označenou jako "Target".

Po identifikaci kolize s cílovým objektem, skript dynamicky vytvoří vizuální reprezentaci díry po kulce v místě dopadu. Toto je realizováno vytvořením instance prefabu bulletHolePrefab na pozici prvního bodu kontaktu, kterou lze najít uloženou v proměnné collision.contacts[0].point a s orientací kolmou na povrch nárazu (Quaternion.LookRotation(collision.contacts[0].normal)), což simuluje realistické usazení díry po kulce na povrchu cíle. V této podmínce je také tento bod uložen a zaslán do logu. To je zde implantováno primárně kvůli měření efektu jednotlivých faktorů, které střelbu ovlivňují.

Dalším krokem je integrace díry po kulce s cílovým objektem tím, že se nastaví jako jeho potomek objektu bulletHole.transform.SetParent(collision.transform), což zajišťuje, že jakékoli pohyby nebo transformace cílového objektu se přenesou i na objekt zásahu a ten se bude hýbat i transformovat společně se zasaženým objektem. To v projektu není nijak podstatné, protože testy probíhají na statických cílech, avšak jelikož je tento postup korektní pro vývoj jakékoliv herní aplikace, byl použit i zde.

Nakonec, po vytvoření vizuálního efektu díry po kulce, metoda DisableBullet je volána k deaktivaci fyzikálních a vizuálních komponent střely, což efektivně odstraňuje střelu ze hry bez okamžitého zničení objektu. Následné zničení objektu střely (Destroy(gameObject)) tedy působí zbytečně, avšak v průběhu testování byl v této části kód dynamicky upravován podle aktuální potřeby, proto jsou v této práci ukázány obě varianty.

Další funkce, která v scriptu upravujícího fyziku kulky je, se nazývá DisableBullet. Tato funkce postupně deaktivuje všechny vlastnosti kulky, které by jakkoliv mohli nadále

ovlivňovat cokoliv ve hře. První je nastavena rychlost objektu na 0 a parametr vlastnosti rigidbody na true. Jeli tento parametr nastaven na hodnotu true, nebude objekt automaticky ovlivněn silami, kolizemi a jinými dynamickými interakcemi v rámci fyzikální simulace Unity. Dále je kulce deaktivován collider, a nakonec i renderer, takže kulka není vidět a nemůže reagovat s žádným dalším objektem. Na poslední řádce je zakomentován příkaz pro úplné vypnutí scriptu, avšak názorná ukázka toho, co všechno objekt ve hře tvoří a možnost vypnout jen některou z částí také není k zahození

```
private void DisableBullet()
    {
        if(rb != null)
        {
            rb.velocity = Vector3.zero;
            rb.isKinematic = true;
        }

        Collider collider = GetComponent<Collider>();
        if(collider != null)
        {
            collider.enabled = false;
        }

        Renderer renderer = GetComponent<Renderer>();
        if(renderer != null)
        {
            renderer.enabled = false;
        }

        // Zavřít skript
        // this.enabled = false;
    }
```

Poslední dvě funkce jsou v tomto scriptu pouze za účelem měření trajektorie kulky. První z nich zaznamenává pozici kulky každou sekundu, kterou kulka existuje, což z poskytnutých dat dá velké možnosti, jak podrobně zkoumat její letovou dráhu.

```
    IEnumerator LogBulletData()
{
    string filePath = Path.Combine(Application.persistentDataPath,
"bulletDataLog.txt");

    while (true)
```

```
    {
        LogDataToFile(filePath);
        yield return new WaitForSeconds(1f); // Počkej vteřinu
    }
}
```

Na ní navazující a také poslední funkce tohoto scriptu LogDataToFile následně všechny pozorovaná data zapisuje do souboru, jenž se nachází na přede definované adrese filePath. Funkce také zasílá hlášku do konzole, že data do souboru zalogovala.

```
void LogDataToFile(string filePath)
{
    // Kontrola komponenty rigidbody na objektu
    if (rb != null)
    {
        Vector3 position = transform.position; // Současná pozice
kulky

        // Příprava stringu
        string dataToLog = $"Time: {Time.timeSinceLevelLoad}; Position:
{position} \n";

        // Zápis dat do souboru
        File.AppendAllText(filePath, dataToLog);

        // Zápis dat do konzole
        Debug.Log($"Data logged to {filePath}");
    }
}
```

3.5 Implementace HitScan pomocí Raycastu

Vytvoření scriptu pro implementaci střelby pomocí Raycastu, bylo daleko méně náročné. Jelikož tato metoda nevyžaduje žádné vytvoření další objektů a prostředí Unity je předefinovaná, stačí pouze vytvořit script, kde funkci zavoláme a nastavíme její parametry, jak bylo diskutováno v teoretické části.

3.5.1 Script WeaponRayCast.cs

Jelikož má script pouze 30 řádků je zde uveden celý najednou.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class WeaponRayCast : MonoBehaviour
{

    public Transform firePoint; // Bod odkud letí kulka
    public float range = 1000f; // max vzdálenost hitu
    public LayerMask hitLayers; // Layers s kterými reaguje

    void Update()
    {
        if (Input.GetButtonDown("Fire1"))
        {
            Shoot();
        }
    }

    void Shoot()
    {
        RaycastHit hit;
        if (Physics.Raycast(firePoint.transform.position,
firePoint.transform.forward, out hit, range, hitLayers))
        {
            Debug.Log(hit.transform.name + " hit");
            Debug.Log(hit.point + " Raycast");
        }
    }
}
```

Na začátku jsou opět definovány knihovny a prostředí, které třída využívá. Protože se nijak neliší od těch použitých v předchozích metodách, není zde potřeba nijak podrobný popis, až na použití UnityEngine, který je v kódu zásadní i z jiných důvodů než před tím, a to hlavně kvůli své knihovně Physics, která mimo jiné obsahuje i pro tuto metodu podstatný Raycast.

Jelikož mají proměnné opět vysvětlující komentáře a v podstatě nejsou nijak zásadně odlišné od proměnných použitých v předchozím scriptu nebudou zde zásadně vysvětleny. Funkce Update kontinuálně sleduje, zda hráč stiskl tlačítko pro výstřel, pomocí Input.GetButtonDown("Fire1"). Pokud ano, je vyvolána metoda Shoot, která provádí samotný Raycast.

Metoda Shoot je jádrem této třídy. Zde se provádí Raycast z bodu firePoint směrem vpřed. To definuje použití proměnné firePoint.transform.forward. Paprsek putuje až do nastavené vzdálenosti neboli range a reaguje pouze v rámci definovaných vrstev hitLayers. Funkčnost byla detailně popsána v teoretické části této práce. Pokud Raycast zasáhne objekt, je zaznamenán jako RaycastHit hit, což umožňuje další zpracování, jako je aplikace

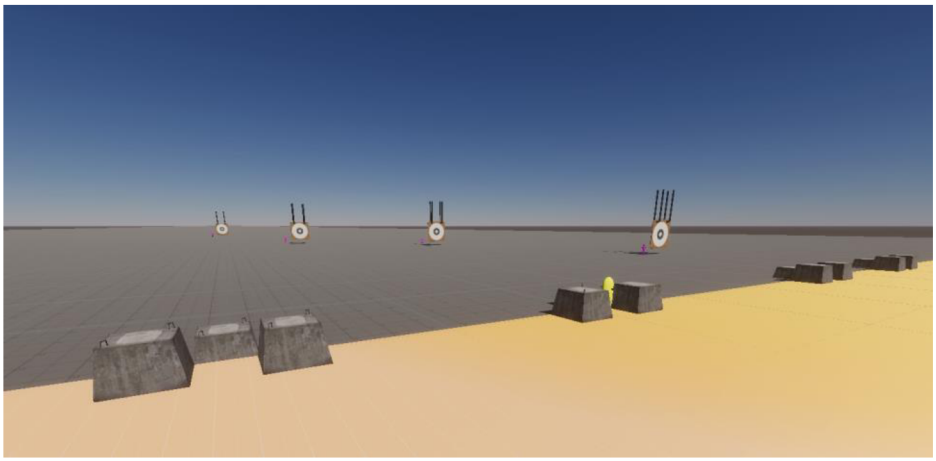
poškození na zasažený objekt nebo vizualizace dopadu střely. Záznam o zásahu a jeho lokaci je opět zaslán do logu, aby při testování byla naměřena přesná data.

3.6 Tvorba scény

Pro účely měření je nutné scénu modelovat velmi stručně a minimalisticky. Jako zem byl zvolen primitivní model z Unity 3D objects s názvem plain. Na něj byla zvolena textura z výše zmíněného texture packu s názvem Scalable Grid Protype Materials, která vypadá profesionálně a zlepšuje vizuální interpretaci hloubky. Objekty, na které se v testu střílí, jsou dva. Jeden má model armádního terče, který byl taktéž zmíněný výše. Druhý je model dummy figuríny představující postavu hráče. Tento model byl vybrán z důvodu, že nejčastější cíl palby je nějaká postava. Dále byly do modelu přidány betonové bloky, pro snadnou indikaci míst střelby.

Oba terče byly postaveny vedle sebe ve vzdálenostech 30, 80, 100 a 200 metrům. Scéna ve své finální podobě vypadá takto.

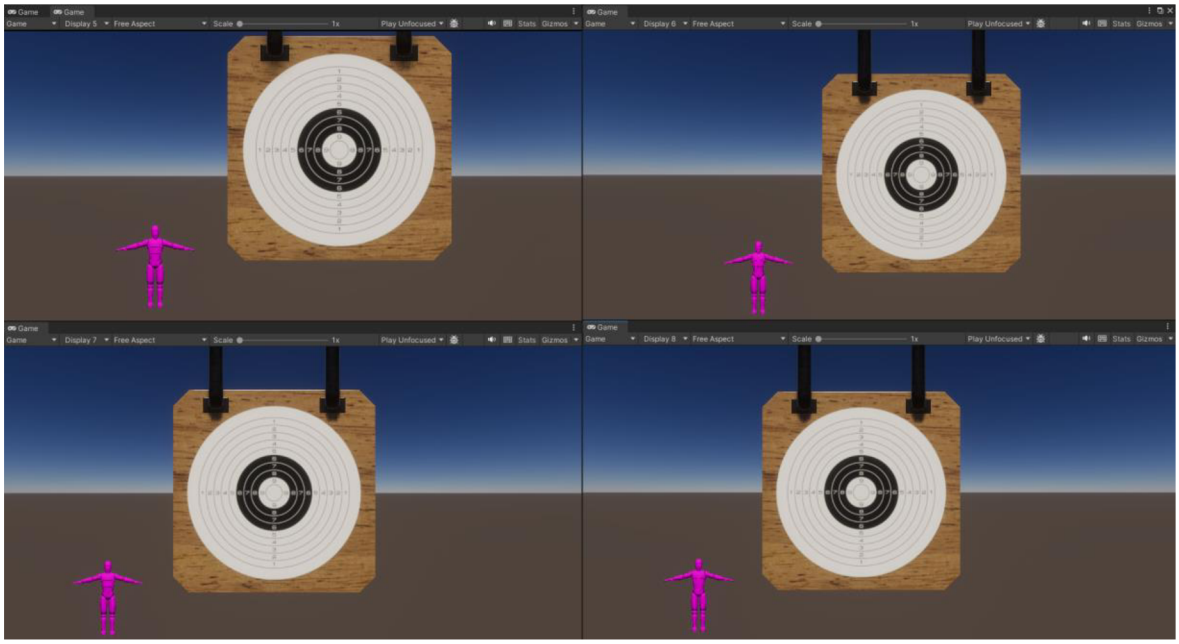
Obrázek 12 - Scéna testovacího prostředí



Zdroj: Vlastní zpracování, Unity(2023)

Dále byla ke každému z cílů přiřazena kamera pro lepší vizualizaci zachycení zásahů. Dále musela proběhnout korekce pozice zbraně při míření, aby pohled přes mířidla odpovídal reálnému místu dopadu.

Obrázek 13 - Ukázka kamer sledujících cíle



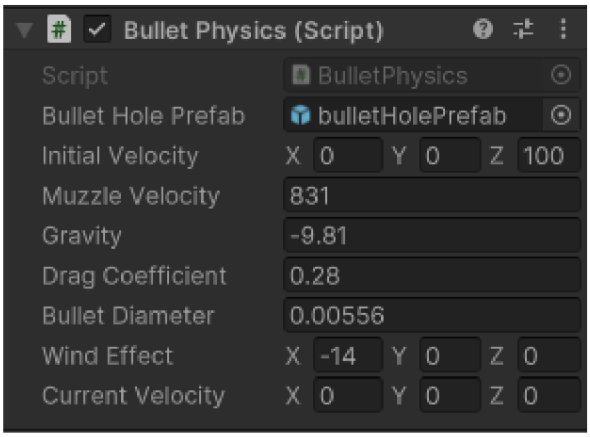
Zdroj: Vlastní zpracování, Unity(2023)

3.7 Měření

3.7.1 Nastavení parametrů

Jelikož jsou všechny důležité parametry public, lze je nastavovat v inspektoru bez nutnosti otevírat a přepisovat kód. Nutno podotknout, že při přepisování těchto parametrů při spuštěné aplikaci se po každém skončení aplikace hodnoty resetují, takže je lepší nastavovat hodnoty mezi jednotlivými spuštěními. První sada testů je prováděna střelbou na terče a figuríny. Druhá sada testů bude vystřelení pod úhlem 45 stupňů a následné sledování trajektorie projektilu. Nastavení parametrů kulky je pro první sadu tesů následující:

Obrázek 14 - Nastavení parametrů fyziky kulky



Zdroj: Vlastní zpracování, Unity(2023)

Všechny tyto hodnoty se snaží simulovat reálné podmínky. Úsťová rychlost je nastavená na 831 m/s, což odpovídá realitě. Gravitační zrychlení je nastaveno na -9,81 z důvodu směru síly. Koeficient odporu má hodnotu 0,28 a průměr kulky je nastaven na 5,56mm, což je ráže, kterou zbraň AK-74 používá.

3.7.2 Střelba pod úhlem 45°

Při prvním měření vystřelil hráč v 45° úhlu do vzduchu. V tabulce lze vidět hodnoty, které byly během letu zaznamenané.

Tabulka 1 - Data měření výstřelu, 45°

			x	y	z
Time	48,67	**Position**	0,03	2,07	0,37
Time	49,67	**Position**	-10,75	569	606,8
Time	50,68	**Position**	-35,67	1125,52	1212,73
Time	51,68	**Position**	-74,55	1669,46	1815,75
Time	52,68	**Position**	-127,31	2201,49	2416,46
Time	53,68	**Position**	-194,28	2724,77	3018,47
Time	54,69	**Position**	-274,98	3235,66	3617,54
Time	55,69	**Position**	-369,68	3736,28	4216,04
Time	56,69	**Position**	-478,49	4227,07	4814,58
Time	57,69	**Position**	-600,84	4705,65	5410,17
Time	58,70	**Position**	-737,71	5175,81	6007,55
Time	59,70	**Position**	-887,57	5632,45	6600,23
Time	60,71	**Position**	-1052,35	6081,51	7195,85
Time	61,71	**Position**	-1230,11	6517,63	7787,37
Time	62,71	**Position**	-1421,99	6943,92	8378,93
Time	63,71	**Position**	-1627,43	7359,1	8968,76
Time	64,72	**Position**	-1846,8	7764,03	9557,9
Time	65,72	**Position**	-2078,92	8156,77	10143,55
Time	66,72	**Position**	-2325,12	8539,7	10729,23
Time	67,72	**Position**	-2585,46	8912,81	11314,97
Time	68,72	**Position**	-2858,3	9273,91	11897,3
Time	69,73	**Position**	-3145,22	9625,21	12479,7
Time	70,73	**Position**	-3445,98	9966,34	13061,56
Time	71,73	**Position**	-3759,95	10296,66	13641,67
Time	72,73	**Position**	-4087,39	10616,52	14220,7
Time	73,74	**Position**	-4428,29	10925,94	14798,63
Time	74,74	**Position**	-4783,34	11225,53	15376,63
Time	75,74	**Position**	-5150,73	11513,86	15951,81
Time	76,74	**Position**	-5531,87	11792,1	16526,23
Time	77,74	**Position**	-5926,38	12060,01	17099,64

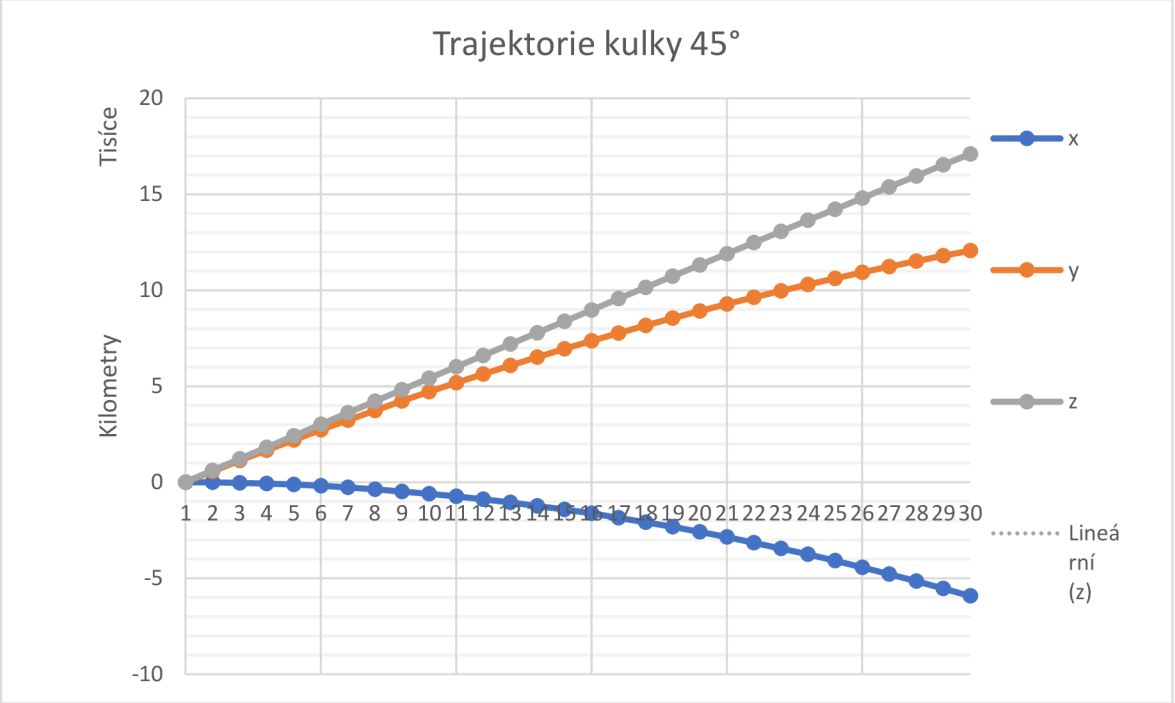
Zdroj: Vlastní zpracování, Unity(2023)

Úhel 45° je znám tím, že teoreticky poskytuje maximální dosah ve vakuu, když se zanedbává odpor vzduchu, což je důsledkem ideální rovnováhy mezi dosaženou výškou a dosaženým horizontálním dosahem.

Z počáteční polohy téměř u nuly je vidět, že střela nejprve získává na výšce a horizontální vzdálenosti, což je očekávané, protože úvodní fáze letu je charakterizována vzestupem pod vlivem počáteční rychlosti a úhlu výstřelu. To je dobře vidět na rychlém nárůstu hodnot y a z, zatímco posun x vlivem větru zůstává relativně nízký.

Jak projektil postupuje, zaznamenáváme zvýšený horizontální posun (x) spolu s pokračujícím, ale zpomalujícím růstem ve výšce (y), což naznačuje, že střela dosahuje vrcholu své trajektorie. Tento bod, kdy vertikální složka rychlosti projektilu dosáhne nuly a začne klesat zpět k zemi, je kritickým momentem v letu střely.

Graf 1 - Poloha kulky vystřelené v úhlu 45° vzhledem k horizontu



Zdroj: Vlastní zpracování

Princip těchto měření spočívá ve výstřelu na libovolné místo cíle a porovnání míst dopadu obou mechanik. U tohoto testu je využito přesnosti HitScan metody, podle které byly také srovnávána mířidla a pozice zbraně, takže je možno tento bod brát jako bod střely, která nebyla nijak ovlivněna. Na každý terč bylo vystřeleno v celku třikrát. Jak bylo možno vidět ve scriptech, obě mechaniky se pouštějí stejným stiskem levého tlačítka myši, takže také začnou v úplně stejný okamžik. Díky implementaci kódu pro zaznamenání místa dopadu do logu lze zanést do tabulek přesné hodnoty x a y souřadnice dopadu střely.

U střelby na terč je souřadnice **z** vždy stejná pro všechny výstřely. Při střelbě na figurínu se však tato souřadnice lišila. Po několika pokusech bylo zjištěno, že tento rozdíl dělá funkčnost mechaniky fyzického projektilu a colliderů. Ty totiž zaznamenají zásah pouze ve chvíli, kdy program kontroluje jejich střety. Může se tedy stát, že mezi těmito kontrolami v letí projektil do collideru a zásah se zaznamená až po několika centimetrech

průniku. To byl u collideru figurínu, který není vůbec velký zpočátku problém. Řešením se ukázalo být několik způsobů. První byl zpomalení letící kulky, avšak jelikož je zrovna rychlost velmi zásadní pro toto měření, byl hledán způsob jiný. Zvětšení modelu kulky a tím i jejího collideru se ukázalo jako ideální způsob, jak tento problém vyřešit. Nejen, že jdou doteky colliderů daleko lépe kontrolovat, ale také jde projektil v letu vidět, což pomáhá při případné korekci střelby.

Důležitý pro test jsou primárně rozdíly hodnot než čísla samotné. Všechny hodnoty v tabulkách jsou uvedeny v metrech.

3.7.2.1 **Měření střelby na 30 metrů**

Tabulka 2 - Záznam polohy koliz výstřelu na 30 m s terčem

30 metrů terč	RayCast		Physical	
z	*x*	*y*	*x*	*y*
28,73	50,04	3,49	50,03	3,47
28,73	50,08	3,69	50,07	3,67
28,73	50,05	3,09	50,04	3,07

Zdroj: Vlastní zpracování

Tabulka 3 - Průměrná hodnota rozdílu složek při střelbě do terče na 30 m

x diff	y diff
0,01	0,02
0,01	0,02
0,01	0,02
0,01	***0,02***

Zdroj: Vlastní zpracování

Obrázek 15 - Terč se zásahy, 30 metrů

Zdroj: Vlastní zpracování, Unity(2023)

Na 30 metrů je průměrná odchylka při realistickém nastavení velmi malá, a to jeden centimetr pod a dva centimetry doleva od místa, na která bylo mířeno. Tento malý rozdíl je možné dát za vinu posunu, jenž nastal kvůli colliderům jednotlivých těles znázorňujících zásah. Na obrázku z pokusu jde vše názorně vidět.

3.7.2.2 **Měření střelby na 80 metrů**

Tabulka 4 - Záznam polohy koliz výstřelu na 80 m s terčem

80 metrů terč	RayCast		Physical	
z	*x*	*y*	*x*	*y*
79,53	30,03	4,07	29,96	4,04
79,53	30,03	2,95	29,96	2,92
79,53	29,96	3,65	29,89	3,62

Zdroj: Vlastní zpracování

Tabulka 5 - Průměrná hodnota rozdílu složek při střelbě do terče na 80 m

x diff	y diff
0,07	0,03
0,07	0,03
0,07	0,03
0,07	***0,03***

Zdroj: Vlastní zpracování

Obrázek 16 - Terč se zásahy, 80 metrů

Zdroj: Vlastní zpracování, Unity(2023)

Na 80 metrů je průměrná odchylka o něco větší, a to tři centimetry pod a sedm centimetrů nalevo od míst, na která bylo mířeno. Na těchto datech už jde vidět začátek působení gravitační síly a efektu větru, který je díky vysoké hodnotě silnější. To všechno je by mělo být umocněno odporem vzduchu. Ten však, jak je zřejmé z předchozího měření má na takto malou vzdálenost zanedbatelný efekt.

3.7.2.3 **Měření střelby na 100 metrů**

Tabulka 6 - Záznam polohy koliz výstřelu na 100 m s terčem

100 metrů terč	RayCast		Physical	
z	*x*	*y*	*x*	*y*
109,41	-0,61	3,68	-0,74	3,6
109,41	-0,04	3,68	-0,16	3,6
109,41	0,44	4,44	0,32	4,37

Zdroj: Vlastní zpracování

Tabulka 7 - Průměrná hodnota rozdílu složek při střelbě do terče na 100 m

x diff	y diff
0,13	0,08
0,12	0,08
0,12	0,07
0,123333	***0,076667***

Zdroj: Vlastní zpracování

Obrázek 17 - Terč se zásahy, 100 metrů

Zdroj: Vlastní zpracování, Unity(2023)

Na 100 metrů je průměrná odchylka zase o něco větší, a to 7 centimetrů pod a 12 centimetrů doleva od místa, na která bylo mířeno. Efekt je i na obrázku z měření znatelný a jestli že by hráč, chtěl střílet nepřítele trefit, musel by provést korekci střelby. Co je však

zajímavé je nárůst průměrných odchylek skoro o dvojnásobek oproti předchozímu měření. Na těchto hodnotách lze vidět, exponenciální působení jednotlivých faktorů, které na projektil v letu působí. Faktem také je, že viditelnost se začínala značně zhoršovat. To mohlo být nastavením textur nebo shaderů, ale například čísla nebyla ani po zmenšení FOV vůbec viditelná.

Obrázek 18 - ukázka viditelnosti terče při testování

Zdroj: Vlastní zpracování, Unity(2023)

3.7.2.4 **Měření střelby na 200 metrů**

Tabulka 8 - Záznam polohy koliz výstřelu na 200 m s terčem

200 metrů terč	RayCast		Physical	
z	*x*	*y*	*x*	*y*
199,56	-20,45	4,22	-20,86	3,94
199,56	-20,11	3,52	-20,51	3,24
199,56	-18,54	2,65	-18,94	2,37

Zdroj: Vlastní zpracování

Tabulka 9 - Průměrná hodnota rozdílu složek při střelbě do terče na 200 m

x diff	*y diff*
0,41	0,28
0,4	0,28
0,4	0,28
0,403333	***0,28***

Zdroj: Vlastní zpracování

Obrázek 14 - Terč se zásahy, 200 metrů

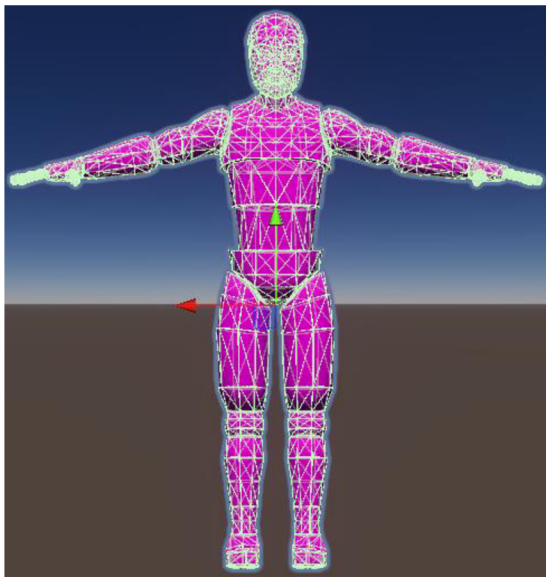
Zdroj: Vlastní zpracování, Unity(2023)

Na 200 metrů je rozdíl mezi jednotlivými technikami opravdu výrazný. Průměrná horizontální odchylka způsobená větrem má hodnotu 40 cm nalevo od místa dopadu paprsku. Vertikální průměrná odchylka dosáhla 28 centimetrů pod místo dopadu. Rozdíly, které jednotlivé vlivy způsobily, jsou na obrázku z měření velmi jasně vidět. Nárůst odchylek oproti minulým měřením je obrovský a lze na něm demonstrovat jejich exponenciální působení. Viditelnost u tohoto testu byla limitována na bílou a černou oblast terče. Krom těchto dvou oblastí nebylo k přečtení absolutně nic.

3.7.3 **Střelba na figurínu**

U tohoto testu je zjišťováno, jak tyto vlivy ovlivňují střelbu mířenou na model člověka a tedy protihráče. U tohoto modelu byl problém se zachycováním doteku objektů značný, jelikož tento model reprezentuje figurínu člověka, nebyl nijak velký.

Obrázek 19 - Ukázka collideru figuríny



Zdroj: Vlastní zpracování, Unity(2023)

3.7.3.1 **Měření střelby na 30 metrů**

Tabulka 10 - Záznam polohy koliz výstřelu na 30 m s terčem

RayCast		
x	***y***	***z***
45,89	1,7	29,15
45,89	1,44	29,15
45,91	1,17	29,16

Physical		
x	***y***	***z***
45,73	1,63	29,46
45,73	1,43	29,47
45,71	1,16	29,4

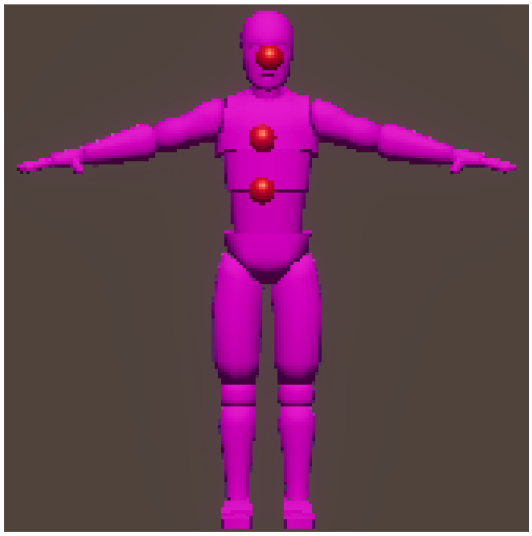
Zdroj: Vlastní zpracování

Tabulka 11 - Průměrná hodnota rozdílu složek při střelbě na figurínu na 30 m

x diff	y diff	z diff
0,16	0,07	-0,31
0,16	0,01	-0,32
0,2	0,01	-0,24
0,173333	0,03	-0,29

Zdroj: Vlastní zpracování

Obrázek 20 - Figurína se zásahy, 30 metrů



Zdroj: Vlastní zpracování, Unity(2023)

První střely na 30 metrů nepřináší stejné výsledky, jako při střelbě na terč. Rozdíl x-ových souřadnic je v průměru 17,33 centimetru. To však lze vysvětlit následovně. Metoda, která u fyzických projektilů inicializuje model zásahu ho inicializuje kolmo na daný objekt. Jelikož má tento model složitý collider s mnoha různě natočenými plochami, je možné, že když zjišťuje kolmý směr na plochu collidru, posune díky tomu souřadnice. Cíl však šel zasáhnout absolutně přesně bez větších problémů. Z dat jde také vidět, že označení dotyku fyzickým modelem je v průměru 29 centimetrů za objektem vytvořeným RayCastem. To velmi názorně demonstruje výše popsaný problém a vyšší hodnotu tohoto rozdílu lze přikládat vysoké počáteční rychlosti projektilu.

3.7.3.2 **Měření střelby na 80 metrů**

Tabulka 12 - Záznam polohy koliz výstřelů na 80 m s figurínou

RayCast		
x	***y***	***z***
25,85	1,84	80,02
25,85	1,63	79,95
25,85	1,35	79,95

Physical		
x	***y***	***z***
25,84	1,73	80,12
25,86	1,59	80,08
25,86	1,3	80,08

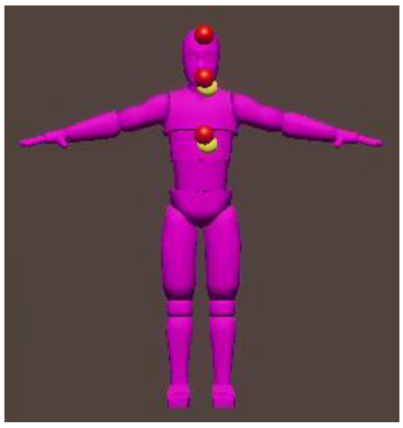
Zdroj: Vlastní zpracování

Tabulka 13 - Průměrná hodnota rozdílu složek při střelbě na figurínu na 80 m

x diff	**y diff**	**z diff**
0,01	0,11	-0,1
-0,01	0,04	-0,13
-0,01	0,05	-0,13
-0,00333	0,06667	-0,12

Zdroj: Vlastní zpracování

Obrázek 21 - Figurína se zásahy, 80 metrů



Zdroj: Vlastní zpracování, Unity(2023)

Střelba na 80 metrů také proběhla v pořádku. Cíl bylo možné zasáhnout v celku dobře. Na naměřených hodnotách se potvrzuje fakt, že na tělese s takto tvarovanými collidery, musí být souřadnice dopadu nějakým způsobem zkreslené. Diference na horizontální ose je pouhé 3 milimetry. Je tedy na místě uvést, že všechny testy proběhly za stejných podmínek a se stejným nastavením všech proměnných. Na obrázku lze také zpozorovat, že collider zaznamenal kolizi dříve než u střelby na 30 metrů a z figuríny tak vykukují kousky sfér označující kolizi s projektilem. Dále stojí za zmínku hodnota z, jež se značně snížila. To lze připisovat klesající rychlosti projektilu.

3.7.3.3 **Měření střelby na 100 metrů**

Tabulka 14 - Záznam polohy koliz výstřelů na 100 m s figurínou

RayCast			Physical		
x	***y***	***z***	***x***	***y***	***z***
-4,24	1,67	109,83	-4,27	1,58	109,9
-4,24	1,48	109,83	-4,27	1,39	109,9
-4,24	1,38	109,83	-4,27	1,29	109,9

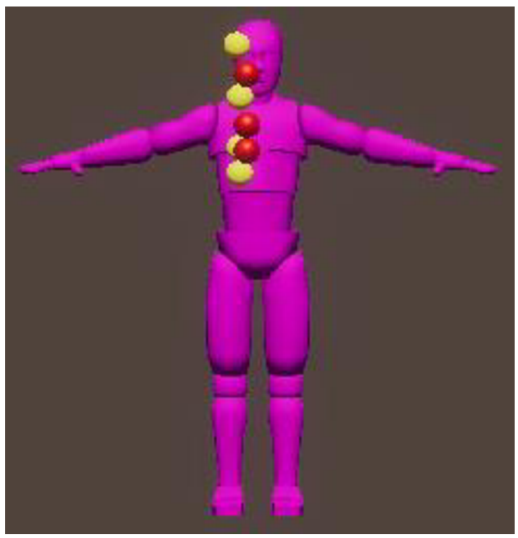
Zdroj: Vlastní zpracování

Tabulka 15 - Průměrná hodnota rozdílu složek při střelbě na figurínu na 100 m

x diff	***y diff***	***z diff***
0,03	0,09	-0,07
0,03	0,09	-0,07
0,03	0,09	-0,07
0,03	0,09	-0,07

Zdroj: Vlastní zpracování

Obrázek 22 - Figurína se zásahy, 100 metrů



Zdroj: Vlastní zpracování, Unity(2023)

Střelba na 100 metrů rozhodně nešla tak lehce jako předchozí měření. Z dat se může zdát, že to střelbu prakticky neovlivnilo. Největší rozdíl ukazuje souřadnice y, kde je naměřený rozdíl jen 9 centimetrů. Hodnota rozdílů x-ových souřadnic je pouhé 3 centimetry a na souřadnici z potvrzuje teorie o snížené rychlosti, protože se rozdíl opět o pár centimetrů zmenšil. Avšak na obrázku 16 si je možné všimnout, že je na figuríně je o jeden žlutý balonek

více. Důvod tohoto jevu, který se může zdát jako chyba, je prostý. Trefit figurínu bylo vcelku obtížné. První střela mířila přímo na hlavu, ale jak vidíte zasáhl jí pouze fyzický projektil. To je asi jediný důkaz, který obtížnost střelby na tuto vzdálenost obhajuje.

3.7.3.4 **Měření střelby na 200 metrů**

Tabulka 16 - Záznam polohy koliz výstřelů na 100 m s figurínou

RayCast		
x	***y***	***z***
-24,28	1,6	199,98
-24,28	1,25	199,98
-24,25	1,43	200

Physical		
x	***y***	***z***
-24,5	1,32	200,13
-24,5	0,97	200,13
-24,49	1,14	200,27

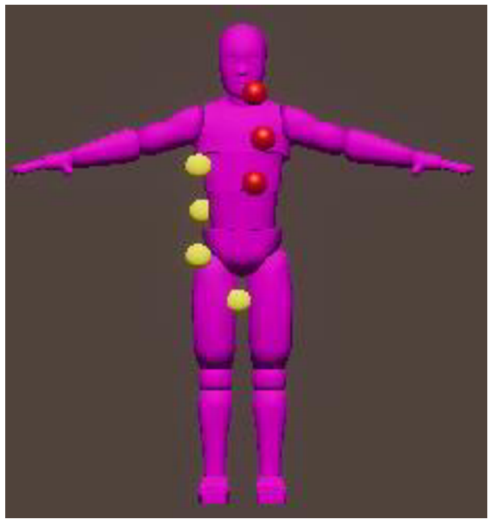
Zdroj: Vlastní zpracování

Tabulka 17 - Průměrná hodnota rozdílu složek při střelbě na figurínu na 200 m

x diff	y diff	z diff
0,22	0,28	-0,15
0,22	0,28	-0,15
0,24	0,29	-0,27
0,22667	0,28333	-0,19

Zdroj: Vlastní zpracování

Obrázek 23 - Figurína se zásahy, 200 metrů



Zdroj: Vlastní zpracování, Unity (2023)

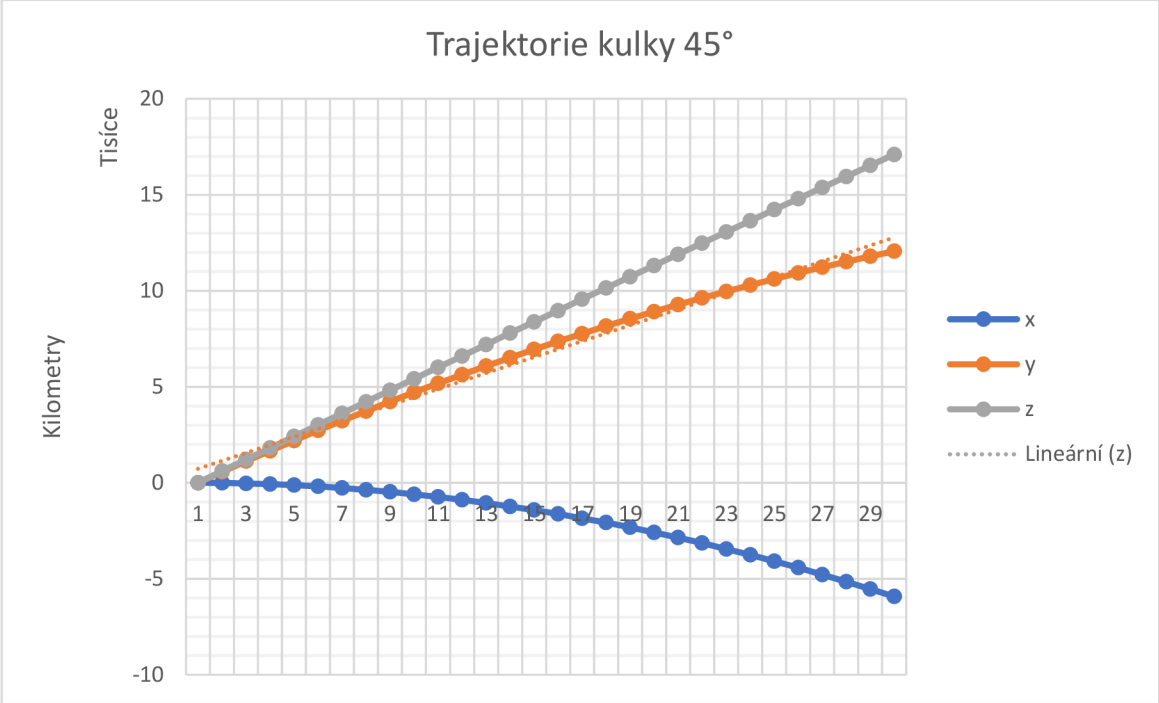
Střelba na 200 velmi názorně ukazuje působení jevů na dlouho vzdálenost. Už z obrázku 17 je na první pohled poznat značný rozdíl míst dopadu. Podle naměřených dat je průměrný rozdíl hodnot horizontální souřadnice 22,66 centimetrů. To je však oproti naměřeným 40 centimetrům u terče skoro poloviční hodnota. Tento rozdíl však lze velmi jednoduše vysvětlit. Projektil netrefil postavu přímo, ale pouze o ni zavadil. To však odpovídá definici kolize v collideru, a proto collider správně zásahy inicioval. Tímto lze vysvětlit i náhle zvětšený rozdíl na souřadnicích hloubky. Protože, projektil do modelu nenarazil, ale pouze zavadil, muselo se tak stát dále než přímý náraz paprsku, to tento rozdíl vysvětluje.

Výsledky a diskuse

3.8 Grafy

3.8.1 Graf znázorňující let kulky vystřelené v úhlu 45°

Graf 2 - Poloha kulky vystřelené v úhlu 45° vzhledem k horizontu



Zdroj: Vlastní zpracování

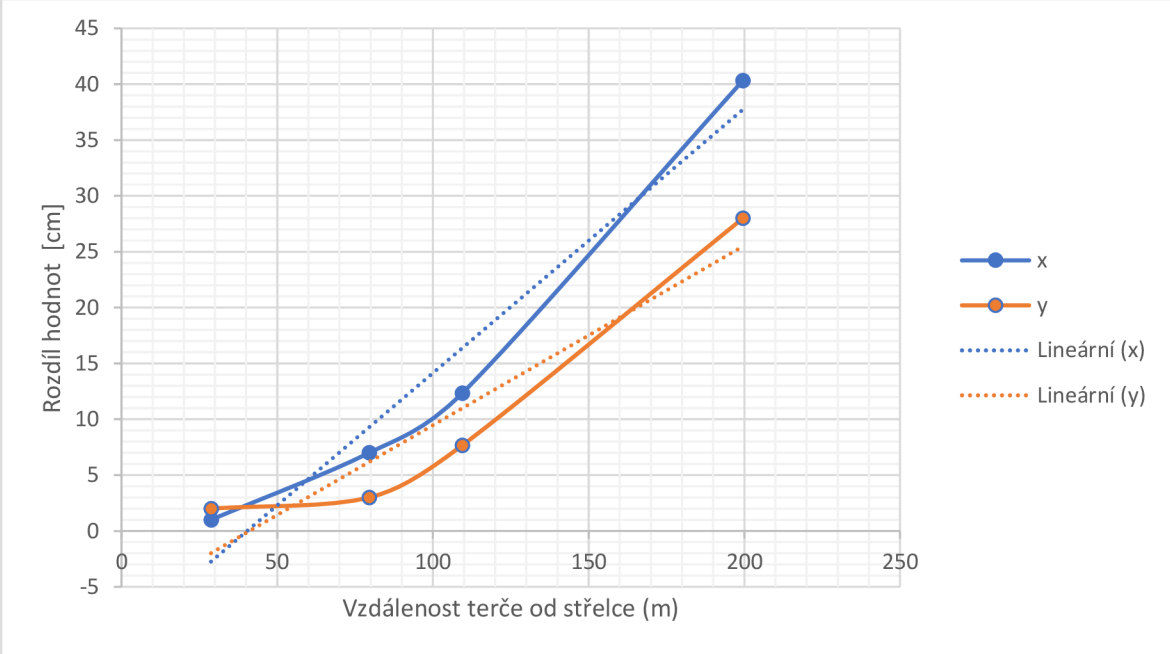
Z grafu je na první pohled patrné, že toto měření sleduje typickou parabolickou křivku, kde maximální dosah a výška jsou dosaženy díky optimálnímu výstřelovému úhlu. Úhel 45° je ideální pro dosažení maximálního dosahu ve vakuu, kde se zanedbává odpor vzduchu, což dokazuje teoretické principy balistiky. Z počáteční polohy blízké nule je zřejmé, že kulka rychle získává na výšce a vzdálenosti, což reflektuje vzestupnou fázi letu charakterizovanou počáteční rychlostí a úhlem výstřelu. S postupem letu kulka dosahuje svého vrcholu, kde vertikální složka rychlosti se snižuje na nulu a následně začíná kulka klesat zpět k zemi, což odpovídá teoretickým očekáváním. Co však realitě neodpovídá je lineární zvětšení souřadnice z. To by mělo být zásadně ovlivněno odporem vzduchu. Na tomto testu však vidíme, že jednoduchá rovnice pro odpor vzduchu, nemá s reálnými parametry v tomto konkrétním případě požadovaný efekt. Tomu odpovídá fakt, že realistické

simulace externí balistiky se musí provádět na velmi výkonných počítačích. Jestliže by vývojář tedy chtěl odpor vzduchu lépe demonstrovat doporučuje se jiné nastavení parametrů funkce.

3.8.2 Graf znázorňující odchylky při střelbě na figurínu

Hodnoty tohoto měření už realitě odpovídají o něco více. Exponenciální růst rozdílu x-ových hodnot je podle funkce, kterou byl tento faktor do simulace vnesen, očekávaný. Tak je tomu i u hodnoty gravitace. Na ní lze poznat i efekt vzdálenosti a odporu vzduchu. Jelikož projektil letěl déle, gravitace měla větší časový úsek na to ho vychýlit, což společně se zpomalením efektem odporem vzduchu odpovídá realistickému chování projektilu. To, jestli hodnoty na tyto vzdálenosti odpovídají reálné střelbě, nelze říci, protože, jak je napsáno v teoretické části, reálné hodnoty jsou velmi dynamické a bylo by za potřebí definovat, kde střelecké testy probíhaly, jakou silou a jakých směrem foukal vítr a v jaké nadmořské výšce. Výsledky by šly pak v simulaci korigovat nastavením jednotlivých parametrů.

Graf 3 - průměrný rozdíl dopadu obou metod na terč

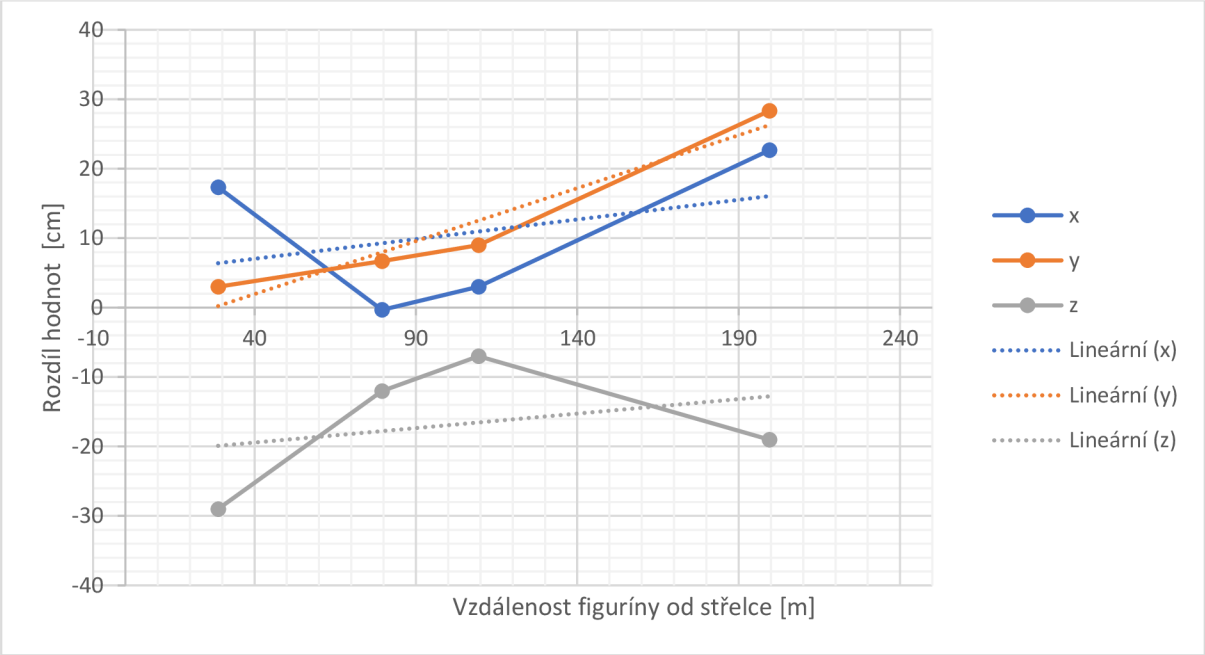


Zdroj: Vlastní zpracování

3.8.3 Graf znázorňující odchylky při střelbě na figurínu

Tento test přinesl nejzásadnější výsledky z pohledu úrovně hratelnosti a obtížnosti střelby. Než dojde k samotnému hodnocení výsledků je nutné upozornit, že v simulaci nebyl žádným způsobem modelován zpětný ráz nebo kývání zbraně při míření, což jsou v žánru Tactical velmi často zahrnuté mechaniky. Také stojí za zmínku, že projektil střílený měl průměr 20 centimetrů, a tedy zásah figuríny byl o to lehčí. U střelby na 30 metrů nebylo při měření zaznamenáno žádných potíží a dopad projektilu odpovídal pohledu skrze mířidla. Na tuto vzdálenost tedy vlivy nijak zásadně střelbu na protihráče neovlivňují. Střelba na 80 metrů také proběhla v pořádku a kulka i když s mírnou odchylkou dopadala na místa, na které zbraň mířila. Situace by se však mohla zásadně změnit, kdyby se cíl během letu střely pohyboval. I když kulka letí velmi rychle, určitá prodleva oproti metodě HitScan je znatelná a museli by být provedeny další korekce míření. Však v tomto testu šlo zasáhnout cíl snadno. U měření střelby na 100 metrů začaly problémy s viditelností a bylo nutné počítat s odchylkou, kterou jednotlivé funkce na výsledku způsobí. Díky velikosti projektilu bylo možné figurínu zasáhnout, avšak při pokusu o trefu do hlavy, šla první rána vedle. Z hlediska vývojářského je zde velmi názorně vidět zpomalení projektilu a tím i rychlejší detekce colliderem. U střelby na 200 metrů jsou všechny působící vlivy velmi dobře vidět. Pro zasáhnutí cíle je zásadní správná korekce zaměřovače. Při rozdílu skoro 30 centimetrů na vertikální ose, je nutné mířit alespoň 15 centimetrů nad hlavu, aby byl zasažen nejspodnější bod hlavy a zásah mohl být fatální. Na obtížnosti ještě přidává efekt větru, který kulku posunul průměrně o 22 centimetrů, tudíž bychom museli do korekce zahrnout i tento vliv. Kdyby se ještě k tomu cíl pohyboval a míření bylo ztíženo kýváním zbraně a zpětným rázem, bylo by zasáhnout cíl o dlouhém nastavování korekce a dobře načasovaném výstřelu, což je v tempu, ve kterém se bitvy odehrávají, skoro nemožné.

Graf 4 - průměrný rozdíl dopadu obou metod na figurínu



Zdroj: Vlastní zpracování

3.8.4 Zamyšlení nad porovnáním se skutečností

Z výsledků lze konstatovat několik následujících tvrzení. První z nich je obtížnost naprogramování externí balistiky alespoň na úroveň základní verze hry Arma 3. Z praktické části lze vyčíst, jak těžké je správné nastavení parametrů. Při nastavování parametrů byly všechny hodnoty určeny tak, aby co nejvíce odpovídali realitě. Výsledky však realitu simulují jen z dálky. Principiálně síly fungují stejně, avšak jejich účinek se oproti realitě liší. To lze přičíst zjednodušení funkcím, které v scriptu tyto složky zastupují. Pro reálnější výsledky by bylo nutné nastavovat parametry na základě měření. To by však zahrnovalo rozsáhlý výzkum s několika desítkami měření a změn nastavení, což by velmi přesáhlo rozsah této práce.

3.8.5 Ovlivnění gameplaye a srovnání s vybranými tituly

Z naměřených dat je zřejmé, že zahrnutí těchto faktorů do vývoje ztěžuje i samotnou hratelnost. Hráč musí během akce velmi rychle vyhodnotit všechny vlivy, které jsou v simulaci obsaženy. Dále musí brát v potaz pohyb nepřítele v následujících sekundách. Kdyby vývojáři chtěli dosáhnout ještě větší uvěřitelnosti, musel by hráč brát v potaz i zakřivení terénu kolem něj.

U prvního zmiňovaného titulu Call of Duty je nyní zřejmé, proč studio zvolilo řešení této metody pomocí přístupu HitScan. Jelikož jsou mapy v hrách této značky rozlohově v řádech stovkách metrů a většinou neposkytují možnost střílet z jednoho konce mapy na druhý, není pro gameplay implementace fyzického projektilu absolutně nutná. Tyto síly se podle měření v této práci viditelně projevují až při střelbě na velkou vzdálenost a tu v této hře lze zažít jen velmi zřídka. Naopak ztížení střelby ostatními vlivy jako zpětným rázem a velikostí zásobníku je pro hru, jejíž tempo je velmi rychlé a kde většina kontaktů s nepřítelem nastane přibližně do 100 metrů, ideální.

U druhého zmiňovaného titulu Battlefield je rozhodnutí vývojářů také pochopitelné. Nutno před vysvětlením opět podotknout, že tato značka cílí na širokou veřejnost. Ve hře se od titulu Battlefield 1 k klasickému působení gravitace přidal ještě odpor vzduchu. Ten nijak neovlivňuje souboje na krátkou vzdálenost, avšak u odstřelování nepřátelů na několik stovek metrů je díky zakomponování této mechaniky velmi obtížné. Jelikož je klesání nelineární, velmi špatně se bez přesného nastavení hledí hledá správné místo, kam vystřelit, aby všechny výše zmíněné faktory nasměrovaly kulku na cíl. Hráči tato změna byla uvítána velmi vřele, jelikož se zamezilo přesnému dostřelování a hráč musel získat opravdové zkušenosti, aby ne víc než 150 metrů přesně trefil cíl. Efekt větru a nadmořské výšky do této hry zanesen nebyl nejspíše proto, že by střílení na dálku bylo ještě více složité a hráče by přestalo bavit za průzkumnickou třídu hrát. Tím by rozhodily celý balanc systému tříd a hra by byla nevyvážená.

Poslední zmíněný titul Arma 3 svou komplexnost v oblasti externí balistiky také obhájí. Jelikož tato hra nemíří na masový trh, ale je spíše na nadšence a lidi, kteří od hry vyžadují věrnou simulaci, jak armádního prostředí, tak faktorů, které pěchotu a vojenský personál působí. Nutno podotknout, že i tak v základní verzi hry, nemá vliv na externí balistiku ani teplota ani nadmořská výška. Tento fakt, že hráči této hry, chtějí co nejrealističtější prostředí, potvrzuje i to, že dříve zmíněný mód ACE3, balistiku dělá ještě více komplexní, než jakou jí vývojáři z Bohemia interaktiv naprogramovali v základní hře, a přesto je v komunitě nejedním hráčem považován za nejlepší mód do hry.

Závěr

V rámci této bakalářské práce bylo cílem prozkoumat vývoj 3D herní aplikace v Unity s důrazem na simulaci externí balistiky projektilů. Práce se zaměřila na analýzu fyzikálních jevů ovlivňujících trajektorii projektilů v reálném světě a jejich aplikaci v prostředí Unity, s cílem dosáhnout co nejrealističtějšího chování projektilů ve videoherním prostředí. Byla provedena kombinace teoretického výzkumu a praktického programování, přičemž byl kladen důraz na chování projektilu a porovnání metod.

Hlavním přínosem práce je porozumění dynamice letu projektilu a jeho implementace do herního engine Unity, čehož bylo dosaženo vytvořením prototypu hry, který integruje realistické balistické modely. Výsledky ukazují, že pečlivá aplikace fyzikálních zásad na modelování trajektorie projektilů může významně zvýšit realismus ve videohrách, aniž by bylo nutné přistoupit k nadměrně složitým a výpočetně náročným simulacím. Práce také odhalila, že i přes technologická omezení současného hardwaru je možné dosáhnout uspokojivých výsledků, které odpovídají očekávání hráčů po realistické simulaci střelby.

Dalším klíčovým zjištěním je potvrzení, že vyšší úroveň realismu chování projektilů je dobrá v případě, když ve hře může běžně nastat situace střelby na cíl vzdálený nad 100 metrů, což přispívá k hráčovu celkovému zážitku ze hry a zároveň hru vyvažuje. Jestliže ve hře tato situace nenastává často a hra je spíše žánru action, je přístup řešení zásahů pomocí HitScan a použití funkce RayCast zcela na místě. Zpětná vazba od komunity žánru tactical ukázala, že i drobné detaily v chování projektilů, jako je ovlivnění trajektorie větrem nebo gravitačním polem, mohou hráče značně více ponořit do herního světa. Tato práce tak přináší také cenné vhledy pro vývojáře FPS her, kterým ukazuje, z jakých základních technik mají při vývoji své hry na výběr a jaký postup by pro ně mohl být lepší. Také v práci bylo obsaženo, jak postupovat při měření kvůli debuggingu.

V praktické části bylo zjištěno, že nastavením reálných hodnot do parametrů naprogramovaných funkcí nemá stejný efekt jako v realitě a je za potřebí přesná kalibrace a nastavených takových hodnot, aby vývojář dosáhl požadovaného efektu. Také byly ukázány funkce na zachycení nárazu střely a jejich základní porovnání.

Výzvy spojené s tímto výzkumem zahrnovaly zejména potřebu vyvážit mezi realistickým chováním projektilů a hratelností hry. Bylo zjištěno, že přílišný

realismus může být pro některé hráče odrazující a v některých situacích zcela zbytečný, což vyžaduje pečlivé kalibrování simulace, aby byla zajištěna dostatečná výzva, aniž by hra byla frustrující. Práce také identifikovala možnosti pro další výzkum, včetně rozšíření na další aspekty fyziky ve hrách a zároveň byla přínosem pro kohokoliv, kdo chce do svého projektu implementovat realistickou externí balistiku.

Seznam použitých zdrojů

Activision. Call of Duty: Modern Warfare 2 [videohra]. Activision. Místo: Activision, 2009-11-10, poslední aktualizace 2023-04-XX [cit. 2024-02-17]. Dostupné z: https://www.callofduty.com/modernwarfare2

ACE3 TEAM, 2015. ACE3 mod - Advanced ballistics. ACE3 [online]. [cit. 2024-02-29]. Dostupné z: https://ace3.acemod.org/wiki/feature/advanced-ballistics

BOHEMIA INTERACTIVE, 2013. Weapons settings. Bohemia Interactive Community Wiki [online]. 2022-12-14 [cit. 2024-02-17]. Dostupné z: https://community.bistudio.com/wiki/Weapons_settings

BROWN B.SC., PH.D., F.INST.P, 1969. The Acceleration Due to Gravity. In: BROWN B.SC., PH.D., F.INST.P, B. General Properties of Matter. 1. Springer, Boston, MA, pp 54–81. ISBN 978-1-4899-6501-1.

CREATION WASTELAND, 2020. AK-74 (HDRP): AK-74 Assault Rifle for the HD Render Pipeline. In: UNITY. Unity Asset Store [online]. [cit. 2024-02-13]. Dostupné z: https://assetstore.unity.com/packages/3d/props/guns/ak-74-hdrp-177205#description

DEBACCO UNIVERSITY, 2020. Bullet Trajectory. In: GOOGLE. Youtube [online]. [cit. 2024-02-27]. Dostupné z: https://www.youtube.com/watch?v=rvE0aJVVVXU

GUINNESS WORLD RECORDS LIMITED 2024. First first-person shooter (FPS) videogame. In: Guinness World Records [online]. [cit. 2023-09-30]. Dostupné z: https://www.guinnessworldrecords.com/world-records/95409-first-first-person-shooter-fps-videogame

IGLESIAS, Kevin, 2020. 3D Character Dummy: Version 1.0. In: UNITY. Unity Asset Store [online]. [cit. 2024-02-13]. Dostupné z: https://assetstore.unity.com/packages/3d/characters/humanoids/humans/3d-character-dummy-178395#releases

INCARNATE, 2019. Consequences of BF1 air drag for players. Forum Sym gg [online]. 2019-05-01 [cit. 2024-02-17]. Dostupné z: https://forum.sym.gg/t/consequences-of-bf1-air-drag-for-players/183

JUNG, Tristan, 2019. How Do Bullets Work in Video Games? In: Game Developer [online]. 2019-06-12 [cit. 2023-09-30]. Dostupné z: https://www.gamedeveloper.com/programming/how-do-bullets-work-in-video-games-

KOZHEMYAKIN, Aleksey, 2019. Modular self-stand fence: Small self-stand fence model and base block for demarcation areas. In: Https://assetstore.unity.com/ [online]. [cit. 2024-

03-13]. Dostupné z: https://assetstore.unity.com/packages/3d/props/modular-self-stand-fence-105862

KRISHNA REDDY, D. Siva, Bibhu PRASAD PADHY a Bharani KUMAR REDDY, 2019. Flat-Fire Trajectory Simulation of AK-47 Assault Rifle 7.82-mm Bullet. In: Emerging Trends in Mechanical Engineering. Lecture Notes in Mechanical Engineering. 2019-12-12. Singapore: Springer, s. 10. ISBN 978-981-32-9931-3. Dostupné z: doi:https://doi.org/10.1007/978-981-32-9931-3_40

LATHI, Evan, 2013. An explanation of Arma 3's ballistics modeling: everything that happens when you fire a gun. In: PC Gamer [online]. [cit. 2024-02-20]. Dostupné z: https://www.pcgamer.com/what-happens-in-arma-3-when-you-fire-a-gun/

LODE, 2004. Lode's Computer Graphics Tutorial Raycasting. VANDEVENNE, Lode. Lode's Computer Graphics Tutorial [online]. [cit. 2023-09-30]. Dostupné z: https://lodev.org/cgtutor/raycasting.html

MLAGENT, 2024. Grid Prototype Materials: A set of prototyping materials that tile with the object scale to always correctly represent size/scale. In: UNITY. Unity Asset Store [online]. [cit. 2024-03-13]. Dostupné z: https://assetstore.unity.com/packages/2d/textures-materials/grid-prototype-materials-214264

NASA, 2006. Beginner's Guide to Aeronautics - Shape effects of drag. NASA. NASA Beginner's Guide to Aeronautics [online]. [cit. 2024-02-28]. Dostupné z: https://www.grc.nasa.gov/www/k-12/VirtualAero/BottleRocket/airplane/shaped3.html

NIKIYANI, 2019. Military target. In: Unity Asset Store [online]. [cit. 2024-03-13]. Dostupné z: https://assetstore.unity.com/packages/3d/environments/military-target-136071#releases

OBERLIN, GREGORY, 2017. The Effect of Humidity on Small Arms Trajectory. 30th International Symposium on Ballistics [online]. Lancaster, PA: DEStech Publications, - [cit. 2024-02-11]. ISBN 978-1-60595-419-6. Dostupné z: doi:10.12783/ballistics2017/16802

OPENSTAX COLLEGE, 2013. Lumen Learning. LUMEN LEARNING. Lumen Learning [online]. [cit. 2024-02-27]. Dostupné z: https://courses.lumenlearning.com/suny-physics/chapter/5-2-drag-forces/

OXFORD LEARNER'S DICTIONARY, 2000. Definition of FPS. In: OXFORD LEARNER'S DICTIONARY. Oxford Learner's Dictionary FPS abbreviation [online]. [cit. 2023-10-05]. Dostupné z: https://www.oxfordlearnersdictionaries.com/definition/english/fps_2

PAYNE, Matthew Thomas, 2008. Interpreting Game-Play Through Existential Ludology. In: FERDIG, Richard E. Handbook of Research on Effective Electronic Gaming in Education. 1. IGI Global, s. 15. ISBN 9781599048086.

PENG, XIA, 2014. 3D Game Development with Unity A Case Study: A First-Person Shooter (FPS) Game. Helsinky. Thesis. Helsinki Metropolia University of Applied Sciences. Vedoucí práce Markku Karhu.

SHERWOOD, A. E., 1967. Effect of air drag on particles ejected during explosive cratering. Journal of Geophysical Research [online]. 1967-03-15, 72(6), 1783-1791 [cit. 2024-02-11]. ISSN 01480227. Dostupné z: doi:10.1029/JZ072i006p01783

SIERRA BULLETS, 2020. EFFECTS OF WINDS. Sierra Bullets [online]. 2020 [cit. 2024-03-12]. Dostupné z: https://sierrabullets.com/exterior-ballistics/3-2-effects-of-winds/

STEAM, 2003. Steam Charts [online]. 2023-09-30 [cit. 2023-09-30]. Dostupné z: https://store.steampowered.com/charts/

SYM.GG, 2018. Battlefield V Weapon Mechanics. Sym game science [online]. 2020-11-17 [cit. 2024-02-04]. Dostupné z: https://sym.gg/index.html?game=bfv&page=weapon-mechanics

TACTIGAMER, 2005. What Makes A Tactical Shooter? - The Difference Between Tactical Shooters and Casual Shooters. In: GOOGLE. Youtube [online]. [cit. 2024-03-13]. Dostupné z: https://www.youtube.com/watch?v=x8jpOP5V2Ls

THE ARMS GUIDE, 2020. Long range shooting ballistics terms. THE ARMS GUIDE. The Arms Guide [online]. 2023-10-04 [cit. 2024-02-29]. Dostupné z: https://thearmsguide.com/6026/long-range-shooting-ballistics-terms/

THE EDITORS OF ENCYCLOPAEDIA, 2023. Shotgun | Hunting, Home Defense & Self-Protection |. BRITANNICA. Britannica [online]. 2023-9-23 [cit. 2024-02-10]. Dostupné z: https://www.britannica.com/technology/shotgun

UNITY, 2013. Unity manual - Primitive Objects. UNITY. Unity [online]. 2013-04-22 [cit. 2024-02-25]. Dostupné z: https://docs.unity3d.com/430/Documentation/Manual/PrimitiveObjects.html

UNITY, 2023. Unity User Manual 2022.3 (LTS)Physics. UNITY. Unity Documentation [online]. 2023-11-01 [cit. 2023-11-07]. Dostupné z: https://docs.unity3d.com/Manual/PhysicsSection.html

UNITY. Unity [online]. [cit. 2023-05-13]. Dostupné z: https://learn.unity.com/tutorials/

URONE, Paul Peter a Roger HINRICHS, 2012. College Physics. 1. Houston, Texas: OpenStax. ISBN 978-1938168000.

WARLOW, T. A., c2005. Firearms, the law, and forensic ballistics. 2nd ed. Boca Raton, Fla.: CRC Press. ISBN 0415316014.

WOLF, Mark J. P. a Bernard PERRON, 2014. The Routledge companion to video game studies. New York, NY: Routledge. ISBN 9781138657052.

Seznam obrázků, tabulek, grafů a zkratek

3.9 Seznam obrázků

3.10 **Seznam tabulek**

3.11 **Seznam grafů**

Přílohy

Příloha 1 – Metoda Update v FPSController.cs

```
    // Update is called once per frame
    void Update()
    {
        #region Ovladani pohybu
        Vector3 forward = transform.TransformDirection(Vector3.forward);
        Vector3 right = transform.TransformDirection(Vector3.right);
        //Běh levým shiftem
        bool isRunning = Input.GetKey(KeyCode.LeftShift);
        //zjištění momentální rychlosti
        float curSpeedX = canMove ? (isRunning ? runSpeed : walkSpeed) *
Input.GetAxis("Vertical") : 0;
        float curSpeedY = canMove ? (isRunning ? runSpeed : walkSpeed) *
Input.GetAxis("Horizontal") : 0;
        float movementDirectionY = moveDirection.y;
        moveDirection = (forward * curSpeedX) + (right * curSpeedY);
        #endregion
        #region Ovladani skoku
        if(Input.GetButton("Jump") && canMove &&
characterController.isGrounded){
            moveDirection.y = jumpPower;
        }
        else{
                moveDirection.y = movementDirectionY;
        }
        if(!characterController.isGrounded){
            moveDirection.y -= gravity * Time.deltaTime;
        }
        #endregion

        #region Ovladání rotace a pohybu kamery
        characterController.Move(moveDirection*Time.deltaTime);

        if(canMove){
            rotationX +=-Input.GetAxis("Mouse Y") * lookSpeed;
            rotationX = Mathf.Clamp(rotationX, -lookXLimit, lookXLimit);
            playerCamera.transform.localRotation =
Quaternion.Euler(rotationX, 0, 0);
            transform.rotation *= Quaternion.Euler(0, Input.GetAxis("Mouse
X") * lookSpeed, 0);
}

        #endregion
    }
}
```