

Optimization of Tank Battle Simulation for Gaming Industry

Bc. Zapletal Michal

Master's Thesis
2024



Tomas Bata University in Zlín
Faculty of Applied Informatics

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Bc. Michal Zapletal**
Osobní číslo: **A22598**
Studijní program: **N0613A140022 Informační technologie**
Specializace: **Softwarové inženýrství**
Forma studia: **Prezenční**
Téma práce: **Optimalizace simulace tankového boje pro herní průmysl**
Téma práce anglicky: **Optimization of Tank Battle Simulation for Gaming Industry**

Zásady pro vypracování

- Vypracujte literární rešerši na zadané téma.
- Zaměřte se na simulaci pohybu tanku, balistiky, poškození a zranitelnosti.
- Analyzujte existující videoherní simulace tankových bitev.
- Navrhněte optimalizaci pro simulaci tankové bitvy v rámci herního prostředí s důrazem na dosažení nejvyšší možné realističnosti při zohlednění vhodných výkonnostních požadavků.
- Implementujte navržené řešení do simulátoru UTW.
- Otestujte nové funkce a výkonnostní zatížení simulátoru.
- Vyhodnoťte výsledky testování.

Forma zpracování diplomové práce: **tištěná/elektronická**
Jazyk zpracování: **Angličtina**

Seznam doporučené literatury:

1. ANDREWS, William. *Tank Gun Systems: The First Thirty Years, 1916-1945: A Technical Examination*. V1. Pen and Sword Military, 2023. ISBN 1399042378, 9781399042376.
2. AVERSA, Davide a DICKINSON, Chris. *Unity game optimization: enhance and extend the performance of all aspects of your Unity games*. Third edition. Birmingham: Packt, 2019. ISBN 978-1-83855-651-8.
3. BORROMEO, Nicolas Alejandro. *Hands-on Unity 2021 game development: create, customize, and optimize your own professional games from scratch with Unity 2021*. Second edition. Birmingham: Packt, 2021. ISBN 978-1-80107-148-2.
4. *Unity user manual* [online]. 2023 [cit. 2023-11-12]. Dostupné z: <https://docs.unity3d.com/Manual/index.html>.
5. OKUN, Nathan. *Major Historical Naval Armor Penetration Formulae* [online]. 1998 [cit. 2023-11-12]. Dostupné z: <http://www.combinedfleet.com/formula.htm>.

Vedoucí diplomové práce: **Ing. Tomáš Vogeltanz, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **5. listopadu 2023**
Termín odevzdání diplomové práce: **13. května 2024**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan



prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

I hereby declare that:

- I understand that by submitting my Master's Thesis, I agree to the publication of my work according to Law No. 111/1998, Coll., On Universities and on changes and amendments to other acts (e.g. the Universities Act), as amended by subsequent legislation, without regard to the results of the defence of the thesis.
- I understand that my Master's Thesis will be stored electronically in the university information system and be made available for on-site inspection, and that a copy of the Master's Thesis will be stored in the Reference Library of the Faculty of Applied Informatics, Tomas Bata University in Zlín, and that a copy shall be deposited with my Supervisor.
- I am aware of the fact that my Master's Thesis is fully covered by Act No. 121/2000 Coll. On Copyright, and Rights Related to Copyright, as amended by some other laws (e.g. the Copyright Act), as amended by subsequent legislation; and especially, by §35, Para. 3.
- I understand that, according to §60, Para. 1 of the Copyright Act, TBU in Zlín has the right to conclude licensing agreements relating to the use of scholastic work within the full extent of §12, Para. 4, of the Copyright Act.
- I understand that, according to §60, Para. 2, and Para. 3, of the Copyright Act, I may use my work - Master's Thesis, or grant a license for its use, only if permitted by the licensing agreement concluded between myself and Tomas Bata University in Zlín with a view to the fact that Tomas Bata University in Zlín must be compensated for any reasonable contribution to covering such expenses/costs as invested by them in the creation of the thesis (up until the full actual amount) shall also be a subject of this licensing agreement.
- I understand that, should the elaboration of the Master's Thesis include the use of software provided by Tomas Bata University in Zlín or other such entities strictly for study and research purposes (i.e. only for non-commercial use), the results of my Master's Thesis cannot be used for commercial purposes.
- I understand that, if the output of my Master's Thesis is any software product(s), this/these shall equally be considered as part of the thesis, as well as any source codes, or files from which the project is composed. Not submitting any part of this/these component(s) may be a reason for the non-defence of my thesis.

I herewith declare that:

- I have worked on my thesis alone and duly cited any literature I have used. In the case of the publication of the results of my thesis, I shall be listed as co-author.
- That the submitted version of the thesis and its electronic version uploaded to IS/STAG are both identical.

In Zlín; dated:
10.5.2024

Bc. Michal Zapletal
Student's Signature

ABSTRAKT

Cílem této práce je vytvořit balistický modul pro modulární simulátor tankových bojů UTW, který rozhoduje o úspěchu proražení pancíře v reálném čase a vychází ze vzorců, které byly navrženy pro aproximaci reálných procesů, jež probíhají mezi projektilem a pancířem v době jejich kolize. K tomu je využit fyzikální engine Unity, který je již využíván v projektu UTW. Simulace je schopna počítat šanci na odražení, úspěšnost probití pancíře, jak kinetickou, tak chemickou střelou a generovat poškozující částice, které mohou zničit vnitřní moduly zasaženého vozidla. Takto sestavený modul je nakonec porovnán s existujícími řešeními.

Klíčová slova: UTW, Unity, Simulátor, Tanky, Modularita, Balistika

ABSTRACT

The objective of this thesis is to develop a ballistic module for the UTW modular tank battle simulator. This module determines the result of armor penetration in real-time, based on formulas designed to approximate real-world processes that occur between a projectile and armor upon impact. It utilizes the Unity physics engine, which is already employed in the UTW project. The simulation is capable of calculating ricochet probability, armor penetration success for both kinetic and chemical projectiles, and generating damaging particles that can destroy internal modules of the struck vehicle. The developed module is compared to existing solutions.

Keywords: UTW, Unity, Simulator, Tanks, Modularity, Ballistics

ACKNOWLEDGEMENTS

This work would not have been possible without the invaluable support of the Project UTW team, with their unending patience, which they proved, while awaiting the ballistic module, as it took much longer, than everyone has anticipated.

I cannot forget to thank my grandparents, who allowed me to stay at their house in times of need and the rest of my family for their bottomless support.

Finally, I cannot forget to thank my thesis supervisor Ing. Tomáš Vogeltanz, Ph.D., who provided me with heaps of useful feedback and was supportive from the beginning to the very end.

I hereby declare that the print version of my Master's thesis and the electronic version of my thesis deposited in the IS/STAG system are identical.

CONTENTS

INTRODUCTION	10
THEORY	12
1 TANK ARMOR AND WEAPONRY	13
1.1 BIRTH OF A TANK.....	13
1.2 INTERWAR TANK DEVELOPMENT.....	15
1.3 PHYSICAL PROCESS OF ARMOR PENETRATION	16
1.4 FACE HARDENED ARMOR AND ARMOR PIERCING AND BALLISTIC CAPS.....	17
1.5 CONVERTIBLE TANKS AND OTHER DEVELOPMENTS IN TANK MOBILITY	19
1.6 CHEMICAL ENERGY PROJECTILES.....	20
1.7 ARMOR OBLIQUITY	21
1.8 MODERN TANK DEVELOPMENT	23
2 UNIVERSAL ARMOR PENETRATION FORMULAE	25
2.1 GENERAL ARMOR PENETRATION FORMULA.....	26
2.2 DE MARRE NICKEL-STEEL ARMOR PENETRATION FORMULA.....	26
2.3 KRUPP ALL-PURPOSE ARMOR PENETRATION FORMULA	27
3 EXISTING TANK SIMULATORS.....	29
3.1 WARTHUNDER	29
3.1.1 GAMEPLAY.....	29
3.1.2 BALLISTICS AND DAMAGE MODEL	30
3.2 SQUAD 44.....	31
3.2.1 GAMEPLAY.....	32
3.2.2 BALLISTICS AND DAMAGE MODEL	33
3.3 IL-2 STURMOVIK: TANK CREW – CLASH AT PROKHOROVKA.....	34
3.3.1 GAMEPLAY.....	34
3.3.2 BALLISTICS AND DAMAGE MODEL	35
4 UNITY PHYSICS ENGINE	36
4.1 RIGIDBODY	36
4.1.1 RIGIDBODY PARAMETERS	36
4.2 COLLIDERS	37
4.2.1 TRIGGERS.....	37
4.3 COLLISIONS.....	37
4.4 RAYCASTS	38
4.4.1 RAYCASTALL.....	38
4.5 FIXEDUPDATE	38
ANALYSIS	39
5 UTW AND ITS SYSTEMS.....	40
5.1 GAMEPLAY.....	40

5.2	BALLISTIC SIMULATION REQUIREMENTS	41
6	UTW BALLISTICS MODULE	43
6.1	NECESSARY MECHANICS	43
6.1.1	DRAG AND GRAVITY	43
6.1.2	RICOCHET	43
6.1.3	PENETRATION.....	43
6.1.4	SPALLING GENERATION	44
6.1.5	TANK CREW AND MODULES	44
6.1.6	KINETIC AND CHEMICAL ENERGY PENETRATION DIFFERENCES.....	44
6.2	UNITY MODELS.....	45
6.2.1	KINETIC SHELL.....	45
6.2.2	CHEMICAL SHELL.....	46
6.2.3	ARMOR PLATE	46
6.2.4	TANK MODULE	47
6.3	DRAG AND GRAVITY IMPLEMENTATION	48
6.4	RAYCAST PROJECTION	49
6.5	RICOCHET HANDLING	51
6.6	KINETIC PENETRATION PROCESS.....	53
6.6.1	DEMARRE FORMULA.....	53
6.7	CHEMICAL PENETRATION PROCESS.....	56
6.7.1	BACKCAST PROJECTION.....	56
6.8	SPALLING GENERATION	57
6.9	DAMAGE HANDLING.....	60
7	UTW BALLISTIC MODULE TEST BUILD.....	62
7.1	TEST BUILD CONTROLS.....	62
7.2	PLAYER-CONTROLLED VEHICLE	63
7.3	TEST TARGET	63
7.4	SHOOTING RANGE.....	63
8	BALLISTIC SIMULATION TESTING	65
8.1	MEASURED DATA.....	65
8.2	WARTHUNDER BALLISTIC VALUES	66
8.3	UTW BALLISTIC VALUES	66
8.3.1	45MM 20-K (BR-240)	66
8.3.2	45MM 20-K (BR-240P).....	67
8.3.3	57MM ZIS-4 (BR-271).....	67
8.3.4	57MM QF-6PDR Mk.III (Mk.8).....	67
8.3.5	50MM KWK38 L/42 (PZGR 39)	68
8.3.6	88MM KWK36 L/56 (PZGR 39).....	68
8.4	RESULT TABLE	69
	CONCLUSION	70
	BIBLIOGRAPHY	71

LIST OF ABBREVIATIONS	74
LIST OF FIGURES	76
LIST OF TABLES	77
APPENDICES.....	78

INTRODUCTION

The unmarked Pzkw IV Ausf G slowly crests over the horizon, searching for potential threats. “They don’t see us yet.” whispers the commander, as the gunner slowly turns the turret towards the vehicle. “Wait until they turn their back to us.” continues the commander, and the turret stops moving for a moment. The unaware tank overcomes the last few meters of the climb and tilts downwards, gaining speed as it travels forward directly towards the little M3 Stuart, silently lying in ambush in the thick shrubbery. “Wait, wait...” the engine noise of the German tank is growing louder and louder, as the vehicle passes the shrubbery. “Ok go!” shouts the commander, as the driver starts the ignition and the little tank’s turret briskly turns to face the back of the enemy. First shell strikes the back of the panzer’s hull, and its engine goes up in flames. “Quick another one! Aim for the turret!” Shouts the commander as the loader loads another APCBC shell into the breech. Second shell strikes the turret ring, and the German vehicle stops moving altogether.

The outcome of a tank battle often hinges on a single well-placed shot, as such accurately simulating armor penetration and component damage is crucial for creating realistic and engaging tank simulators, allowing the players to enjoy tank simulation games much more.

Tanks and armored vehicles had been used in military conflicts all around the world for more than a hundred years. As such, they are often depicted in media as an inseparable part of modern armies, often playing major roles in movies and games. The latter one currently enjoys a massive boom, as the living standards of humanity slowly increases, allowing more people to spend more money on games and game regularly, which allows developers to create more complex and enjoyable games. The computers are becoming more powerful, allowing developers to create stunning visuals and to simulate intricate processes, greatly increasing the quality of newly created games and the tank simulators are not an exception.

Out of these thoughts a new project came to life. Project UTW is a student-driven project based on TBU FAI, which aims to create a modular tank simulation framework, based on Unity game engine, which could be used by various community developers to create their own, fully customizes tank simulator game. As such a need arose for a solid ballistics simulation system, that could be used as an integral part of UTW, which would allow detailed

customization of anti-tank projectiles and tank armor and internal modules, while keeping the whole system simple, lightweight, and gameplay oriented.

The proposed system should be able to handle both kinetic and chemical energy penetration process, based on real-life formulae, include a semi-stochastic ricochet chance calculation, which factors in the overmatch ratio, allow the system to describe complex shell types and provide rudimentary damage system, that the system can be showcased on.

This thesis aims to adduce the reader into the vast and complex problematic of armor penetration process, explain the evolution of tank development and some of the design choices and improvements, that were devised to increase the effectivity of armor vehicles, compare some of the existing tank simulators and explain the functionality of Unity Physics engine, which the UTW is built upon. In second part of the thesis a proposed ballistics system is explained in great detail, together with testing, which was conducted on the finished UTW ballistic system, and its results.

Despite the armor penetration process being rather stochastic and unpredictable, it is still possible to create a somewhat realistic model even in a very simple environment which the Unity Physics engine undoubtedly is. Main goal of the UTW ballistic system is to provide fully customizable, somewhat predictable penetration evaluation model, which, despite not being exactly accurate, works on the similar principles, that the real-life penetration process stands on.

To better visualize the results, a test build of UTW ballistics system, with a rudimentary shooting range, player-controlled vehicle, and a transparent test target, was set up for the reader, which should better illustrate the proposed system.

I. THEORY

1 TANK ARMOR AND WEAPONRY

On 15th September of 1916, German forces stationed near Flers-Courcelette witnessed first ever tank assault. This newborn weapon was ultimately designed to break through unending trenches of Great War and end the “War to end all wars” [1]. It eventually succeeded in this role and these steel behemoths paved the way for modern armored vehicles, that are being used by the militaries all over the world for years to come.

1.1 Birth of a tank

The First World War proved to be an immense challenge for both Central powers and Allied soldiers. New inventions spearheaded by Hiram Maxim’s machinegun, concrete emplacements, barbed wire, and modern artillery changed the way the war was fought from numerous mobile clashes between infantry, artillery, and cavalry into brutal, endless trench warfare. Each mile of ground had to be paid in blood and situation on Great War massive western front came to a halt.

"The present war has swept away all previous military theories. Machine-gun fire is so powerful that a hundred yards is enough to stop any attack by the enemy, who, to escape artillery fire, digs trenches in the rear... The war, instead of being fought over long distances, as was supposed, is fought over short distances... Therefore, the most important thing is not the long-distance onslaught, but the overcoming of a hundred or two hundred meters of open ground or nets of wire obstacles. It would therefore be advantageous to arm a certain number of tracked tractors as quickly as possible, to secure them against machine-gun fire, and to modify them to accommodate men and machine guns."

- Winston Churchill 1915

By the 1915, British captain Murray Sueter witnessed potential effectiveness of newly designed Rolls-Royce armored cars which, despite failing miserably in muddy and wet “No man’s land”, proved to be quite useful on solid surfaces, where they could move unhindered by omnipresent mud [2]. At that time, it was already more than 10 years since American inventor Benjamin Holt unveiled his great invention, a tracked agricultural tractor, capable of crossing flooded fields of north California [3]. His patented tracks, the so called

“Caterpillar” tracks, were to be used as the basis for a brand-new weapon to break through the impervious frontline, a landship.

On 20th February 1915 “Landship Committee”, created by Winston Churchill, came into existence. Their task was simple, design a dreadnought warship, that could “swim” through the mud of western front, destroy machinegun nests, raze down barbed wire obstacles, and provide cover for advancing infantry, that could retake the trench line. To avoid alerting Central powers about their new creation, Landship committee agreed to mask true purpose of landships and referred to them as “Water carriers” or simply “Tanks” [2].

The first ever tank was built by the Foster’s Wellington Works and named “Little Willie”, after its creator William Tritton, among other reasons. After extensive testing, it was decided to create a second prototype with a heavier armament using rhomboid track frame. This second model, that went by many names, such as Big Willie, HMLS (His Majesty’s Land Ship) Centipede, Mother or simply Mark I, was the first landship ever used in combat. Its armament originally consisted of several Hotchkiss Portative machineguns and 6-pounder field cannons. Design was later split into two variants, Male, equipped with cannons and machineguns, and Female, with machineguns only.

While the Mark 1’s proved to be very effective at destroying helpless German machinegun emplacements, they suffered from many imperfections, as is traditional with new inventions. Chief among them was the lack of ventilation, so both engine, which was mounted directly in the middle of the vehicle, and onboard cannons, spewed gases directly inside of the vehicle making the operation of such tank uncomfortable at best. Weaponry was mounted in the sponsons so both gunner’s and loader’s position were too cramped, commander sat too far from both gunners to effectively mark their targets and suspension damping was almost non-existent, hence the accuracy when the tank was in motion suffered greatly [2]. Armor plating was thick enough to stop conventional rifle rounds but could be easily penetrated by artillery or dedicated armor-piercing rounds.

Despite a few teething problems, introduction of tanks onto First world war battlefield changed trench warfare dramatically [1]. War became much more mobile, newly designed tanks with onboard radios allowed troops to operate farther from their base and the frontline slowly started to move again. Tanks were now deemed necessary to successfully accomplish any sort of assault and British, French, and German manufacturers started developing new models. The most groundbreaking tank of First world war was the French Renault FT [4].

Unlike most of the other tanks, that were designed in similar fashion to warships, FT introduced several important features, that are now considered standard, like separating crew and engine compartment, drastically increasing quality of life and safety for the tank crew, or mounting armament in turret, that could rotate 360°. Thanks to its size, it was considerably lighter than most landships at the time and could carry thicker armor, making it much more resilient against German anti-tank weaponry, like for example their anti-tank rifle “Tankgewehr” [5]. It was this very tank that set the trend of tank development for the next 100 years.

1.2 Interwar tank development

Tank’s success on battlefields of Great war birthed global interest in these steel monstrosities and sooner or later every developed nation integrated at least a few tanks into the ranks of their military [1]. Not every nation could afford to develop their own vehicles, so they usually opted to either buy tanks from other nations or bought the license to produce foreign types locally. As tanks became standard in many militaries, it became necessary to develop ways to counter them with devices like anti-tank rifles, anti-tank grenades, armor-piercing cannon rounds, anti-tank mines and so on. It was also necessary to better understand the physics of armor penetration to determine if a newly designed weapon is adequate to fight armored vehicles. Armor piercing formulae were already well known to military experts at the time, since they were used to measure the armor and cannon effectiveness of pre-WWI dreadnought battleships, which gave life to early landships [6].

Early tanks were mostly built as a skeleton chassis, which was then plated with steels and alloys of different composition and quality and riveted to hold in place. Such tanks were quite cheap and easy to make, but the metallic skeleton increased the weight of these vehicles and riveted armor was prone to shear off rivets on shell impact, which caused harmful spalling even if the armor managed to stop the actual shell [7].

To address these issues, new technological procedures like casting and welding were used instead of riveting. The most common type of armor became Rolled Homogeneous Armor (RHA), which was manufactured by squeezing heated steel block between two rolls multiple times, ensuring that the material properties of rolled metal were consistent throughout the whole armored plate [7]. RHA eventually became so prevalent that its properties became a standardized unit for armor thickness. The effectivity of modern composite armor is often recalculated into thickness of RHA plate with identical effectivity (RHAe).

When the tanks crept over the desolate battlefield, it was field cannons that managed to combat them most effectively [1]. Artillery has proven many times to be the most powerful and decisive part of army in any large-scale combat and so it was decided to supplement field cannons with new munitions to better combat armored vehicles, the Armor Piercing round (AP). These fully metallic shells omitted (or at least reduced) the use of explosive filler to punch through the armor via raw kinetic energy, generating shrapnel-like spalling, that would damage the equipment and wound the crew of the tank on impact.

1.3 Physical process of Armor penetration

When fast moving metallic projectile strikes the armor plate, many destructive processes happen together at once and the results vary depending on armor plate hardness and thickness, and shell shape, mass, velocity, and material hardness. As soon as the projectile hits the target, the projectile tip is subjected to enormous stress concentration, which is much more detrimental to sharp projectiles, that may crack throughout the whole projectile and shatter it, severely decreasing its penetrative power. Softer armor subjects the projectile to lower stress concentration and is worse at breaking the projectile down [8].

Once the projectile digs into armor, the temperature of the contact point, generated by projectile friction, spikes so much, that the armor in proximity of contact point can't dissipate the heat and melts. As the projectile progresses deeper into the armor, the melted armor surface is cut off. This process is called Adiabatic shearing and causes the projectile to slow down as it shears the armor. Hardened armor tends to shear faster than softer homogenous armor and performs worse when slowing the projectile down. As the projectile slows down, burrowed into the armor plate, the heat generated by friction is not hot enough to cause adiabatic shearing and projectile starts to squeeze the armor sideways.

Clashing of projectile with armor plate is also accompanied by shock waves and since the speed of sound in steel is very fast, the initial shockwave usually reaches the back of the armor faster than the projectile itself and reflects towards the contact point. The reflected and original shock waves then meet somewhere in the middle and create a local stress concentration, which can split the inside layer of armor and in worst cases tear off a chunk of metal and fling it inside of the vehicle. This tends to happen to armor plates with insufficient quality and may harm the tank controls and crew even if the actual projectile doesn't penetrate. Some shells, namely High Explosive Squash Head (HESH), in English nomenclature,

or High Explosive Plastic (HEP), in American nomenclature, were designed with this type of damage specifically in mind.

1.4 Face hardened armor and armor piercing and ballistic caps

Once the projectile overcomes the armor, the projectile will damage the backside of armor plate depending on armor ductility. Brittle armor tends to fragment, while more ductile armor is usually penetrated by a slug or a disc, sheared off the armor plate by the projectile. Softer armors try to change their shape to keep the adiabatic shearing process going for a longer period, which results in projectile losing its velocity to the state where it can't generate enough heat to shear off a solid slug and instead pushes the armor mass forward. If the projectile manages to go through the whole thickness of the armor, the armor opens at the end via the radial crack and the projectile pushes through the armor accompanied by small metal fragments, known as spalling.

Brittle armors are usually very hard, and their main purpose is to shatter the projectile, but once they have been penetrated, they generate considerable amount of fragmentation, which is lethal to vehicle crew. As the hardness decreases and ductility increases, armor becomes worse at shattering the projectile but better at slowing the projectile down. Softer armor is then better suited to protect heavy vehicles with sizable armor thickness, while the harder armor works better on light vehicles, that wouldn't have necessary armor thickness to slow down the projectile with ductile armor. Ductile armor could be paired with hard armor in the form of additional armor skirts or by hardening the outer side of ductile RHA plate either via heat treatment or via induction [7].

This combination of materials was known as Face Hardened Armor (FHA), and it combines properties of both hard and soft armors. The incoming projectile first strikes the High hardness layer, which attempts to shatter the shell. Once the shell overcomes the High hardness layer, it continues forward, damaged by the impact and attempts to overcome the soft layer of armor, which slows the broken projectile down. This armor proved to be very effective against the many interwar era munitions, but it was quite expensive and complicated to produce.

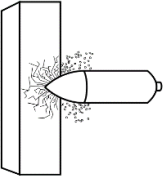
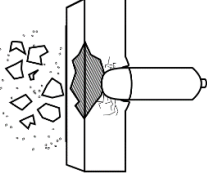
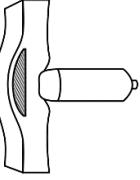
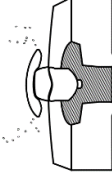
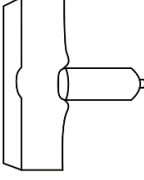
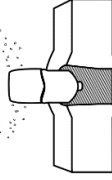
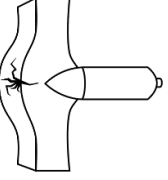
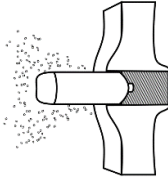
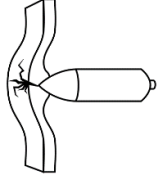
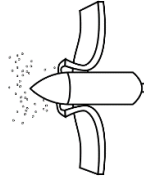
	Projectile strikes the armor plate	Projectile overcomes the armor plate
Very brittle		Fragmentation 
Brittle		Discing 
Ductile		Plugging 
Very ductile		Ductile hole growth 
Ductile & thin		Petaling 

Fig. 1. - Projectile behavior upon striking an armor plate

To counter FHA, which was by the time used on contemporary warships, Admiral Makarov invented a soft metal cap, that protected an armor piercing projectile upon the impact with armor and increased its resistance against shattering considerably. Projectile fitted with this so called “Makarov tip” is nowadays known as Armor Piercing, Capped (APC) shell and it was later shown that not only is this projectile more resistant against shattering, it also performs better against high oblique targets than standard AP rounds which were prone to ricochet [9].

Time has shown that the shape of the APC projectiles is not ideal, and they tend to lose their velocity because of the increased drag that the blunt headed projectiles suffered from. To solve this problem, it was devised that a Ballistic cap should be added to the APC projectiles, which would make the shape of the projectiles more aerodynamic. This additional cap was made from soft metal or plastic and was completely hollow [9]. On impact this ballistic cap broke off and the projectile itself continued penetrating the plate without hindrance. To differentiate shells with added ballistic cap, the BC was added to the APC acronym forming APCBC (Armor Piercing, Capped, Ballistic Capped)

1.5 Convertible tanks and other developments in tank mobility

As the armored vehicles shrunk to save weight and materials, it became clear that they could excel in more roles than just breaking through lines of defense. Small and nimble light tanks, tankettes and armored cars were often used in reconnaissance missions, patrols and skirmishing, tactics mainly popularized by German Panzerwaffe, which helped considerably in swift and decisive victories throughout the early years of Second World War [1].

To differentiate between the proposed usage of armored vehicles, British came up with new concept of “Infantry tank” a slow moving, heavily armored vehicle, which main purpose is to advance with infantry at walking pace and provide cover and fire support to infantrymen, and “Cruiser” or “Cavalry tank” which behave like pre-WW1 cavalry, which main goal was to find unprotected gaps in enemy defense, push through and harass enemy back lines, working independently of artillery and infantry. As such the Infantry tanks were often built to travel at small speeds and equipped with supple suspension to improve accuracy when firing on the move, taking advantage of their heavy armor and nearby infantry to protect the tank from other armored vehicles or anti-tank guns, while the Cruisers had to rely on their speed and maneuverability to escape from potential threats. As such the Cruisers were often lightly armored and armed.

One of the biggest pioneers of fast-moving tanks was an American inventor John Walter Christie, which invented a special type of adaptable tank, that could travel either on its tracks, which was the preferred option in combat situations and off-road driving, or directly on its road wheels, after its tracks were taken down. Since the metallic tracks suffered from serious wear, especially when travelling on paved roads, Christie's system promised to dramatically increase their lifespan and allowed the "Convertible tank" to travel great distances on its own power.

Similar experiments were made in Europe, namely in Sweden, with their Landsverk L-30, or Czechoslovakia, with their Kolohousenka series, which used large truck wheels, fitted over the tracks, lifting the vehicle higher, so that the tracks don't touch the ground when the truck wheels rotate. Both concepts were later scrapped because of its unnecessary complexity and added cost to each vehicle, without gaining any remarkable benefits.

Another improvement to tank mobility came with the introduction of amphibious tanks, that were designed to allow armored columns cross rivers and lakes without the need of a bridge. These vehicles were often poorly armored and armed to allow the metallic vehicle to float, severely reducing their combat effectiveness and as such were quickly phased out of military service. Even though the light amphibious vehicles were not successful, they laid down foundation to armored landing crafts, which played major role in Pacific theater of WW2 and are used until today.

1.6 Chemical energy projectiles

So far, every mentioned projectile (except the HESH shell) belongs to the group called Kinetic energy projectiles, i.e. projectiles that penetrate the armor via kinetic energy. The other way to penetrate the armor is via chemical energy, which is generated via chemical reaction (explosion) directly at the point of contact with target. This allows chemical shells to not depend on muzzle velocity which is an important factor in determining the power of Kinetic energy projectiles. Chemical shells are then better suited for light vehicles that can't handle heavy recoil, which is often necessary to launch projectiles with high muzzle velocity. As the projectile flies, it loses its velocity because of the drag, which makes Kinetic energy projectiles lose its power when firing at range, unlike chemical energy projectiles [10].

The most prevalent chemical round is a modified High Explosive (HE) shell which utilizes a shaped charge, the High Explosive Anti-Tank round (HEAT). Despite its acronym, the

shaped charge doesn't melt the armor via the immense heat, but instead utilizes the explosive to blast in a single direction, towards the armor. When HEAT projectile strikes the target, the explosive charge is detonated starting from the side opposite of the contact point and the explosion starts to spread through the conical cavity in a shell, gaining considerable pressure before it is released through an opening in a form of jet of high velocity metal particles. This effect of focusing the blast energy through a hollow cut is called Munroe effect and it was extensively weaponized throughout the Second world war in the form of tank shells and anti-armor grenades, mines, and projectors. The power of this effect scales with the diameter of HEAT shell, making it better suited for cannons with large calibers.

Despite its effectiveness against solid steel armor, HEAT rounds are easily countered by spaced armor or applique plates or meshes, which detonate the HEAT shell prematurely and the round fails to generate long enough jet to reach behind the vehicle's main armor and modern composite materials, which offer greatly increased protection against HEAT jet [10].

Similarly to HEAT, the Explosively formed penetrator (EFP) utilizes power of explosive charge to propel soft metal, usually copper, lining with great speed. The metallic disc-shaped liner is propelled forwards via the blast, and deformed into the shape of a slug, like standard AP rounds. This metallic slug can reach farther than standard HEAT jet, but it is still weak against spaced armor, which damages the slug when it penetrates first layer of armor.

As already mentioned above, the HESH shell works differently to usual anti-tank shells. The plastic explosive, which makes up majority of the shell, "squashes" itself onto the armor on impact, covering large surface area of the armor, before detonating, which creates a shock wave, that is supposed to stress the armor and generate scabbing and spalling of the inside layers of armor. This spalling can injure tank crew, disable tank modules, and even detonate stowed ammunition.

1.7 Armor obliquity

Most tanks designed by the time of Second world war made use of angled armor which improves the chance to ricochet projectiles and increases armor "Line of sight" [7]. By increasing the angle of impact, either actively, by rotating the vehicle on battlefield or passively by mounting certain armor plates, mainly the upper front plate and lower front plate, in a V pattern, the distance that the projectile must travel through the armor increases, while the actual thickness of the armor stays the same. Vehicle utilizing angled armor can stay

lighter than the vehicle with same amount of armor effectivity with flat armor. Some tanks, mainly in the second half of the Second World War, were designed with armor angling specifically in mind. The trend was set by the Soviet T-34 tank, which offered angled armor plates not only on its front but on its sides and back as well. This angling obsession culminated in 1945 when was the Soviet IS-3 tank developed, featuring a “pike-nose” design, which was aimed at providing maximum possible angling directly from the front of the vehicle, drastically increasing its armor protection when engaged directly head-on.

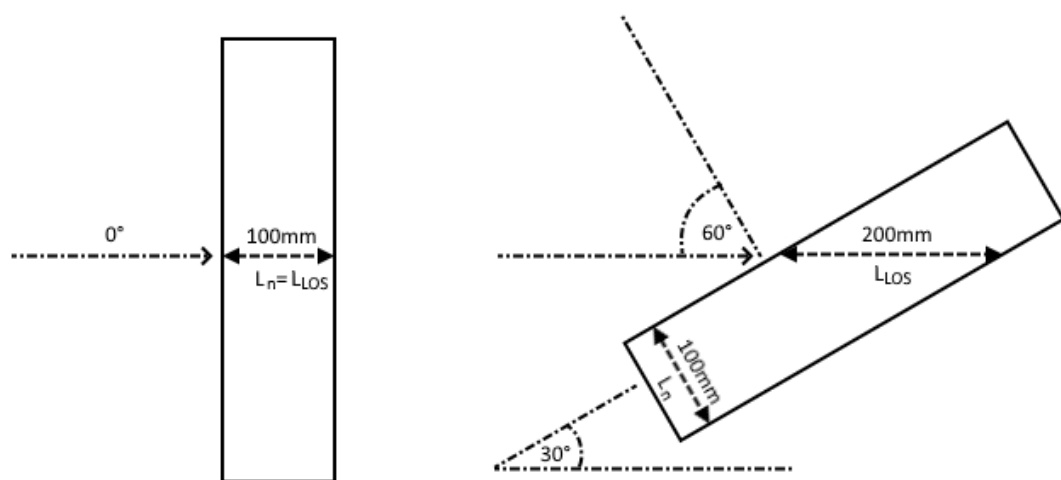


Fig. 2. – Armor LOS

As already mentioned, the armor angling doesn't only increase its line of sight but increases the chance of projectile ricochets. The bigger the angle of impact, the bigger chances the projectile will to ricochet away from the armor surface. There are other parameters, that dictate how well is the projectile suited for penetrating angled armor. Namely it is the shape of the projectile and the ratio of armor thickness and projectile caliber. When the projectile attempts to penetrate the armor at very high angle and its diameter is smaller than the armor thickness, the projectile gives way upon impact with armor, and slides along the surface of the armor plate, causing ricochet. When the projectile caliber is significantly bigger than the armor thickness, the armor plate gives way to the projectile, denting slightly, which allows projectile to cut into the armor and start the penetrating process. This term is known as Overmatch and thanks to it, projectiles with big diameter are able to penetrate light vehicles despite their highly angled armor.

1.8 Modern tank development

As the steel projectiles reached terminal velocity, i.e. they struck the armor in such speed, that they shattered, some effort was made to introduce new materials to beat the thick armor plates [8]. A core of high hardness material, like tungsten was encased in steel, to protect the core from shattering and used with success against highly armored targets. This projectile, called APCR (Armor Piercing Composite Rigid), or HVAP (High Velocity Armor Piercing) in American nomenclature, used high hardness, heavy core, with smaller diameter than the caliber of the cannon it was fired from, and delivered concentrated pressure into a smaller contact point of the armor. Albeit expensive, the APCR rounds were issued to most of the anti-tank cannons in limited quantities and served throughout the World War 2.

Additional upgrade to APCR shells came in the form of discarding sabot. A sabot is a special shell formed around the inner sub-caliber round, which allows a cannon to fire projectile with narrower diameter than the cannon's caliber. After the cannon fires, the sabot guides the sub-caliber round out of the barrel and falls off from the projectile when it exits the barrel. This allows the cannon to fire projectiles with higher velocity, compared to full caliber rounds [10]. Since many modern tanks are built with smoothbore cannons, to allow firing ATGM (Anti-Tank Guided Missile) munitions, some tweaks were necessary to stabilize the projectile when it exits the barrel and so the modern APDS shells are very similar to medieval arrows, featuring fin-like arrow fletching and are made from solid tungsten or depleted uranium. These APFSDS (Armor Piercing Fin Stabilized Discarding Sabot) or simply Dart projectiles use the fins to stabilize the round as it flies towards its target and allow modern tanks to penetrate insane amounts of RHA, up to the point where it is impossible to protect the vehicle with standard steel armor.

To combat the new APCR, APDS and HEATFS rounds, some changes had to be made to armor structure, because standard RHA plates would have to be enormously thick to protect a vehicle, which would slow it down considerably. Combination of ceramics, steel, plastics or even air was packed into a protective array, creating the Composite armor. These composites range wildly, and each military uses different approach to protect their vehicles, usually keeping their exact combination confidential. Composite armor is usually also considerably lighter than solid RHA plate of same thickness, allowing vehicles to carry thicker armor, without severely impacting its mobility and it is not uncommon that arrays thicker than one meter are mounted on some parts of modern Main Battle Tanks (MBT). There are also many

other contemporary inventions to protect or damage armored vehicles, but they won't be mentioned, since the goal of this thesis is to talk about gamification of interwar to WW2 tank warfare.

2 UNIVERSAL ARMOR PENETRATION FORMULAE

Over the years many different armor piercing formulae were invented to describe theoretical effectiveness of armor plates or armor piercing rounds, to save time, for tank and warship designers, and material, that would be wasted on live ammunition testing. Some of these formulae were very specific, explaining penetration process of armor penetration of projectile striking a specific plate at normal obliquity, but there are some armor penetration formulae, that could be labeled as “universal”. These formulae introduce set of constants and variables, that allow abstract description of both armor plate and projectile and could be theoretically used to determine, albeit not 100% correctly, if the penetration in the environment specified by the user was successful or not. Since these formulae were sufficient for approximation in military environment, they should be more than adequate to be used in gaming industry as well.

The process of armor penetration is quite complicated with many possible outcomes. The term “Penetration” itself varies between additional terms like “Through crack” or “Nose through” and “Base through” or “Complete penetration”, each specifying how far did the projectile manage to advance, and the “penetration” of tank armor doesn’t necessarily mean that the tank was damaged by the projectile. Some sources also claim that the term penetration is used incorrectly [11].

“In considering the effects of missiles on targets it has been found useful to distinguish between penetration and perforation. The term penetration is reserved for the entry of a missile into the armor without passing through it. The term perforation implies the passage of the missile completely through the armor.”- Headquarters, US Army Materiel Command: Elements of Armament Engineering Part Two Ballistics, 10-8.1a

2.1 General armor penetration formula

Armor penetration formulae are usually written in following form:

$$V_L = \frac{(K)(C)T^t D^d}{[W^w \text{Cos}^a(Ob)]}$$

Fig. 3. – General armor penetration formula

V_L specifies the striking velocity of projectile upon hitting the target. K is a numerical constant, that allows to transform the formula between metric, or imperial, units. C is a plate quality factor constant, which may or may not be used depending on the formula. T stands for armor plate thickness, while D describes the diameter of a projectile. W , or sometimes M as Mass, specifies the total weight of the projectile and lastly the Ob stands for obliquity, the angle of impact between normal of armor plate and projectile impact in degrees, where 0° is used for perfectly perpendicular shot and almost 90° for grazing shot. The t , d , w and a represent the power values of their respective parameter, which allows to increase or decrease importance of the parameters. All these parameters have an impact on the result of a successful armor penetration [6].

2.2 De Marre nickel-steel armor penetration formula

One of the oldest and most known universal formulae, De Marre formula was devised by Jacob De Marre in late 1880s and uses the “De Marre Coefficient” or simply C as a comparison of test plate with average French 1890 nickel-steel armor plate [6]. The calculated striking velocity is the velocity needed to just barely fully penetrate a nickel-steel plate.

$$M \cdot \frac{(V \cdot \cos(Ob))^2}{D^3} = C^2 \cdot \left(\frac{T}{D}\right)^n \cdot D^{-s}$$

Fig. 4. – De Marre formula

This formula has been originally invented to describe penetration of pre-WW1 warship RHA armor, but it was perfected throughout the time, with added constants and variables, like for

example obliquity or ability to simulate penetration of FHA, so that by the start of WW2 it was one of the main formulas used by militaries throughout the world.

The C, or K, constant is usually determined by firing test and is a rather comprehensive description of armor resistance and projectile effectivity packed into a single number for easier calculations. Some sources [10] claim that the C value is used to determine armor quality, specifying following values as common:

Armor plate	C value
Low carbon steel plate	1530
Nickel steel plate	1900
Generic homogenous armor	2000 - 2400
Face hardened armor	2400 – 2600

Table 1. - Common values of DeMarre coefficient

Other sources [12] claim that the C value is used to describe the shape and hardness of the projectile, claiming that the C constant for soviet blunt-headed shells is somewhere around 2400. Reality is probably somewhere in between, and the C constant is a combination of both, but the fact is, that as the C value increases, the resulting penetration effectivity decreases.

The n represents a shape and sharpness of the projectile head and has an impact on the effectivity of the projectile when facing angled armor. As the n increases, the effectivity against the angled armor reduces.

2.3 Krupp all-purpose armor penetration formula

Another important historical formula is an all-purpose armor penetration formula invented by Krupp company. Krupp was leading developer and manufacturer of modern artillery in Europe, and it is only logical that they devised their own simple to use armor penetration formula to estimate effectiveness of their weaponry before building and testing the real thing.

$$T = \frac{V \cdot \sqrt{W}}{C \cdot \sqrt{D}}$$

Fig. 5. – Krupp formula

The C constant is a comparison coefficient, which is very similar to constant found in De Marre formula [6]. Unlike the De Marre formula, the Krupp formula doesn't take obliquity into account and only produces results for projectiles hitting the armor perpendicular to the armor plate with 0° obliquity. To remedy this additional dataset was produced for each projectile separately, which could fix the deficiency for military purposes, but make it rather complicated for implementation into a game setting.

3 EXISTING TANK SIMULATORS

Over the years there were many different tank simulators, that offered varying degree of realism with some real-life mechanics getting simplified to afford easier understanding and better entertainment for the casual player. Problems like ballistics, armor penetration, crew management, module system, reduced vision, absence of GUI helpers and others are something that the real-life tank crew has to deal with every day, but majority of gamers are not willing to spend their time learning complex systems and deliberately reduce the effectivity of their vehicle by sharing it with other teammates. Because of this, the majority of tank simulator games on the market right now offer very simplified systems to allow users to enjoy driving a tank without a need for a steep learning curve.

3.1 WarThunder

Created in 2012 by the company Gaijin Entertainment, the WarThunder started as an aerial combat simulator and was later expanded with additional modules for ground (tank), naval and helicopter battles. Its original goal was to compete with World of Tanks, a game by another Russian company, Wargaming, which was evident from number of ads aimed at degrading the simplifying mechanics of World of Tanks and luring the playerbase towards the WarThunder [13]. WarThunder is currently one of the most played tank simulators.

3.1.1 Gameplay

WarThunder combines vehicle combat simulation, MMORPG, and mobile idle game mechanics to create a rather repetitive gameplay loop of researching vehicles of higher tiers and expanding the user virtual garage through extensive replaying of the game's single gamemode. This process is called "Grind" and represents spending time playing a barely enjoyable game to obtain a reward, in this case a vehicle [14]. To generate revenue, WarThunder allows players to reduce the grind by many in-game shortcuts, be it premium vehicles, orders and boosters, or premium subscription, which greatly increases the number of in-game resources generated by playing [15].

Like World of Tanks, the WarThunder is aimed mainly towards the casual players, despite incorporating number of complex mechanics into the base gameplay loop. Combat vehicles are compromised from number of modules, be it mechanical tank parts of crew members, which keep the vehicle running and the vehicle is knocked out by damaging these modules,

unlike other casual games, like World of Tanks or Armored Warfare, that set hit points pool for the vehicle which slowly depletes on receiving damage. This system allows for quick and decisive engagements, which are often solved with a single placed shot. To increase the playtime per battle, the player doesn't attend the battle with a single vehicle but with a roster of similarly powerful vehicles called "lineup".

Since the WarThunder offers vehicles spanning from interbellum period up to contemporary armored vehicles used in modern militaries, the need to somehow balance these vehicles to offer fair fight arose and each vehicle has a set level, called "battle rating". Lineups are then assembled to accommodate vehicles of similar battle ratings, since the final battle rating of a lineup is as high as is the highest rated vehicle in a lineup.

Battles are fought in random environment, generated out of list of possible maps and environmental conditions, with randomly matched adversaries and teammates, which creates an illusion of each battle being unique, while keeping the basic mission objectives same throughout the battles.

Each player controls a whole tank, unlike real-life armored vehicles, that are usually crewed by multiple soldiers. To compensate for the workload necessary for a single person to control a complex vehicle, most of the vehicle systems are greatly simplified, which allows a single player to control an armored vehicle much easier than an experienced crew would in real life settings [15]. This warps the realism effect even further, because certain vehicle strengths and weaknesses, like for example number of viewports, transmission controls, or size of blind angles have minimal effect on gameplay.

3.1.2 Ballistics and damage model

To calculate penetration success, WarThunder used several historical data tables to resolve penetration success, but since the scale of the game became quite big and details of some of the vehicles represented in game are still classified, an empirical formula is now used instead [16]. This improves the modularity of WarThunder, allowing developers to easily add new vehicles.

WarThunder offers range of different munition types from HE shells to tungsten kinetic penetrators, with each of the shells having their own benefits and disadvantages. The chemical shells are detonated upon the collision with buildings and shrubbery and the kinetic shells lose their velocity and penetration effectivity as they travel. Armor impacted at steep angle

can deflect the shell, which continues to fly with new trajectory and could potentially endanger a different target. Explosions, either generated by chemical munitions or by detonation of vehicle ammo rack, generate shrapnel, which can damage or destroy lightly armored vehicles.

Each cannon is fitted with ballistic sights or even scope, that allows the player to easily engage targets at distance without needing any sort of virtual assistant, usually provided in competing games, enhancing the realism. Projectiles have different attributes, like mass and initial velocity, which dictates how well will the projectile slow down over its lifetime and some calculations had to be made by the player to account for the projectile drop-off.

Successful penetration of vehicle armor usually results in damaging tank modules, like cannon breech, engine, transmission, turret ring controls etc., or wounding crew members, that are eventually knocked out. Each module has its own hit point pool, which is depleted by taking damage, causing an effect when the hit point pool reaches zero, usually disabling the module's functionality or even destroying the whole vehicle in fiery explosion. Thanks to this system, firefights are usually swift and brutal, sending the targeted vehicle back to garage in a matter of milliseconds.

Repairing of the damaged module is done by stopping the vehicle in place to perform repairs and waiting for a set period of time, depending on the scale of damage dealt to a vehicle and the amount of healthy crewmembers. Tank locked in this state can rotate its turret and use its weaponry but cannot move until the repairs are done. Same applies to firefighting, which is a necessary countermeasure to combat fires, usually ignited by projectiles hitting engine or fuel tank.

Despite any issue with the gameplay aspect of WarThunder, its ballistic system is extremely detailed and offers very well-written wiki [17], which contains great deal of information, making the WarThunder a great source of information. It also offers great deal of playable vehicles, which are modeled with insane attention to detail. Combination of above mentioned makes the WarThunder excel as some kind of virtual museum, filled with historical gems.

3.2 Squad 44

The Squad 44 started in 2018 as a Post Scriptum [18], initially developed as a WW2 mod for Squad, a modern realistic combat simulator, it became a standalone game for few years,

before it was eventually acquired by the developers of the original Squad game, the Offworld Industries, in 2023 and rebranded to better fit into the company's repertoire [19].

3.2.1 Gameplay

Squad 44 is a realistic World War 2 combat simulator, offering both infantry and tank combat mechanics, allowing players to not only combat other tanks, which is what the other competing tank games aim at most of the time, but infantry as well, greatly increasing immersion and variety [18].

Unlike the WarThunder, Squad 44 doesn't match players into an automatically created lobbies, but instead allows users to host their own servers, each with their custom setting, and map rotation, allowing users to connect to their favorite servers or play on their favorite maps.

Instead of battle ratings, the warzone is split into several theaters, like Normandy, Invasion of France, Invasion of Greece and so on, each offering its own unique maps, vehicles, and infantry equipment, which are somewhat balanced. The fact is that real-life WW2 combat differed greatly, theater by theater, and in some cases some armies pitted against themselves in combat were better equipped than the opposing ones, creating a slight imbalance on certain theaters. While this can ruin the fun for casual players, for more experienced soldiers it is a minor drawback that must be played around to prevail [18].

Each player plays as a single crewmember or infantryman, which moves the tank combat closer to realism. While the tank crew is locked inside a crew compartment of armored vehicle, its view outside is lackluster and it can easily become prey to infantry tank hunters, anti-tank guns or other tanks, attacking from unexpected angle. By splitting the controls of a single vehicle between many players, the total reaction time of the vehicle plummets greatly and it is only up to vehicle crew to communicate well enough to be able to react to potential targets and dangers. Crew members have an option to open the hatches and climb out to improve the tank awareness, allowing infantrymen to engage tanks without armor piercing weaponry, picking the crewmembers one by one as they stick out of the armored vehicle [20].

When it comes to mobility, the Squad 44 features immersive transmission, which allows the player to finesse the throttle and manually change gears to either achieve better agility of armored vehicle while going slowly or dramatically increase the speed at which it can drive,

sacrificing its ability to quickly turn. This system is both challenging, ensuring that the player playing as a driver must learn proper tank driving methods and improve their skill, and rewarding, since harder the gameplay is, the better is the satisfaction of player, when they play well.

3.2.2 Ballistics and damage model

Like WarThunder, the Squad 44 also uses empirical formula to calculate armor penetration success, which is extremely useful for community driven development, that has access to rather powerful tools, allowing modders to create and distribute their own content easily [21].

Squad 44 offers wide range of period accurate cannon shells, from generic kinetic AP shells to chemical shells like HEAT, each with its own preferred use case and limitations, forcing vehicle commanders to count their rounds and select the perfect shell for the moment.

Each time the projectile comes into contact with the armor, a spalling is generated both inside, if the projectile overcomes the armor, and outside, which has most effect when the shell shatters, causing serious damage to infantrymen in close proximity to vehicle. Some shells, namely the HE and AP shells with explosive filler, generate considerably more shrapnel and spalling than generic AP rounds, but usually offer reduced penetration, making them more suitable against soft targets or after a successful flanking maneuver, firing against target weaker back or side armor [22].

Modules are divided into vital, that cannot be fully repaired, and non-vital components, that can be repaired by dismounted crew member, and they use similar hit point pool system, like WarThunder modules do. Crew can only repair destroyed modules into barely functional state and to fully repair the vehicle, it must be driven back to repair station, which returns all the hit points back to 100%. Vehicle can be knocked out by killing its crew, blowing up the ammo compartment or destroying 3 or more vital components, which renders the vehicle unreparable, and the crew will have to bail out.

Additional mechanic to make destroying light vehicles with strong anti-tank weaponry is “hull-break”. Vehicles have a dedicated hit pool, that tracks the state of their hull integrity, which is damaged every time a vehicle is penetrated by a projectile, even if no important module was damaged. This makes destroying light vehicles like armored cars much easier [20].

Modules like engine, fuel tank or ammunition stowage can ignite a fire when damaged, that can spread to another modules and harm the crew. To combat the fire a crew member has to use the fire extinguisher, abandoning their battle station while firefighting [21].

3.3 IL-2 Sturmovik: Tank Crew – Clash at Prokhorovka

IL-2 Sturmovik: Tank Crew – Clash at Prokhorovka is a part of Il-2 Great Battles series and can be played both separately and integrated with other Great Battles titles, offering aerial combat. This integration allows players to experience combined arms operations where warplanes work in tandem with ground forces, dramatically increasing immersion of the game [23].

3.3.1 Gameplay

IL-2 Tank crew offers both singleplayer and multiplayer combat, putting players into role of a tank crew, fighting on eastern front of WW2. Each player can either command the whole vehicle, by taking role of a tank commander and issuing orders to their AI crewmembers, or by multi-crewing a vehicle with other players. The player takes control of a single soldier in both cases, watching the battlefield from soldiers' eyes, greatly reducing visibility out of an armored vehicle.

AI is well made and provides sufficient challenge for player/s as other armored vehicles, artillery and even warplanes. Light vehicles try to aim their weapons at tracks and viewports, cracking them in the process, which dramatically reduce visibility and heavy vehicles turn their thickest armor towards the player/s.

Singleplayer campaign consists of only a handful scripted missions, where the player plays as either Soviet or German tank commander, leading its vehicle to a battle taking place near the Prokhorovka village in 1943, but thanks to community-made mods, the replayability can be increased by using one of the “mission generators”, which allow players to generate random missions.

Vehicle roster is with its 10 playable vehicles, and a handful of collector vehicles for additional price, quite limited, especially compared to WarThunder, which offers hundreds of vehicles to choose from, but IL-2 Tank crew offers detailed depictions of vehicles, including fully modeled crew compartment, with functional gauges, instruments, and animated crew. It is also possible to play with a VR headset, greatly enhancing the immersion [23].

3.3.2 Ballistics and damage model

IL-2 Tank crew uses a combination of reference data values and empiric De Marre formula, that supports the reference data. This makes the armor penetration behave almost identical to real-life tanks, which the in-game vehicles are modeled off.

Vehicle doesn't have a single health pool and is split into a number of modules, which can be destroyed by the enemy shells, and in some cases, like for example the periscopes, viewports and scope, even with small arms fire. Damaged vehicle can be repaired by stopping for a while, similar to how the repairs work in WarThunder, but the repairs can take much more time in IL-2 Tank crew [24].

Each vehicle, or more specifically each cannon, has its own sights or scope, which is specifically calibrated for cannon's ballistics, which makes firing at larger distances less of a chore and is extremely satisfying.

Some of the modeled tanks carry a serious amount of armor, which is most noticeable when engaging German heavy tanks and SPG's and can survive multiple hits without an issue, especially when they rotate the hull into a diamond shape, forcing the incoming projectiles to meet the armor at sharp angle. It is evident that the game was not balanced as hard as WarThunder and Squad 44 are, mainly because the player doesn't engage in combat with other players, as all of the enemy vehicles are controlled by artificial intelligence, which makes the balance between sides unnecessary, and the simulator can stick to realism.

Vehicle can be destroyed by blowing up the ammunition rack, burning down or by killing its crew, which is very well depicted in-game by vehicles continuing to travel in the same direction even after its driver has been killed, forcing the player to either choose to save their ammunition and risk enemy vehicle "coming back to life" later on, or firing a pointless shell into a dead husk of a vehicle. This adds a certain level of uncertainty into the game, very similar to problems that the real-life tank crews have to deal with.

4 UNITY PHYSICS ENGINE

The goal of this thesis is to design a ballistics module for Project UTW, a modular tank simulation game, based on Unity engine. As such, it is necessary to understand how the physics is simulated in Unity.

Unity offers wide range of components to make the development of complex physical systems much simpler, while allowing user to create their own components, mainly via C# Scripts. Each of the components is described in great detail in the official Unity documentation [25].

4.1 Rigidbody

One of the most prominent components, used for physics simulation in Unity, is the Rigidbody. This component is used to describe an object which is susceptible to gravity and other forces, generated either by collision with other colliders or by applying direct force to the object. Attaching a Rigidbody component to an object allows it to react to any changes in the scene without an explicit scripted behavior, which is perfect for a lightweight ballistic simulation.

4.1.1 Rigidbody parameters

Each Rigidbody component specifies number of parameters, which allows the user to better describe how should the object travel through the air or how much force it should generate in case of a collision with other Rigidbody or a static collider [25].

One of such parameters is Mass, which dictates how heavy the object is. Heavier objects are subjected to bigger gravitational force and transmit more force in case of an impact. Mass is also one of the parameters needed to successfully calculate the penetration success via the DeMarre formula.

Another very important parameter is Velocity, which is stored as a Vector3 variable. This parameter describes the current speed and direction at which the object is currently traveling. The DeMarre formula needs Velocity as an input parameter as well, but these two velocities are not directly compatible. Since the Rigidbody velocity is stored as a three-dimensional vector, it first needs to be translated into a single value speed. To do that Unity includes a special sub-parameter, the Velocity.Magnitude, which returns the actual object speed in meters per second, which is perfectly compatible with the DeMarre formula.

There are many other important parameters of RigidBody, that allows the user to describe physical details of a moving object, but for the sake of the Unity driven ballistics module, the final important parameter is the Drag. Drag parameter allows the user to specify the rate at which the object loses its velocity. The higher the drag, the stronger the slowing effect is and each FixedUpdate the object loses velocity based on this formula:

```
newVelocity = oldVelocity * 1 - deltaTime * drag;
```

It is possible to switch the IsKinematic flag to true, which will disable any physical interaction with other objects. Another useful flag is the UseGravity, which enables to turn on and off the influence of gravitational force on the RigidBody.

4.2 Colliders

Collider is a description of physical bounds of the object [26]. Colliders allow interaction between objects, especially if they are combined with RigidBody Component. Colliders that are part of an object that doesn't include the RigidBody are called Static colliders, while the colliders that are part of an object that includes the RigidBody are either Dynamic or Kinematic, depending on the RigidBody IsKinematic flag.

Unity offers a wide range of different collider shapes, like for example BoxCollider, SphereCollider, CapsuleCollider or even custom MeshCollider, which takes the shape of a user-made model.

4.2.1 Triggers

A special form of a collider is a trigger, which is a standard collider with its IsTrigger flag switched to true. Triggers don't cause collisions but allow the Physics engine to detect other Colliders and broadcast events, such as the OnTriggerEnter, which allow the user to put into effect custom scripts, without physics engine handling the collision.

4.3 Collisions

Whenever two colliders collide, a collision event occurs, which handles the direction and force of repulsion for both colliding objects. Collisions happen automatically and there is no need for the user to handle them, but Unity offers the ability to catch the events, broadcasted when the collision happens and apply some specific rules. One such event is named OnCollisionEnter and it is called in a nearest FixedUpdate, after two or more objects collided [26].

The `OnCollisionEnter` allows the user to implement special rules for collisions, like for example switching certain parameters in one of the colliding object's scripts, spawning new objects, increasing score counters and many other possible cases.

4.4 Raycasts

Raycast allows the Unity Physics to send out a Ray, which main goal is to check for any potential targets in its path and return either `True`, if a collider was hit, or `false` if there are no viable targets in its path. A successful hit creates a `RaycastHit` instance, which contains many useful data, like for example distance to target, the hit collider, or the point in space where the Raycast hit the target. Its main goal is to obtain an information about the hit object and allow for instantaneous action with hit object, directly within the script, which called for a Raycast [27].

4.4.1 RaycastAll

A Raycast function sends out a ray, that will try to hit a single collider, but if the need arises to check for multiple targets in a single time frame, a `RaycastAll` should be used instead. Unlike the simple Raycast, `RaycastAll` returns all of the hits as a list of `RaycastHit`s, allowing the user to handle each `RaycastHit` independently one after another, in an order in which the `RaycastAll` hit the colliders, starting from the closest to the Raycast origin point.

4.5 FixedUpdate

Each frame an `Update` event is raised, which serves as the main loop of most of the components, inheriting from the `MonoBehaviour` class. `Update` calls allow the game to check for player input or handle basic operations but are not appropriate to be handle any Unity Physics based calculations, as the `Update` function is called depending on the frames per second, that the Unity client currently runs in, which may cause the calculations to return inaccurate results. As such the `FixedUpdate`, an `Update` variant, should be used instead [28].

Unlike the standard `Update`, the `FixedUpdate` is called at a consistent rate, which allows it to better cooperate with other Unity Physics engine functions. The default rate at which the `FixedUpdate` is called is 50 frames per second, but it can be easily customized in Project settings, allowing the user to slow down or speed up the rate at which the `FixedUpdate` is called.

II. ANALYSIS

5 UTW AND ITS SYSTEMS

UTW is Unity based project developed by students of TBU [29], which aims to create a free-to-play, open source, modular, semi-realistic tank combat simulator, which will be released on Steam in upcoming years. Main purpose of UTW is to create an in-house large-scale Unity project, which could serve students of TBU, interested in game development, as their starting point of game development career. As such the scope of UTW is rather complex and the main 3 pillars, that the whole project stands on are modularity, playability, and realism, in that order.

Whole project is built from the scratch, with several important libraries and asset packs, which are the Unity Fish Networking, which handles the server-client communication and allows master servers to handle multiple lobbies at the same time, and Physics Tank Maker, which offers well-made track simulation and is heavily built upon in UTW [30].

5.1 Gameplay

Unlike the above-mentioned games, the UTW will feature a different approach to connecting a player to a battle. Instead of a matchmaking system or a server list, player first has to join one of the “Shards”, which is a term specifying a server, hosting an instance of UTW, which will be fully customizable by its owner, allowing them to implement different vehicle modules, maps, factions or even gamemodes. After a player’s first connection to a new Shard, player will be asked to join one of the resident factions, which each has its own players, lore, tank presets and other progression meters, like score, resources, victory points or other meters introduced by the Shard owner.

Once the player joins a faction, they can attend battles, that are fought in one of many lobbies, that the Shard can provide. Players will be able to organize their own battles, specifying the map, that will be used to host the battle, which factions can participate in a battle or even detailed settings, like for example maximum effectiveness of a tank preset.

Vehicle assembly will be another unique part of UTW. Instead of storing vehicles as a bundle, that the players can choose to play with, the asset database, critical part of UTW, that holds the loaded assets, stores vehicle parts like turrets, hulls, suspensions, weaponry and other modules, separately. This system allows players to create unique combinations of combat vehicles, even with a few of different modules.

Tank modules will be assembled via a blueprint called “Preset”, which holds information about a specific pattern of vehicle and the modules required to assemble it. These Presets will be serialized on the server in the form of JSON file with minimal size, allowing players to create hundreds of presets without sacrificing too much of a disc space on the server.

UTW will be strictly multiplayer and doesn't feature any AI enemies or allies. All of the crew member positions will be manned by the players, who have to communicate to control the vehicle effectively, each contributing to the vehicle wellbeing by manning their respective battle stations.

Each Shard will have its own whitelist/blacklist, which will allow Shard owners to protect their Shard from cheaters and other problematic individuals. Giving the power to protect their own server frees the hands of official developers, who would have to police the servers by themselves.

5.2 Ballistic simulation requirements

A number of requirements were specified prior to making the ballistic simulation module [29]. The UTW is aimed mainly at tank warfare of vehicles, loosely based on tanks of “Interbellum” period, spanning from years 1918 to 1938. As such, it is safe to assume, that most of the modeled vehicles won't offer much protection to their crew and internal modules, and anti-tank weaponry will be rather weak.

Perception of players, controlling the vehicles from inside of the vehicle, seeing outside only thanks to a rudimentary slits and visors, will be severely reduced and as such, there is no need for the simulation to be too detailed since the chances are that players won't be able to notice the details.

Since each Shard has to host multiple lobbies, each filled with number of players and combat vehicles, the simulation should be relatively undemanding, to allow shards to run more lobbies without a visible lag.

The simulation must be modular, and every calculation must be universal, to allow community developers to create wide variety of their own content. Hardcoded parameters should be reduced to a minimum and preferably stored as an adjustable variable, that can be modified by the developers.

While the equations and formulas don't have to be exactly identical with formulas used in real life, they should be at least roughly similar to the existing simulations, to allow players recognize patterns from different games and allow community developers to create somewhat realistic depictions of historical vehicles, even though it is not the main goal of UTW.

Armor penetrating process has to be predictable and even though it is possible to include a stochastic element into the simulation, the player must be able to determine if they are able to achieve their planned objective, be it penetrating a certain armor plate at certain angle, or deflecting a projectile aimed at a part of vehicle they are exposing to enemy.

As already mentioned above, the playability of the UTW is slightly more important than the realism. As such, it is possible to offer players a certain degree of slightly unrealistic features or mechanics, that will increase the quality of life for the players. These should be however reduced to minimum to keep the game from straying into arcade territory.

6 UTW BALLISTICS MODULE

To better describe and execute processes happening when the projectile strikes the target, without hampering the engine computational power and overloading player with information, UTW ballistics module should implement some form of simplified mechanics, based on physical processes that happen in real life. Since the armor penetration process is rather stochastic and every single nuance cannot be possibly pre-calculated in real time even with modern specialized equipment, a decent approximation will be more than sufficient for the needs of UTW.

6.1 Necessary mechanics

6.1.1 Drag and gravity

When the shell is fired from the tank cannon, it should travel through the air using the Unity engine Rigidbody component, which is perfect for such task. It allows the system to track the velocity and mass of the projectile at any given point in time and thanks to the drag parameter, allows the engine to slow the projectile down as it travels.

6.1.2 Ricochet

As soon as the projectile comes into contact with an armored plate, a ricochet simulation should occur that will determine if the projectile can continue on its path or if it is diverted away from the armor plate. As this process is stochastic in nature, a simplification of some sort should be used to determine the ricochet result, possibly by taking an inspiration from one of the above-mentioned existing tank simulators.

Despite being simplified, the process should still reflect real life ricochet chance and allow some sort of modularity. Another important aspect is taking into account overmatch effect, which will allow high caliber cannons to better engaged targets sporting highly angled armor.

6.1.3 Penetration

If the projectile overcomes the angle of the armor plate, the penetration simulation should take place, determining if the projectile is stopped by the armor or manages to pass through into the target vehicle. At this point it is advised to use one of the above-mentioned armor

penetration formulas, preferably the DeMarre, as it allows the formula to take into account the angle at which the projectile struck the armor plate.

As the DeMarre formula describes the maximum armor thickness at which the projectile can pass through completely, there is a need to implement another state, a “partial penetration”, to complement the “complete penetration”, when the shell overcomes the armor and continues forward, and “non-penetration”, where the projectile is stopped by the armor. Partial penetration should describe the state when the projectile manages to overcome the armor and opens the armor plate in such a way that harmful spalling, which damages the target vehicle, is generated, but won't keep any energy to continue pushing forward.

6.1.4 Spalling generation

Since the spalling generated by penetrating projectile occurs randomly, it is almost impossible to calculate scientifically. As such a simplified system should be devised, that allows the module to automatically generate a number of spalling fragments of variable count and size, showering the insides of the target vehicle at a variable angle.

This system should take into account thickness of armor plate, and velocity, mass and caliber of the shell, generating more spalling when shell and armor plate displace bigger volumes of metal. Explosive fillers should generate additional amount of spalling on top of the strictly kinetic collisions and cover wider area with metallic shrapnel.

6.1.5 Tank crew and modules

The vehicle will be split into modules, each describing an important part of the tank, that is necessary to efficiently commandeer the vehicle on a battlefield. Each of the modules will sport its own hit point pool, which describes the current status of a module.

To better describe the current state of a module, they should be split into three states. An operational module is able to do its job whether it is providing horsepower to transmission in case of an engine, or ability to rotate turret in case of turret horizontal controls. A damaged module has its hit points reduced to 0 and is not operational. A destroyed module has been damaged beyond repairs and cannot be fully restored back to the operational state.

6.1.6 Kinetic and chemical energy penetration differences

As the DeMarre formula describes the penetration power of kinetic projectile, the chemical shells need a different approach to determine the result of their penetration process. Since

the Munroe process success is determined solely by the distance of target from the point of explosion, it would be possible to implement armor penetration directly as an input parameter of the chemical shell. Even though the Munroe effect is only applicable for shaped charges, it should be theoretically possible to reuse the mechanic for standard high-explosive rounds, which will sport a much shorter jet length, compared to projectiles with shaped charges.

6.2 Unity models

Before the whole penetration process is described, it is important to mention the models, i.e. classes, that hold a number of parameters, that are necessary to complete any sort of calculation. Each of the important objects, be it the projectile, armored plate, or tank module, have their own model. Parameters in these models are fully customizable, allowing the UTW developers to create their own variants and will be often mentioned in numerous functions.

6.2.1 Kinetic Shell

The kinetic shell class describes the functionality of an armor piercing kinetic shell. It includes number of parameters, like shell caliber in decimeters, minimum and maximum angle in degrees and shape factor, which is used as “n” parameter in DeMarre’s formula, which are necessary to handle the penetration calculation and the impact behavior itself, which is handled in FixedUpdate function.

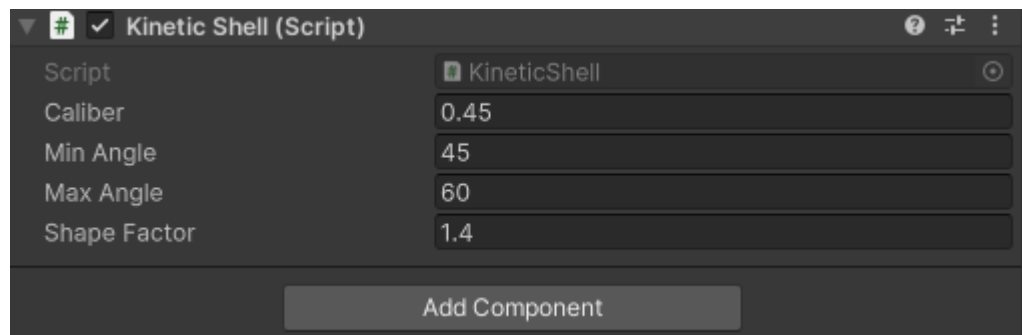


Fig. 6. – Kinetic Shell script

Notice that the muzzle velocity is not a shell parameter, since the same shell could be fired from a cannon with longer barrel, generating higher muzzle velocity. As such the muzzle velocity is not linked to the shell but to the cannon itself. The default type of a kinetic shell is a standard Armor piercing (AP) round, but by finessing the shell parameters, it is fairly easy to create a specialized rounds like the APC, which would sport higher ricochet angles

and different shape factor, but suffering from higher Rigidbody drag, or the APCR, which would sport a much higher muzzle velocity and its caliber diameter would be decreased, since the DeMarre formula (despite not being exactly recommended for subcaliber rounds) counts the tungsten core diameter as a caliber in this case.

6.2.2 Chemical Shell

A similar model, but using a different calculation and sporting different parameters, is the Chemical Shell. This class describes the penetration process of a projectile that doesn't penetrate the armor with its kinetic energy, but uses chemical compound to create a directed explosion, which forms a metallic jet, that penetrates the same amount of armor, independently of the projectile velocity. As such, the chemical projectile doesn't need the shape factor, which is instead replaced with the Jet Length Modifier, which describes the level of sophistication of the shell.

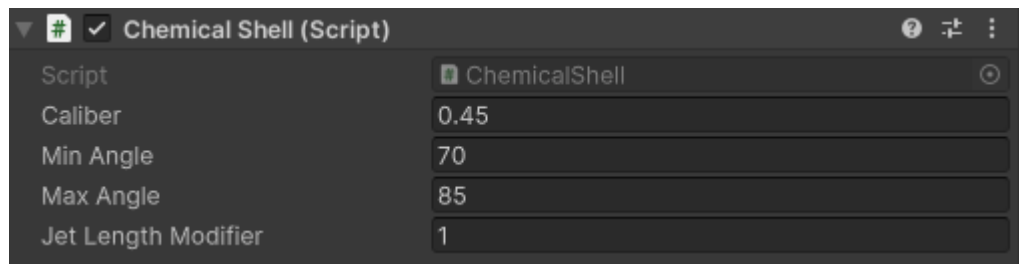


Fig. 7. – Chemical Shell script

Modern HEATFS shells can generate metallic jet up to seven times of their diameter, but as the crude Second world war and pre-Second world war HEAT projectiles are nowhere near optimized, they offer much shorter jet length.

6.2.3 Armor plate

The armor plate class describes the effectivity of a single armor plate, which is modeled out of a box object, with its Z axis aiming outwards from the vehicle, providing protection against incoming projectiles. Similar to projectiles, which require a Rigidbody component to be able to function, the armor plate requires a BoxCollider component, which is used to obtain armor thickness. As the resulting penetration is calculated in decimeters, the armor thickness, obtained by multiplying the Z scale of the BoxCollider and the Z scale of the armor plate gameobject itself, must be additionally multiplied by 10 to convert meters into decimeters.

```
private void Start()
{
    ArmorThickness = boxCollider.size.z * gameObject.transform.localScale.z
    * 10;
}
```

As the armor thickness is calculated automatically, there is no need to specify it in the armor plate parameters. What has to be specified though is the Armor Quality, which is used as a “C” constant by DeMarre’s formula, and the fragment chances, allowing the UTW developers to specify how fragile the armor is and how much big fragments should be generated upon penetration.

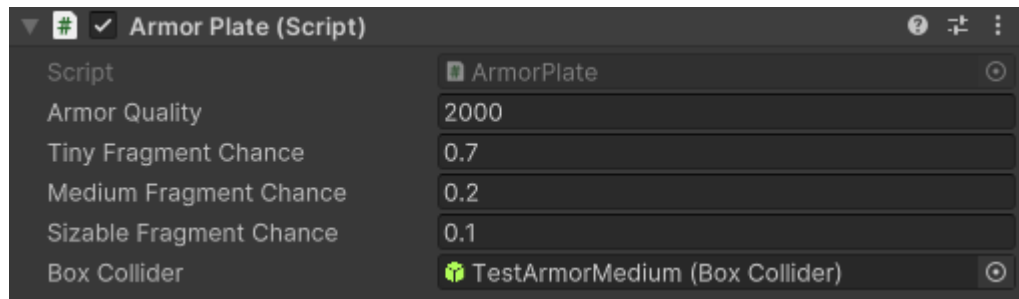


Fig. 8. – Armor plate script

The default parameters describe a rather ductile rolled homogenous armor (RHA) with low chance of generating sizable fragments. A face-hardened armor (FHA) would sport a higher armor quality of about 2500 and higher chances of medium and sizable fragment generation.

6.2.4 Tank module

When the armored vehicle is engaged, the primary goal is to destroy its internal modules, which could be an engine, ammo stowage, cannon breech, vertical turret drive and many more, or its crew and as such it is important to model such damage in ballistic simulation. As the process of module destruction is strictly stochastic, this part of the system is completely fabricated and instead of providing realistic simulation aims mainly at providing a predictable and balanceable system, that would fit the gamers needs.

As such it uses traditional hit point system, which subtracts hit points from a damaged module, until it reaches 0 upon which it is destroyed. To meet specifications from the UTW team a special state system was created, that describes the functionality of modules by assigning a state to each module, which changes as the module receives the damage.

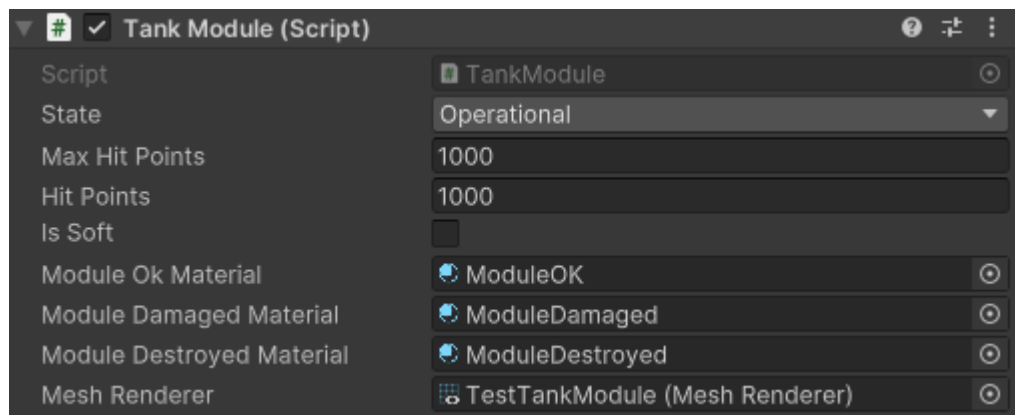


Fig. 9. – Tank Module script

The module State can switch between Operational, when the module works as intended and despite not having full hit point pool, can provide its functionality, Damaged, when the module has been hit so badly, that it can no longer provide its functionality, or Destroyed, when the module received such damage that it is beyond repairs and cannot be restored only by its crew. This system was devised to work in tandem with repair points, which will have to be reached by the vehicle to be able to repair Destroyed modules.

To better present this system in test build, the module also changes its material depending on its current state, which provides the tester more clarity while firing at the test target.

Another additional parameter is the IsSoft boolean variable, which is used to model tank crew, which also use the TankModule class. When the TankModule state changes to Destroyed and the module has IsSoft variable switched to True, its collider is deleted, which allows the spalling to pass through.

6.3 Drag and gravity implementation

As the drag and gravity is handled solely by the Rigidbody component, the implementation of such system is very straightforward. The projectile prefab is simply fitted with a Rigidbody component, and its parameters are filled with appropriate data. For the purpose of the testing build a BT-7 light tank was selected together with its 45mm 20-K cannon as the player-controlled vehicle. As such the Rigidbody parameters should somewhat like this:

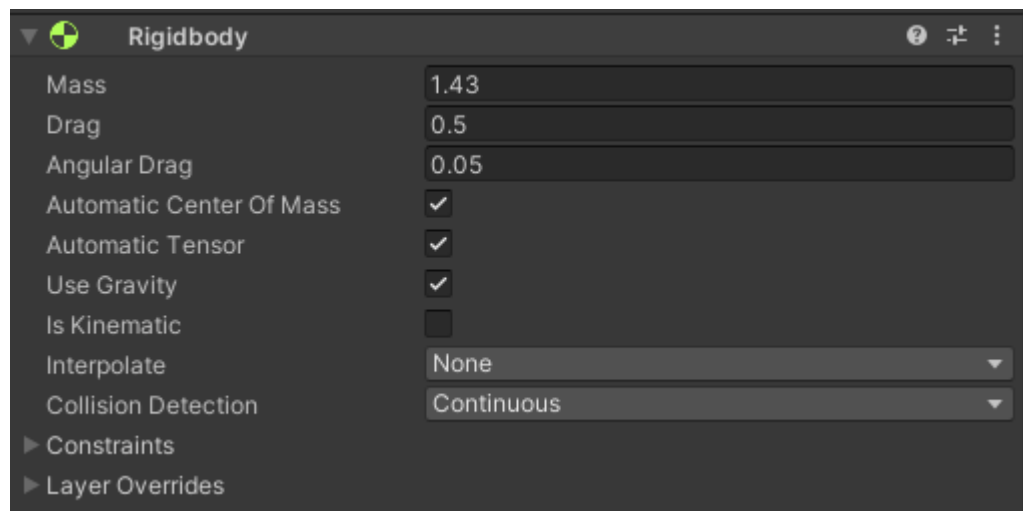


Fig. 10. – Rigidbody component

As the modelled BR-240 shell, which was used at the time as standard shell for 20-K cannon, weights around 1.43 kg. Once the Rigidbody is set up, and the projectile is fired, it should automatically reduce its velocity and drop closer to the ground every fixedUpdate.

6.4 Raycast projection

Whenever the projectile meets an object with a collider, the Rigidbody will automatically resolve the collision and ricochets the projectile away from the surface. If the colliding object sports a Rigidbody of its own, each Rigidbody projects its colliding force onto the other, which may cause both of the object to ricochet away from each other, when their forces are somewhat equal, or force weaker object away from the object with more force. As the force is dictated by both mass and velocity, the light projectile would probably be the object that will ricochet away from the surface.

As the ricochet of the projectile is only one of the many possible outcomes, when it comes to armor penetration process, some sort of collision handling script is necessary to allow the projectile to pass through the armor instead of ricocheting away. Sadly, the OnCollisionEnter event, which is called in the nearest FixedUpdate, after the collision happens, is called too late to be of use, as the projectile had already bounced away from the plate at the time. To remedy this issue an alternative solution was devised to allow projectile to pass through the armor.

Instead of letting the projectile handle its collisions using the Unity Physics engine, each FixedUpdate a special Raycast is projected forward into the path of the projectile, to such distance, that it will cover the whole travel distance of the projectile in the following

FixedUpdate. As such the collision is always handled one FixedUpdate earlier than it would be possible by awaiting the OnCollisionEnter event.

To properly measure the distance which the projectile travels in a FixedUpdate a simple formula was devised:

```
var distance = Time.fixedDeltaTime * Rigidbody.velocity.magnitude;
```

Thanks to this formula a perfect distance is counted for every FixedUpdate, depending on the current velocity of the moving projectile.

Since the fast-moving projectile could easily hit more than one target in one FixedUpdate, traveling distance of several meters between each FixedUpdate method calling, a RaycastAll is used instead to capture every possible target the projectile could possibly hit.

```
var hitTargets = Physics.RaycastAll(transform.position, transform.forward,  
    Time.fixedDeltaTime * Rigidbody.velocity.magnitude);
```

```
if (!hitTargets.Any()) return;
```

Each of the collisions is then handled one by one, in the order the Raycast passed through them. Thanks to this system, if the projectile is stopped by the armor, the function is returned, which stops any other collisions from happening. Each of the possible targets is then handled separately. When the projectile hits the ground, it is automatically stopped and the function returned. Current implementation stops the projectile in its tracks and disables the gravity, so that the projectile trail is not lost, but once the module is fully implemented into the game, the projectile is simply destroyed.

```
if (target.transform.TryGetComponent(out TerrainCollider _))  
{  
    LogManager.Instance.LogMessage("Projectile hits the ground!");  
    TurnOffRigidbody();  
    gameObject.transform.position = hit.point;  
    return;  
}
```

Current implementation stops the projectile in its tracks and disables the gravity, so that the projectile trail is not lost, but once the ballistic simulation module is fully implemented into the game, the projectile will be simply destroyed.

```
private void TurnOffRigidbody()  
{  
    Rigidbody.isKinematic = true;
```

```
Rigidbody.velocity = new Vector3(0, 0, 0);
Rigidbody.useGravity = false;
GetComponent<TrailRenderer>().material = stoppedMaterial;
}
```

Hitting the tank module will automatically disable the tank module collider, allowing the projectile to pass through, while inflicting some damage. As the proper balancing of the vehicle module system is not part of this thesis a temporary hardcoded value is used, but depending on the whims of UTW development team, the value can be either directly specified as a shell parameter or calculated dynamically from the projectile's weight, velocity and/or caliber.

```
if (hit.collider.transform.TryGetComponent(out TankModule tankModule))
{
    tankModule.TakeDamage(200);
    Ignore(hit.collider);
}
```

Ignore function is a function which handles the case when the projectile should simply pass through the collider.

```
private void Ignore(Collider collider)
{
    Physics.IgnoreCollision(GetComponent<Collider>(), collider);
    lastHitCollider = collider;
}
```

The `Physics.IgnoreCollision()` function allows to specify exceptions for Physics collision handling. Input parameters of `IgnoreCollision` function are two colliders, which will now simply pass through each other without Physics engine taking any sort of action.

6.5 Ricochet handling

Whenever the Raycast sent out from the travelling shell detects an armor plate in its path, a penetration process begins, starting with the ricochet calculation. A ricochet is a semi-stochastic phenomenon, that occurs randomly when a fast-moving object strikes an angled plate. The bigger the angle is, the bigger the chance of a ricochet.

As this phenomenon is almost impossible to scientifically predetermine, a game mechanic was devised, that will determine if the ricochet should happen or if the projectile cuts into the armor and starts penetrating the armor plate.

First step is to load several necessary data, that describes the projectile velocity, angle between the incoming projectile and the armor normal, the projectile mass, C constant, armor thickness and especially the overmatch parameter, which is a necessary part of the ricochet calculation.

```
// Load data
var velocity = Rigidbody.velocity.magnitude;
var impactAngle = Vector3.Angle(transform.forward, hit.collider.transform.forward);
if (impactAngle > 90) impactAngle = 180 - impactAngle;
var impactMass = Rigidbody.mass;
var armorQuality = armorPlate.GetArmorQuality();
var armorThickness = armorPlate.GetArmorThickness();
var overmatch = Caliber / armorThickness;
```

Once these parameters are extracted, a ricochet chance is calculated via the CalculateRicochetChance() function, which needs several parameters to be able to calculate the ricochet result.

```
// Handle ricochet and overmatch
if (CalculateRicochetChance(impactAngle, MinAngle, MaxAngle, overmatch))
return;
LogManager.Instance.LogMessage("Projectile attempts to penetrate the armor!");
```

If the impact angle is smaller than the MinAngle parameter of the shell, the projectile cannot ricochet and starts penetrating the armor. If the impact angle is bigger than the MaxAngle, the ricochet is guaranteed. When the impact angle lands somewhere between MinAngle and MaxAngle, a random number is generated, which will determine the result of ricochet.

```
private static bool CalculateRicochetChance(float impactAngle,
float minAngle, float maxAngle, float overmatch)
{
var overmatchModifier =
Mathf.Clamp01(Mathf.InverseLerp(1.3f, 7f, overmatch));
var minRicochetAngle =
minAngle + (overmatchModifier * (90f - minAngle));
var maxRicochetAngle =
maxAngle + (overmatchModifier * (90f - maxAngle));

var clampedImpactAngle =
Mathf.Clamp(impactAngle, minRicochetAngle, maxRicochetAngle);
var normalizedAngle =
```

```

        (clampedImpactAngle - minRicochetAngle)
        / (maxRicochetAngle - minRicochetAngle);

        var random = new Random();
        var randomValue = random.NextDouble();
        LogManager.Instance.LogMessage($"Ricochet calculation: Impact an-
        gle={impactAngle},Min angle={minRicochetAngle}," +
        $"Max angle={maxRicochetAngle}, Chance ={normalizedAngle}, Gener-
        ated={randomValue}");
        return randomValue <= normalizedAngle;
    }

```

Important factor when it comes to ricochet handling is the Overmatch parameter. When the shell diameter is larger than the thickness of the armor plate, it becomes easier for the shell to dig into the armor plate, as it shifts bit under the impact, giving more space for the projectile to burrow into the plate. The above mentioned WarThunder uses system similar to this, which benefits the shell with Overmatch factor being larger than 1,3. Thanks to the Overmatch, the Min and Max angles are automatically increased by the Overmatch factor, resulting in less ricochets for massive projectiles.

6.6 Kinetic penetration process

The kinetic penetration process uses the DeMarre's formula to resolve the penetration success, which is dependent on many different parameters of both shell and armor plate. As such the effectivity of kinetic shell drops with distance, since longer the shell flies the lower the velocity drops.

6.6.1 DeMarre formula

The whole penetration success depends mainly on DeMarre's formula. Once the projectile strikes the plate and the projectile fails to ricochet, a result based on DeMarre's formula is calculated.

```

// DeMarre penetration approximation
var dmPenetrated = Math.Abs(
    Math.Pow(velocity * Math.Pow(impactMass, 0.5)
        * Math.Pow(Math.Cos(DegToRad(impactAngle)), ShapeFactor)
        / (armorQuality * Math.Pow(Caliber, 0.75)), 1D));

LogManager.Instance.LogMessage($"A shell with a velocity of {velocity}m/s,
with angle of {impactAngle}°,"

```

```
+ $"mass of {impactMass}kg, caliber of {Caliber}dm and penetration power  
{dmPenetrated}dm, just hit an armor plate!";
```

The formula results in a floating-point number, which describes the distance in decimeters that could be effectively penetrated by shell travelling at specified velocity and angle, with specified parameters like shell caliber, mass and shape factor, and plate quality. This distance is then compared to the actual armor thickness and can result in either complete or partial penetration, or in failure to penetrate, which effectively destroys the projectile.

6.6.1.1 *Partial penetration*

As the DeMarre's formula describes the thickest possible armor thickness, that can be fully penetrated by the projectile, there is a need to implement a special case of partial penetration, when the shell manages to overcome the armor, but is too weak to continue forward and only generated fraction of spalling.

```
if (dmPenetrated < armorThickness && dmPenetrated > armorThickness / 1.1)  
{  
    LogManager.Instance.LogMessage("Partial penetration!");  
  
    //Offset the hit.point a bit to ignore armor plate collider  
    armorPlate.Shatter(hit.point + Rigidbody.velocity.normalized * 0.01f,  
Rigidbody.velocity.normalized, (Caliber+armorThickness)*10);  
  
    TurnOffRigidbody();  
    gameObject.transform.position = hit.point;  
    return;  
}
```

The partial penetration happens when the resulting amount of penetration distance is lower than the actual armor thickness, but not by a big margin. The Shatter function is used to generate spalling from the armor plate and will be described in its own chapter.

6.6.1.2 *Full penetration*

If the shell penetration potential is higher than the thickness of penetrated plate, the shell successfully managed to overcome the armor in all its glory and could keep on moving deeper into the vehicle.

```
else if (dmPenetrated > armorThickness)  
{
```

```
LogManager.Instance.LogMessage("Full penetration!");

Ignore(hit.collider);

//Offset the hit.point a bit to ignore armor plate collider
armorPlate.Shatter(hit.point + Rigidbody.velocity.normalized * 0.01f,
Rigidbody.velocity.normalized, (Caliber+armorThickness)*20);

gameObject.transform.position = hit.point;
Slowdown((float)(armorThickness / dmPenetrated));
}
```

Once the projectile overcomes the armor plate, the Ignore function is called so that the body of the shell can go through a defeated armor plate. Once the physics between two colliders is disabled, the Shatter function is called, which generates the spalling. Lastly the projectile velocity is reduced depending on how hard it was for the projectile to penetrate the armor plate.

```
private void Slowdown(float amount)
{
    Rigidbody.velocity -= Rigidbody.velocity * amount;
    LogManager.Instance.LogMessage($"New velocity = {Rigidbody.velocity.magnitude}");
}
```

At this point a problem arose, when the projectile was slowed so severely, that it couldn't pass through the armor in a single FixedUpdate frame, which resulted in repeating the armor penetrating process next FixedUpdate, as the IgnoreCollision function ran out of its time and the physics engine restored the collision handling between the two colliders. To solve this problem a simple variable was introduced, which remembers the last collision and won't allow repeated penetration of the same collider.

```
if (lastHitCollider == hit.collider) continue;
```

6.6.1.3 Failure to penetrate

If the armor proves to be too thick to be penetrated by the shell, the projectile is stopped in its tracks and the process ends.

```
else
{
```

```
LogManager.Instance.LogMessage("Failed to penetrate!");

TurnOffRigidbody();
gameObject.transform.position = hit.point;
return;
}
```

6.7 Chemical penetration process

The main difference between the kinetic and chemical penetration is that the chemical shells can always engage only a single collider. Once the shell comes into contact with any hard surface, it detonates, firing out its superheated jet even if other object was hit prematurely. This makes the shell much less effective when engaging targets in cover, or targets with multiple layers of protection, like for example applique armor plates or meshes.

The ricochet system is the same as for the kinetic penetration process, but since the chemical shells are quite volatile, they tend to be able to go off even at sharper angles.

6.7.1 Backcast projection

Since the DeMarre's formula is only applicable to kinetic penetration projectiles, a different solution was devised to handle chemical penetration. This system tries to simulate Munroe effect by Unity components, while keeping the whole system as much lightweight and modular as possible.

The impact is first offset forward at the direction which the shell is currently traveling, for distance which equals the shell caliber multiplied by JetLengthModifier. This point serves as the point of maximum reach of the superheated jet. From this point a Raycast is sent back towards the original projectile contact point, which tries to find the exit point of the deepest armor plate. If the jet end point is located inside of a collider, the Raycast starting from this point will automatically ignore the collider and attempts to find a new one. This results in a durable system, which always either finds the exit point or nothing, which means that the armor plate was too thick and jet too short.

```
var munroeLength = 0.1f * JetLengthModifier * Caliber;
var backcastOrigin = hit.point + Rigidbody.velocity.normalized * munroeLength;
if (Physics.Raycast(backcastOrigin, -Rigidbody.velocity.normalized,
out var backRaycastHit, munroeLength))
{
```



```
Debug.DrawLine(backcastOrigin, backRaycastHit.point, Color.red,1200);
if(backRaycastHit.transform.TryGetComponent<ArmorPlate>(out var originalPlate))
{
    originalPlate.Shatter(backRaycastHit.point, Rigidbody.velocity.normalized, 10 * Rigidbody.mass);
    LogManager.Instance.LogMessage("Shaped charge punched through the armor!");
}
} else LogManager.Instance.LogMessage("Shaped charge failed to penetrate!");

TurnOffRigidbody();
gameObject.transform.position = hit.point;
return;
```

If the point is found, a spalling is generated and the projectile stops.

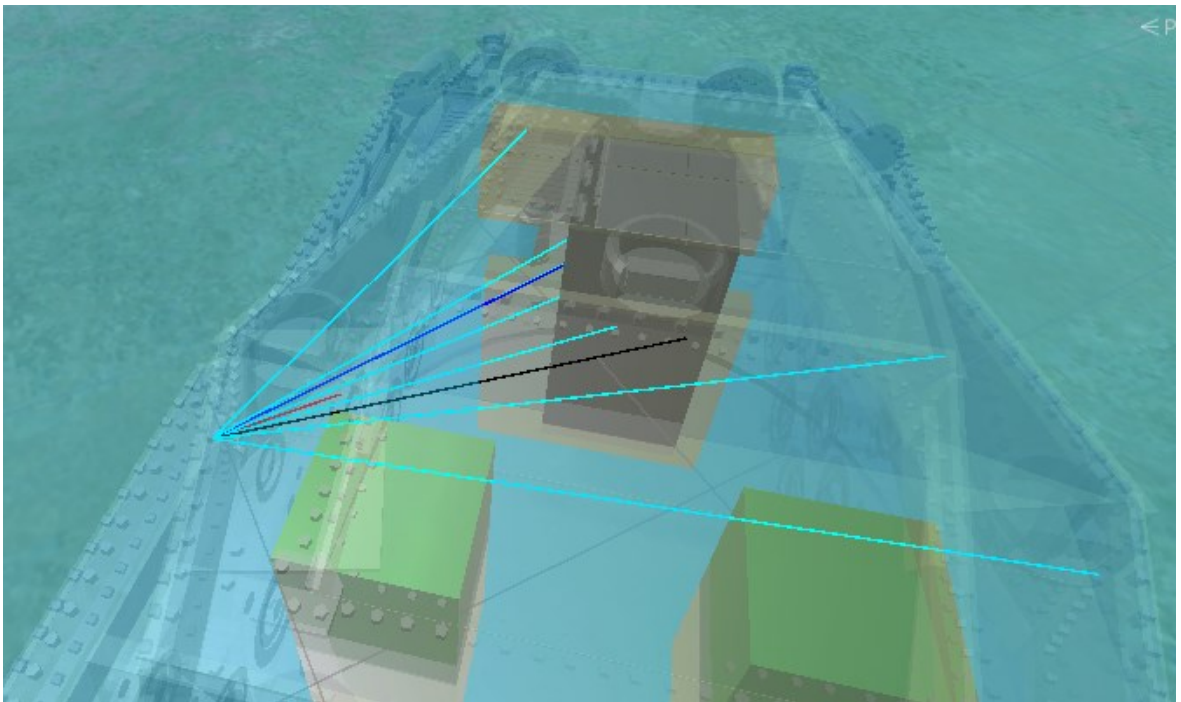


Fig. 11. – Chemical penetration (red) with spalling (cyan, blue and black)

6.8 Spalling generation

Spalling generation is a stochastic process, which is hard to scientifically describe and as such uses a simplified system, which randomly generates one of the three spalling types, one at a time, until all of the spalling points are depleted.

```
public void Shatter(Vector3 origin, Vector3 direction, float points)
{
    var pointsLeft = points;
    while (pointsLeft > 0)
    {
        var randomValue = UnityEngine.Random.value;
        if (randomValue < SizableFragmentChance && pointsLeft >= SizableCost)
        {
            GenerateSpalling(origin, AddRandomOffset(direction, 1300), 300,
            Color.black );
            pointsLeft -= SizableCost;
        }
        if (randomValue < MediumFragmentChance && pointsLeft >= MediumCost)
        {
            GenerateSpalling(origin, AddRandomOffset(direction, 1700), 120,
            Color.blue );
            pointsLeft -= MediumCost;
        }
        else
        {
            GenerateSpalling(origin, AddRandomOffset(direction, 2100), 50,
            Color.cyan );
            pointsLeft -= TinyCost;
        }
    }
}
```

The tiny spalling represents birdshot-sized metallic pellets, which could be potentially harmful when they hit a critical part of a tank module but are usually non-lethal by themselves. Medium spalling describes a bullet-sized metallic shards, that can seriously wound a crew-member or damage a tank module. Sizable spalling represents a solid chunk of metal, which could easily destroy an unprotected module and has similar damaging effectivity as the projectile itself. Each round of the spalling generation process one of the spalling types is selected and generated until the points are completely depleted.

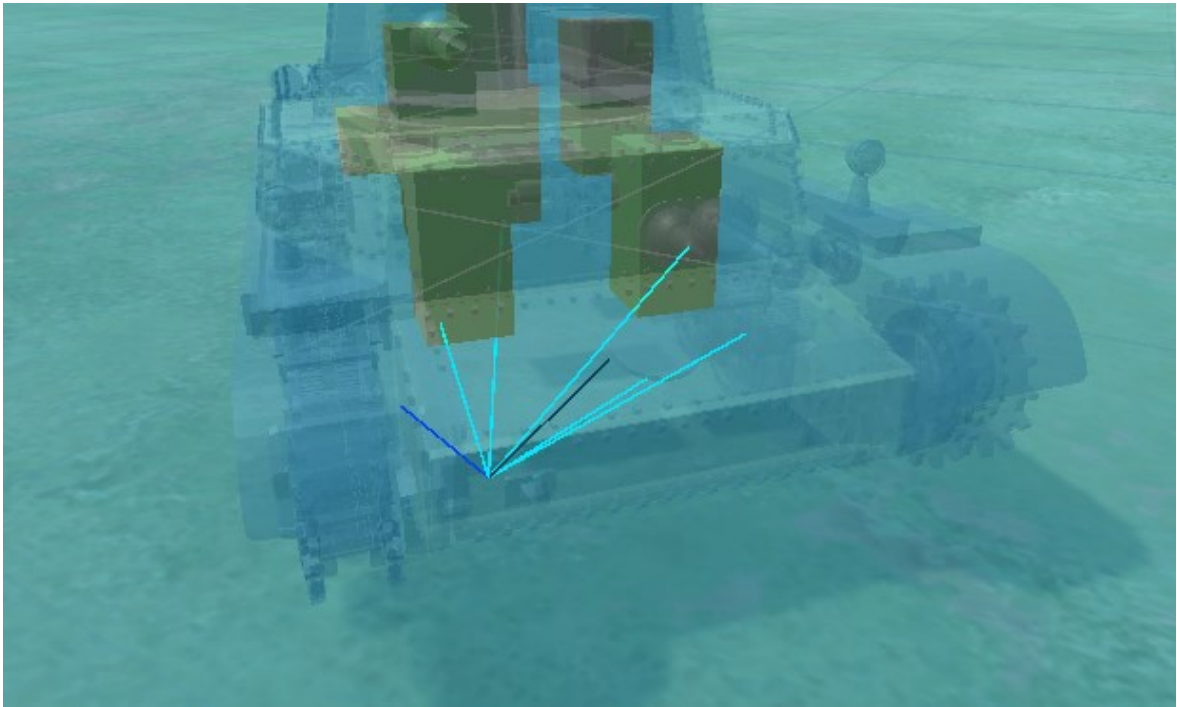


Fig. 12. – Spalling simulation

To increase the area that the spalling can cover, a random offset is added before the GenerateSpalling function is generated. First the offset is converted from angles to radians, then a random range is generated and combined to form an offset vector, which is then used to multiply the direction, forming a new offset direction, which is later used to direct the newly generated spalling.

```
public Vector3 AddRandomOffset(Vector3 direction, float maxOffsetAngle) {  
    var maxOffsetRadians = Mathf.Deg2Rad * maxOffsetAngle;  
  
    var randomXOffset = UnityEngine.Random.Range(-maxOffsetRadians, maxOffsetRadians);  
    var randomYOffset = UnityEngine.Random.Range(-maxOffsetRadians, maxOffsetRadians);  
    var offset = new Vector3(randomXOffset, randomYOffset, 0f);  
  
    var rotatedDirection = Quaternion.Euler(offset) * direction;  
    return rotatedDirection.normalized;  
}
```

The GenerateSpalling function handles generation of a single shard of spalling. It takes origin, direction and damage as a parameter, together with color, which is used to draw spalling lines in debug mode.

```

public void GenerateSpalling(Vector3 origin, Vector3 direction, int damage,
Color color)
{
    var hit = Physics.Raycast(origin, direction, out var target, 10);
    if (!hit) return;
    if (target.transform.TryGetComponent<TankModule>(out var module)) mod-
ule.TakeDamage(damage);
    Debug.DrawLine(origin, target.point, color, 1200);
    LogManager.Instance.LogMessage($"Spall has struck the {target.trans-
form.name}");
}

```

6.9 Damage handling

Whenever the tank module object is struck, either by spalling or by the projectile itself, a number of hitpoints is subtracted from the module hit point pool. Whenever the hit point pool drops below zero, the module state changes from Operational to Damaged. Whenever the Damaged object receives another damage, it changes the state to Destroyed. A special was added, when the Operational object receives the damage which is bigger than twice the maximum hit point pool. In this case the object is exposed to considerable force and is instantly destroyed, skipping the Damaged state completely.

```

public void TakeDamage(int damage)
{
    if (state.Equals(TankModuleState.Destroyed)) return;
    if (damage - hitPoints > 2 * maxHitPoints || (hitPoints - damage < 0 &&
state.Equals(TankModuleState.Damaged)))
    {
        hitPoints = 0;
        state = TankModuleState.Destroyed;
        meshRenderer.material = moduleDestroyedMaterial;
        if (isSoft)
        {
            if (TryGetComponent<Collider>(out var collider))
            {
                collider.enabled = false;
            }
        }
        LogManager.Instance.LogMessage($"{{this.gameObject.name}} has been
struck and took {damage} points of damage and is now destroyed");
    } else if (hitPoints - damage < 0)
    {
        hitPoints = 0;
        state = TankModuleState.Damaged;
    }
}

```

```
        meshRenderer.material = moduleDamagedMaterial;
        LogManager.Instance.LogMessage($"{this.gameObject.name} has been
struck and took {damage} points of damage and is now damaged");
    }
    else
    {
        hitPoints -= damage;
        LogManager.Instance.LogMessage($"{this.gameObject.name} has been
struck and took {damage} points of damage");
    }
}
```

7 UTW BALLISTIC MODULE TEST BUILD

Part of the thesis is the UTW Ballistic module test build, which allows the user to test the nuances of the proposed UTW Ballistic module. It represents a shooting range with the player-controlled vehicle, several test targets, including a completely armored target vehicle with few of the internal modules, and an action log.

7.1 Test build controls

The ballistic module test is controlled by the standard combination of mouse and keyboard, where the mouse is used to control player camera and engage targets, while the keyboard is traditionally reserved for movement controls and additional functionalities, like for example switching shells. These controls are the default controls provided by the PTM package, which the UTW is built upon.

The camera is controlled by mouse movement, allowing the user to zoom in and out with the mouse wheel, or switch into the gunner's view with right mouse button, which will show the gun reticle overlay, allowing the user to easier engage targets at distance. It is also possible into a commander's or driver's view by pressing the F key.

Player-controlled vehicle is able to traverse the testing ground by pressing the combination of W, S, A and D keys. W increases the throttle, and the vehicle is propelled forward, while the S propels the vehicle backwards. The A and D buttons are used to steer the vehicle left and right. As the transmission is automatic in test build, the user doesn't have to worry about it and use only the 4 basic keys to control the vehicle.

The vehicle controlled by the player is equipped with two sets of shells. The default is the kinetic projectile, but by pressing the V button the user can switch to chemical shells, which offer different approach to penetration.

To better inform the user about the background processes, that are being calculated by the test build, the user can access a special log, by pressing the P key. Each projectile's processes are written in this log in real-time as the projectile progresses. Each time a new projectile is launched, the log is erased to improve readability.

As the user can potentially destroy every single tank module in testing area, fall out of the map or somehow get stuck in certain colliders, a restart, which is engaged by pressing the TAB key, was added as well. To exit the build a simple press of ESC key is all that is needed.

7.2 Player-controlled vehicle

The player is represented with a BT-7 convertible tank, equipped with 20-K cannon, that is able to fire either kinetic AP shell, which is modelled after the BR-240 APBC round, completely with proper parameters, or a fictional HEAT shell, as the BT-7 was not historically equipped with HEAT shells, since they were usually issued to large caliber cannons.

The vehicle parameters are rather exaggerated and its speed and reload speed are much faster than what could the real-life vehicle offer. This was done mainly to reduce the time it takes the user to get into proper position and to quickly fire again if they miss.

7.3 Test target

To offer some sort of approximation of how the combat could look like with the UTW ballistic module implemented a test target in the form of 41M Turán medium tank was modelled, completely with accurate armor thickness, shapes, and angles, including few of the internal modules, so that the user can observe the damage done by the firing.

The Turán doesn't offer much protection, with its UFP (Upper front plate) and frontal turret plate reaching up to 50mm, with little to no angling. The rest of the vehicle is armored with 35mm of armor, especially the sides of hull and turret. The roof and bottom are armored only with 13mm of armor. As the BR-240 can potentially penetrate up to 66mm from 100m or even more from point blank distance, it is rather easy to penetrate the Turán without much issue. When the distance between the player-controlled vehicle and the test target increases to about 500m, the penetration starts to be rather difficult especially if the projectile strikes the armored plate at an angle of 40° or higher.

7.4 Shooting range

To easily showcase the effectivity of BR-240 (and any subsequent shells, used for comparison) a rudimentary shooting range, which allows the testing of flat (90°/0°) or angled (60°/30°) impacts at distances of 100, 300 and 500 meters. Proper firing position is marked by the white line, which allows the user to test the effectivity of the shell at somewhat precise distance.

Each of the target plates is colored red for better visibility and is 70mm thick, meaning that the BR-240 can achieve only partial penetration. The achieved armor penetration effectivity

can be displayed in the log, allowing the user to observe how the changes of distance decrease the effectivity of the projectile.

8 BALLISTIC SIMULATION TESTING

To better visualize the accuracy of the ballistic simulation, respectively the kinetic penetration process, calculated by the version of DeMarre formula, it was decided to compare the results of UTW ballistic to a real-life data and data from another existing tank simulator.

8.1 Measured data

To represent a real-life measured data, a following document was selected [31]:

Дистанция в метрах	45 мм танков пушка		57 мм пушка ЗИС-4		76 мм пушка ка ф 34		85 мм пушка		85 мм пушка		100 мм пушка		122 мм пушка		152 мм пушка		152 мм пушка		57 мм пушка		75 мм пушка		50 мм пушка		75 мм пушка		75 мм пушка		88 мм пушка		88 мм пушка		
	60°	90°	60°	90°	60°	90°	60°	90°	60°	90°	60°	90°	60°	90°	60°	90°	60°	90°	60°	90°	60°	90°	60°	90°	60°	90°	60°	90°	60°	90°			
100	43/68	57/96	93/130	115/175	61/105	75/128	97	119	116	143	140	170	135	165	114	130	230	194	244	69	83	52	64	60/89	74/110	83	103	114	150	98	120	137	164
300	38/54	53/72	89/115	110/160	59/89	72/110	94	114	110	136	136	167	131	161	110	136	222	187	232	61	78	48	59	55/76	67/94	77	95	111	136	95	116	134	164
500	35/46	48/51	86/100	108/140	56/75	69/92	91	111	105	130	132	162	128	157	108	130	217	182	224	52	74	45	55	49/66	61/82	72	89	107	132	91	112	130	164
1000	28/30	40	78	96	49	61	83	102	98	118	121	149	120	147	101	120	204	170	205	39	65	39	46	38/43	47/53	60	74	93	114	84	103	121	144
1500	24	-	70	86	44	54	76	93	82	102	109	137	112	138	95	117	197	165	202	29	36	32	39	29	36	50	62	79	96	78	96	108	138
2000	20	-	68	77	39	48	69	85	72	89	102	125	105	129	89	103	184	160	197	-	-	-	-	-	-	-	-	65	82	72	88	100	124

Примечание: 1. В числителе указано бронепробиваемость бронебойным снарядом, в знаменателе подкалиберным снарядом.
2. Толщина брони 6 мм.

Fig. 13. – Armor penetration data table

It was gathered by the engineers of the Soviet Union in 1943, and it describes penetration values of a selection of tank cannons, which were at the time present for evaluation by the soviet engineers. Since the document was created in the times of Stalin’s dictatorship, it is quite possible, that the numbers were tampered with in the favor of soviet weapons, but as the table was not publicly accessible at the time, chances are that the document describes at least approximately valid penetration values.

Little is known about the target, which was used to catch the projectiles, nor its material composition, while the same can be said about the quality of the shells, especially when it comes to the looted German weaponry. Nevertheless, as the UTW doesn’t require a 100%

precise penetration values for its vehicles, it is safe to say, that the document is more than adequate to serve as a UTW ballistics comparison table.

8.2 WarThunder ballistic values

Since the WarThunder offers a complex wiki, which describes the vehicles and their weaponry in great detail, it was selected as an existing system comparison. Each of tank cannons have their own dedicated page, which describes types of ammunition, which was historically used for a cannon, with necessary information, like the caliber, projectile mass, muzzle velocity, and approximate penetration values at different distances.

The only downside of WarThunder comparison is its lack of penetration values for angled targets, as the WarThunder uses a different system than the UTW and instead of including the angle variable in the DeMarre formula, the penetration values are calculated only for the flat surface and the armor thickness is increased depending on the impact angle.

8.3 UTW ballistic values

As the ballistic simulation for UTW is modular in such a way that almost any shell can be modelled simply by changing projectile mass, caliber, initial velocity, shape factor and drag, the process of creating a testing models for every shell was very simple. Following data was collected and applied to unity shell model.

Shell	Caliber	Mass	Initial velocity	N	Drag
45mm 20-K (BR-240)	0,45	1,43	760	2	0,4
45mm 20-K (BR-240P)	0,28	0,85	960	1,7	0,8
57mm Zis-4 (BR-271)	0,57	3,14	990	1,6	0,35
57mm QF-6pdr Mk.III (Mk.8)	0,57	2,87	837	1,6	0,35
50mm KwK38 L/42 (Pzgr 39)	0,5	2,05	835	1,65	0,45
88mm KwK36 L/56 (Pzgr 39)	0,88	9,5	810	1,65	0,45

Table 2. – UTW Ballistic module parameters for various test shells

8.3.1 45mm 20-K (BR-240)

A default shell for the 20-K cannon, which was historically mounted on the BT-7 a vehicle, which is accessible to user in UTW ballistics testing build, the BR-240 is relatively common APHEBC (Armor-Piercing, High-Explosive, Ballistic-Cap) shell, which was used extensively in the early stages of Second World War by the tank forces of Soviet Union [17]. As

the main time period, that the UTW plans to simulate, is the Interbellum era, the BR-240 is a perfect candidate for a testing round, as it was developed in year 1932.

The shell itself is quite powerful for its age and offers decent penetration in both WarThunder and real-life. Its major disadvantage is severe reduction of penetration while penetrating angled plates. To better simulate this issue an N factor was raised to whole 2.0.

8.3.2 45mm 20-K (BR-240P)

The BR-240P is an APCR (Armor Piercing Composite Rigid) shell, which was developed as an upgrade for aging 20-K cannon. Some sources claim that thanks to this new shell, the 20-K cannon was able to penetrate a Pzkwf.VI Tiger at the distance of 200 meters when aimed at the lower hull from the side [32].

Since this shell uses a subcaliber tungsten core, the DeMarre formula dictates to include only the mass and caliber of the subcaliber core, which was provided by the WarThunder wiki. As the DeMarre formula was devised to describe the armor penetration process of solid steel projectiles, it is not optimal to use it for subcaliber projectile, however the UTW simulation managed to provide rather accurate despite this issue.

8.3.3 57mm Zis-4 (BR-271)

Unlike the 20-K which is a universal lightweight tank cannon, the Zis-2 was developed as a dedicated anti-tank cannon, which could penetrate thick armors, greatly sacrificing its ability to effectively engage softer targets [17]. The Zis-4 was a tank variant of the Zis-2, which was experimentally mounted on several platform, like for example the T-34, forming the T-34-57. As the Zis-4 fires shells with very high muzzle velocity and relatively small caliber, the penetration values are very high.

8.3.4 57mm QF-6pdr Mk.III (Mk.8)

The Quick-firing 6 pounder is a notorious British anti-tank cannon, which was sent to Soviet Union through the Lend-Lease program, mounted on Valentine Mk.IX tanks. Similar to Zis-4, the QF-6pdr was designed mainly as an anti-tank cannon and as such was able to penetrate sizable amounts of armor, for its size [17].

The table differs to other sources, stating different muzzle velocity than all of the possible shells, which were available to the cannon at the time, which could be caused by the inconsistency in unit conversion, as the British use feet per second, while the Soviets used meters

per second. This causes an inaccuracy in testing, as the data diverges, so as an approximation the Mk. 8 shell was used, which offered higher velocity (846 m/s) than the shell in Soviet ballistic table (837 m/s).

8.3.5 50mm Kwk38 L/42 (Pzgr 39)

The Kwk38 cannon was mounted primarily on German Pzkwf III Ausf F to Ausf J and saw extensive use against the Red Army throughout the Operation Barbarossa [33]. It was more powerful than the soviet 20-K cannon, with higher penetration and heavier shell, capable of delivering more explosives, but still inferior to cannons like British 6pdr or Soviet Zis-4.

Similar to the British 6pdr, the Kwk38 Pzgr 39 is listed with much higher muzzle velocity (835 m/s) than the other sources mention (685 m/s). The higher number was used in UTW simulation which resulted in almost perfect results.

8.3.6 88mm Kwk36 L/56 (Pzgr 39)

While the UTW is aimed mainly at simulating the interwar vehicles, the Kwk36 cannon was selected as a last test subject both for its renown, as a cannon used by the notorious Pzkwf VI Tiger, and to offer a test subject with much more penetrating power than the pre-war vehicles had at its disposal [33].

As expected, the resulting penetration power of this mighty weapon in UTW simulation is much lower than the same weapon modelled in WarThunder and the one described in real-life test table. As such the formula would need more tweaking to be able to simulate powerful late-war weaponry, but since that is not the requirement for the UTW, it is safe to proclaim the testing as successful.

8.4 Result table

To better illustrate the differences between the Soviet testing values, WarThunder simulation and UTW simulation a following table was assembled:

Projectile testing (Soviet post-WW2 testing values WarThunder ballistics system UTW ballistics system)																		
Shell	100m 90°			100m 60°			300m 90°			300m 60°			500m 90°			500m 60°		
45mm 20-K (BR-240)	57	67	66	43	X	50	53	62	58	38	X	44	48	58	51	35	X	38
45mm 20-K (BR-240P)	96	87	88	68	X	70	72	76	71	54	X	56	51	64	55	46	X	43
57mm Zis-4 (BR-271)	115	142	108	93	X	86	110	135	100	89	X	80	106	128	92	86	X	73
57mm QF-6pdr Mk.III (Mk.8)	83	106	91	69	X	73	87	98	83	61	X	66	74	89	75	52	X	60
50mm KwK38 L/42 (Pzgr 39)	74	80	79	60	X	63	67	69	71	55	X	56	61	62	62	49	X	49
88mm KwK36 L/56 (Pzgr 39)	120	162	113	98	X	90	116	157	110	95	X	88	112	151	107	91	X	85

Table 3. - Results of projectile testing of real-life data, WarThunder ballistic simulation and UTW ballistic simulation

As was already mentioned, the accuracy of tests concluded by the Red Army is dubious at best, nevertheless it offers a broad comparison of WW2 shells of various effectivity, which were apparently tested against the same target plates in similar conditions. It is also evident that the WarThunder simulation offers higher penetration numbers than the real-life data, which may suggest that either the WarThunder uses more powerful shells to even out the balance between vehicles, or that the real-life data were conducted against stronger armor plates, with shells of dubious quality or with different acceptance criteria for a shell that is able to “penetrate”.

The UTW simulation leans closer to real-life values than the WarThunder values, fluctuating somewhere between the WarThunder and real-life values, depending on the type of shell. As was already mentioned the DeMarre formula is not optimal for simulating sub-caliber shells, which tend to be less effective, which is evident on the results of BR-240P shell.

As the results are quite accurate, creating a visible difference between the penetration power of individual shells, it is safe to say that the testing of the UTW ballistics module was successful.

CONCLUSION

This thesis aimed at understanding the complex process of armor penetration and developing a solid armor penetration system using Unity Physics engine, that could be used by the Project UTW development team. The resulting system offers somewhat accurate data, that are in accordance with the real-life testing data and data collected from other tank simulation game, which are generated in real-time, allowing the projectile to easily calculate numerous penetrations without hindering the performance, fulfilling the determined requirements of the UTW. The system is also completely modular, allowing developers to easily implement new shells of any possible variety, both kinetic and chemical.

There are number of shortcomings of the system, namely its severe underestimation of penetration values of heavy WW2-era cannons, which is in accordance with the UTW requirements, as the UTW aims mainly at simulating the combat of older and lighter vehicles and their weaponry, which are simulated rather accurately by the proposed UTW ballistic system. Another potential problem could be caused by the Raycast driven penetration handling. As every contact of projectile and armor plate or internal module is determined one FixedUpdate ahead of the actual path of projectile, there may be some issues with sudden drastic changes of velocity, where the projectile travels at a different path than the Raycast projected in front of it. This issue can decrease the predictability of certain, mainly fast moving APCR, projectiles, which may behave incorrectly. Nevertheless, the Project UTW developers have been acknowledged of this problem and are content with using the system despite this issue.

The resulting system achieves a balance between accuracy and real-time performance, providing game developers with a reliable tool to create immersive tank battles. While calibrated for lighter vehicles to align with UTW's focus, the system lays the groundwork for future refinements to encompass heavier weaponry. Regardless, the UTW development team has embraced the system, and it is currently being integrated into the UTW framework. By providing a foundation for accurate armor and projectile representation, UTW paves the way for more immersive and strategic tank battles in the ever-evolving world of video games.

BIBLIOGRAPHY

- [1] WEIR, William. 50 Weapons that changed warfare. V1. New Page Books, 2005. ISBN 1564147568.
- [2] NOVOTNÝ, František. Churchillovy "nádrže". Online. 2003. Dostupné z: <https://www.valka.cz/742-Churchillovy-nadrze>. [cit. 2024-03-01].
- [3] Příběh jména Caterpillar: Jen se podívej, jak se při pohybu vlní. Online. 2020. Dostupné z: <https://www.stavebni-technika.cz/clanky/pribeh-jmena-caterpillar-jen-se-podivej-jak-se-pri-pohybu-vlni>. [cit. 2024-03-01].
- [4] NIEUWINT, Joris. From The Tank Museum – The World’s First Modern Tank, the Re-nault FT-17. Online. 2018. Dostupné z: <https://www.warhistoryonline.com/whotube-2/the-renault-ft-17-modern-tank.html>. [cit. 2024-03-01].
- [5] SY SIMULATIONS. TANK GEWEHR vs FT-17 | 13.2mm TuF AP | WW1 Armour Piercing Simulation. Online. 2023. Dostupné z: <https://www.youtube.com/watch?v=bLcne2W1aPs>. [cit. 2024-04-04].
- [6] OKUN, Nathan. MAJOR HISTORICAL NAVAL ARMOR PENETRATION FORMULAE©. Online. 1998. Dostupné z: <http://www.combinedfleet.com/formula.htm>. [cit. 2024-04-04].
- [7] HONNER, David M. Armour Hardness and Quality. Online. 2020. Dostupné z: https://web.archive.org/web/20111123195123/http://www.freeweb.hu/gva/weapons/introduction.html#Armour_Hardness_and_Quality. [cit. 2024-04-04].
- [8] The principle of armor piercing. Online. 2019. Dostupné z: <https://forum.il2sturmovik.com/topic/52058-the-principle-of-armor-piercing/>. [cit. 2024-04-04].
- [9] OKUN, Nathan. History and Technology: Projectile AP Caps. Online. 2000. Dostupné z: http://www.navweaps.com/index_tech/tech-055.php. [cit. 2024-04-04].
- [10] ONTARIO REGIMENT RCAC MUSEUM. Modern anti tank ammunition. Online. 2018. Dostupné z: <https://www.ontrmuseum.ca/tankmuseum/blog-post/modern-anti-tank-ammunition/>. [cit. 2024-04-04].
- [11] Elements of Armament Engineering: Ballistics, Part 2. Online. 107. Headquarters, U.S. Army Materiel Command, 1963. Dostupné z: https://books.google.cz/books/about/Elements_of_Armament_Engineering.html?id=OcuLvgAACAAJ&redir_esc=y. [cit. 2024-04-05].

- [12] Armor simulator. Online. UNKNOWN. Dostupné z: <https://armor-sim.web.app/>. [cit. 2024-04-07].
- [13] Warthunder ads throwing shade at WoT, while still having to unlock "adjustment of fire" modification to accurately shoot at range for every tank lol. Online. 2020. Dostupné z: https://www.reddit.com/r/Warthunder/comments/i1qazw/warthunder_ads_throwing_shade_at_wot_while_still/. [cit. 2024-04-11].
- [14] FIŠER, Jakub. Raději umřít než hrát War Thunder. Fanoušci přistoupili k review bombing, studio Gaijin vrací úder. Online. 2023. Dostupné z: <https://hrej.cz/article/raději-umrit-nez-hrat-war-thunder-fanousci-pristoupili-k-review-bombingu-studio-gaijin-vraci-uder>. [cit. 2024-04-13].
- [15] ZACNY, Rob. War Thunder review. Online. 2014. Dostupné z: <https://www.pcgamer.com/war-thunder-review/>. [cit. 2024-04-13].
- [16] GAIJIN GAMES KFT. Improved Calculation of Armour Penetration in the game. Online. 2019. Dostupné z: <https://warthunder.com/en/news/6010-development-improved-calculation-of-armour-penetration-in-the-game-en>. [cit. 2024-04-13].
- [17] GAIJIN GAMES KFT. Warthunder Wiki. Online. 2024. Dostupné z: https://wiki.warthunder.com/Main_Page. [cit. 2024-04-13].
- [18] What is Post Scriptum?. Post Scriptum game [online]. Unknown: Periscope Games, 2022 [cit. 2024-05-04]. Dostupné z: <http://postscriptumgame.com/the-game>
- [19] SHEEHAN, Gavin. Post Scriptum Rebrands As Squad 44 With New Content. Online. 2023. Dostupné z: <https://bleedingcool.com/games/post-scriptum-rebrands-as-squad-44-with-new-content/>. [cit. 2024-05-06].
- [20] THE FRESH BAKED GOODS. Post Scriptum - Tank Support In New Armor Overhaul. Online. 2021. Dostupné z: <https://www.youtube.com/watch?v=718Bmams8Uw>. [cit. 2024-04-16].
- [21] [Official] How to download the Post Scriptum SDK for modding. Online. 2020. Dostupné z: <https://steamcommunity.com/app/736220/discussions/0/2967272951895219087/>. [cit. 2024-04-16].
- [22] PERISCOPE GAMES. Armour Devblog #3 - "Shell Mechanics". Online. 2021. Dostupné z: <https://steamcommunity.com/games/736220/announcements/detail/2977426980059880961>. [cit. 2024-04-16].

- [23] STORMBIRDS. Tank Crew – Clash at Prokhorovka full review. Online. 2021. Dostupné z: <https://stormbirds.blog/2021/01/17/tank-crew-clash-at-prokhorovka-full-review/>. [cit. 2024-04-19].
- [24] 1C GAME STUDIOS. Dev blog #113. Online. 2015. Dostupné z: <https://il2sturmovik.com/news/192/dev-blog-113/>. [cit. 2024-04-19].
- [25] UNITY TECHNOLOGIES. Unity documentation. Online. 2024. Dostupné z: <https://docs.unity3d.com/Manual/index.html>. [cit. 2024-04-24].
- [26] Introduction to collision detection in Unity. Online. 2024. Dostupné z: <https://www.educative.io/answers/introduction-to-collision-detection-in-unity>. [cit. 2024-04-24].
- [27] FRENCH, John. Raycasts in Unity, made easy. Online. 2024. Dostupné z: <https://gamedevbeginner.com/raycast-in-unity-made-easy/>. [cit. 2024-04-24].
- [28] FRENCH, John. How to use Fixed Update in Unity. Online. 2024. Dostupné z: <https://gamedevbeginner.com/how-to-use-fixed-update-in-unity/>. [cit. 2024-04-24].
- [29] *UTW Official discord server*. Online. 2021. Dostupné z: <https://discord.gg/e6QRc4NKjJ>. [cit. 2024-04-27].
- [30] *Project UTW*. Online. 2023. Dostupné z: <https://github.com/Thomas-Bata-University/UTW>. [cit. 2024-04-27].
- [31] Penetration. Online. 2013. Dostupné z: <https://www.tankarchives.ca/2013/03/penetration.html>. [cit. 2024-05-01].
- [32] SAMSONOV, Peter. 45 mm APCR. Online. 2013. Dostupné z: <https://www.tankarchives.ca/2013/05/45-mm-apcr.html>. [cit. 2024-05-01].
- [33] ANKERSTJERNE, Christian. Armor penetration table. Online. 2021. Dostupné z: <https://panzerworld.com/armor-penetration-table>. [cit. 2024-05-01].

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
AP	Armor Piercing
APC	Armor Piercing, Capped
APDS	Armor Piercing Discarding Sabot
APFSDS	Armor Piercing Fin Stabilized Discarding Sabot
APCBC	Armor Piercing, Capped, Ballistic Cap
APCR	Armor Piercing Composite Rigid
ATGM	Anti-Tank Guided Missile
FAI	Faculty of Applied Informatics
FHA	Face Hardened Armor
GUI	Graphic User Interface
HE	High Explosive
HEAT	High Explosive Anti-Tank
HEATFS	High Explosive Fin Stabilized
HEP	High Explosive Plastic
HESH	High Explosive Squash Head
HMLS	His Majesty's Land Ship
HVAP	High Velocity Armor Piercing
JSON	JavaScript Object Notation
LOS	Line of Sight
MBT	Main Battle Tank
MMORPG	Massive Multiplayer Online Role-Playing Game
PTM	Physics Tank Maker
RHA	Rolled Homogenous Armor

RHAe	Rolled Homogenous Armor equivalency
SPG	Self-Propelled Gun
TBU	Thomas Baťa University
UFP	Upper Frontal Plate
VR	Virtual Reality
WW1	First World War
WW2	Second World War

LIST OF FIGURES

Fig. 1. - Projectile behavior upon striking an armor plate	18
Fig. 2. – Armor LOS	22
Fig. 3. – General armor penetration formula	26
Fig. 4. – De Marre formula	26
Fig. 5. – Krupp formula	27
Fig. 6. – Kinetic Shell script	45
Fig. 7. – Chemical Shell script.....	46
Fig. 8. – Armor plate script.....	47
Fig. 9. – Tank Module script.....	48
Fig. 10. – Rigidbody component	49
Fig. 11. – Chemical penetration (red) with spalling (cyan, blue and black).....	57
Fig. 12. – Spalling simulation.....	59
Fig. 13. – Armor penetration data table	65

LIST OF TABLES

Table 1. - Common values of DeMarre coefficient	27
Table 2. – UTW Ballistic module parameters for various test shells	66
Table 3. - Results of projectile testing of real-life data, WarThunder ballistic simulation and UTW ballistic simulation.....	69

APPENDICES

Appendix P I: CD with a digital version of the thesis, test build and source codes.

APPENDIX P I: UTW BALLISTICS MODULE CD

CD which contains following:

- Digital version of Master's thesis: DP_ZapletalMichal_2024.pdf
- UTW Ballistics module test build: UTW_Ballistics_TestBuild.zip
- Source codes, which were developed as part of the Ballistics module:
UTW_Ballistics_Source.zip