Gymnázium J. K. Tyla Hradec Králové



3D hra v Javě

Maturitní práce z IVT

Autor práce: Ondřej Česák, 4.C

Vedoucí práce: Mgr. Michal Dočekal

Gymnázium J. K. Tyla, 2022

Prohlášení

Prohlašuji, že jsem maturitní práci vypracoval samostatně pod dohledem vedoucího uvedeného na první straně. Všechny použité zdroje jsou uvedeny v seznamu zdrojů a informace z nich získané jsou v textu řádně označeny odkazem na zdroj. Souhlasím s tím, aby tištěná forma práce byla uchována na Gymnáziu J. K. Tyla a tam používána jako tištěný zdroj např. pro další studentské práce či pro prezentaci vzdělávání na GJKT.

V Hradci Králové dne …………………….. Podpis autora práce:

### Anotace

ČESÁK, Ondřej: 3D hra v Javě. Hradec Králové. Gymnázium J. K. Tyla. 2022. Počet stran 15. Maturitní práce.

Maturitní práce se zabývá tvorbou 3D hry v Javě. V práci je uvedeno pár druhů renderingů, které byli nebo jsou používány, z nichž jeden byl právě využit ve zmíněné hře. Součástí práce je také vysvětlení, na jakém principu hra funguje, jak se ovládá, zčásti popis její programové části a seznámení s prvky, které se týkají hry samotné, jakými jsou příběh a scénář.

### Klíčová slova

Ray Casting, Ray Tracing, Rendering, Videohra

## Obsah

[1 Úvod 1](#_Toc99203895)

[2 Studium Problematiky 2](#_Toc99203896)

[3 Popis přípravy projektu 6](#_Toc99203897)

[4 Popis tvorby Projektu 8](#_Toc99203898)

[5 Použité zdroje a literatura 14](#_Toc99203899)

## Seznam obrázků

[Obrázek 1: Výsledek Ray castingu 2](#_Toc99203873)

[Obrázek 2: Výsledek Ray tracingu 3](#_Toc99203874)

[Obrázek 3: Atributy třídy LevelObject 8](#_Toc99203875)

[Obrázek 4: Konstruktor třídy LevelObject 9](#_Toc99203876)

[Obrázek 5: Metody třídy LevelObject 10](#_Toc99203877)

[Obrázek 6: Metoda zajišťující pohyb hráče 11](#_Toc99203878)

[Obrázek 7: Pohled na zeď s rybím efektem 12](#_Toc99203879)

[Obrázek 8: Pohled na zeď bez rybího efektu 12](#_Toc99203880)

[Obrázek 9: Řešení problému rybího efektu 13](#_Toc99203881)

## Seznam tabulek

[Tabulka 1: Příklady metod Java knihovny Swing 5](#_Toc99304920)

[Tabulka 2: Příklady metod Java knihovny AWT 5](#_Toc99304921)

[Tabulka 3: Ovládání hry 12](#_Toc99304922)

# Úvod

Tématem mé maturitní práce je 3D hra naprogramovaná v Javě. Projekt je inspirovaný starou hrou Wolfenstein 3D z roku 1992, jedná se tedy o single-player střílečku z první osoby. Jakožto hráč počítačových her jsem fanoušek těchto starých známých titulů, jakými jsou TES: Daggerfall, DOOM nebo Warcraft a Wolfenstein 3D je z hlediska programu ne tolik složitá hra a stejně tak i způsob renderingu v něm použitý je poměrně jednoduchý. To byl ostatně důvod, proč jsem si tuto hru vybral jako svoji inspiraci a idol pro svoji práci. Cílem celého projektu bylo hlavně sestrojit funkční rendering založený na principu renderingu ze hry Wolfenstein 3D a samozřejmě naprogramovat hru, ve které by byl použit. Programovacím jazykem, jenž jsem ve hře použil, je Java, protože to je jazyk, který jsem za poslední dva roky intenzivně využíval, a tudíž to pro mě byl nejvhodnější kandidát. V práci se zprvu v kapitole **Studium Problematiky** zabývám teorií, tedy obecně, co je to rendering a pár druhů, které existují, jakými způsoby lze v dnešní době hry vytvářet a něco málo ke knihovnám, které jsem využil k tvorbě svého programu. V další kapitole, **Popis přípravy projektu**, se věnuji cíli samotné hry a jejímu příběhu. Poslední část práce, **Popis tvorby projektu**,se již zaobírá tvorbou projektu a obsahuje příklady tříd a problémů, na které jsem narazil a musel je vyřešit.

# Studium Problematiky

## Rendering

Rendering je v počítačové grafice proces, během kterého program vezme data a informace o dané scéně a převede je do požadovaného formátu, což většinou bývá obrázek nebo sekvence obrázků. Jedná se vlastně o převedení zpravidla 3D prostředí do 2D podoby, která je poté vykreslena na obrazovku. (1)

Existuje několik druhů renderingu, které se zaměřují na různé úkoly. V této práci jsou zmíněné jen dva druhy, Ray casting, použitý právě ve hře Wolfenstein 3D, a Ray tracing.

## Druhy Renderingů

### Ray Casting

Ray casting je algoritmus schopný přetvořit omezený formát dat do trojrozměrné projekce za pomocí vysílání paprsků z jednoho bodu (kamery) do pohledové výseče (oblast, vyznačená úhlem pohledu). Základním principem ray castingu je, že paprsky mohou být poslány a sledovány ve skupinách za použití určitých geometrických podmínek. V ray castingu je poslán paprsek za každý pixelový sloupec na obrazovce a získávají se všechna jeho protnutí s objekty. Jako další krok se uloží hodnota nejbližšího střetu. (2)

Ray casting je rychlý, jelikož je třeba spustit pouze jeden výpočet za každý sloupec pixelů na obrazovce. Je to také důvod, proč byl tak populární v dobách starých 3D her. Je zde však daň. Obrázky získané touto cestou nejsou příliš realistické a kvůli geometrickým výpočtům použitých za pochodu není možné vykreslit určitá trojrozměrná tělesa. (2)

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 1: Výsledek Ray castingu

### Ray Tracing

Ray tracing je velmi podobný ray castingu, ale jak název napovídá, jedná se o algoritmus, který se zabývá sledováním světla. V této metodě se sleduje nejen cesta paprsku, ale také to, co se s ním stane, když dopadne na nějaký povrch, kde se může odrazit nebo vstřebat. Jako tomu není u ray castingu, zde vystřelujeme paprsek za každý pixel a sledujeme, zda narazí na nějaký povrch. Při kolizi se pak podle vlastnosti daného povrchu, např. zda se jedná o průhledný nebo o odrazivý povrch, spustí spoustu dalších výpočtů, které vedou k vytvoření osvětlení a stínů. (3)

Obsah obrázku stůl, víno, interiér, brýle

Popis byl vytvořen automaticky

Obrázek 2: Výsledek Ray tracingu

## Herní Enginy

Herní engine je program sloužící k vývoji videoher. Jedná se o software, který má v sobě už zabudované systémy obecně využívané hrami, mezi jež patří například právě renderování ale také i fyziku nebo zvuky. Díky tomu se vývojář může soustředit na konkrétní prvky hry a nemusí vkládat tolik času do těchto základních recyklovatelných systémů. Jsou enginy, které jsou více univerzální, ale vývojář je nucen si vytvořit některé věci sám, a jsou enginy, které jsou zaměřené na konkrétní herní žánr a uživatel se nemusí některými mechanismy už zaobírat, ale zároveň je tento engine oproti tomu univerzálnímu více limitovaný. (4)

Ačkoliv nevyužít herní engine v dnešní době pro tvorbu vlastní hry, zvláště pokud hru vyvíjí jeden nebo malá skupina, může být velký krok zpět, cílem mé hry bylo naprogramovat hru s užitím vlastního renderingu. To automaticky vyškrtlo tuto možnost ze seznamu, a právě proto také nebyl během tvorby použit žádný herní engine. (4)

## GUI a Obsluha událostí

Hra je rozdělena do dvou scén/ploch, hlavního menu a samotné hry. Grafické rozhraní hlavního menu je statické, obsahuje pouze název hry, návod, jak hru ovládat, a pokyn k započetí hraní. Druhá scéna se už skládá z více prvků. Nejhlavnějším je pohled, který je vykreslován přes celé okno každou jednu šedesátinu sekundy (0,0166). Přes pohled, na vrch, se potom zobrazují informace, které hráči poskytují informace o jeho stavu a vlastnictví. Tyto informace se tu nacházejí celkem tři: Hráčova hodnota zdraví, resp. životy, jaké zbraně vlastní a může je tedy používat a číslo reprezentující počet nábojů do právě držící zbraně.

## Vybrané Java API

Grafické uživatelské rozhraní (GUI) je ve hře zpracováno pomocí dvou knihoven platformy Java, Swing a AWT (**Abstract Window Toolkit**). Knihovna Swing byla použita pro vytvoření samotného okna a jeho nastavení, a to za pomoci rozhraní **JFrame**. Dále bylo také využito jejích metod, mezi které patří například:

Tabulka 1: Příklady metod Java knihovny Swing

|  |  |
| --- | --- |
| setSize (**int** width, **int** height) | nastaví velikost okna |
| setLocation (**int** x, **int** y) | nastaví pozici okna na obrazovce |
| addKeyListener (**KeyListener** l) | program zaregistruje **KeyListener**, ze kterého poté může číst hráčovy klávesové vstupy |

Knihovna AWT posloužila jako nástroj k vytvoření více scén, mezi kterými lze přepínat, hlavní menu a samotná hra, za pomoci jednoho z jejích rozložení (**Layout**), která se dají do daného okna nastavit, **CardLayout**. Hlavním účelem AWT v programu však bylo vykreslení projekce prostředí, tedy pohled hráče, kam ale patří i podoby herních objektů nebo hráčovy zbraně, jejichž obrázky byly vytvořené na základě třídy **BufferedImage**. Příklady metod použité v programu jsou:

Tabulka 2: Příklady metod Java knihovny AWT

|  |  |
| --- | --- |
| show (**Container** parent, **String** name) | zobrazí scénu/plochu, které přísluší přiřazený název **name**,v daném **Containeru** |
| first (**Container** parent) | v daném **Containeru** zobrazí jeho první scénu/plochu |
| paintComponent (**Graphics** g) | za použití třídy **Graphics** se uvnitř této metody dají vykreslit různé objekty, ať už se jedná o text, linky nebo i mnohoúhelníky |

# Popis přípravy projektu

## Cíl hry

Hra je dělaná na motivy Wolfensteina 3D a starých her. Princip hry je tedy naprosto jednoduchý. Porážet nepřátele, snažit se přežít a objevovat skryté místnosti. Na konci však hráče nečeká nějaký hlavní nepřítel nebo „Boss“, nýbrž pouze statický objekt, po jehož zničení je hra ukončena.

## Příběh

Hra je posazená do neznámé doby, avšak do našeho světa. Místo, kde se vše odehrává je budova školy Gymnázia J. K. Tyla, blízko které dopadl nevelký meteorit z černé horniny. Po dopadu onoho záhadného kamene se okolní fauna a flóra začala chovat úplně jinak, než jak bylo doposud na Zemi známo. Rostliny, jako by dostali vlastní vědomí, začali vylézat z půdy a spolu se zdivočelou zvěří útočili na nevinné občany. Tyto rostliny poté obsadily budovu gymnázia vytvořili si zde něco jako svoji základnu, kde ukrývají a chrání Srdce rostlin, zvláštní druh květiny, který na této planetě nikdy nerostl. Vypadá to, že funguje jako jedno společné nervové centrum všech „obživlých“ rostlin. A tak se jeden odvážný voják, hráč, odváží na vlastní pěst do budovy vstoupit, probít si cestu na vrch, kde je tajemná květina ukryta a skoncovat tento chaos.

Při spuštění hry je hráči prezentován příběh ve formě následujícího textu:

„A month ago a black stone meteorite has landed on the Earth. People thought it was just a harmless rock from the space, but it turned out they were wrong. Strange things started to happen since then. Animals were aggressive and plant's fruit was inedible. However, things got even crazier. Plants got up from the ground and started to attack citizens. They transformed into some kind of mutated monsters. But there is a way to destroy them. You could of course eliminate every single mutated plant, however that would take a long time. Lucky for us, these mutants have their "heart"-like thing which they all are bound to. If you destroy the heart they all should fall. Their heart is hidden in local high school find it and destroy it.“

## Scénář

Na začátku hry se hráč objevuje v přízemí budovy gymnázia, které je z půlky zavalené sutinami zdí nebo i vyšších pater. Jak hlavní postava prochází zdevastované patro, naráží na rostlinné mutanty a je nucen použít zbraň. Poté, co projde celou tuto oblast, nachází na jejím zvláštní díru, kterou obklopují stěny připomínající povrch rostlin. Při interakci to hlavní postavu přesune do vyššího patra. První patro neskrývá téměř žádné zajímavosti, zato je zde dostatek nábojů. které mohou hráči posloužit jako užitečná podpora při prozkoumávání dalších částí školy. Na konci tohoto podlaží se hráč setkává s dosud neznámým druhem nepřítele, který je rychlejší a nebezpečnější. Cesta do dalšího patra je opět tunel tvořen rostlinnou pokožkou. V této úrovni již všechny zdi nevypadají tak, jak by měli. Jsou pokryté tím samým materiálem jako průchody mezi patry. Hlavní postavě to tedy dokazuje, že se pomalu blíží k cíli. Většina podlaží stále připomíná školní budovu, po jisté době ale hráč vstupuje do dlouhé široké chodby tvořené čistě jen rostlinným materiálem, na jejíž konci se nachází průchod další a poslední části hry. Jedná se o prostor vymezený školními kabinety, avšak to není poznat, protože celá oblast je zarostlá. Hráč se musí prostřílet skrze skupiny nepřátel, aby dorazil k nemalé místnosti, kde se ukrývá hlavní rostlinný mutant. Pokud hráč uspěje a porazí Srdce rostlin, hra končí.

# Popis tvorby Projektu

## Příklad třídy

### Třída LevelObject

LevelObject je třída, definující jakýkoli objekt ve hře, který má svou danou pozici ve světě a vizuální podobu, obrázek, kterou je prezentován hráči.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 3: Atributy třídy LevelObject

Mezi její malé množství atributů patří desetinné hodnoty x a y, které reprezentují souřadnice pozice tohoto objektu ve scéně. Dalšími desetinnými hodnotami jsou screenX a screenY. Tyto vlastnosti objektu říkají programu, kde na obrazovce se daný objekt nachází. Hitbox každého objektu je na výšku postavený sloup a hodnota hitbox udává poloměr tohoto hitboxu. Hodnota size je zakomponována do algoritmu, který zajišťuje vykreslení bojektu na obrazovku a ovlivňuje ji tím způsobem, že prakticky určuje jak velký či malý objekt bude. Posledním atributem Level Objektu je myImage. Ten drží takový obrázek, který se má zrovna ukazovat hráči, protože objekty mohou mít více obrázků, které střídají na základě toho, zda například útočí, nebo jsou zraněni. Atribut myImage je poté proměnná, do které je uložen obrázek prezentující právě probíhající stav objektu.

Konstruktor třídy Level Objekt pouze najde pomocí třídy ResFileReader soubor ve zdrojové složce a získá si z toho informace, které potřebuje pro vytvoření nového objektu, což je například počáteční pozice a poloměr hitboxu.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 4: Konstruktor třídy LevelObject

Jedinými dvěma metodami této třídy jsou distToPlayer a distToPlayerTan. Obě metody vracejí hodnotu, která se rovná vzdálenosti objektu od hráče, ale ne tu stejnou vzdálenost, a ne stejným způsobem. Metoda distToPlayer vrací jednoduše způsobem Pythagorova trojúhelníku vzdálenost objektu hráče. Hodnota je ale vrácená v nezmenšeném tvaru a při jejím použití je třeba nejprve vložit získanou hodnotu pod odmocninu. Metoda distToPlayerTan je lehce odlišná. Tato metoda nemá sloužit k tomu, abychom zjistili, jak daleko je tento objekt od hráče ve scéně, ale potřebujeme vědět, jak daleko od něj je z pohledu kamery, abychom mohli poté příkladným matematickým způsobem vykreslit na obrazovku objekt správné velikosti. Jedná se vlastně o metodu rozšiřující k metodě distToPlayer, akorát s tou výjimkou, že kdybych vytvořil rovinu na hráčském pohledu tak, aby směr pohybu byl normálovým vektorem pro tuto rovinu, poté metoda distToPlayerTan určuje vzdálenost od této roviny. Závislost na této funkci je zapříčiněna způsobem, jakým vykresluji zdi, protože i u zdí používám vzdálenost od této roviny, a ne vzdálenost od samotného bodu pohledu proto, aby při vykreslení obrazovky nevznikal tzv. efekt rybího oka.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 5: Metody třídy LevelObject

## Zpracování pohybu

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek 6: Metoda zajišťující pohyb hráče

Pohyb hráče je velice jednoduchý. Program si zjistí s využitím hráčovy nastavené rychlosti a jeho úhlu pohledu, kam by se měl v příštím snímku hráč pohnout. Hráč se na tuto pozici pohne a proběhne kontrola prostředí, zda hráč nestojí ve zdi nebo se za procesu nedostal dovnitř cizího hitboxu. Pokud ano, je vypočítán nejbližší bod, na kterém může hráč stát. V případě tzv. sebratelných předmětů (**Pickable**) to hráče neposune, ale při kolizi je daný objekt ze současné hry odstraněn, a tudíž není třeba hráče nikam posouvat. Metoda performMovement má jeden parametr a tím je celočíselná hodnota a, která reprezentuje směr pohybu vůči otočení hráče ve stupních. Pokud se například chceme pohnout doleva, a nese hodnotu 90.

## Ovládání

Ovládání hry je omezené pouze na klávesnici, ale během hry se objevují i vyskakující okna, která mají tlačítko a lze je zavřít, nebo potvrdit klasicky myší. Ovládání, které je vysvětleno i ve hře, vypadá následovně:

Tabulka 3: Ovládání hry

|  |  |
| --- | --- |
| Pohyb | W, S, A, D |
| Otáčení | Levá a pravá šipka |
| Střelba / Útok | Mezerník |
| Aktivace / Interakce | E |
| Odejít ze hry | klávesa Escape |

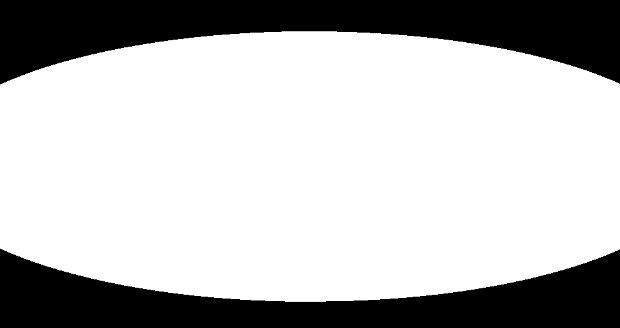
## Zpětná vazba

### Hodnocení 1

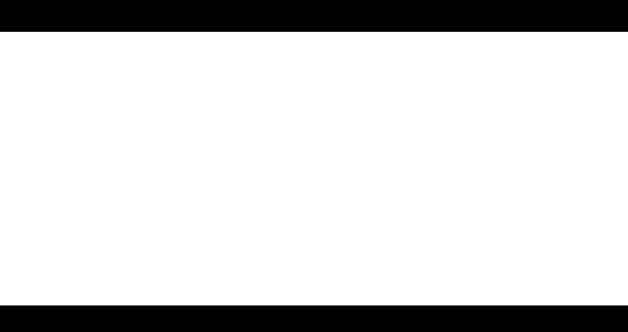
Hráč hodnotil hru převážně pozitivně, akorát mu bylo lehce nevolno při delším pohledu na stále renderující se obraz. Chválil grafické zpracování, game play jej bavil a příběh mu přišel i docela zajímavý. Vytkl nedostatek zvuků.

## Řešení problémových částí

Největším problémem mé práce bylo odstranit vznikající rybí efekt, který se objevoval v důsledku použitého renderovacího algoritmu.

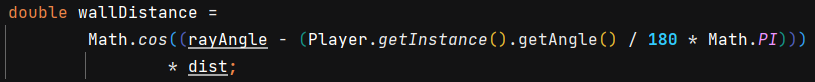


Obrázek 7: Pohled na zeď s rybím efektem



Obrázek 8: Pohled na zeď bez rybího efektu

Ve hře jsem použil renderovací algoritmus na způsob hry Wolfenstein 3D, což znamená, že jsem za každý pixelový sloupec na obrazovce vyslal jeden paprsek do 2D prostoru a sledoval jeho trasu a zda se nestřetl se zdí. V případě, že nastala kolize se zdí, se nezíská barva jako u klasického Ray Castingu, ale hra si uloží vzdálenost hráče a bodu střetu a během vykreslovacího procesu, kde v každém pixelovém sloupci vybarví určitý počet pixelů, podle toho jak daleko daný paprsek doletěl. Čím větší vzdálenost, tím méně pixelů je vybarveno. Problém s rybím efektem nastával práve v procesu zpracovávání vzdálenosti, kterou paprsek urazil. Jak je patrné z levého obrázku, krajní paprsky urazili větší vzdálenost než došlo ke střetu se zdí, než paprsky ležící uprostřed. To dává smysl, neboť jestliže náš pohled směřuje kolmo na rovnou zeď, tak potom náš pohled vytváří jakýsi trojúhelník, kde krajní paprsky opravdu urazí větší vzdálenost. Takové prostředí ale působí docela nepřírozeně a tak bylo třeba se tohoto efektu zbavit tak, aby byli všechny paprsky jakoby sjednocené, reps. aby měli všechny paprsky stejnou vzdálenost, pokud je hráč otočen přímo proti rovné zdi, tak aby pohled vypadal stejně jako na obrázky vpravo. Toho jsem docílil jednoduše pomocí krátkého výpočtu, kde každý paprsek násobím funkcí cosinus úhlu, o který se liší od úhlu otočení hráče. Níže je ukázka kódu, který celý problém vyřešil.



Obrázek 9: Řešení problému rybího efektu

## Cizí pojmy

Rendering – algoritmus pro získání dat, většinou 3D, a jejich následné projekce do 2D (viz podkapitola Rendering)

Herní engine – viz kapitola Herní Enginy

# Použité zdroje a literatura

(1) What is Rendering? (For 3D & CG Work). *Concept Art Empire - Tips & Resources For Artists* [online]. Copyright © 2022 Concept Art Empire [cit. 24.03.2022]. Dostupné z: <https://conceptartempire.com/what-is-3d-rendering/>

(2) What is Ray Casting? - Definition from Techopedia. *Techopedia: Educating IT Professionals To Make Smarter Decisions* [online]. Copyright © 2022 [cit. 24.03.2022]. Dostupné z: <https://www.techopedia.com/definition/21614/ray-casting>

(3) What is ray tracing? Everything you need to know, from games to graphics cards | TechRadar. *TechRadar | The source for tech buying advice* [online]. Copyright © [cit. 24.03.2022]. Dostupné z: <https://www.techradar.com/news/ray-tracing>

(4) Co je to herní engine [Obecné téma] - CESKEMODY.cz . *CESKEMODY.cz - MODIFIKACE A MAPY PRO POČÍTAČOVÉ HRY* [online]. Copyright © Petr Skovajsa [cit. 24.03.2022]. Dostupné z: <https://www.ceskemody.cz/clanky.php?clanek=56>

## Obrázky

(Obrázek 1: Výsledek Ray castingu) AUTOR NEUVEDEN. *raywenderlich.com* [online]. [cit. 26.3.2022]. Dostupný na WWW: <https://www.raywenderlich.com/books/metal-by-tutorials/v2.0/chapters/18-rendering-with-rays>

(Obrázek 2: Výsledek Ray tracingu) AUTOR NEUVEDEN. *electronicdesign.com* [online]. [cit. 26.3.2022]. Dostupný na WWW: <https://www.electronicdesign.com/technologies/displays/article/21801219/whats-the-difference-between-ray-tracing-ray-casting-and-ray-charles>