**STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST**

**Obor č. 18-20-M/01: Informační technologie**

# 3D hra tvořena v Unity

**Dominik Malinovský**
**Pardubický kraj**

**Pardubice 21.3. 2022**

# STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

**Obor č. 18-20-M/01: Informační technologie**

# 3D hra tvořena v Unity

# 3D game made in Unity

**Autoři:** Dominik Malinovský
**Škola:** DELTA – SŠIE, Ke Kamenci 151, 530 03 Pardubice
**Kraj:** Pardubický kraj
**Konzultant:** Mgr. Jan Mottl

Pardubice 21.3. 2022

## Zadání maturitního projektu z informatických předmětů

Jméno a příjmení: *Dominik Malinovský*

Školní rok: *2021/2022*

Třída: *4.A*

Obor: *Informační technologie 18-20-M/01*

Téma práce: *3D hra tvořena v Unity*

Vedoucí práce: Mgr. Jan Mottl

## Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:

Cílem projektu bude vytvořit 3D hru v prostředí Unity. Samotná hra se bude odehrávat v budoucnosti (ponese v sobě prvky sci-fi). Postava, kterou bude hráč ovládat bude mít různé schopnosti, kromě základních jako pohyb, možnost útočit. Hráčova postava bude moct získat různé schopnosti, které upraví ovládání postavy, různé nadpřirozené schopnosti. Hra bude mít svou storyline, podle které se hráč bude řídit a vše budou doprovázet AI nepřátelé (obyčejní, silní, unikátní (bossové)).

Hra bude obsahovat animace různých zbraní. Postava hráče se bude také upravovat skrz možnost nákupu různých vylepšení za herní peníze, které bude moct v jednotlivých úrovních získávat. Bude možnost si i hru uložit a nahrát zpět, aby mohl hráč pokračovat tam, kde naposledy skončil. V menu si hráč bude moci nastavit: grafiku hry, hlasitost, rozlišení, fullscreen, a další.

Celá hra bude jen singleplayer, žádný multiplayer ve hře nebude.

Game Engine pro hru bude Unity a hra bude napsána v C#.

## Stručný časový harmonogram (s daty a konkretizovanými úkoly):

**Září 2021**
Postava a animace pro ni; Základní storyline příběhu hry
**Listopad 2021**
AI a UI pro hru; Finální představení příběhu hry
**Prosinec 2021**
Level design pro hru a možnost uložit hru a nahrát ji
**Leden 2022**
Finální podoba většiny herních levelů; Testování hry
**Únor 2022**
Doladění chyb, které vyjdou z testování; Sepisování dokumentace k projektu

# Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Pardubicích dne 21.3. 2022 ……

# Poděkování

Děkuji panu Mgr. Janu Mottlovi za vedení maturitní práce a za pomoc ohledně projektu a interpretovi „Qillix“ za dovolení použít jeho hudbu v tomto projektu.

# Anotace

Tato práce popisuje využití programu Unity a její skriptovací knihovny. Za pomocí mnoha užitečných komponentů lze vytvořit objekty. Zároveň tato práce přináší náhled do průběhu vývoje a vysvětluje funkce použitých komponent. Dále práce rozebírá tématiku grafických módů a jejich využití k dosažení konkrétních vizuálních efektů. Výsledkem této práce je 3D hra nesoucí název „Most Wanted", která je zaměřena na žánr FPS (First Person Shooter), sci-fi a celou hrou je hráč provázen skrz dramatický příběh.

# Klíčová slova

C#; Unity Engine; FPS; Animace; Komponenty

# Annotation

This work describes the use of the Unity programming engine and it's scripting library. With the help of several useful components, Unity can be used to create functional objects. This document also describes the process of development and explains why and how the different components work. It also discusses the use of the graphic modes that was used to achieve the visual effects used. The produck of this work is a game called "Most Wanted", which combines elements of FPS (First Person Shooter), sci-fi genre and story line.

# Keywords

C#; Unity Engine; FPS; Animations; Components

# OBSAH

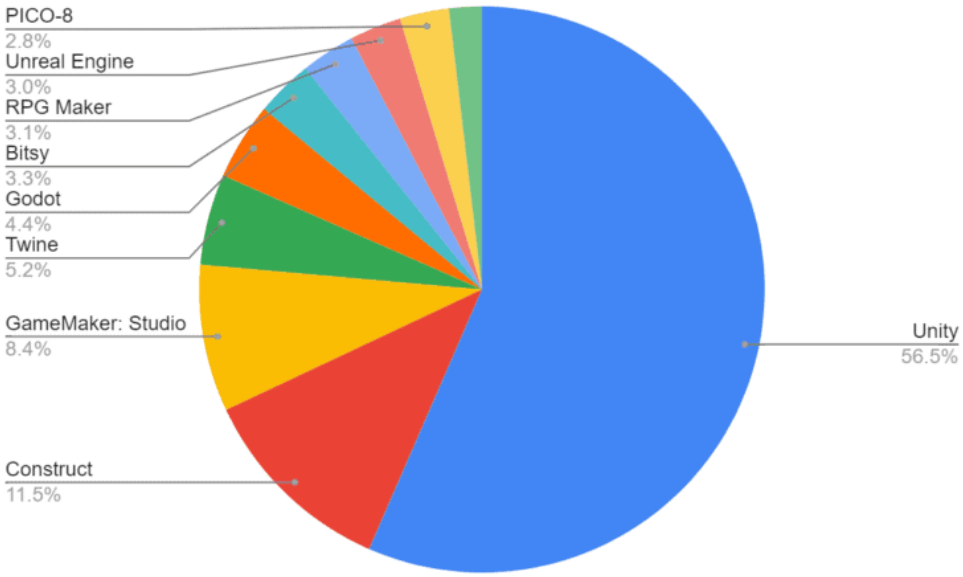
# 1. Úvod

Cílem tohoto projektu bylo vytvořit hru v 3D prostoru s žánrem FPS (first person shooter). Hra jako taková obsahuje více střelných zbraní od útočné pušky po pistoli. Tyto zbraně lze ve hře samotné vylepšit na konkrétním místě. Uživatel si bude moci vylepšit i množství zdraví a staminu (výdrž, která ubývá při sprintování a skákání, při chůzi se opět přidává) a rychlost získávání staminy při chůzi. V rámci hry se hráč střetne s velkým množstvím nepřátel, kteří si sami najdou hráče a snaží se ho zneškodnit. Skrz hru provází hráče různé úkoly, jejich plnění posouvá hráčskou postavu v rámci příběhu. V hlavním menu, které slouží jako úvod hry s možností si nastavit kvalitu modelů, osvětlení, textur (obrázek pro daný model, aby vypadal realisticky a bylo poznat co za daný model to je), hlasitost a režim v okně nebo celá obrazovka. Menu má více možností, jako je spustit novou hru a načíst na úroveň, kde byla naposledy uložena. Během hraní je hru možnost pauznout pomocí klávesy „escape“, podobné k hlavnímu menu, kde uživatel si může nastavit kvalitu, hlasitost a další již zmíněné možnosti. Uživateli je dáno UI (uživatelské rozhraní, znázorňující počet zdraví, staminy, nábojů v zásobníku, kterou právě drží, ale také i úkoly co má hráč v danou chvíli splnit, počet mincí, pro možný nákup vylepšení a crosshair (ukazatel ve středu obrazovky, který uživateli napoví kam hráč střílí)).

Když jsem se dozvěděl, jak funguje vývoj nějaké hry, ihned jsem si chtěl zahrát na profesionálního vývojáře ze známých herních studií po celém světe. Když mi byla naskytnuta možnost si zkusit nějaký vývoj, ihned jsem si vybral vývoj nějaké hry v nějakém herním enginu (program ulehčující práci při tvorbě náročných aplikací a her, u většiny z nich se vyskytují 3D rozhraní, kde si uživatel může nastavit potřebné objekty, více v celé kapitole o Unity Engine). Vybral jsem si Unity Engine a téma, které mi sedělo a to bylo „3D hra tvořena v Unity“. A díky tomuto programu jsem schopný naplno využit mé znalosti ze světa programování a z programovacího jazyka C#.

# 2. UNITY ENGINE

Unity je populární engine pro vývoj 2D, ale i 3D her a aplikací, jeho první vydání bylo v roce 2005. Dříve byl známý kvůli hrám typu „Indie (Hry vytvořeny malým herním studiem, nejsou náročné na výkon počítače a nejsou tak příběhově nebo hratelně dlouhé)“, ale v dnešní době, díky nové technologii s názvem HDRP (High Definition Render Pipeline, toto bude vysvětleno v podkapitole o Render Pipeline), dokážeme udělat vysoce realistické stíny, textury, záření, ale i sluneční volumetrické efekty (efekty napodobující slunečním paprskům) [1]




*Obrázek 1 Graf popularity [2]*

## 2.1 Render Pipeline

Render Pipeline je označení pro renderování (zobrazení) světel a stínů. Momentálně Unity podporuje 3 druhy pipelinů. Nejstarší (SRP), jinak označován build-in, se v dnešní době už skoro vůbec nepoužívá, už jen z toho důvodu, že se v rámci tohoto pipelinu musí veškeré efekty nastavit a naprogramovat ručně. [3]

### 2.1.1 Standard Render Pipeline (SRP)

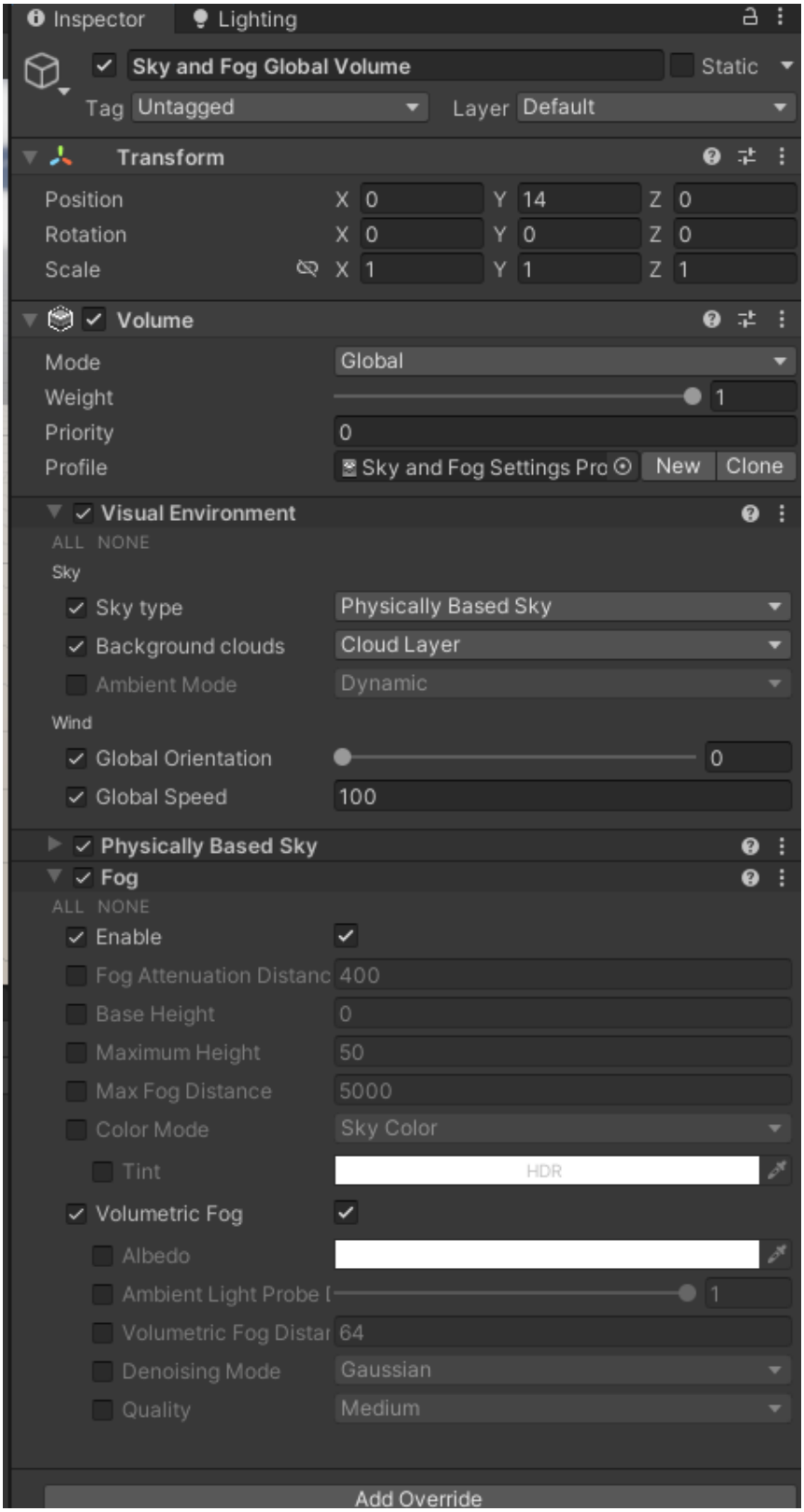
SRP, je nejvíce základní render pipeline v Unity. Pokud bychom chtěli nastavit počet světel, které se budou zobrazovat na textuře, rozlišení, povolení HDR (High Dynamic Range), kvalita textur, Post-Processing (možnost upravit barvy, jas, rozostření a celkové zlepšení obrazu) a dalších plno nastavení, tak bychom z velké části museli napsat skript podle sebe. [4]

### 2.1.2 Universal Render Pipeline (URP)

URP, je pipeline, který se dá lehce změnit v GUI (grafické uživatelské prostředí) nabídce, ale má výhodu, že se dají měnit funkce pomocí již vytvořeného skriptu, který URP používá tak, aby správně fungovalo. Dnes se používá pro jednoduchost, mobilní aplikace a hry s žánrem „Indie“. [5]

### 2.1.3 High Definition Render Pipeline (HDRP)

HDRP, je podobný pipeline k URP, ale tento má o hodně funkcí navíc. Jak už bylo zmíněno, HDRP dokáže za pomocí pár kliknutí vytvořit pěkné „sluneční světlo“ a to díky objektu „Sky and Fog Global Volume (můžeme nastavit jaký typ oblohy budeme využívat a jak moc hustou mlhu potřebujeme, jakou barvu bude mít a jakou kvalitu chceme)“. Všechny světelné objekty nyní mají možnost nastavit teplotu a intenzitu osvícení. Objekty a komponenty použité v projektu budou popsány v kapitole o komponentech. Na další straně lze vidět komponentu „Sky and Fog Global Volume “. [6]




*Obrázek 2 Sky and Fog Global Volume*

## 2.2 Skriptovací jazyk

Jako hlavní výhodou Unity enginu je ta, že veškeré skripty fungují na C# jazyce. Samotný engine používá vlastní knihovnu, ve které jsou metody, proměnné a všechny reference, které pomohou při vývoji v unity. S danou knihovnou se dají programovat mechaniky hráče, funkčnost objektů, AI a mnoho dalších. Do skriptu můžeme vložit hlavní metody „Start()“ a „Update()“, poté můžeme použit i metody „Awake()“, „FixedUpdate()“ a „LateUpdate()“.[7]

### 2.2.1 Metoda Start()

Proběhne ihned v prvním snímku od aktivování skriptu ještě předtím než je metoda „Update()“ zavolána. [8]

### 2.2.2 Metoda Awake()

Liší se od metody Start() tím, že se inicializuje (vykoná), při načtení scény, vytvořením nového objektu přes „Instantiate (metoda, která umožní vytvořit nový objekt, podle objektu který již existuje, tzv. duplikování objektu)“ a nebo pokud objekt byl dříve neaktivní a za pomocí skriptu se aktivoval. Metodu tedy použijeme v případě, že chceme kód inicializovat ještě předtím, než se nám aplikace spustí. [9]

### 2.2.3 Metoda Update()

Probíhá každým snímkem za sekundu. Využijeme to v případě, že chceme, aby se kód neustále vykonával dokola. Na následujícím obrázku, použijeme metodu Update() na kontrolování hráče jestli splňuje podmínku. V případě, že splňuje, hráč se nebude moci hýbat, odemkne se kurzor (bude se moci volně pohybovat po obrazovce) a zobrazí se kurzor, aby byl viditelný. V opačném případě se kurzor uzamkne do středu obrazovky a následně se kurzor skryje. [10]

```
void Update()
{
    if (UI_PauseMenu.gameIsPaused || P_UpgradeShop.isUpgradeMenuOpen || dialogues.isDialogue)
    {
        canMove = false;
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
    }
    else
    {
        canMove = true;
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
    }
}
```

*Obrázek 3 Update Metoda*

### 2.2.4 Metoda FixedUpdate()

Vypočítá fyziku (např. gravitace, chůze, běh), která se používá hlavně u komponentu „Rigidbody (chová se jako reálné těleso, můžeme nastavit hmotnost, gravitaci, sílu pohybu hráče)“. FixedUpdate() se provede jednou za 0.02 vteřiny, což znamená, že se tato metoda za 1 celou sekundu zavolá 50krát.[11]

### 2.2.5 Metoda LateUpdate()

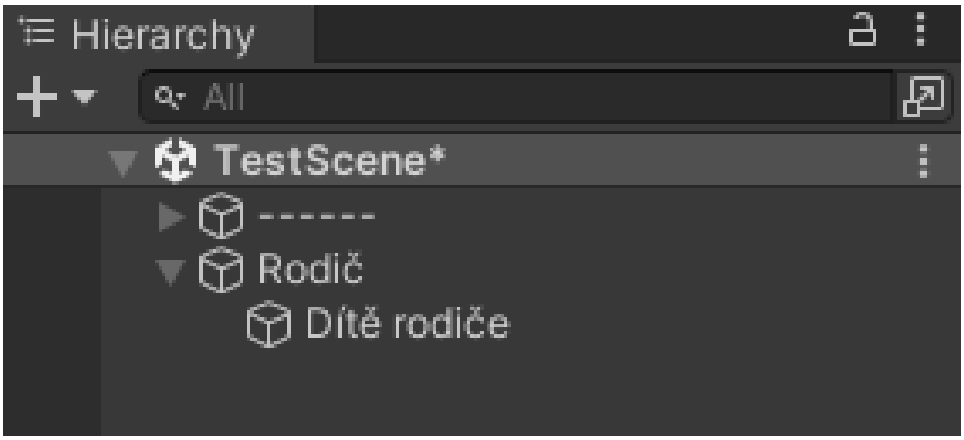
Provede se pokaždé po metodě Update(). Tato metoda je doporučena pro práci s kamerou, z toho důvodu, že se aktualizuje až po vykonání hlavního kódu v Update().[12]

## 2.3 User Interface

Unity má také vlastní uživatelské rozhraní. Uživatel si ho může přenastavit na to, jak sám potřebuje a jak mu to vyhovuje.

### 2.3.1 Hierarchy

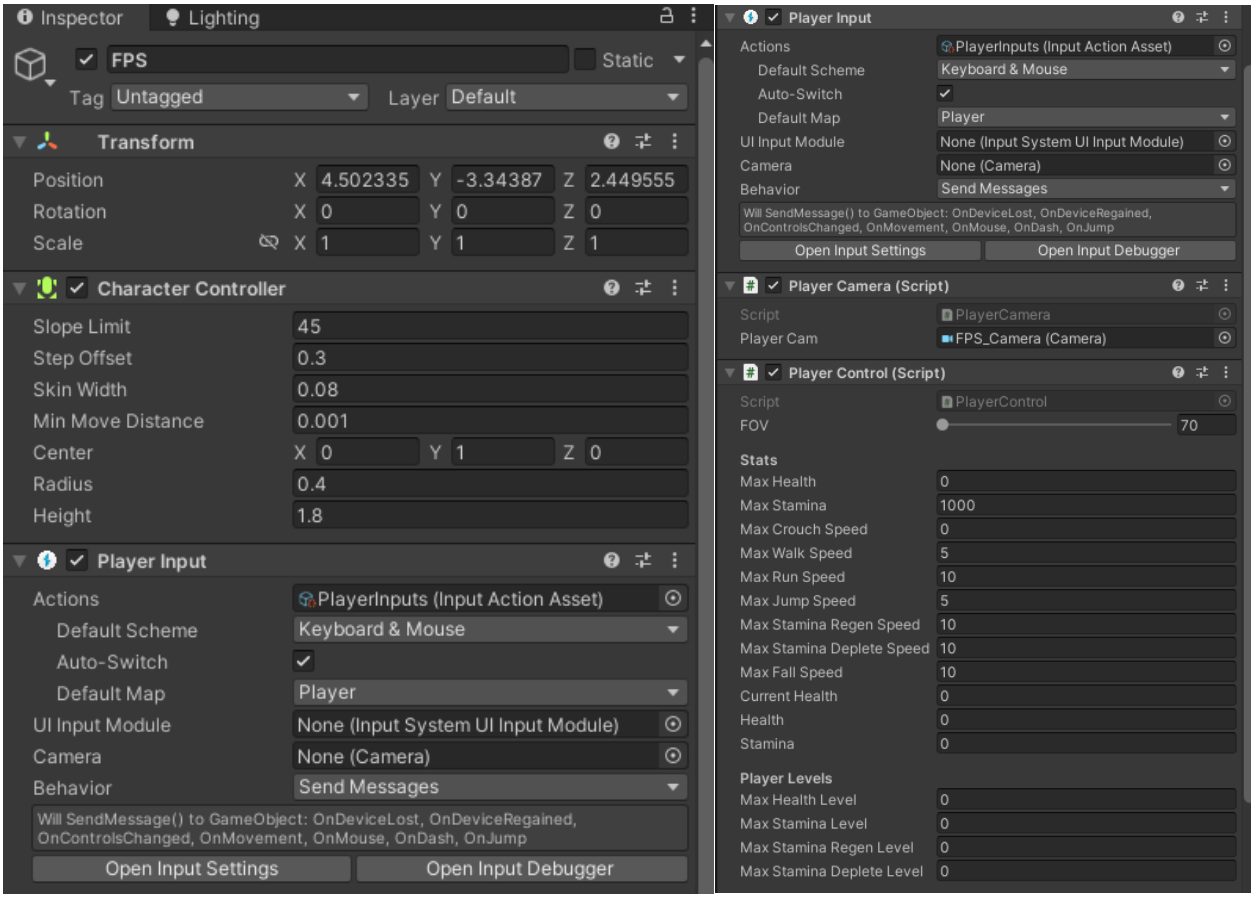
Hierarchie slouží k zobrazení objektů, které jsou v dané scéně (viz. níže) vložené, a které se renderují (zobrazují). Uživatel si může z daných objektů udělat parent (rodič) a child (dítě rodiče) objekty. Pokud uživatel pohne s celým objektem, což je většinou parent, tak pohne i s child objektem, a tak objekt zůstane nenarušen a může se používat dál. Trochu jinak to je u child objektů. Uživatel si musí dávat pozor, jelikož si tímto činem může pokazit celý objekt a může mu to omezit práci, například tím, že se daný objekt bude zobrazovat jinde i když parent je na místě kde ho uživatel chce mít. [13]




*Obrázek 4 Rodič a Dítě*

### 2.3.2 Inspector

Toto okno se využívá k nastavení pozice objektu, přidání komponent a nastavení daného skriptu. Uživatel si s objektem může hýbat, přemisťovat na dané místo, upravovat, přidávat komponenty podle využití, jelikož je dynamický a volně nastavitelný.[14]




*Obrázek 5 Inspector*

### 2.3.3 Project

Okno je podobné k prohlížeči souborů od Windowsu, jelikož uživatel si může upravovat složky, přidávat objekty a pohybovat se mezi assetama (vložené modely, skripty, nastavení a ruzné hodnoty přidané do okna Project). I z tohoto místa lze využít okno Inspector k upravení daného objektu, prefabu (objekt, který má nastavené hodnoty podle uživatele, může to být model, různé herní nastavení, ale i skupina modelů, například sestavené město se může dát do jednoho prefabu, poté se dá znovu použít bez nutného upravování) anebo přidání textur pro daný materiál. [15]

### 2.3.4 Console

Konzole slouží pro zobrazení chyb, notifikací a debugování (zjišťování funkčnosti aplikace, hledání chyb v živé verzi) aplikace. Do této konzole se psát nedá. [16]

### 2.3.5 Scene

Díky scéně můžeme do levelu(scény) vkládat modely a různé prefaby se skriptem a tím sestavit celou mapu podle sebe. Každý pohyb se ukládá a aktualizuje svoji pozici (X, Y, Z). Veškeré umístěné objekty jsou viditelné i v hierarchii. [17]

### 2.3.6 Game

Okno Game využijeme k testování aplikace, jelikož zobrazuje funkčnost živě před vývojářem, a tak může otestovat jak daný skript nebo objekt funguje. V živé verzi se dají upravovat všechny objekty, umístění, měnit hodnotu proměnných ve skriptu, mazat objekty, ale jen po určitou dobu, co je živá verze spuštěna, po ukončení se vše vrátí do původního stavu, než aplikace byla spuštěna.[18]

## 2.4 Lightning

Zobrazení nastavení pro úpravu osvětlení v dané scéně(levelu), jako celé unity i toto malé okno má kategorii scene, ale také i Environment, Realtime Lightmaps a Baked Lightmaps. [19]

### 2.4.1 Scene

Obecné nastavení pro danou scénu. Lze nastavit, jestli chceme, aby se světla aktualizovali v průběhu živé verze aplikace anebo, aby se osvětlení „zapeklo (vytvoří se textura, která bude mít hodnoty světel a přidá se na texturu určitého modelu“ do textury daného objektu, který musí být nastaven jako „Statické“, což znamená, že se objekt nikdy v živé verzi nepohne. Pokud by se objekt pohnul mohly by se objevit různé artefakty (tečky, černé vybarvení na textuře, jelikož engine předpokládá, že se s objektem nebude hýbat). Jako poslední možnost je „mixovaná“ osvětlení, dynamické objekty se aktualizují v reálném čaše a „static“ objekty jsou „zapečené“. A tak se nemusí řešit různé problémy a zvažování, jestli použit první nebo druhou možnost. Jako vedlejší, ale zároveň i hlavní nastavení je výběr pro „baking system“, kde můžeme použít CPU nebo GPU. V dnešní době je využití GPU lepší a minimálně desetkrát rychlejší, než CPU. Nastavení také umožňuje nastavit počet odrážení světel, velikost, vzdálenost mezi danými objekty, které se ukládají do takzvané „Lightmap“ textury, filtry a kvalitu.[20]

### 2.4.2 Environment

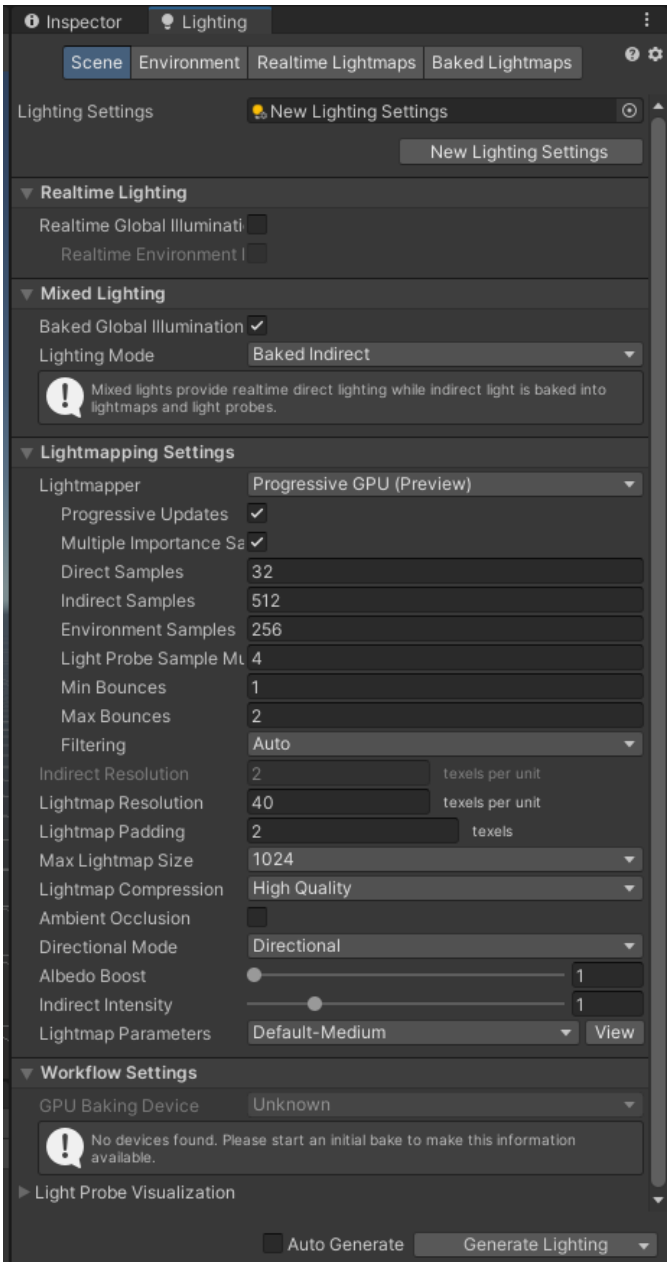
Možnost upravení světel z dynamického profilu na static. Lze nastavit obloha, mraky a volumetrické osvětlení. [21]

### 2.4.3 Realtime Lightmaps

List textur pro „realtime“ osvětlení. [22]

### 2.4.4 Baked Lightmaps

List textur pro „baked“ osvětlení. [23]




*Obrázek 6 Lightning*

# 3. MODELY

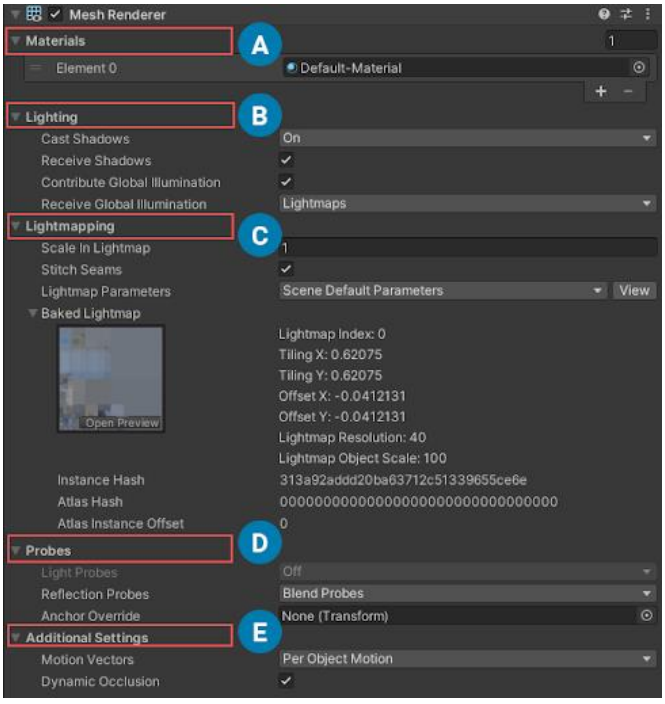
Modely jsou soubory, které v sobě mají data ohledně tvaru(mesh) a vlastností. Skládají se z polygonů (jednotlivé části/body, které se spojí a vzniknou strany. Pomocí stran už se pak jednoduše vytvoří modely. U modelu může být velké množství bodů, ale musíme si dát pozor na počet, jelikož pokud máme více bodů, může se nám začít program sekat anebo se může celý počítač zmrazit a bude vyžadován tvrdý restart). Jedná se o 3D soubor, který lze použít v různých programech a dají se jednoduše upravovat.  Lze je vytvořit v programu Blender. Model obsahuje vlastnost „mesh (tvar modelu)“, který zobrazuje přesný tvar, jak byl model vytvořen. Má 2 strany, první strana je viditelná a druhá není vidět (Backface Culling). V Unity existuje možnost nastavit takzvaný „Two Sided“, který vypočítá stranu tak, aby přes model nešlo osvětlení, které zde nepotřebujeme, ale samotnou stranu nám to nepřídá / nezobrazí. [24]

## 3.1 Mesh

Data a vlastnosti modelu. V Unity se nachazejí 4 možnosti, jak daný mesh zobrazit a upravovat.

### 3.1.1 Mesh Renderer Component

Nese v sobě informaci, jak daný model zobrazit.




*Obrázek 7 Mesh*

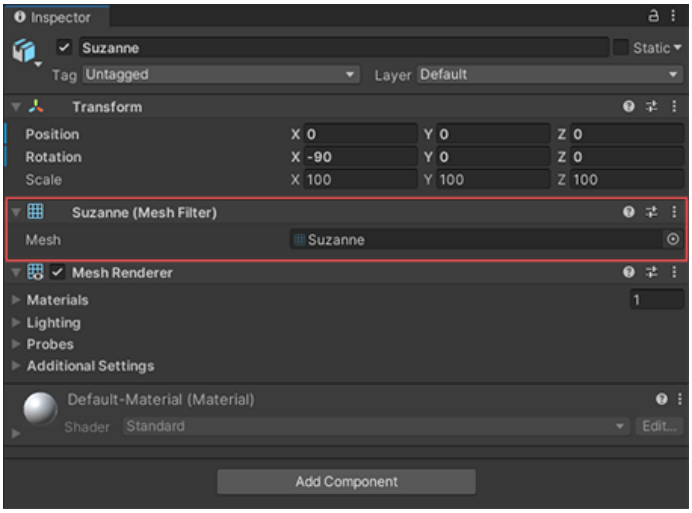
A) Sekce, která slouží k přidání a upravování materiálu pro daný model. V programu na tvorbu modelů lze nastavit počet materiálů a v Unity se nám to automaticky zobrazí. Následně jen vytvoříme materiál s určenými texturami a vložíme jej na danou pozici
B) Nastavení pro získávání a tvoření stínů a celkové nastavení pro osvětlení
C) Velikost lightmapy (textura, která nese data s pozicí a velikostí osvětlení, který se pak přenese na reálnou texturu a vznikne z toho textura, která na nějakém místě bude světlejší a jiném místě tmavší)
D) Nastavení pro odrážení světel od ostatních objektu, např. pokud máme bílou kostku a dáme ji ke žluté stěně, kostka bude mít na dané straně žlutý odlesk, ale na opačné bude mít barvu bílou, pokud tedy nemáme žlutou stěnu i na druhé straně.
E) Možnost nastavit, jestli se objekt skryje, pokud není před kamerou, tím se ušetří náročnost na výkon, pokud by vše bylo aktivované, mohlo by se stát, že by se aplikace sekala a byla náročná na výkon. [25]

### 3.1.2 Skinned Mesh Renderer Component

Podobný komponent k „Mesh Renderer Component“, ale tento se využívá, pokud se jedná o model, který se bude v aplikace deformovat (např. pokud chceme vytvořit objekt oblečení, vlajky) [26]

### 3.1.3 Mesh Filter Component

Drží v sobě data, jak přesný tvar má model mít a jak ho má zobrazovat. Pokud bychom tento komponent neměli, s větší pravděpodobností bychom model neviděli. [27]



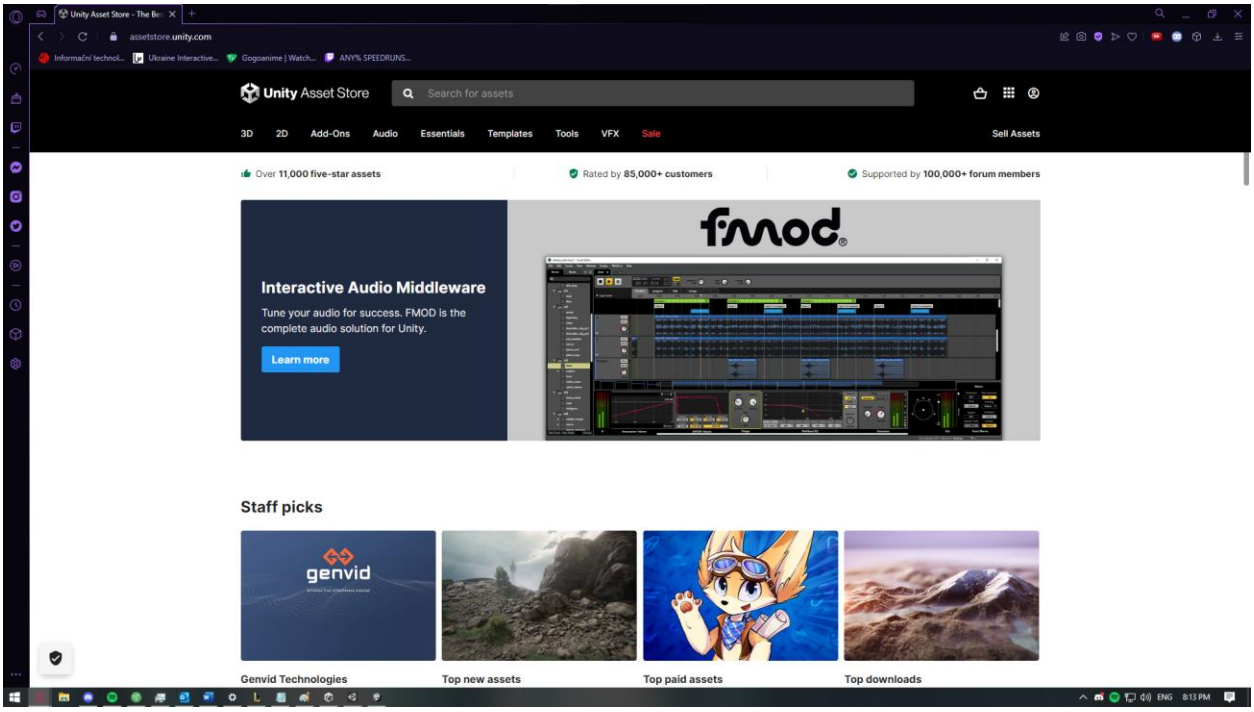

*Obrázek 8 Mesh Filter*

### 3.1.4 Text Mesh Component (Legacy)

Komponent, který dříve byl používán na zobrazení 3D textu ve scéně. Dnes se už tento komponent nepoužívá. [28]

# 4. UNITY ASSETS

Abychom nemuseli vše vytvářet sami, dokáže nám pomoci stránka Unity Assets. Místo, kde si můžeme zakoupit nebo zdarma stáhnout různé modely, aniž bychom je museli dělat ručně. Tímto způsobem bylo v projektu vytvořeno město z budov, které byly zdarma. [29]




*Obrázek 9 Asset Store*

# 5. JAK HRA FUNGUJE

## 5.1 Hráč

### 5.1.1 Pohyb

Hráč je nejdůležitější objekt a nejtěžší část celého projektu. Aby nám hráč fungoval dobře musíme mít kameru. Pomocí kamery můžeme vidět co je před námi. Kamera je připevněna k celému objektu hráče a to proto, aby vždy při hýbání byla na stejné pozici hráče a tímto způsobem se nám vytvoří tzv. první pohled (pohled reálného člověka). Pohyb je vytvořen pomocí rychlostí (proměnné typu float). Je potřeba využít „CharacterController“ (komponenta, která nám pomůže s pohybem hráče). V Unity si přednastavíme vstupy pro pohyb (rovně a doprava, jelikož každý vstup má hodnotu plus a minus) Následně v kódu sečteme a vynásobíme rychlosti, jakým směrem chceme.

### 5.1.2 Skrčení

Pro skrčení musíme znát velikost hráče, ta je dána v komponentě „CharacterController“ a hodnotu vstupu, poté jen v kódu při stisknutí daného tlačítka nastavíme základní výšku na poloviční a naopak.

### 5.1.3 Skok

Při skoku využijeme vertikální pozici (Z) a tu následně zvýšíme určeným počtem. Aby skok byl plynulý vynásobíme celý kód pomocí „Time.deltaTime (kód neproběhne instantně, ale bude po částech po jednom snímku vykonávat celý kód“, poté jen aplikujeme gravitaci, aby hráč dopadl na zem.

### 5.1.4 Zdraví

Zdraví se při každém novém načtení levelu nastaví na maximální hodnotu, která je určena podle potřeby vývojáře (základ je 100 v proměnné float). Hlavní využití životů je takové, aby si hráč hlídal zdraví, zdali má více než 0 životů. Pokud hráč má méně jak 0 životů, uživateli se objeví „Game Over (konec hry)“ okno na obrazovce. Aby hráč měl možnost si zdraví zase zvýšit, při zranění nepřítele

### 5.1.5 Stamina (výdrž)

Výdrž se využívá při běhu a skákání, ale musíme si dát pozor, jelikož každá akce může stát určitý

počet výdrže, a tak nás může v nějaké obtížné situaci znevýhodnit. Pokud hráč používá jen chůzi a nepoužívá skok, tím se nám výdrž opět doplní a můžeme ji znovu využívat (výdrž má také základní hodnotu 100). Pokud výdrž bude na 0, hráč musí počkat, než se doplní alespoň 25 %, aby mohl znovu běhat. Toto ale neplatí při skoku, pokud má hráč výdrž na skok, může kdykoliv vyskočit, pokud má menší, nebude moci skákat, a tak se musí počkat na doplnění.

## 5.2 Zbraně

Hráč má na výběr ze dvou zbraní (útočná puška a pistole). Každá zbraň má svoje hodnoty, které se dají jednoduše nastavit za pomocí skriptu.

### 5.2.1 Typy hodnot

- Poškození – nastaví minimum a maximum poškození do nepřítele
- Maximální počet nábojů – nastaví maximální počet nábojů
- Počet nábojů – zobrazí počet nábojů v aktuálním zásobníku
- Rychlost střelby – nastaví se počet střel za vteřinu (pokud je hodnota 1, vystřelí se 1 náboj, pokud je hodnota nastavena např. na 4, zbraň vystřelí každou střelu za 0.25 vteřiny)
- Rychlost přebíjení – nastaví se rychlost přebíjení v sekundách
- Typ zbraně – možnost zaškrtnout, jestli je zbraň manuální (nevystřelí více něž 1 náboj při držení levého tlačítka na myši)

Pomocí kategorie (útočná puška a pistole), lze nastavit podle ID, kterou zbraň v danou chvíli používáme. Pokud zadáme vstup pro změnu zbraně, změní se nám ID (0 nebo 1) a pomocí těchto ID nastavíme, jaký typ zbraně se nám zobrazí.

## 5.3 Vylepšení

V jednotlivých úrovních lze najít postavu, u které lze otevřít menu pro vylepšení. Uživatel uvidí, které vylepšení má koupené, a které si může koupit. K provedení nákupu hráč musí vlastnit určitý počet mincí, které lze získat zabíjením nepřátel.

### 5.3.1 Druhy vylepšení

- Primární zbraň – vylepšení pro útočnou pušku
- Sekundární zbraň – vylepšení pro pistoli
- Zvýšení maximálního zdraví – Zvýší se maximální zdraví hráče, tím je schopen vydržet větší poškození
- Zvýšení maximální staminy (výdrž) – Zvýší se maximální stamina hráče, tím je schopen déle běhat a skákat.
- Zvýšení regenerace staminy – Hráči, který je v klidném stavu (neběhá a neskáče), se bude doplňovat stamina rychleji.

## 5.4 Úkoly

Ve hře jsou různé typy úkolů. Pokud hráč splní první, odemkne se ihned druhý, který bude navazovat na příběh. Ve hře existují místa a předměty, které hráč musí sebrat, navštívit, ale, někdy je potřeba k dosažení cíle porazit několik nepřátel.

### 5.4.1 Typy úkolu

- Najdi daného člověka
- Zabij pár nepřátel
- Sběr předmětů
- Povídání si s přátelskou postavou

Pro dialogy je vytvořeno pole, do kterého se vloží věty a v kódu se každá věta rozdělí na písmena a podle daného času se budou v dialogu přidávat nová písmena, a tím vytvoříme pěknou animaci pro dialogy.

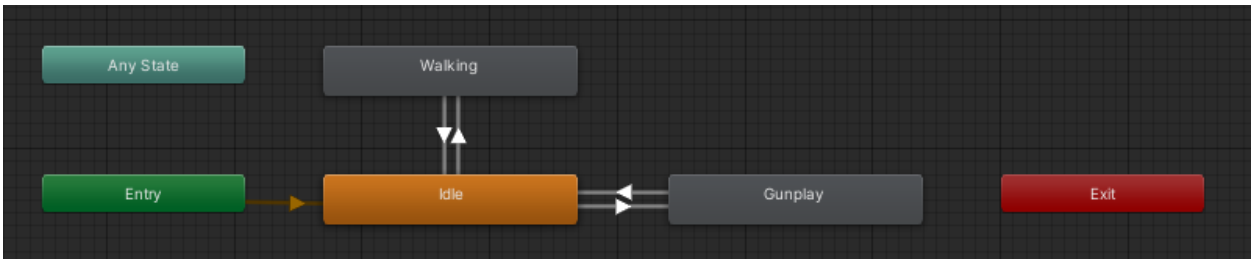
## 5.5 Nepřátelé

Základ nepřítele je jeho skript, který v metodě Update() hledá pozici hráče. Pro střelbu je využit

Raycast (dokážeme na určitou vzdálenost v jednom směru hledat co je před námi nebo před nepřítelem, v testovacím režimu můžeme vidět čáru, která určuje tento raycast). Pokud raycast nalezne nepřítele, kód se změní na režim střelby, tím se nepřítel zastaví a začne po hráčovi střílet. Nepřítel má nastaven „Error Margin (přesnost)“, kterou vynásobíme s celým kódem a vznikne nám nepřesnost nepřítele, jelikož nechceme, aby nepřítel měl 100 % přesnost. Pro pohyb po mapě je použit NavMesh, který nám umožní na celou mapu vložit místa, kde daní nepřátelé mohou chodit. Pokud nenastavíme tuto hodnotu, nemůžeme vytvořit AI, které by dokázalo chodit.

## 5.6 Animace

Pokud chceme použít animace, nemusíme je těžce programovat, slouží nám k tomu okno Animator, které se stará o všechny animace, které máme. Pokud máme všechny animace, můžeme je spojit k sobě dohromady jako na obrázku pod tímto textem, poté jen v kódu nastavíme, kdy a jak se nám budou animace prolínat.



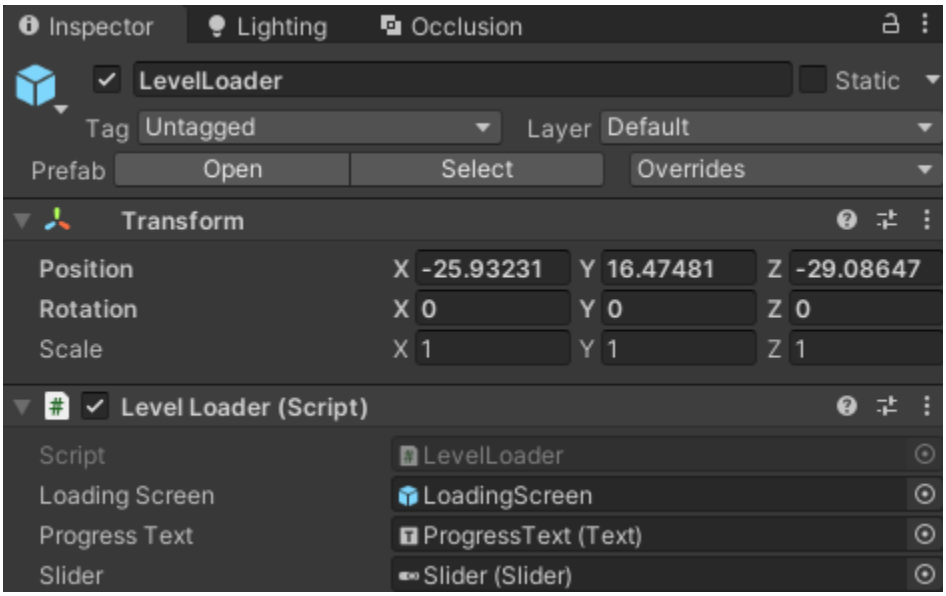

*Obrázek 10 Animace*

## 5.7 Systém hry

Aby hra správně fungovala, využívá se tzv. „Game Manager“, který se stará o to, aby se načetla správná úroveň, ukázal se „Loading Screen (načítací obrazovka), umístění hráče při načtení úrovně, aby se spustila správná hudba

### 5.7.1 Level Loader (manager pro načítání levelů)

Pokud zavoláme metodu na načtení levelu, ihned se nám spustí načítání. Abychom neměli černou obrazovku, nastavíme obrazovku pro načtení s ukazatelem, kolik procent máme načteno. Pokud budeme mít 100% načítací obrazovka se schová a objevíme se na nové načtené mapě.




*Obrázek 11 Level Loader*

### 5.7.2 Level Manager

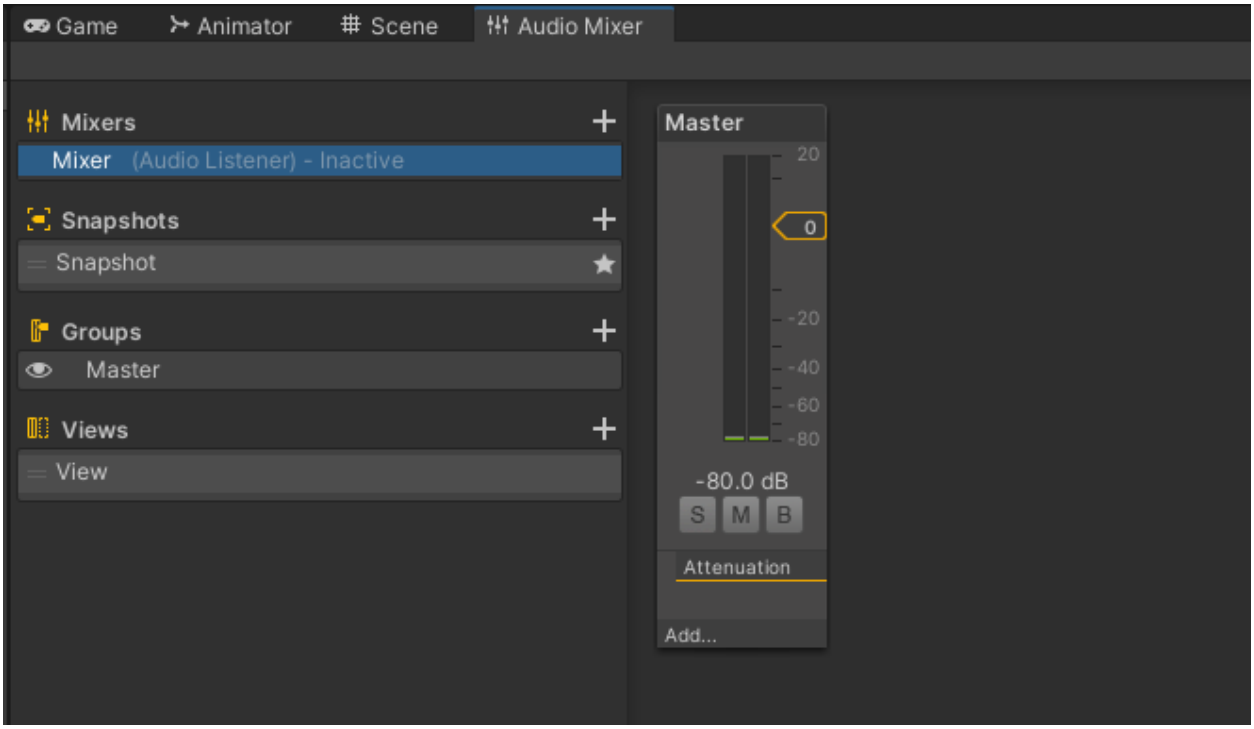
Pokud je tento manager v levelu, musíme nastavit startovací pozici hráče, tím dáme hře vědět, že má našeho hráče přemístit na určenou pozici a také nám to vyřeší problém s umístěním na nulté pozice (střed mapy, pozice v X (0), Y (0), Z (0)).

## 5.8 Uživatelské rozhraní pro hru

Aby se hra nenačetla přímo do levelu, kde již hrajeme za hráče, je přidána scéna s hlavním menu. Nachází se zde tlačítka, která mají svou jasně určenou funkci, dle popisku. V Unity si můžeme tlačítka, pozadí a různé funkce nastavit podle sebe. Také se dá u každého objektu nastavit zarovnání, tím uděláme prevenci, aby při změně rozlišení se naše uživatelské rozhraní nepokazilo a bylo na stejném místě. Ve hře si tlačítkem můžeme nastavit kvalitu, obraz a hlasitost.

# 6. ZVUKY

Nejdříve jsem si našel hudbu od interpreta Qillix, který mi sám zpřístupnil tuto hudbu pro tento projekt. Vytvořil jsem si mixer pro hudbu, který dokáže ovládat všechny určité zvuky v tomto mixeru najednou. Poté jsem si vytvořil skript na pouštění hudby a poté jsem jen v Level Manageru nastavil, jaký zvuk se má přehrát. Díky tomuto skriptu se zvuky nepřekrývají a pokud se pustí zvuk vícekrát najednou tak se nepřeruší.




*Obrázek 12 Audio Mixer*

# ZÁVĚR

V této práci jsem vytvořil hru za pomoci Unity Engine. Mým cílem bylo zhotovit hru svého výběru, tím, aby se mi hra líbila a vyzkoušel si, jak funguje vývoj her.

Nejdříve jsem začal s hráčem, jelikož jsem si řekl, že to je to nejdůležitější, co bych měl v první řadě udělat. Naprogramoval jsem pohyb hráče za použití komponenty „CharacterController“ a postupně přidával skok, kde jsem musel aplikovat gravitace, skrčení a střelbu, která má vlastní třídu, ve které lze nastavit hodnoty podle výběru. Po dokončení hráče jsem začal s tvorbou nepřátel, kteří budou hledat hráče, jaké zbraně budou mít, jaké poškození a jak velkou přesnost bude mít nepřítel. Pro nepřítele platí stejná třída pro zbraně jako pro hráče, ale pro přesnost má vlastní.

Po dokončení dvou hlavních objektů jsem začal vytvářet nějaké prostředí jako je město využitím assetů z Asset Storu a snažil jsem se zjistit, jak fungují osvětlení, abych mohl do temných uliček přidat světla a hlavně, aby hráč viděl, kam jde. Postupně jsem programoval úkoly, sbírání předmětů a přechod mezi levely.

Když jsem viděl, že mi hra funguje, snažil jsem se ji optimalizovat, přidal jsem nové modely pro nepřítele a postavy, které jsou využity k příběhu. Nakonec jsem dostal nápad vytvořit dialogy pro rozhovory s postavami. Toto byl největší problém, který se mi při programování vyskytl. Musel jsem vyřešit způsob zadávání vět a následné rozdělení určitých vět na písmena, aby byly dialogy plynulé a měly animaci.

Na úplný konec jsem zprovoznil hlavní menu a všechno uživatelské rozhraní, aby mohl uživatel z menu načíst level, uložit hru a popřípadě si nastavit kvalitu. Vše funguje pomocí tlačítek a menu je přehledné, tak aby uživatel nebyl zmatený.

Ve hře vzniklo vše, co by hra měla obsahovat (funkčnost, nepřátelé, příběh, uživatelské rozhraní, hudba). V aplikaci se nacházejí i chyby, ale nejsou takové, které by ovlivnily hru. Jedná se o optimalizaci, která již je, ale ne kompletní, rozdělení kódu do metod a lepší graficky zpracované uživatelské rozhraní.

Jsem rád, že jsem si mohl tento projekt vyzkoušet a naučit se, jak vytvořit hru. Určitě jsem se i naučil, jak používat různé programy, které byly potřeba při práci. Tuto práci nehodlám publikovat, jelikož je velmi krátká a nachází se zde plno funkcí, které by šli napsat jinak. Proto jsem ihned po dokončení začal pracovat na nové hře a doufám, že již bude publikovatelná a bude mít lépe napsané funkce.

# Citace

[1] Unity. *Unity* [online]. [cit. 2022-03-15]. Dostupné z: https://unity.com

[2] Graf popularity. *Indie Game Cloud* [online]. [cit. 2022-03-15]. Dostupné z: https://indiegamecloud.com/most-common-software-used-for-game-development/

[3] Render pipeline. Unity – Manual [online]. [cit. 2022-03-15]. Dostupné z: https://docs.unity3d.com/Manual/render-pipelines-overview.html

[4] Unity - SRP. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

[5] Unity - URP. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/universal-render-pipeline.html

[6] Unity - HDRP. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/high-definition-render-pipeline.html

[7] Unity - Metody. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

[8] Unity - Metoda Start. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html

[9] Unity - Metoda Awake. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html

[10] Unity - Metoda Update. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html

[11] Unity - Metoda FixedUpdate. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html

[12] Unity - Metoda LateUpdate. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html

[13] Unity - Hierarchy. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/Hierarchy.html

[14] Unity - Inspector. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/UsingTheInspector.html

[15] Unity - Project. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/ProjectView.html

[16] Unity - Console. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/Console.html

[17] Unity - Scene. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/UsingTheSceneView.html

[18] Unity - Game. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/GameView.html

[19] Unity - Lightning. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/lighting-window.html

[20] Unity - Lightning Scene. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/lighting-window.html#Scene

[21] Unity - Lightning Environment. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/lighting-window.html#Environment

[22] Unity - Lightning RealTime Lightmaps. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/lighting-window.html#RealtimeLightmaps

[23] Unity - Lightning Baked Lightmaps. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/lighting-window.html#BakedLightmaps

[24] Unity - Models. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/540/Documentation/Manual/FBXImporter-Model.html

[25] Unity - Mesh Renderer Component. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/class-MeshRenderer.html

[26] Unity - Skinned Mesh Renderer Component. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/class-SkinnedMeshRenderer.html

[27] Unity - Mesh Filter. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/class-MeshFilter.html

[28] Unity - Text Mesh. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://docs.unity3d.com/Manual/class-TextMesh.html

[29] Unity - Asset Store. *Unity* [online]. [cit. 2022-03-21]. Dostupné z: https://assetstore.unity.com

# Seznam obrázků