# Implementace 3D modelů ve vybraném enginu

Miroslav Zapletal

Bakalářská práce
2020



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

# Univerzita Tomáše Bati ve Zlíně
## Fakulta aplikované informatiky
### Ústav automatizace a řídicí techniky

Akademický rok: 2019/2020

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **Miroslav Zapletal** |
| Osobní číslo: | **A17207** |
| Studijní program: | **B3902 Inženýrská informatika** |
| Studijní obor: | **Informační a řídicí technologie** |
| Forma studia: | **Kombinovaná** |
| Téma práce: | **Implementace 3D modelů ve vybraném enginu** |

## Zásady pro vypracování

1. Vypracujte literární rešerši na téma 3D grafika a modelování.
2. Seznamte se s prostředím programů na 3D modelování a s vývojovým prostředím 3D enginů. Popište jejich vlastnosti a základní práce v jejich aktuálních verzích.
3. Teoreticky prostudujte možné způsoby převodů 3D modelů z 3D programů do 3D enginů.
4. Vytvořte komplexní 3D model ve vybraném 3D modelovacím programu.
5. Vytvořený model převeďte do vhodného 3D enginu. Navrhněte nejoptimálnější formáty pro tento převod.
6. V prostředí tohoto 3D enginu vytvořte jednoduchou vizualizační aplikaci s tímto modelem.

Rozsah bakalářské práce:
Rozsah příloh:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:


1. BLAIN, John M. The complete guide to Blender graphics: computer modeling & animation. Fifth edition. Boca Raton: Taylor & Francis, a CRC title, part of the Taylor & Francis imprint, a member of the Taylor & Francis Group, the academic division of T&F Informa, 2019. ISBN 9780367184742.
2. POKORNÝ, Pavel. Blender: naučte se 3D grafiku. 2., aktualiz. a rozš. vyd. Praha: BEN – technická literatura, 2009. ISBN 978-80-7300-244-2.
3. SHANNON, Tom. Unreal Engine 4 for design visualization: developing stunning interactive visualizations, animations, and renderings. Boston: Addison-Wesley, [2018]. ISBN 9780134680705.
4. Blender 2.80 Reference Manual [online]. [cit. 2019-11-20]. Dostupné z: https://docs.blender.org/manual/
5. Unreal Engine 4 Documentation [online]. [cit. 2019-11-20]. Dostupné z: https://docs.unrealengine.com/


Vedoucí bakalářské práce: **Ing. Pavel Pokorný, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **20. prosince 2019**
Termín odevzdání bakalářské práce: **15. května 2020**


**doc. Mgr. Milan Adámek, Ph.D.**
děkan

**prof. Ing. Vladimír Vašek, CSc.**
ředitel ústavu

Ve Zlíně dne 20. prosince 2019

**Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 6. 8. 2020

Miroslav Zapletal, v.r.
…….
podpis diplomanta

## ABSTRAKT

Tato práce se zabývá implementací 3D modelů ve vybraném enginu. Cílem je vytvořit komplexní 3D model a vyhodnotit, jaké formáty jsou nejoptimálnější pro konverzi do enginu. Teoretická část představí problematiku 3D grafiky, 3D modelovacích programů a 3D enginů a možné způsoby převodů mezi nimi. V praktické části je vytvořeno prostředí s dílčími 3D modely, včetně kvadrokoptéry se zabudovanými PID regulátory. Pro implementaci do enginu jsou použity konverzní formáty vyhodnocené jako nejvhodnější.



Klíčová slova: 3D grafika, modelování, 3D enginy, konverzní formáty, PID regulátory



## ABSTRACT

This thesis is focused on implementation of 3D models in a selected engine. The goal is to create a complex 3D model and evaluate which formats are most optimal for conversion to the engine. The theoretical part introduces the topics of 3D graphics, 3D modelling programs and 3D engines and possible ways of conversion between them. In the practical part, an environment with partial 3D models is created, including a quadcopter with built-in PID controllers. Conversion formats evaluated as the most suitable are used for implementation in the engine.



Keywords: 3D graphics, modelling, 3D engines, conversion formats, PID controller

Tímto bych chtěl vyjádřit poděkování vedoucímu své bakalářské práce panu Ing. Pavlu Pokornému, Ph.D., za spolupráci a za cenné připomínky a rady k obsahu práce. Dále také své rodině za jejich podporu nejen v průběhu vypracovávání této práce, ale především po celou dobu studia.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

# OBSAH


**ÚVOD** ..... **10**

**I TEORETICKÁ ČÁST** ..... **11**

**1 POČÍTAČOVÁ GRAFIKA A MODELOVÁNÍ** ..... **12**

1.1 2D GRAFIKA ..... 12

1.2 3D GRAFIKA ..... 13

1.2.1 Modelování ..... 13

1.2.1.1 Souřadnicový systém ..... 14

1.2.1.2 Geometrická transformace ..... 14

1.2.1.3 Vzhled objektu ..... 14

1.2.2 Nastavení scény ..... 15

1.2.2.1 Keyframing ..... 15

1.2.2.2 Osvětlení ..... 15

1.2.3 Renderování ..... 15

1.2.3.1 Ray tracing ..... 16

1.2.3.2 Scanline ..... 17

1.2.3.3 Radiosita ..... 17

1.3 HARDWAROVÁ REALIZACE POČÍTAČOVÉ GRAFIKY ..... 17

**2 3D MODELOVACÍ PROGRAMY** ..... **18**

2.1 NEJROZŠÍŘENĚJŠÍ 3D PROGRAMY PRO TVORBU MODELŮ ..... 18

2.1.1 Blender ..... 18

2.1.2 Autodesk Maya ..... 18

2.1.3 Autodesk 3ds Max ..... 19

2.1.4 Cinema 4D ..... 20

2.2 POPIS PROSTŘEDÍ VYBRANÉHO 3D PROGRAMU ..... 21

2.2.1 Uživatelské prostředí ..... 21

2.2.2 Modelování ..... 23

2.2.3 Renderovací enginy v Blenderu ..... 25

2.2.3.1 Eevee ..... 25

2.2.3.2 Cycles ..... 25

2.2.3.3 Workbench ..... 25

**3 3D ENGINY** ..... **26**

3.1 ZÁKLADNÍ FUNKCE 3D ENGINŮ ..... 26

3.1.1 Renderování a grafický engine ..... 26

3.1.2 Herní rozhraní ..... 27

3.1.3 Animace ..... 28

3.1.4 Skriptování ..... 28

3.1.5 Audio ..... 28

3.1.6 Umělá inteligence ..... 29

3.1.7 Fyzikální engine ..... 29

3.2 NEJROZŠÍŘENĚJŠÍ 3D ENGINY ..... 30

3.2.1 Unity 3D ..... 30

3.2.2 Unreal Engine ..... 30

3.2.3 CryEngine ..... 31

3.3 Popis prostředí vybraného 3D enginu ... 32
3.3.1 Pracovní prostředí Unreal Enginu ... 32
3.3.2 Blueprints ... 34
3.3.3 Vizuál ... 34
**4 KONVERZE MODELU Z 3D PROGRAMU DO 3D ENGINU ... 36**
4.1 FBX (.fbx) ... 36
4.2 Wavefront OBJ (.obj) ... 36
4.3 Alembic (.abc) ... 36
4.4 glTF 2.0 (.glb/.gltf) ... 37
4.5 Addon „Blender for Unreal Engine“ ... 37
4.6 Universal Scene Description (.usd) ... 37
**5 ŘÍZENÍ 3D MODELU ... 38**
**II PRAKTICKÁ ČÁST ... 42**
**6 VYTVOŘENÍ 3D MODELU ... 43**
6.1 Statické mesh objekty ... 43
6.1.1 Model automobilu ... 44
6.1.2 Model kvadrokoptéry ... 44
6.1.3 Částicové systémy ... 44
6.2 Materiály a textury ... 45
6.3 UV mapping ... 45
6.4 Animace postavy (Rigging) ... 46
6.5 Rigid body animace ... 47
6.6 Fluid simulace ... 47
**7 KONVERZNÍ FORMÁTY ... 49**
7.1 FBX (*.fbx) ... 49
7.2 Alembic (*.abc) ... 50
7.3 Wavefront (*.obj) ... 50
7.4 glTF 2.0 (*.glb/*.gltf) ... 51
7.5 Blender for UnrealEngine 4 ... 52
7.6 USD ... 52
7.7 Výsledky testování ... 52
**8 IMPLEMENTACE MODELU V UNREAL ENGINU ... 54**
8.1 Kvadrokoptéra s PID regulátory ... 54
8.1.1 Integrace 3D modelu ... 55
8.1.2 Ovládání kvadrokoptéry ... 55
8.1.3 Realizace PID regulátorů ... 58
8.2 Import prostředí ... 60
8.2.1 Statické objekty ... 60
8.2.2 UV mapa, materiály a textury ... 61
8.2.3 Částicové systémy ... 61
8.2.4 Rigging postavy ... 62
8.2.5 Rigid body animace ... 63
8.2.6 Fluid simulace ... 63

**ZÁVĚR** ..... **65**
**SEZNAM POUŽITÉ LITERATURY** ..... **66**
**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK** ..... **70**
**SEZNAM OBRÁZKŮ** ..... **71**
**SEZNAM TABULEK** ..... **72**
**SEZNAM PŘÍLOH** ..... **73**

# ÚVOD

S rozvojem moderních technologií se stala počítačová 3D grafika nedílnou součástí běžného života. Snížením výpočetního času modelovacích a vykreslovacích procesů se stala přístupnějším prostředkem nejen odborným grafickým studiím, ale i běžným uživatelům s přístupem k výkonnějším technologiím. Trojrozměrná grafika je v současnosti neodmyslitelnou součástí několika odvětví. Ať už se jedná o architekturu a stavebnictví, kde slouží k vizualizaci budov, nebo filmový a videoherní průmysl. Její výskyt v oblasti klasického marketingu je čím dál častější, a to většinou v těch případech, kdy je problém rozlišit skutečnost od grafické simulace. Záměrem této práce je využít současných trendů 3D grafiky a nastínit její použití i pro potřeby informačních a řídící technologií. Vizualizace postupů může přispět k pochopení složitých automatizačních procesů řízení, či propojit klasické programovací části s vizualizační simulací.

Cílem práce je vytvořit komplexní 3D model a vyhodnotit možné způsoby jeho převodu do 3D enginu. V této práci se vybere jeden typ 3D modelovacího programu, ve kterém bude vytvořen komplexní 3D model prostředí. Dále bude vybrán jeden typ 3D enginu a popsán způsob jeho komunikace s modelovacím programem pomocí konverzních formátů. Následně bude model implementován v grafickém enginu pro jeho použití při simulacích.

Úvod teoretické části této práce se věnuje literární rešerši na téma problematiky 3D grafiky a jejího začlenění do moderního grafického oboru. Ústředním bodem je popis v současnosti nejběžněji používaných 3D modelovacích programů a enginů a výběr konkrétních programů pro použití v praktické části. Na závěr teoretické části je popsán způsob implementace modelů ve vybraném enginu pomocí konverzních formátů a teorie automatizace pro návrh řízení modelů pomocí regulační soustavy.

Praktická část práce je zaměřena na proces tvoření modelů a simulací ve vybraném modelovacím programu a na následné testování, jakým způsobem různé konverzní formáty převádějí vytvořené modely. Nakonec dojde k implementaci těchto modelů do vybraného grafického enginu pomocí nejoptimálnějších formátů. V prostředí enginu bude vytvořena jednoduchá vizualizační aplikace s řiditelným modelem s využitím PID regulátorů, který se bude pohybovat v konvertovaném prostředí. Očekávaným výsledkem této práce by měl být seznam doporučení, jakým způsobem a s využitím jakých formátů lze nejvhodněji převést vytvořený 3D model z modelovacího programu do prostředí enginu, kde má sloužit k praktické simulaci.

# I. TEORETICKÁ ČÁST

# 1 POČÍTAČOVÁ GRAFIKA A MODELOVÁNÍ

Pojem „počítačová grafika“ zahrnuje vše, co se podílí na tvorbě nebo manipulaci s obrázky v počítači, včetně animovaných obrázků. Tato práce se zaměřuje především na trojrozměrnou, neboli 3D grafiku, jejímž hlavním produktem je vytvořený 3D model. Avšak konečným výsledkem projektu počítačové grafiky bývá dvourozměrný obraz, respektive snímek vytvořeného prostředí. Samotná tvorba 2D obrazů a manipulace s nimi jsou tudíž také důležitými tématy, které nelze opomenout.

## 1.1 2D grafika

Každý 2D obrázek vyobrazený na počítačové obrazovce je složen z pixelů. Obrazovka se skládá z obdélníkové mřížky pixelů uspořádaných v řádcích a sloupcích. Pixely jsou zároveň dostatečně malé, aby je nebylo možné vidět jednotlivě, ale jako spojený celek. Ve skutečnosti se pro mnoho displejů s vysokým rozlišením stávají v podstatě neviditelnými. V daném čase může každý pixel zobrazovat pouze jednu barvu. Většina obrazovek v dnešní době používá 24bitovou barevnou hloubku, přičemž barvu lze specifikovat třemi 8bitovými čísly, které udávají intenzitu červené, zelené a modré barvy (RGB). Jakákoli barva, která může být zobrazena na obrazovce, je tvořena určitou kombinací těchto tří „primárních“ barev. Hodnoty barev pro všechny pixely na obrazovce jsou vždy uloženy ve vyrovnávací paměti známé jako „framebuffer“. Úprava obrázku na obrazovce vyžaduje změnu hodnot jeho barev uložených právě v této vyrovnávací paměti. Počítačový obraz, který funguje na tomto principu, je základním modelem rastrové grafiky. [1]

Další možností pro skládání počítačového obrazu je vektorová grafika a ta je zcela odlišná od grafiky rastrové. Zatímco rastrová grafika rozdělí každý obrázek do mřížky s barvami, vektorová grafika je rozdělí na popisy každého dvourozměrného tvaru na obrázku. Díky této informaci může počítač vyprodukovat rastrový obrázek věrným nakreslením každého tvaru na základě jeho uloženého popisu. Skutečnou silou vektorové grafiky je však to, že počítač může použít stejné informace k vytvoření rastrového obrazu libovolné velikosti bez pixelace. Znalost přesných tvarů, které tvoří obraz, umožňuje počítači kreslit jej v jakékoliv velikosti a úrovni detailů. Vektorová grafika tak může být zvětšena nebo zmenšena bez ztráty detailů. [2]

Obraz, který lze specifikovat přiměřeným počtem geometrických tvarů, obsahuje mnohem menší množství informací potřebných ke zobrazení obrázku, než pokud se rozhodneme pro

použití rastrového zobrazení. Pokud zobrazujeme obrázek vytvořený z tisíce řádků, pro vektorové zobrazení tohoto obrázku stačí uložit souřadnice dvou tisíc bodů, respektive úseček, což by zabralo jen několik kilobajtů paměti. Uložení obrazu do framebufferu pro rastrové zobrazení by vyžadovalo více paměti. Podobně by vektorový displej mohl nakreslit čáry na obrazovce rychleji, než by rastrový displej mohl zkopírovat stejný obrázek z vyrovnávací paměti na obrazovku. [1]

## 1.2 3D grafika

Když se zaměříme na 3D grafiku, zjistíme, že nejběžnější přístupy k ní mají více společného s vektorovou grafikou než s grafikou rastrovou. To znamená, že obsah obrázku je nejčastěji specifikován jako seznam geometrických objektů. Techniku, která se ve 3D grafice využívá, označujeme jako geometrické modelování. Výchozím bodem je v této technice sestrojený „umělý 3D svět“ jako soubor jednoduchých geometrických tvarů uspořádaných v trojrozměrném prostoru. Objekty mohou být charakterizovány pomocí atributů, které určují vzhled objektů na základě přirozených charakteristik v reálném světě. Rozsah základních tvarů je často velmi omezený, zahrnuje spíše pouze body, úsečky a trojúhelníky. Složitější tvary, které nejsou definovány jako základní, například mnohoúhelník nebo koule, mohou být vytvořeny, nebo aproximovány, jako seskupení několika základních tvarů. Abychom vytvořili dvourozměrný obraz vymodelovaného prostředí, je třeba jej promítnout ze tří dimenzí do dvou. Taková projekce je ekvivalentem fotografování. [1]

Proces vytváření 3D počítačové grafiky lze postupně rozdělit do tří základních po sobě jdoucích fází:

- Modelování
- Nastavení scény
- Renderování

### 1.2.1 Modelování

Vytvoření 3D scény vyžaduje alespoň tři klíčové komponenty: modely, materiály a světla. Modelování je zjednodušeně umění a věda o vytvoření povrchu, který buď napodobuje tvar objektu skutečného světa, nebo vyjadřuje představivost abstraktních objektů. Procesy modelování zahrnují mimo jiné úpravy vlastností povrchu objektu, nebo materiálu (např. barev, jasu, difúzních a zrcadlových složek stínování – běžně nazývaných drsnost a lesk, charakteristika odrazu, průhlednost, či index lomu), přidávání textur, úpravu hrbolatosti a úpravu

dalších prvků. Modelování může také zahrnovat různé činnosti související s přípravou 3D modelu pro animaci (i když v modelu s komplexními vlastnostmi jsou tyto činnosti vyčleněny do samostatné fáze známé jako „rigging“). Objekty mohou být opatřeny kostrou, což je hlavní rámec objektu se schopností ovlivňovat tvar nebo pohyby tohoto objektu. Tato kostra zefektivňuje proces animace tím, že její manipulací lze automaticky ovlivnit odpovídající části modelu. [3]

#### *1.2.1.1 Souřadnicový systém*

Modelovací prostor potřebuje souřadnicový systém, který spojuje každý bod v prostoru se třemi hodnotami, obvykle označovanými jako souřadnice X, Y a Z. Tento souřadnicový systém se označuje jako „světové souřadnice“ a jeho cílem je vytvořit prostředí uvnitř umělého světa tvořené geometrickými objekty. Jeho prostřednictvím můžeme definovat úsečku zadáním souřadnic jejích dvou koncových bodů, nebo trojúhelník zadáním souřadnic jeho tří vrcholů. [4]

#### *1.2.1.2 Geometrická transformace*

Pro úpravu jednotlivých 3D objektů při jejich modelování se využívá geometrická transformace. Tři nejzákladnější druhy geometrické transformace se nazývají škálová, rotační a translační. Škálová transformace se používá k nastavení velikosti objektu, tj. k jeho zvětšení, nebo zmenšení o určitý specifikovaný faktor. Rotační transformace se používá k nastavení orientace objektu jeho otočením o určitý úhel kolem konkrétní osy. Translační transformace se používá k nastavení pozice objektu jeho přemístěním z původní polohy o určitou vzdálenost. [4]

#### *1.2.1.3 Vzhled objektu*

Úprava vzhledu geometrických tvarů se provádí přiřazením jednotlivých atributů, např. barvy. Ve 3D grafice se místo barvy obvykle mluví spíše o materiálu. Termín materiál zde chápeme jako vlastnosti skutečného vizuálního vzhledu povrchu, tedy v podstatě způsob, jak daný povrch ovlivňuje světlo, které na něj dopadá. Vlastnosti materiálu mohou zahrnovat základní barvu nebo další vlastnosti, jako je lesk, drsnost a průhlednost. Jednou z nejužitečnějších vlastností materiálu je textura. Obecně řečeno, textura je způsob, jak mezi jednotlivými body měnit vlastnosti materiálu na povrchu. Nejběžnějším použitím textury je přiřazení různých barev pro různé body materiálu. To se provádí pomocí 2D obrazu, jakožto textury, kterou lze aplikovat na povrch tak, aby obraz vypadal, že je na povrch namalovaný.

Textura se však může vztahovat také na měnící se hodnoty průhlednosti, či například hrbolatosti. Materiál je skutečná vlastnost objektu, ale skutečný vzhled objektu závisí také na prostředí, ve kterém je objekt zobrazen. [1]

### 1.2.2 Nastavení scény

Další fází tvorby 3D počítačové grafiky je nastavení scény. Nastavení scény zahrnuje uspořádání virtuálních objektů, světel, kamer a dalších subjektů v prostředí, které budou později použity k vytvoření statického obrázku nebo animace.

#### *1.2.2.1 Keyframing*

Pokud je kýženým výsledkem 3D grafického projektu animace, využívá se v této fázi obvykle technika zvaná „keyframing“, která usnadňuje vytváření komplikovaného efektu pohybu. Místo toho, aby bylo nutné v každé animaci animovat geometrické transformace objektu pro každý snímek zvlášť, stačí díky keyframingu pouze nastavit některé klíčové snímky, mezi nimiž je následně interpolováno nastavení ostatních snímků. [3]

#### *1.2.2.2 Osvětlení*

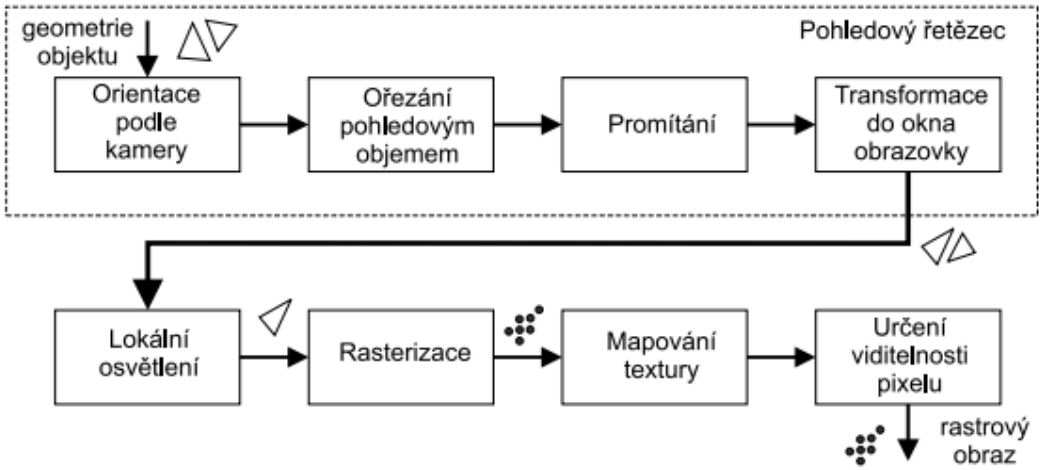
Důležitým aspektem nastavení scény je osvětlení. Ve skutečném světě nic nenajdete, pokud v prostředí není žádné světlo a totéž platí i pro 3D grafiku, kde je zapotřebí scénu opatřit simulovaným osvětlením. V jednom prostředí může zároveň být několik zdrojů světla. Každý světelný zdroj může mít svou vlastní barvu, intenzitu, směr či polohu. Světlo z těchto zdrojů pak bude interagovat s materiálovými vlastnostmi objektů. Podpora osvětlení v grafických systémech se může lišit, od docela jednoduché podpory po podporu velmi složitou a výpočetně náročnou. [1]

### 1.2.3 Renderování

Důležitou úlohou počítačové grafiky je zobrazování prostorových modelů a scén, které jsou uloženy v paměti počítače. Vlastní převod trojrozměrné informace do 2D podoby se v anglické literatuře nazývá „rendering“, v češtině se používá většinou pojem vykreslování. Zobrazování scény, která obsahuje tělesa a zdroje světla, můžeme zjednodušeně charakterizovat jako postupné řešení následujících dílčích úloh:

- zobrazení globálního osvětlení scény, které je závislé především na zdrojích světla a na optických vlastnostech (například odrazu) prostředí a objektů

- nastavení kamery (pohled na scénu z místa pozorovatele) a řešení promítací úlohy, společně s částečným nebo úplným řešením viditelnosti,
- výsledné vytvoření rastrového 2D obrazu, včetně řešení lokálních osvětlovacích modelů, zobrazení nastavených textur a viditelnosti. [5]




Obrázek 1: Schéma klasického zobrazovacího řetězce [5]

Existuje mnoho nástrojů a technik na vykreslení výsledného 3D obrázku. Některé algoritmy jsou vytvořeny na výpočet co nejrychlejšího výsledného obrazu, jiné dávají důraz na fyzikální přesnost. V dnešní době většina 3D programů obsahuje svůj vlastní renderovací engine, případně umožňují instalaci a volbu mezi několika externími vykreslovacími algoritmy.

#### *1.2.3.1 Ray tracing*

Ray tracing, v překladu sledování paprsků, představuje „globální zobrazovací metodu, která vychází ze zpětného sledování paprsku, který je vržen v každém bodu zorného pole kamery. Při této metodě renderování se bere vzájemná interakce objektů. Velice věrohodně se tímto způsobem dají zobrazit stínování, průhlednost nebo zrcadlové odrazy. Výsledkem jsou tedy scény, jejichž vzhled vypadá velice realisticky. Bohužel jde o časově náročnou metodu, neboť je v ní potřeba provádět spoustu matematických výpočtů.“ [6]

#### *1.2.3.2 Scanline*

Scanline renderování se používá při vysokých nárocích na rychlost. Namísto vykreslování pixelu po pixelu využívá totiž tato metoda výpočet po polygonech. Díky tomu se tato technika využívá pro renderování v reálném čase a interaktivní grafiku. [7]

#### *1.2.3.3 Radiosita*

Radiosita je další globální zobrazovací metoda, která se používá při simulování přirozeného osvětlení objektů scény ve snaze co nejvíce se přiblížit podobě reálného světa. Metoda radiosity odstraňuje nedokonalost fotorealistické věrnosti, která se objevuje za použití samotné metody ray tracingu. Na rozdíl od ray tracingu totiž nepředpokládá pouze bodové zdroje světla, ale vychází ze zákona chování energie a předpokládá, že přenos světla probíhá v energeticky uzavřené scéně a světelné záření se odráží od ploch difúzně. Radiosita proto slouží jakožto doplněk ray tracingu pro zdokonalení světelné simulace. [6]

## 1.3 Hardwarová realizace počítačové grafiky

V prvních stolních počítačích byl obsah obrazovky spravován přímo procesorem. Například pro nakreslení úsečky na obrazovce spustil CPU smyčku pro nastavení barvy každého pixelu, který leží podél úsečky. Grafika zabírala hodně času CPU a grafický výkon byl ve srovnání s tím, na jaký jsme zvyklí dnes, velmi pomalý. V porovnání s tím jsou dnes počítače samozřejmě mnohem rychlejší, avšak velkou změnou je, že v moderních počítačích se grafické zpracování provádí pomocí specializované komponenty nazývané GPU, neboli "Graphics Processing Unit". GPU může obsahovat velké množství procesorů pro provádění výpočtů, které pracují souběžně s velmi rychlými nadgrafickými operacemi. Obsahuje také vlastní vyhrazenou paměť pro ukládání informací, jako jsou obrázky a seznamy souřadnic. Procesory GPU mají velmi rychlý přístup k datům uloženým v paměti GPU, který je mnohem rychlejší než jejich přístup k datům uloženým v hlavní paměti počítače. Sada příkazů, kterým GPU rozumí, vytváří API GPU. OpenGL je příkladem grafického rozhraní API a většina GPU OpenGL podporuje v tom smyslu, že rozumí jejím příkazům, nebo že alespoň tyto příkazy dokáže efektivně převést do příkazů, kterým GPU rozumí. Nejznámější alternativou k OpenGL je pravděpodobně Direct3D, 3D grafické API rozhraní používané pro Microsoft Windows. [1]

# 2 3D MODELOVACÍ PROGRAMY

3D modelovací modely vytvářejí objekt ve třech rozměrech. Jsou považovány za specializovaný software, který je schopen vytvořit matematické znázornění povrchů, objektů, scén či animací ve 3D prostoru a zároveň může zobrazovat 3D modely jako „normální“ 2D obrazy. Aplikace 3D modelů se používá téměř v každém odvětví, protože naše představivost je svým způsobem omezená. Například zábavní průmysl využívá 3D modelovací software k vytváření postav a celých filmových sekvencí. Na trhu je k dispozici mnoho modelovacích programů, ovšem každý se může lišit svými funkcemi a primárním zaměřením, ať už se jedná o fotorealistickou grafiku, architekturu, animaci či 3D tisk. Je nutné proto dbát na zvolení toho správného programu.

## 2.1 Nejrozšířenější 3D programy pro tvorbu modelů

### 2.1.1 Blender

Blender je program pro 3D počítačovou grafiku s nástroji pro modelování a animaci objektů, postav a vytváření rozsáhlých scén. Ty mohou být vytvořeny jako statické, nebo lze použít animované sekvence pro tvorbu videa. Modely a scény mají vyladěnou barvu a strukturu, aby vytvářely dokonale realistické efekty. Výsledná prostředí a videa mohou sloužit k uměleckým účelům, nebo jako architektonické či vědecké prezentace. V programu jsou k dispozici také nástroje pro tvorbu 2D animací. Individuální modely lze použít pro 3D tisk. [8]

Blender je spravován Blender Foundation a vydáván jako Open Source Software, který je k dispozici ke stažení a zdarma k použití pro jakékoli účely. Aktuální verzi programu 2.82a lze bezplatně stáhnout na oficiálních stránkách programu (www.blender.org), kde je k dispozici i kompletní dokumentace a návody pro začátečníky. V roce 2019 si tento program stáhlo téměř 10 milionů uživatelů. [8] [9]

Po celém světě se vyskytující uživatelská základna tvoří v podstatě jádro podpůrných mechanismů programu, ať už se jedná o tvorbu rozšiřujících nástrojů, pluginů či jen o videotutoriály pro začínající grafiky. I díky této komunitě je Blender podporovaný většinou operačních systémů a je jedním z nejrozšířenějších programů pro tvorbu 3D grafiky.

### 2.1.2 Autodesk Maya

Autodesk Maya je přední softwarová aplikace pro 3D animace vyvinutá společností Autodesk. Maya umožňuje profesionálům v oblasti tvorby videa, kteří pracují s animovanými

filmy, televizními programy, vizuálními efekty a videohrami, vytvářet vysoce profesionální trojrozměrné filmové animace. Před existencí 2D a 3D animačních softwarů byly používány nástroje pro ruční animaci, jako jsou kreslicí papír a tužky, gumy, barvy a štětce, světelné stoly a průhledné fólie. Ty byly pouze podmnožinou toho, co lze nyní dělat s programy, jako je Maya. [10]

Maya 1.0 byla původně vyvinuta a vydána v roce 1998 společností Alias Wavefront a o sedm let později v roce 2005 byla odkoupena společností Autodesk, Inc. a přejmenována na „Autodesk Maya“. Od svého prvního vydání se Maya stala oblíbenou ve filmovém průmyslu, kde se široce používá k tvorbě grafiky pro filmy oceněnými cenou filmové akademie (Oscary), jako jsou Rango a Hugo. Stála také za tvorbou vizuálních efektů ve známé sáze Harry Potter či ve filmu Matrix. Stále častěji se tento software používá také ve videoherním v průmyslu k vytváření vizuálních efektů. Maya využívá "Maya Embedded Language", neboli MEL, a skriptování v Pythonu, které umožňuje prostřednictvím otevřené architektury programovat komplikované nebo opakující se příkazy. Tyto naprogramované příkazy pomáhají šetřit drahocenný čas a zároveň nabízejí možnost jejich sdílení s jinými uživateli, kteří by je mohli považovat za užitečné. Ve filmovém a televizním průmyslu je Maya de facto standardem pro 3D vizuální efekty, počítačovou grafiku a animaci postav. [10]

### 2.1.3 Autodesk 3ds Max

Další populární volbou od společnosti Autodesk je 3ds Max. 3ds Max je 3D modelovací, animační a renderovací software používaný hlavně pro animaci, modelování, tvorbu videoher, vizuální 3D efekty a architektonickou vizualizaci. Nejčastěji je využíván architekty a inženýry z důvodu jeho schopnosti interagovat s programem AutoCad a dalšími nástroji Autodesk. Pokud jde o modelování, 3ds Max je bezkonkurenční v rychlosti a jednoduchosti. Tento software dokáže zvládnout několik fází animačního procesu, včetně předvizualizace, rozvržení, kamer, modelování, texturování, animace, VFX, osvětlení a vykreslování. [9]

3ds Max je dodáván se dvěma subsystémy pro animované modely postav – CAT a character studio. Každý je plně přizpůsobitelný s celou řadou aplikací. Oba jsou kompatibilní s formáty souborů pro snímání pohybu („motion-capture“) a společně poskytují mocný prostředek k animaci složitých a podrobných scén. 3ds Max je často srovnáván s programem Maya od stejné společnosti. Zatímco Maya je obecně výkonnější ve většině oblastí, 3ds Max je snadnější, a proto se běžně používá k výuce 3D grafiky pro začátečníky. [12]

### 2.1.4 Cinema 4D

Cinema 4D je 3D modelovací, animační a renderovací aplikace vyvinutá společností Maxon a je skvělou aplikací pro pohybovou grafiku, modelování a texturování. Zatímco většina velkých společností má tendenci používat Maya a 3ds Max, Cinema 4D je obvykle nejlepší pro jednotlivé umělce nebo malé týmy. Cinema 4D je považována za jednu z nejintuitivnějších a uživatelsky nejpříjemnějších 3D aplikací. Některé z výhod, které nabízí Cinema 4D jsou „MoGraph" a „Sketch and Toon" umožňující Cinema 4D dělat věci, které jiné programy nemohou, bez velkého skriptování. „MoGraph" je skvělá funkce, která přispívá k rychlému a snadnému pracovnímu postupu. Umožňuje uživateli klonovat řadu objektů, vytvářet vytlačovaný text, přidávat efekty, vytvářet pohyby a další. „Skica a Toon" jsou nástroje pro stínování, karikatury a technické výkresy. [11]

Tabulka 1: Srovnání nejrozšířenějších 3D modelovacích programů [13] (vlastní zpracování)

| | Blender | Maya | 3ds Max | Cinema 4D |
|---|---|---|---|---|
| **Společnost** | Open-source | Autodesk | Autodesk | MAXON |
| **Rok vzniku** | 2003 | 1998 | 1990 | 1990 |
| **Klíčové vlastnosti** | Modifikátor vzhledu, Klávesové zkratka, rozšíření | Zpracování pohybu, „nurbs" modelování[1], vrstvy | Skriptovací jazyk, edit poly modifikátor, spline křivky[2] | Procedurální a polygonální modelování, rigging[3], přívětivé rozhraní |
| **Nejlepší využití** | Animace, nezávislé projekty | Animace a vizuální efekty | Modelování a animace postav | Studio modelování a animace |
| **Náročnost** | Náročná | Průměrná | Jednoduchá | Jednoduchá |
| **Cena** | Zdarma | 1,470 $ ročně (Trial verze zdrama) | 1,470 $ ročně (Trial verze zdrama) | 700 $ licenčně |

[1] „Nurbs modelování" je technika modelování, která využívá spline křivky, vyhlazuje hrany mnohoúhelníků
[2] „Edit poly modifier" je soubor nástrojů, které slouží k úpravám mnohoúhelníků. „Spline" je aproximační křivka, která umožňuje zjemnění povrchu.
[3] „Rigging" dále viz kapitola 1.2.1

## 2.2 Popis prostředí vybraného 3D programu

Pro využití v praktické části byl zvolen program Blender, a to pro jeho výhody, které jsou zmíněny v následujících bodech:

- Licence, která umožnuje využívat program zcela zdarma
- Podpora práce s klávesovými zkratkami
- Real-time renderování pomocí renderovacího enginu Eevee[4]
- Možnost rozšíření programu o již vytvořené pluginy
- Velká uživatelská komunita

Nevýhodou využití programu Blender může být ze začátku složitější užívání programu, kvůli neobvyklému ovládání a uživatelskému rozhraní.

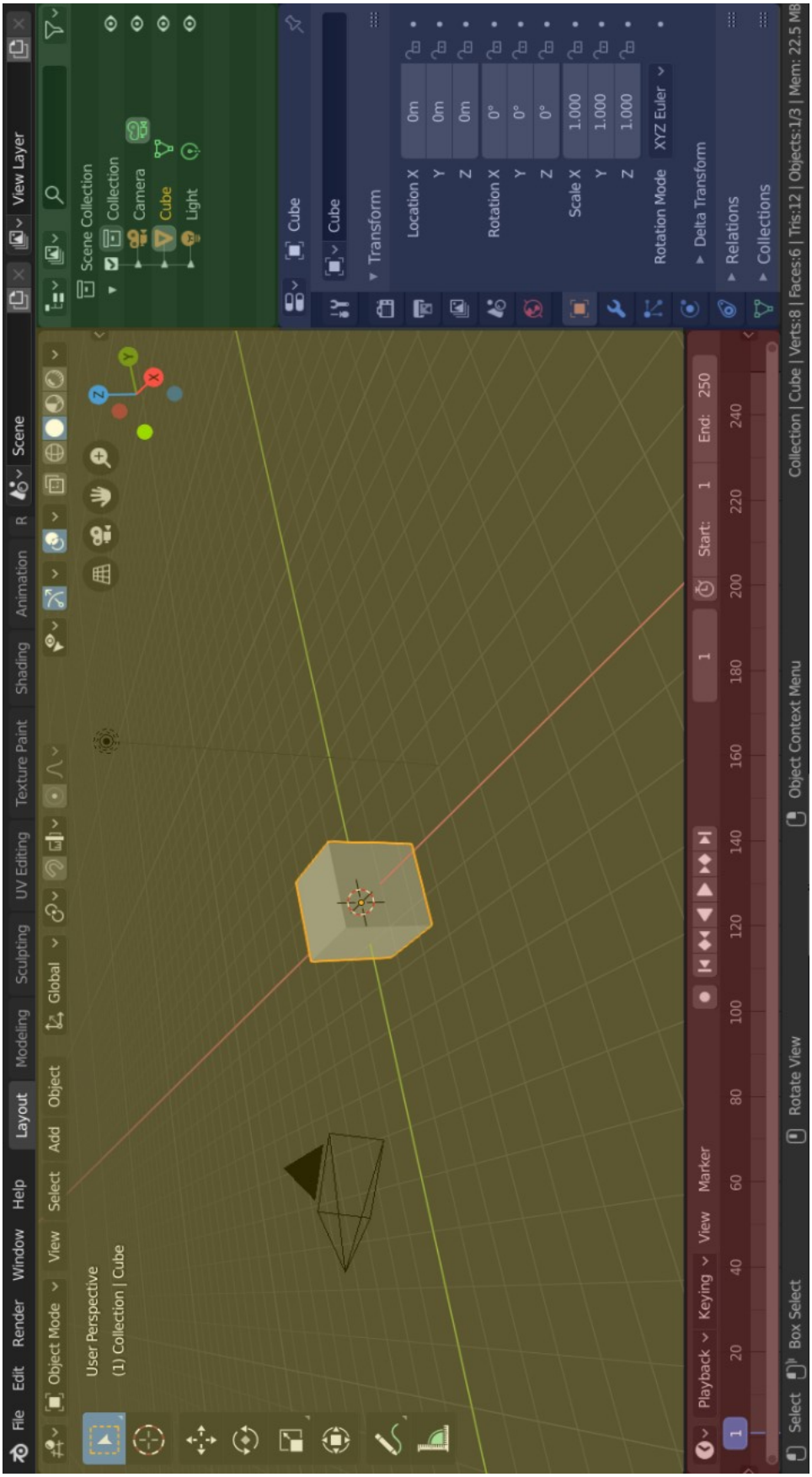
### 2.2.1 Uživatelské prostředí

Uživatelské rozhraní Blenderu si nezískalo příliš pozitivní pověst kvůli jeho komplikovanosti. Pro uživatele tak mohlo být jeho obsluhování zpočátku náročnější. V dřívějších verzích byla tato pověst poněkud oprávněná, ale od verze Blender 2.8 došlo k výrazným aktualizacím rozhraní a nyní je mnohem předvídatelnější a snáze se učí. Většina funkcí má klávesové příkazy pro rychlejší přístup a rozhraní je neblokující, což znamená, že okna a dialogy se nepřekrývají nad sebou. [14]

Výchozí obrazovka ukazuje pracovní plochu „Layout“ v hlavní oblasti. Tento pracovní prostor je obecný prostor pro náhled scény a objektů a obsahuje následující editory: [15]

- 3D pohled vlevo nahoře (žlutá barva)
- Přehled kolekcí vpravo nahoře (zelená barva)
- Editor vlastností vpravo dole (modrá barva)
- Časová osa vlevo dole (červená barva)

[4] O Eevee více v kapitole 2.2.3.1

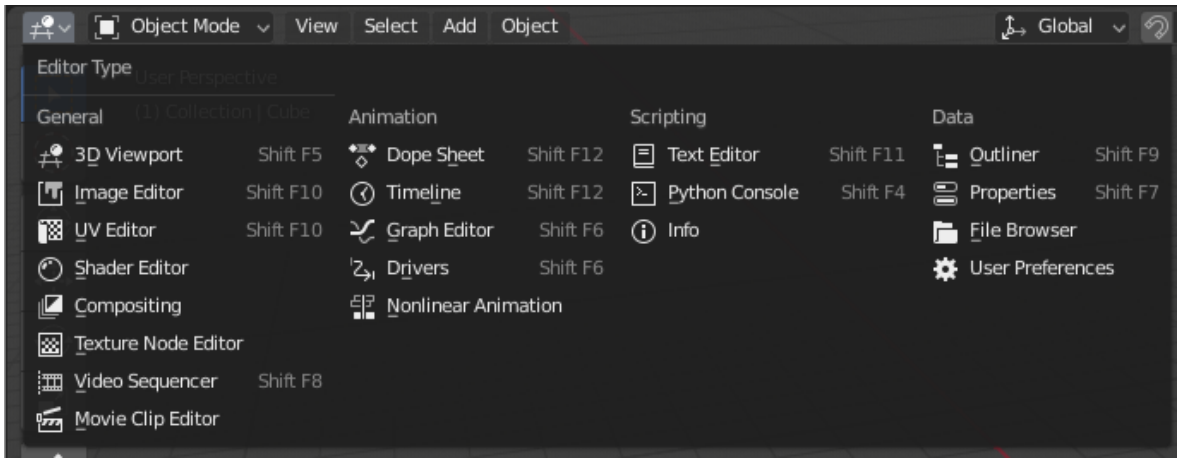



Obrázek 2: Základní pracovní plocha se čtyřmi editory

Pracovní plochy, tzv. „Workspaces“, jsou v podstatě předdefinovaná rozvržení oken. Flexibilita Blenderu s oblastmi umožňuje vytvářet přizpůsobené pracovní prostory pro různé úkoly, jako je modelování, animace a skriptování. Často je užitečné rychle přepínat mezi různými pracovními prostory ve stejném souboru. Ve výchozím nastavení obsahuje Blender také několik dalších pracovních prostorů:

- **Modelling:** pro úpravu geometrie pomocí nástrojů pro modelování
- **Sculpting:** pro úpravu objektů pomocí sochařských nástrojů
- **UV Editing:** mapování souřadnic textury na 3D povrchy objektů
- **Texture Paint:** nástroje pro barvení textur v 3D zobrazení
- **Shading:** nástroje pro určení vlastností materiálu pro vykreslování
- **Animation:** nástroje pro vytváření vlastností objektů v závislosti na čase
- **Rendering:** pro prohlížení a analýzu výsledků vykreslování
- **Compositing:** post-processing obrázků a informací o vykreslování
- **Scripting:** programovací pracovní prostor pro psaní skriptů [15]

Blender poskytuje řadu editorů pro zobrazování a úpravu různých aspektů modelu. Výběr typu editoru, první tlačítko na levé straně záhlaví, umožňuje změnit editor ve vybraném podokně. Každé podokno v Blenderu může obsahovat jakýkoli typ editoru a je také možné otevřít stejný typ editoru vícekrát. [16]



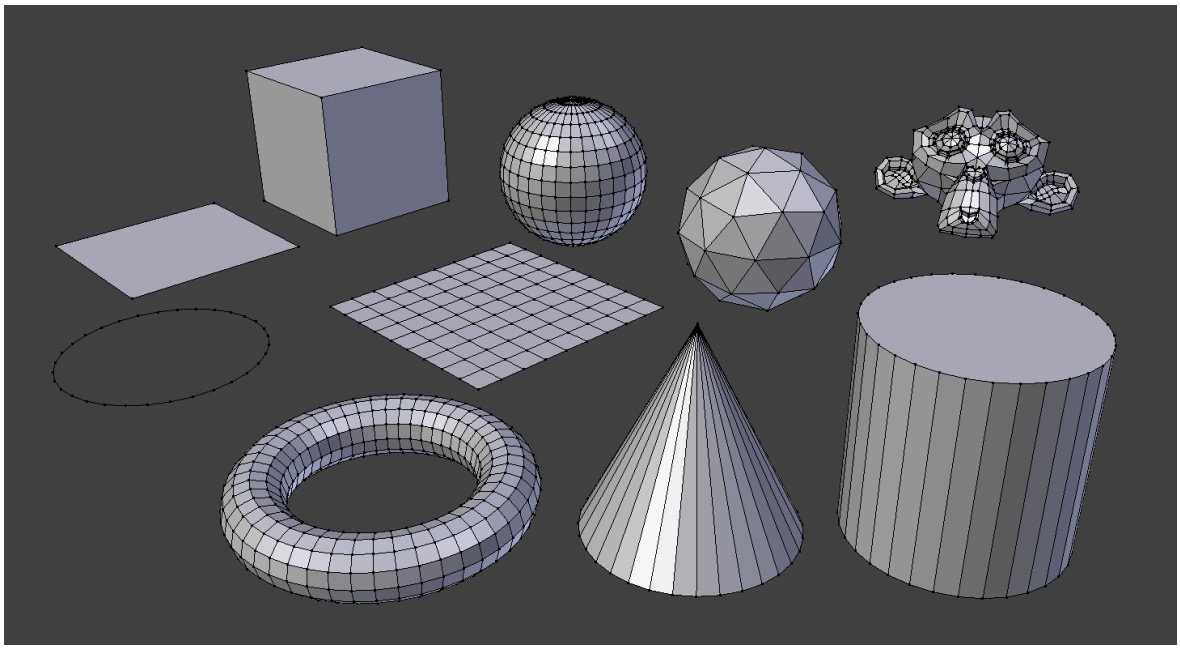

Obrázek 3: Přehled editorů v Blenderu 2.82 [16]

### 2.2.2 Modelování

Běžným základním typem objektu používaným ve 3D scéně jsou mřížkové tzv. „Mesh“ modely, které jsou sestaveny ze tří základních struktur:

- Vrchol – elementární část, která představuje jediný bod nebo polohu v 3D prostoru
- Hrana – spojuje dva vrcholy, používá se pro vytváření a spojení ploch
- Plocha – slouží k vytvoření skutečného povrchu objektu

Blender přichází s řadou „primitivních“ mřížkových modelů (např. kruh, kostka, válec atd.), ze kterých lze začít modelovat. Postupným sestavováním jednotlivých primitivních modelů se vytváří větší a složitější tvary. [17]



Obrázek 4: Přehled primitivních modelů [17]

3D pohled v Blenderu má tři základní režimy, které umožňují vytváření, editaci a manipulaci s mřížkovými modely. Každý z těchto tří režimů má různé nástroje. Některé nástroje lze nalézt ve více režimech.

Tři režimy pro modelování:

- **Object mode:** Podporuje základní operace, jako je vytváření objektů, spojování objektů, správa tvaru, UV / barevné vrstvy.
- **Edit mode:** Používá se pro většinu operací úpravy struktur mřížkových modelů.
- **Sculpt Mode:** Namísto řešení jednotlivých struktur modelu podporujte úpravu pomocí štětce, obdoba odvětví sochařství. [18]

### 2.2.3 Renderovací enginy v Blenderu

Renderovací enginy jsou tou částí programu Blender, která konvertuje aktuální zobrazení displeje do 2D obrázku, případně sekvencí obrázku tvořící animaci. Ve verzi 2.82 jsou k dispozici 3 renderovací enginy, každý s mírně odlišným uživatelským prostředím a vlastnostmi a s jinými parametry renderování. [8]

#### *2.2.3.1 Eevee*

Eevee (zkratka pro „Extra Easy Visual Environment Engine“) je renderovací engine Blenderu, jenž renderuje vytvořený obraz v reálném čase pomocí OpenGL. Je zaměřený na rychlost a interaktivitu, které využívá při vykreslování fyzicky založených PBR materiálů. Lze jej interaktivně používat ve 3D Viewportu, ale také k vytváření finálního vyrenderovaného obrazu ve vysoké kvalitě. [19]

Na rozdíl od Cycles nevyužívá Eevee vykreslovací metodu ray tracing[5]. Místo výpočtu každého paprsku světla používá Eevee proces zvaný rasterizace. Rasterizace odhaduje, jak světlo interaguje s objekty a materiály pomocí řady algoritmů a je méně přesná než ray tracing. I když je Eevee navržen pro použití principů PBR, není dokonalý a Cycles budou vždy poskytovat fyzicky přesnější vykreslení. [19]

#### *2.2.3.2 Cycles*

Vykreslování pomocí Cycles je speciálně navrženo tak, aby produkovalo realistické, vysoce kvalitní, zobrazení obrázků, nebo snímků v animaci, které zahrnuje barvy, textury a speciální osvětlení. Kvalita finálního výtvoru je nastavitelná, protože vykreslování s vysokým rozlišením může zabrat spoustu výpočetního času. [20]

#### *2.2.3.3 Workbench*

Workbench Engine je renderovací engine optimalizovaný pro rychlé vykreslování během modelování a náhledu animace. Není zamýšlen jako engine, který by vykresloval finální obrázky pro projekt. Jeho hlavním úkolem je zobrazit scénu ve 3D Viewport, na které se momentálně pracuje. [21]

---

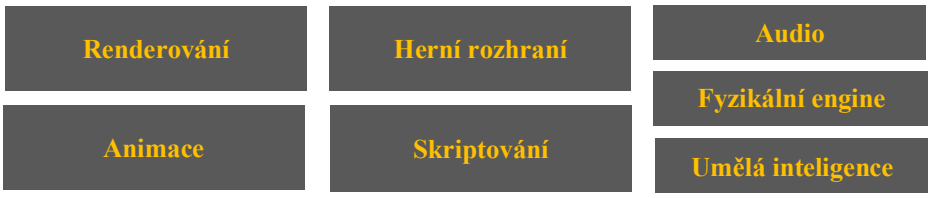
[5] Ray tracing viz kapitola 1.2.3.1

# 3 3D ENGINY

3D enginy jsou složité, víceúčelové nástroje pro tvorbu her a multimediálního obsahu. Nabízejí platformu pro efektivní vývoj interaktivního prostředí, někdy i bez potřeby znalosti programování. Enginy pokrývají mnoho různých oblastí procesu tvorby virtuálního prostředí, jako je vykreslování a fyzika prostředí, zvuk, animace, umělá inteligence či vytvoření uživatelského rozhraní.

## 3.1 Základní funkce 3D enginů

Moderní enginy staví na architektuře komponent. Enginy jsou rozděleny do několika částí, z nichž každá poskytuje speciální funkce a věnuje se určité problematice. Jednotlivé enginy se tedy mohou lišit právě v tom, na kterou z jednotlivých částí kladou největší důraz.



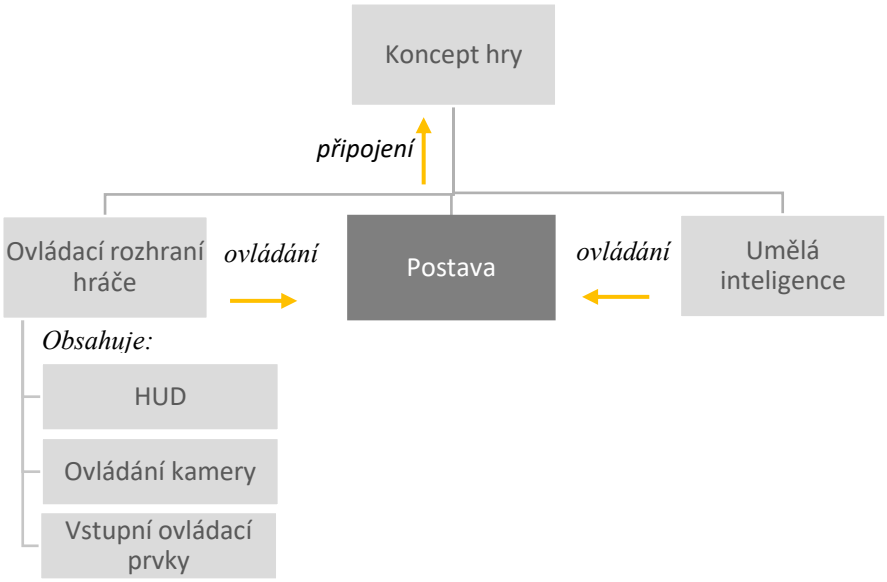

Obrázek 5: Přehled komponent 3D enginů

### 3.1.1 Renderování a grafický engine

Aby se obrázek zobrazil na obrazovce, musí být vykreslen na monitor (např. na počítač, televizi, nebo mobilní zařízení). Grafický engine je zodpovědný za výstup na displeji tím, že přijímá informace o celé scéně, jako je barva, textura, geometrie, stíny jednotlivých objektů, osvětlení a úhel scény. Následkem toho zvažuje vzájemnou interakci faktorů, které ovlivňují celkovou okluzi objektů. Grafický engine poté na pozadí pomocí všech těchto informací podstoupí rozsáhlé výpočty, než bude moci na obrazovku zobrazit konečné pixelové informace. Síla grafického enginu ovlivňuje, jak realisticky bude scéna vypadat. Enginy mají schopnost dodávat výslednému prostředí fotorealistické vlastnosti. Jejich schopnost optimalizovat scénu a zpracovat velké množství výpočtů v reálném čase umožňuje uživatelům vytvářet realistické objekty. [22]

### 3.1.2 Herní rozhraní

Engine se skládá ze vstupního systému, který převádí stisknutí kláves a tlačítka myši ovládané hráčem na akce provedené postavou ve hře. Tento vstupní systém lze nakonfigurovat prostřednictvím herního rámce. Herní rámec je souborem různých funkcionalit, neboli herních tříd. Jeho součástí je herní koncept, do kterého řadíme herní třídu sledování vývoje hry (seznam hráčů, průběžné skóre atp.) a herní třídu kontrolu pravidel hry. Mezi herní třídy dále řadíme také heads-up obrazovka (HUD)[6] a uživatelská rozhraní sloužící k tomu, aby hráči poskytli zpětnou vazbu během hry. Vývojový diagram na Obrázku 6 níže ukazuje, jak jsou tyto základní herní třídy vzájemně provázány. Hra se skládá z herního konceptu. Lidští hráči připojující se ke hře jsou asociováni se svým ovládacím rozhraním hry. Toto ovládací rozhraní umožňuje hráčům vlastnit ve hře postavu, aby mohli mít fyzickou podobu a také poskytuje hráčům vstupní ovládací prvky (jako například skóre), které mohou být přeneseny mezi různými postavami, HUD a ovládání různých pohledů kamery. Postavy mohou být ovládány jak hráči, tak umělou inteligencí. [22]




Obrázek 6: Vztah herních tříd v herním rámci [23] (vlastní zpracování)

[6] Heads-up obrazovka (HUD) je 2D náhled, který je součástí mnoha her a slouží k lepší orientaci ve hře.

### 3.1.3 Animace

V mnoha případech není konečným cílem projektu vytvořit jeden obrázek, ale vytvořit animaci, která se skládá ze sekvence obrázků, které ukazují prostředí v různých časech. V animaci dochází k malým změnám z jednoho obrázku v sekvenci na další. Během ní se může změnit téměř jakýkoli aspekt scény, včetně souřadnic objektů, transformací, vlastností materiálu či pohledu. Objekt může být například vytvořen tak, aby rostl v průběhu animace postupným zvyšováním škálového faktoru ve škálové transformaci aplikované na objekt. A změna pohledu během animace může vytvořit efekt pohybu, nebo létání po prostředí. Při výpočtu potřebných změn existuje mnoho technik. Jednou z nejdůležitějších je použití fyzikálního enginu, který vypočítává pohyb a interakci objektů na základě fyzikálních zákonů. [1]

### 3.1.4 Skriptování

Enginy by měli také zajistit způsob, jak popsat chování jednotlivých interaktivních komponent. Jako komponenty pro objekty mohou sloužit skripty, které objektům umožňují reagovat na různé podněty uživatele a vzájemně spolupracovat. Skriptovací jazyky se napříč enginy liší. Engine Unity používá skriptovací jazyk C# a javascript, Unreal engine nabízí nativní C++ nebo Blueprint vizuální skriptování a CryEngine pak C++ nebo Lua. Nicméně je důležité, aby skriptovací kódy pro objekty v prostředí mohly být přidávány a znovu použity takovým způsobem, aby je mohl použít i návrhář prostředí, který není programátor. Jak je tato problematika řešena, závisí na editoru daného enginu. V Unity enginu může návrhář přidat kód jako součást objektu a nastavit jeho vlastnosti ručně. V Unreal enginu kromě toho existuje celá koncepce vizuálního kódování Blueprint, která uživateli umožňuje vytvářet komplikovanou logiku bez zápisu jediného řádku kódu. [24]

### 3.1.5 Audio

To, co platí pro grafický engine ohledně grafických zdrojů a jejich vykreslování na obrazovku, platí obecně pro správce ozvučení ohledně zvukových zdrojů a jejich přehrávání v reproduktorech systému. Ozvučení zde znamená hudbu i zvuk. Hudba se týká zvuku obvykle trvajícího déle než 1 minutu a určeného k přehrávání ve smyčce jako pozadí pro jakýkoli jiný přehrávaný zvuk. Zvuk odkazuje na krátké zvuky, jako jsou klepání na dveře, výstřely a kroky, které se pravděpodobně budou pouštět v reakci na konkrétní událost, a které trvají méně než 1 minutu. Úkolem správce zvukových zdrojů je načítat zvuk ze souborů do paměti,

ale úkolem správce ozvučení je tyto zdroje přijímat a pouštět je do reproduktorů podle potřeby hry. Správce ozvučení je rovněž zodpovědný za komunikaci mezi enginem a zvukovým hardwarem, nastavení celkové úrovně hlasitosti jednotlivých zdrojů a použití zvukových efektů, jako jsou efekty dozvuku, efekty zesilování a zeslabování či efekty ozvěny. [25]

### 3.1.6 Umělá inteligence

Umělá inteligence (anglicky „Artificial Intelligence“, dále jen „AI“) může pomocí naprogramových instrukcí napodobovat skutečný svět. Programátoři vytvořili umělou inteligenci, která dává objektům mozek, tedy schopnost myslet a rozhodovat se. V zásadě je AI tvořena složitými sadami naprogramovaných pravidel, které pomáhají objektům rozhodovat se a vykonávat jejich navrženou funkci nebo chování. [22]

Prvky umělé inteligence obsahují také konečné automaty, regulátory a jednají podle určitých pravidel. Například v Unreal Engine, který si představíme později, jsou implementovány behaviorální stromy. Tyto stromy jsou vývojové diagramy, které se používají k rozhodování o tom, jaký úkol má AI provést. Enginy však nemají programovat umělou inteligenci namísto vývojářů. Nabízí pouze sadu nástrojů a systémů, které mohou vývojáři použít, aby se jejich umělá inteligence chovala podle potřeby. [24]

### 3.1.7 Fyzikální engine

Ve skutečném světě jsou objekty ovládány fyzikálními zákony. Srazí se a jsou uvedeny do pohybu podle Newtonových zákonů. Přitažlivost mezi objekty se řídí zákonem gravitace a Einsteinovou teorií obecné relativity. Aby objekty ve virtuálním světě reagovaly podobně jako ve skutečném, musí mít stejné vlastnosti vytvořené pomocí programování ve fyzikálním enginu. Každý objekt má své ohraničení, zjednodušenou hranici, na kterou působí fyzikální zákony, a také nějaký fyzický materiál, který určuje hmotnost objektu a vlastnosti povrchu. Engine vypočítá pohybové vektory a kolize mezi objekty. Můžeme rozlišit mezi statickými objekty, které se vůbec nepohybují, jako země a pevná tělesa, a dynamickými objekty, které mají hmotnost, materiál reagující na působící síly, padají gravitací atd. Navíc v enginu můžeme simulovat měkká tělesa, která mění svůj tvar podle vnějších sil (tkanina, kapalina apod.). Ačkoli fyzikální simulace nemusí být extrémně přesné, je to důležitá součást vývoje virtuálního prostředí. Realistický fyzikální engine nám umožňuje soustředit se na to, abychom vytvářeli samotnou scénu namísto trávení času nad vytvářením správně interagujících objektů ve virtuálním světě. [22]

## 3.2 Nejrozšířenější 3D enginy

V této kapitole jsou popsány tři v současné době nejrozšířenější 3D enginy – Unity 3D, Unreal Engine 4 a CryEngine. Na základě zjištěných informací pak bude vybrán vhodný engine pro použití v praktické části.


Obrázek 7: Nejrozšířenější 3D enginy

### 3.2.1 Unity 3D

Engine Unity 3D vyvíjí společnost Unity Technologies aktuálně ve verzi 2019.3.5. Je to nejpoužívanější vývojový engine na světě. První verze, která byla zaměřena pouze na vývoj operačního systému MAC OS X, byla vydána na konferenci Apple v roce 2005. Od té doby Unity vyvinula a v současné době podporuje celkem 27 platforem včetně VR. Nejdůležitější výhodou Unity je snadné použití. Vytváření prostředí v Unity je velmi rychlé, zejména pro mobilní platformy. Projekty a sestavené hry mají malý datový objem a proces exportu z enginu je poměrně jednoduchý. Architektura komponent enginu je snadno pochopitelná. Skriptování v jazyce C# je rychlé a efektivní. Unity má velkou uživatelskou komunitu. Existují fóra plná otázek a odpovědí, které usnadňují ladění. Zároveň existuje obchod s externími rozšířeními, který je relativně levný a obsahuje mnoho užitečných prostředků. Pouze pokud jde o grafiku Unity, vypadá o něco hůře ve srovnání s Unreal nebo CryEngine a nemá dobrou podporu tvorby terénů a flóry. [24]

### 3.2.2 Unreal Engine

Unreal Engine 4 (UE4) je poslední verze herního vývojářského nástroje vytvořeného firmou Epic Games. Je to profesionální videoherní software kombinující nástroje pro špičkové vykreslovaní materiálů a prostředí, jejich osvětlení a odraz v reálném čase, čímž vytváří působivé interaktivní zážitky. Tyto nástroje zahrnují simulaci fyzikálního chování předmětů, světla a stínů, uživatelského prostředí a vykreslování, ať už se jedná o velkoplošná prostranství a terény, nebo o komplexní materiály, částicové simulace, animace postav či tvorbu

videí. Unreal Engine obsahuje vše potřebné pro tvorbu největších herních titulů od začátku až do konce. A co víc, každá vlastnost, kterou by Unreal Engine případně postrádal, může být přidána díky kompletnímu přístupu ke zdrojovému kódu, napsaném v modifikatelném C++ kódu. [26]

Navzdory jeho síle a komplexnosti je UE4 neuvěřitelně přístupný širší veřejnosti. Editor tohoto programu se chlubí moderním uživatelským prostředím, pokročilými nástroji, postupně aktualizovanou dokumentací, kompletním přístupem ke zdrojovým kódům a širokou komunitou vývojářů. Každý jednotlivec tak může vytvářet vizualizace, simulace nebo případně hry na profesionální úrovni za velmi krátký čas. [26]

Prostředí UE4 je možné získat pro konečného uživatele bezplatně pomocí licence EULA (End User License Agreement) pod podmínkou, že hrubé příjmy při zpeněžení výsledného produktu nepřesáhnou čtvrtletně hodnotu 3000 dolarů. V opačném případě je nutné zaplatit 5% licenční poplatek. [27]

### 3.2.3 CryEngine

CryEngine je výkonný engine navržený německou společností Crytek. Je zaměřen na PC a konzole. CryEngine má vynikající algoritmy pro osvětlení a jeho vykreslovací schopnosti jsou velmi vysoké. Pokročilé jsou rovněž jeho animační systémy. CryEngine má také výkonné nástroje pro návrh úrovní a podporuje hustou vegetaci. Není však příliš uživatelsky přívětivý a pro začátečníky je nevhodný. Zdrojový kód CryEngine je napsán v jazyce C++. Zajímavostí je, že tento engine využilo také Pražské herní studio Warhorse při tvorbě neúspěšnější české hry Kingdom Come: Deliverance. [24]

Každý z těchto enginů má odlišný důraz na jednotlivé funkce a také různou náročnost, a proto je třeba si před realizací projektu ujasnit, co je jeho hlavním cílem a podle toho vybrat nejvhodnější engine. Srovnání 3 nejpoužívanějších enginů znázorňuje Tabulka 2. Ze srovnání vyplývá, že nejvhodnějším enginem pro použití v praktické části bude Unreal Engine. Hlavní důvody budou popsány v další části.

Tabulka 2: Srovnání nejrozšířenějších 3D enginů

| | Unity 3D | Unreal Engine | CryEngine |
|---|---|---|---|
| **Společnost** | Unity Technologies | Epic Games | Crytek |
| **Rok vzniku** | 2005 | 1998 | 2002 |
| **Klíčové vlastnosti** | Programování v C# a UnityScript (podobné jako JavaScript), jednoduchost | Jazyk C++, Blueprint skripty, postprocessing, dokumentace | Grafické vlastnosti, programovací jazyk C++ |
| **Nejlepší využití** | 2D a 3D engine, mobilní hry, prezentace | 3D herní engine, vizualizace, filmový průmysl | Herní průmysl, vizualizace |
| **Náročnost** | Průměrná | Náročný | Náročný |
| **Cena** | Studentská a individuální licence zdarma, předplatné (od 399 $ ročně) | Zdarma (5% z případného výdělku) | Zdarma (5% z případného výdělku) |

## 3.3 Popis prostředí vybraného 3D enginu

V praktické části byl jako 3D engine zvolen program Unreal Engine 4. Byl vybrán na základě těchto výhod:

- Licence pro vývojáře zdarma
- Výborné vestavěné efekty pro postprocesování videa
- Sada filmových nástrojů umožňující pracovat s animacemi a kamerami způsobem filmové produkce
- Velmi pokročilé nástroje pro tvorbu terénů a flóry
- Zabudovaný editor materiálů, který umožňuje vytváření nových shaderů

Nevýhodou využití Unreal Engine oproti rozšířenějšímu enginu Unity 3D může být jeho komplexnost a složitost při prvním seznámení s programem.

### 3.3.1 Pracovní prostředí Unreal Enginu

Unreal Engine, v aktuální verzi 4.24, je komplexní vývojářský nástroj zaměřený na velké projekty. Jeho pracovní prostředí se nazývá Unreal Editor. V tomto editoru se scény, ve

kterých se vytváří virtuální 3D prostředí, obvykle označují jako úrovně (v angličtině „Levels“). Jakékoliv předměty umístěné v těchto úrovních, ať už je to světlo, objekt nebo postava, jsou považovány za aktéry (v angličtině „Actors“). Technicky vzato, „Actor“ je programovací třída používaná v Unreal Engine k definování objektu, který má své údaje o poloze, rotaci a měřítku. Zároveň existuje několik těchto programovacích tříd v závislosti na jejich použití (např. třída „Character“ pro nastavení chování postavy, kterou uživatel ovládá pomocí klávesnice). Postupné využití jednotlivých tříd slouží k sestavení kompletního prostředí. [28]

Obrázek 8: Základní rozložení Unreal Editoru [28]

Unreal Engine je založený na spolupráci několika komponent, jenž ty nejdůležitější z nich mají vlastní editor z individuálními funkcemi podle zaměření. Nepoužívanější editory jsou tyto: [29]

- **Level editor:** Primární editor pro sestavení jednotlivých úrovní a prostředí. Obecně řečeno je to prostor, kam se přidávají objekty a modely.
- **Material editor:** Editor materiálů slouží jako místo pro tvorbu, nebo úpravu již vytvořených materiálů použitých na objektech.
- **Blueprint editor:** Prostor pro úpravu vizuálních skriptů Blueprint.
- **Persona editor:** Editor se sadou nástrojů pro tvorbu a úpravy animací, včetně animací postav.

### 3.3.2 Blueprints

Systém vizuálních skriptů Blueprints v Unreal Engine je kompletní skriptovací systém založený na konceptu použití rozhraní sestaveného pomocí uzlů sloužících k vytvoření interaktivních prvků z Unreal Editoru. Stejně jako u mnoha běžných skriptovacích jazyků se používá k definování objektově orientovaných tříd, nebo objektů v enginu. Tento systém je mimořádně flexibilní a výkonný, protože umožňuje projektantům využívat prakticky celou řadu konceptů a nástrojů, které jsou obecně dostupné pouze programátorům. Navíc značkování specifické pro Blueprint dostupné v implementaci C++ Unreal Engine umožňuje programátorům vytvářet základní systémy, které mohou návrháři rozšířit. [30]

### 3.3.3 Vizuál

Unreal engine je znám svými vizuály a postprocessingem. Umožňuje uživateli vytvářet nové materiály pomocí uzlů, v zásadě se jedná o vizuální programování shaderu[7] (podobné zpracování jaké se používá v Blenderu pro tvorbu materiálů). Unreal engine je široce používán pro architektonické vizualizace ve formě průchodů v reálném čase. Vývojář nepotřebuje žádné další pluginy, aby hra vypadala jako dnešní nejpopulárnější tituly. Všechny tyto vizuální prvky jsou však výkonné a vyžadují vhodný hardware.

Systém materiálů je tedy jednou z nejlepších funkcí, které Unreal nabízí. Engine sestaví pro každý materiál shader - materiál lze použít tak, jak je, nebo změnit parametry s cílem vytvořit různé variace stejného materiálu. Tato metoda dává tvůrci mocný nástroj pro vytváření ohromujících materiálů. Nevýhodou je, že kompilační časy složitějších shaderů mohou být vyšší (až minuty). [24]

Díky svým vynikajícím obrazovým vlastnostem se Unreal Engine v dnešní době využívá i ve filmovém průmyslu kde nahrazuje tzv. zelené plátno (green screen). Na obrovskou jednolitou LED obrazovku, umístěnou za herci, se v reálném čase renderuje filmové prostředí a ve spolupráci se světelným stropem se docílí výborné simulace prostoru. [31]

Na konci jara 2020 byla oznámena nová pokročilá verze programu Unreal Engine 5, jenž bude využívat výkonný hardware nové generace konzolí (Playstation 5 a Xbox Series X).

[7] Shader je prvek sloužící k řízení jednotlivých částí programovatelného grafického řetězce grafické karty (GPU)

Obecný základ enginu zůstane stejný, rozšíření se dočká především možnost vizuální zpracování prostředí. Virtuální mikropolygonová geometrie Nanite umožní vytvářet neuvěřitelné množství detailů prostředí zahrnující až miliardy polygonů, kterou lze do enginu importovat. Další nová technologie Lumen zase zajistí plně dynamické řešení globálního osvětlení, které okamžitě reaguje na změny scény a světla. Plánované vydání nového enginu je stanoveno na začátek roku 2021. [32]

Obrázek 9: Využití vyrenderováného prostředí v UE4 místo zeleného plátna během natáčení seriálu The Mandalorian (Disney+, 2019) [31]

# 4 KONVERZE MODELU Z 3D PROGRAMU DO 3D ENGINU

Pro konverzi z vybraného modelovacího programu Blender do Unreal Enginu budou využity formáty teoreticky popsány níže. V praktické části se bude ověřovat, jaké formáty jsou pro konverzi nejvhodnější a jaké je jejich praktické využití, včetně omezení, při převodu. Jedná se tedy o formáty, ve kterých je možné z programu Blender vytvořené modely vyexportovat.

## 4.1 FBX (.fbx)

Tento formát je využíván především pro výměnu znaků reprezentujících 3D model mezi aplikacemi a je podporován širokým spektrem aplikací jako jsou například modelovací programy Cinema4D, Maya, Autodesk 3ds Max, Wings3D a enginy jako Unity3D, Unreal Engine 3 a Unreal Engine 4. Exportovací rozhraní může do FBX formátu zakomponovat modifikátory modelů a animací, tak aby se konečný výsledek co nejméně lišil od vytvořeného modelu. [33]

## 4.2 Wavefront OBJ (.obj)

Formát OBJ, který je ve 3D grafice široce používaným standardem, je populární formát prostého textu, který však má pouze základní geometrii a materiálovou podporu. Mesh objekty přeneseny pomocí tohoto formátu obsahují vrcholy, plochy, hrany, normály, UV separaci podle skupin, či objektů, NURBS křivky a povrchy. Neexistuje ovšem podpora pro materiály objektů, armatury, animací, světel či kamer, prázdných objektů, nebo formy transformace. [34]

## 4.3 Alembic (.abc)

Alembic je otevřený systém pro výměnu počítačové grafiky. Umí „destilovat“ složité, animované scény do neprocedurálního, na aplikaci nezávislého souboru vymodelovaných geometrických výsledků. Tato „destilace“ scén do vymodelované geometrie je přesně analogická s rozptýlením světelných a renderovacích scén do vykreslených dat obrázku. Alembic je zaměřen na efektivní ukládání výsledků výpočtů složitých procedurálních geometrických konstrukcí. Konkrétně se nezabývá ukládáním komplexního závislostního grafu procedurálních nástrojů použitých k vytvoření výsledků výpočtů. Systém bude například efektivně ukládat animované pozice vrcholů a animované transformace, jež jsou výsledkem libovolně složitého procesu animace a simulace, kupříkladu fluid simulací, látkových a objektových animací atd. [35]

## 4.4 gITF 2.0 (.glb/.gltf)

glTF™ (GL transmisní formát) se používá pro přenos a načítání 3D modelů do webových a nativních aplikací. glTF snižuje velikost 3D modelů a dobu zpracování potřebnou k rozbalení a vykreslení těchto modelů. Tento formát se běžně používá ve webových aplikací a má podporu v různých 3D enginech, jako jsou Unity3D, Unreal Engine 4 a Godot. Tento importér/exportér podporuje následující funkce glTF 2.0: materiály, nesvítivé textury bez stínu, kamery, bodová světla, animace, dokáže převést ale i statické objekty. [36]

## 4.5 Addon „Blender for Unreal Engine“

Toto rozšíření umožňuje exportovat obsah vytvořený pomocí Blenderu do Unreal Enginu. Zjednodušuje způsob exportu z Blenderu do UE4 tím, že umožňuje exportovat všechny objekty scény současně. Dokonce je automaticky distribuuje ve správné stromové struktuře v korelaci s Unreal Engine 4 pipeline[8]. Kolizní tvary jsou vytvářeny přímo v Blenderu. Lze přesně vybrat, které animace se mají exportovat. Toto rozšíření zahrnuje kontrolu chyb, aby se předešlo potenciálním problémům, a obecné skripty v Pythonu, které lze použít v UE4 k importu kamerových objektů a animací z Blender projektu do úrovní v enginu. Všechny kamerové objekty a jejich animace se importují jako kamerový aktéři. [37]

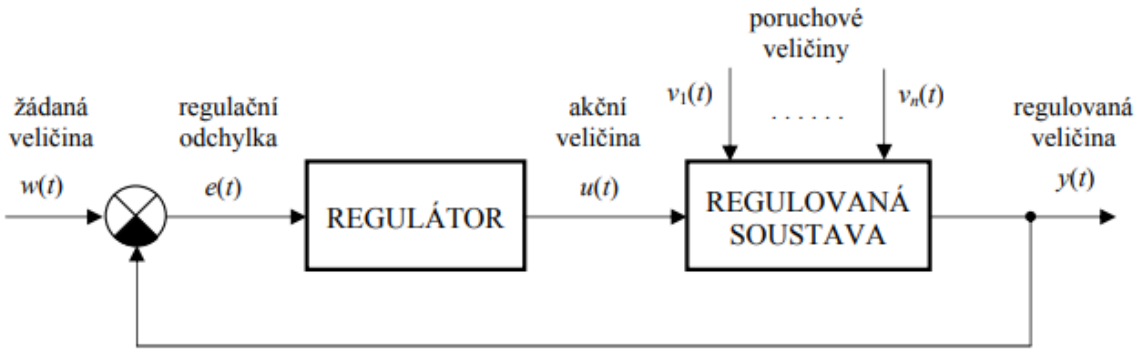
## 4.6 Universal Scene Description (.usd)

Soubory Universal Scene Description (USD) mohou obsahovat složité vrstvení, přepisování a odkazy na jiné soubory. Exportér USD Blenderu má mnohem jednodušší přístup. Při exportu se exportují všechny viditelné podporované objekty ve scéně, případně omezené jejich výběrem. Blender nepodporuje export neviditelných objektů, USD vrstev, variant, animací koster atd., tedy alespoň prozatím. Do USD lze exportovat následující objekty: Mesh různých druhů, kamery (pouze perspektivní), světla (všechny typy kromě plochých), částicové systémy typu Hair (exportované jako křivky a omezené dědičností). [38]

[8] Unreal Engine je vybaven FBX pipeline pro import, která umožňuje jednoduchý přenos obsahu z libovolného počtu aplikací pro tvorbu digitálního obsahu, které podporují tento formát. [39]

# 5 ŘÍZENÍ 3D MODELU

V praktické části bude vytvořen model kvadrokoptéry, který bude poté, s využitím vhodného formátu, převeden do Unreal Enginu. Následně by mělo dojít, pomocí vizuálních skriptů Bleuprint, k nastavení řízení tohoto modelu. Jak je uvedeno ve vybraných statích od P. Navrátila „Řízení je cílené působení na řízený objekt (řízený systém) tak, aby se dosáhlo určitého předepsaného cíle.“ [40] Na základě způsobu provedení řízení se řízení rozlišuje na ruční a automatické. Mezi automatické řízení lze zařadit řízení letadla autopilotem nebo řízení teploty regulátorem (řídící systém), mezi ruční pak řízení letadla či teploty člověkem. Z hlediska zpětné kontroly výsledku řízení, neboli zpětné vazby při řízení, můžeme řízení dělit na ovládání, regulaci a vyšší formy řízení. Zatímco ovládání je řízení bez zpětné vazby, regulace je řízením se zpětnou vazbou, při kterém dochází k udržování hodnot fyzikální veličiny na požadované úrovni. Například při regulaci polohy kvadrokoptéry jsou zjištěné aktuální hodnoty srovnávány s hodnotou, která je požadována a podle zjištěných odchylek se zasahuje do regulačního procesu, ve kterém jsou odchylky. [40]




Obrázek 10: Blokové znázornění regulátoru a regulované soustavy [40]

Podle principu působení řídícího systému na řízený systém lze automatické řízení rozdělit na logické, spojité a diskrétní. Logické využívá k řízení dvouhodnotových veličin (vypínač je sepnut nebo vypnut), které jsou formálně vyjádřeny hodnotami 0 a 1. Spojité řízení je takové, kde jak řídící systém, tak i řízený systém jsou spojité, tj. vstupně/výstupní veličiny těchto systémů jsou spojitě proměnné v čase a vytvářejí (na rozdíl od diskrétního systému) nepřetržitou vazbu mezi vstupy a výstupy. Diskrétní řízení je důsledkem nasazení počítačů jako regulátorů (řídících systémů). U řídících počítačů je nutné spojitý signál převádět na diskrétní, který počítače umějí zpracovávat. Diskrétní řídící systém vytváří vztah mezi vstupy a výstupy jako vztah mezi posloupnostmi impulsů, snímaných v časovém sledu

daném tzv. vzorkovací periodou. Mezi okamžiky vzorkovaní není regulovaná veličina měřena a ani akční veličina není upravována. [40]

Další členění regulátorů může být podle příkonu (přímé, nepřímé), či podle nositele signálu (pneumatické, hydraulické, elektrické). Většina průmyslových regulátorů jsou nepřímé regulátory, které vyžadují vnější energii pro svou funkčnost. Další možné rozčlenění regulátoru je podle jejich struktury, tj. regulátory s pevně danou strukturou (regulátory typu PID) a regulátory s obecnou strukturou (obecný lineární regulátor).

### Spojité regulátory

Dynamické vlastnosti regulátoru lze popsat rovnicí: [39]

$$\cdots + T_2^2 u''(t) + T_1 u'(t) + u(t) = r_0 e(t) + r_{-1} \int_0^t e(\tau) d\tau + r_1 \frac{de(t)}{dt} \tag{1}$$

Kde $r_0 e(t)$ je proporcionální složka regulátoru, $r_{-1} \int_0^t e(\tau) d\tau$ je integrační složka regulátoru, $r_1 \frac{de(t)}{dt}$ je derivační složka regulátoru a $\cdots + T_2^2 u''(t) + T_1 u'(t) + u(t)$ jsou setrvačné členy regulátoru. Tato rovnice vyjadřuje skutečný regulátor, kde se můžou uplatňovat různá zpoždění. To vyjadřují členy na levné straně diferenciální rovnice, které jsou způsobené setrvačností, pasivními odpory, kapacitou apod. Rovnice ideálního regulátoru má tedy podobu:

$$u(t) = r_0 e(t) + r_{-1} \int_0^t e(\tau) d\tau + r_1 \frac{de(t)}{dt} \tag{2}$$

### Diskrétní regulátory

Pro použití v praktické části pro řízení kvadrokoptéry se využije diskrétní (číslicové) formy PID regulátoru. Od číslicového regulátoru lze očekávat stejnou funkci jako od spojitého, kterou je schopnost vstupující regulační odchylku zesilovat, integrovat a derivovat. Proto při sestavování algoritmu pro číslicový regulátor lze vycházet z funkce a tím i z rovnice spojitého PID regulátoru. Úpravou na tvar s časovými konstantami lze získat výchozí rovnici PID regulátoru: [41]

$$u(t) = r_0 \left[ e(t) + \frac{1}{T_i} \int_0^t e(t) dt + T_d \frac{de(t)}{dt} \right] \tag{3}$$

Číslicové verze regulátoru se získá z této rovnice diskretizací integrace a derivace. Integraci lze provést náhradou spojitého signálu tzv. stupňovitou náhradou zleva (obdélníky zleva, další možností náhrady jsou obdélníky zprava či sečná náhrada lichoběžníky). Určení hodnoty integrálu se provádí jako součet ploch pod náhradním průběhem. Derivaci se získá nahrazením diferencemi.

$$\int_0^{kT} e(t)dt \cong T\sum_{i=0}^{k} e(i) \qquad \frac{de}{dt} \cong \frac{e(k)-e(k-1)}{T} \qquad (4), (5)$$

Po dosazení těchto vztahů do rovnice spojitého PID regulátoru a také dosazením diskrétního času kT respektive k, vypadá rovnice následovně:

$$u(k) = r_0\left\{e(k) + \frac{T}{T_i}\sum_{i=0}^{k} e(i) + \frac{T_d}{T}[e(k)-e(k-1)]\right\} \qquad (6)$$

Tomuto algoritmu číslicového regulátoru se říká polohový algoritmus. Hodnota integrálu se zde získává sumací a hodnota derivace se získává pomocí diference. Proto se tyto regulátory nazývají proporcionálně-sumačně-diferenční, a označují zkratkou PSD. Ale také se nazývají číslicové PID regulátory. Pro použití v Unreal Enginu se využije forma s diskrétním časem $\Delta t$ v následující formě: [41]

$$u(k) = K_P e(k) + \sum K_I e(k)\Delta t + K_D \frac{e(k)-e(k-1)}{\Delta t} \qquad (7)$$

**Proporcionální člen P**

Proporcionální člen je primární proměnou pro regulaci chyby kdy nejjednodušším případem je pouhé zesilování. V tom případě je akční veličina úměrná regulační odchylce. Přímo ovlivňuje regulaci chyby, takže s nízkou hodnotou proporčního členu bude regulátor provádět malé pokusy minimalizovat chybu, se zvyšováním bude regulátor dělat větší pokusy. Pokud je hodnota členu příliš malá, nemusí dojít k minimalizaci regulační odchylky vůbec (pokud nejsou použity členy D a I) a regulátor nebude schopen reagovat na změny ovlivňující systém. Pokud je člen příliš velký, začne docházet k destabilizaci modelu (oscilaci) a překmitu. [42]

### Integrační člen I

Integrační regulátor je takový regulátor, jehož akční veličina je přímo úměrná integrálu regulační odchylky. Využívá se pro zpracování chyb v ustáleném stavu, které se hromadí s časem. Problém může nastat při velké hodnotě integračního členu, kdy se v průběhu času opravují chyby ještě dříve, než se stihnou projevit aktuální změny. Špatné nastavení tohoto členu je často příčinou nestability regulátoru. [42]

### Derivační člen D

V případě složky D je akční veličina úměrná derivaci regulační odchylky. Člen D se tedy zabývá tím, jak se systém chová mezi časovými intervaly. To pomáhá tlumit systém a zlepšit tak stabilitu. V praxi není technická realizace regulátoru D možná (došlo by k rozpojení regulačního obvodu v ustáleném stavu), a proto se ve většině případů jedná o ideální D regulátor. [42]

# II. PRAKTICKÁ ČÁST

# 6 VYTVOŘENÍ 3D MODELU

Obsahem praktické části této práce je vytvoření 3D modelů a prostředí v programu Blender a převedení těchto modelů do Unreal Enginu. Pro otestování konverze mezi programy se využije několik výstupních formátů z programu Blender, tak, jak byly v krátkosti popsány v teoretické části v kapitole č. 4. Na základě nejlepších vlastností, tj. největšího počtu korektně převedených modelů a jejich nastavení, se vybere nejvhodnější formát. Pomocí tohoto formátu se poté převede celý model do prostředí Unreal Enginu.

Pro otestování co možná největšího počtu vlastností bylo nutné vytvořit komplexní model prostředí. Jako oblast, kde se budou jednotlivé modely shromažďovat, byl zvolen model ulice s rodinnými domy, cestou a automobilem. Při tvorbě budov se využije UV mapování, pro tvorbu vegetace (stromy, tráva) se zase použijí částicové systémy. Pro otestování animací a simulací poslouží vestavěné funkce Blenderu. Jedná se o tzv. Rigid body animaci (bourací koule na řetězu) a Fluid simulaci (nádoba s tekoucí kapalinou). Do prostředí enginu se bude také testovat import postavy, modelu vytvořeného pomocí funkce rigging, jenž by se měl ve výsledné aplikaci pohybovat pomocí předem definovaných vazeb. Další z podstatných částí této bakalářské práce je pak tvorba modelu kvadrokoptéry a jeho následná konverze do prostředí enginu, která bude sloužit k demonstraci funkcí Unreal Enginu a jeho možného využití při návrhu řízení a regulací složitých soustav.

Některé z modelů bylo nutné rozdělit do samostatných souborů. Jedná se o model postavy, rigid body animaci, model kvadrokoptéry a fluid simulaci. Důvodem je to, že vytvořené modely mohou mít v jednotlivých testovaných formátech odlišné výsledky převodu, což znamená, že na jeden druh vytvořeného modelu se může lépe hodit jiný formát než na druhý model. I když by bylo možné všechny tyto modely vytvořit v jediném souboru, jejich následná konverze by byla složitá, v některých případech i nemožná.

## 6.1 Statické mesh objekty

V programu Blender byl tedy vytvořen model prostředí, na němž se bude testovat především konverze statických objektů a materiálových vlastností. Tento model bude sloužit jako základ, ke kterému se budou ve výsledné aplikaci přidružovat další modely. Jedná se tedy o model ulice, který je tvořen domy, cestou, chodníky, stromy a trávou. Při jeho tvorbě bylo použito volně dostupné rozšíření Archipack, které se používá pro jednoduché architektonické modely typu oken, dveří, střech, plotů apod.

### 6.1.1 Model automobilu

Do prostředí ulice tvořené statickými objekty s navázanými texturami byl také naimportován složitý model automobilu vytvořený uživatelem z komunity zaměřené na program Blender, který byl sdílen přes webovou stránku Blendswap.com s volně šiřitelnou licencí. Samotná tvorba tohoto typu modelu by byla časově velice náročná, ovšem právě na takových složitých model se otestují základní vlastnosti konverze objektů, jako jsou vazby statických objektů (jednotlivé části automobilu nejsou od sebe odděleny a tvoří jednolitý model) a materiálové vlastnosti jednotlivých části.

### 6.1.2 Model kvadrokoptéry

Model kvadrokoptéry je vytvořen jednoduchým způsobem spojením čtyř primitivních prvků typu „Torus“ do jednoho celku. Volný prostor uprostřed těchto torusů bude vyplněn lopatkami rotoru, ty jsou ovšem vyexportovány zvlášť, protože v Unreal Enginu na ně budou navázány jiné (rotační) vlastnosti než na zbytek modelu. Následně je na celý model aplikován modifikátor „Subdivision surface“ pro vyhlazení povrchu. Velmi důležité je dbát na správné rozložení modelu na střed globální osy. V případě nerovnoměrnosti by se nemusela kvadrokoptéra chovat podle představ a mohlo by docházet k oscilaci. Tvorba tohoto modelu v Blenderu je opravdu jednoduchá, o to horší ale je celkové nastavení ovládání v Unreal Enginu.

### 6.1.3 Částicové systémy

Částicové systémy (v angličtině „Particle system") slouží k emitování objektů a mohou být dvojího typu. Typu Emitor, kdy se objekty generují podle jiného objektu, nebo typu Vlasy (Hair). Těmto systémům lze nastavit nejrůznější vlastnosti, které ovlivňují jejich generování. Může se jednat například o rotaci, nahodilost rozměrů jednotlivých objektů či rozložení podle nějakého vzoru. Pro otestování konverze částicových systémů byly v modelů prostředí vytvořeny jak systémy typu „Emitor“ – stromy a jejich náhodné emitování, tak i „Hair“-početné úzké polygony evokující trsy trávy. Pro tyto systémy byly nastaveny i odpovídající materiálové vlastnosti.

Jednou z důležitých částí je právě rozložení částicových systémů. Při generování tisíců částic dochází k vysokému zatížení procesoru a celkovému zvýšení výpočetního času. K nastavení rozložení výskytu emitovaných částic slouží mód programu Blender s názvem Weigh Paint. Jedná se o mapu četnosti, kde se mají částice vyskytovat. Například pokud se pro

renderování snímá kamerou jen určitá část scény, mimo zorné pole kamery se tyto částice vůbec nemusí emitovat.

## 6.2 Materiály a textury

Na vybrané objekty byly použity textury z online databáze dostupné na webové stránce https://www.textures.com. V této databázi jsou dostupné stovky textur a materiálu (včetně například oskenovaných povrchů), které jsou po registraci zdarma ke stažení v omezené kvalitě a s denním limitem stažení.

Pro lepší ztvárnění povrchů byly textury navázány na objekt v několika vrstvách, jedna pro základní barvu materiálu „Albedo“, druhá pro znázornění drsnosti „Roughness“ a poslední vrstva pro normálovou mapu (zvrásnění povrchu). Na ostatní modely se využilo klasických principiálních BSDF materiálů s různým nastavením odstínů barev.


Obrázek 11: Model automobilu vyrenderovaný pomocí Cycles

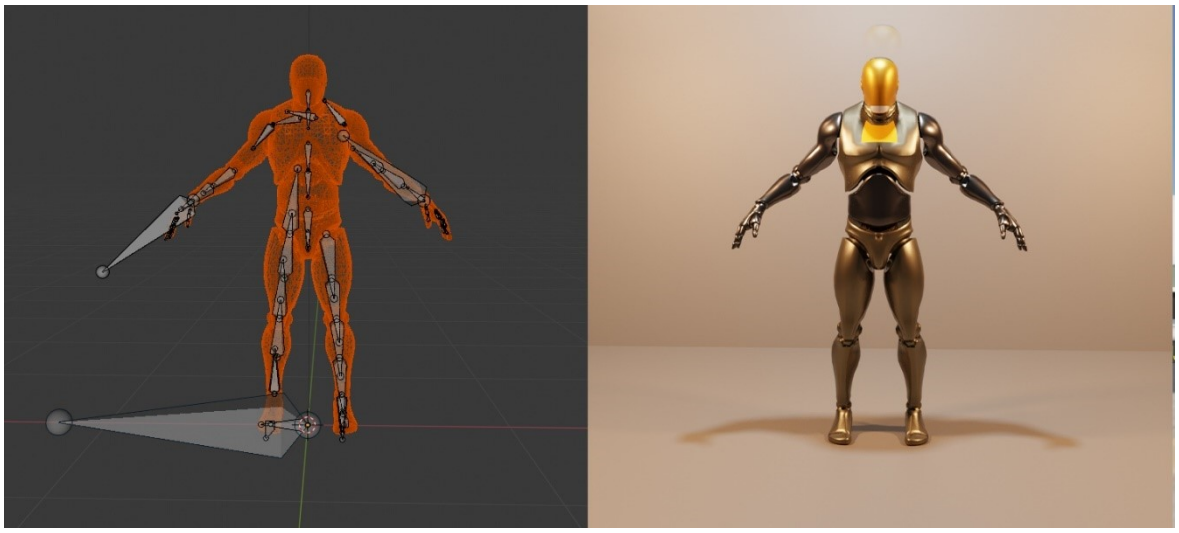
## 6.3 UV mapping

Pomocí UV Editoru lze textury namapovat přímo na objekt tak, aby odpovídaly své předloze. Při tvorbě modelu se využilo textury reálné budovy, jenž se jako obraz namapovala na vytvořený základní objekt krychle. Pomocí "Loop cuts" v Edit módu lze docílit rozdělení základního objektu podle přímek ve svislém a vodorovném směru. Toto rozdělení stěny objektu na více částí povede k vytvoření plastičnosti objektu, a to vybráním částí oken a dveří a jejich vysunutím z objektu, případně dovnitř objektu (funkce Extrude v Edit módu).

## 6.4 Animace postavy (Rigging)

Jak bylo popsáno v kapitole 1.2.1, rigging v programu Blender je proces, kdy se objektu přiřadí kostra, pomocí které se dá objekt ovládat. V Blenderu se tato kostra nazývá „Armature“. Hlavní využití této funkce je při animaci postav, kdy se modelu přiřadí pomyslné kosti, které se pomocí dědičnosti rodič – potomek (případně jiných vazeb) pohybují v požadovaném směru. Zároveň mohou existovat i vodící kostry, kterou nejsou součástí těla postavy, ale slouží jen k ovládání pohybu. Tímto nastavením lze docílit plynulé animace například chůze postavy, kdy se v každém snímku animace změní pozice jednotlivých částí kostry tak, aby co nejrealističtěji simulovaly pohyb lidské postavy. Výhoda Blenderu je, že se nemusí každý snímek animovat samostatně. Při vytvoření dvou od sebe vzdálených klíčových snímků na časové ose se animace mezi těmito snímky automaticky vygeneruje. Kromě tvorby postav se rigging může využít například i při animaci jedoucího automobilu, kdy se na osu každého kola přiřadí kostra a jednoduchým pohybem se docílí rotace kol.

Při samotné tvorbě postavy se tedy využije již zmíněné funkce Armature a dalšího z módů programu Blender a to Pose editoru. Od první vytvořené základní kosti se vytahují (pomocí funkce Extrude) další. Čím větší jsou tyto kosti, tím větší vliv mají na ovládání postavy. Při tvorbě je ovšem nutné brát ohled na správně nastavenou osu na střed postavy. Při modelování končetin stačí totiž vytvořit jen jednu stranu postavy a tu poté zrcadlit pomocí této osy. Dojde nejen k ušetření času, ale i k zamezení případné nesouměrnosti vytvořeného charakteru. I z tohoto důvodu byla postava vytvořena v samostatném souboru.

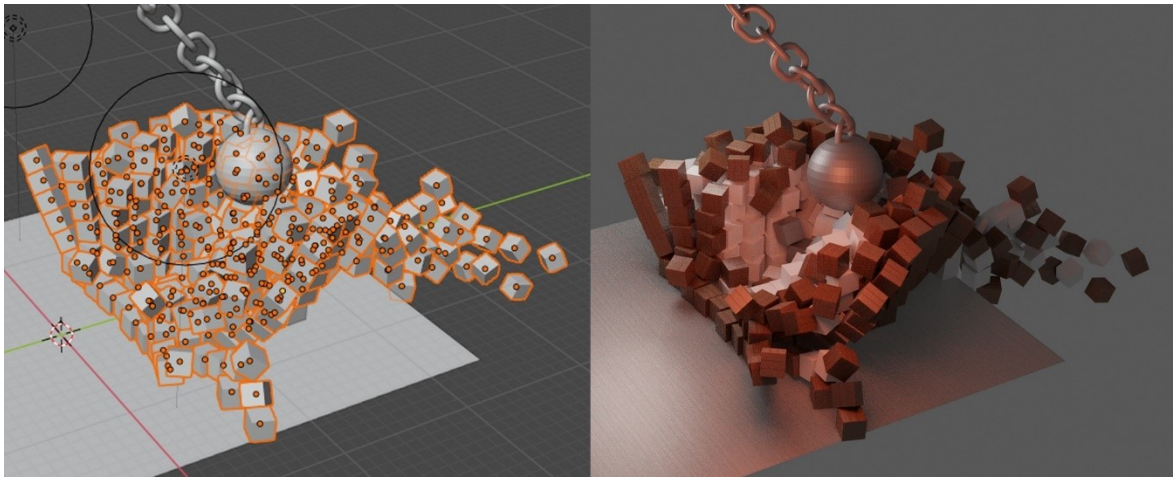


Obrázek 12: Vytvořená postava v programu Blender

## 6.5 Rigid body animace

Ač může název evokovat spojitost s vytvářením postavy, rigid v programu Blender značí animaci, kde na sebe vzájemně působí objekty. Ve většině případů se jedná o kolizi několika provázaných objektů, případně o destrukci modelu sestaveného z jednotlivých částí.

Pro kontrolu převodu takovéto simulace do Unreal Enginu byl vytvořen jednoduchý model bourací koule zavěšené na řetězu. Díky funkci "Rigid body" v záložce "Physics properties" lze každému objektu v simulaci nastavit požadované vlastnosti. Především se jedná o to, zdá má být daný objekt v animaci aktivní či pasivní. Části, které nemají v animaci figurovat, zůstanou pasivní (podložka, poslední člen řetězu sloužící k zachycení). Ostatní prvky jsou aktivní a lze jim také nastavit různé vlastnosti ovlivňující simulaci, a to především hmotnost. Pokud by bourací koule měla příliš vysokou hmotnost, řetěz by se mohl roztrhnout. Při nízké hmotnosti se neprojeví žádné účinky. Takto vytvořená koule působí na soustavu vytvořených kostek, kdy se při nárazů kostky rozletí do prostoru. Základní kostka je jednoduchý mesh objekt Cube rozmnožený pomocí modifikátoru Pole (Modifier – Array).



Obrázek 13: Rigid body animace

## 6.6 Fluid simulace

Pro tvorbu simulací tekutin v programu Blender se využívá vestavěné funkce Fluid v záložce pro nastavení fyzikálních jevů „Physics properties“. Tento prvek neslouží ovšem jen ke tvorbě kapalin s různou hustotou, dají se zde tvořit také simulace kouře a ohně. Všechny tyto elementy jsou založeny na stejném principu tří částí, které tvoří výsledný model a kterými jsou:

**Doména „Domain“**

Určuje, kde se bude samotná simulace odehrávat. Jedná se v podstatě o nádobu, ve které bude působit vytvořená tekutina, odrážet se od stěn apod. Ve vytvořeném modelu se jedná o základní objekt Cube rozšířený na kvádr.

**Zdroj „Flow“**

Zdrojem je myšleno, odkud bude vytvořená tekutina vycházet. Při správném nastavení může být zdrojem jakýkoliv primitivní Mesh prvek. U tohoto prvku se také nastavuje, o jaký typ tekutiny v simulaci půjde. Zdroji tekutiny lze také nastavit inicializační vektor rychlosti pohybu tekutiny při její emitaci. V podstatě se jedná o to, jakou rychlostí a jakým směrem bude tekutina ze zdroje vycházet. Ve vytvořeném modelu je zdrojem toku klasická kapalina připomínající vodu. Na výběr je ovšem více možností zdroje, a to:

- Kapalina
- Kouř
- Oheň
- Oheň + kouř

**Efektor „Effector“**

Poslední částí pro tvorbu korektní simulace tekutin je Efektor. Jde o objekt, na který bude tekutina reagovat. Může být kolizního typu (překážka), nebo vodícího (kudy má tekutina proudit). Efektor musí být uvnitř domény, tak aby na něj mohla tekutina působit.

Pro každý z těchto prvků lze nastavit vlastní materiál, pro vzhled tekutiny je důležité nastavení Domény. Ta je primárním objektem v simulaci, kde se nastavují všechny důležité prvky ovlivňující výsledek. Ať už se jedná o počet částic, ze kterých se bude tekutina skládat, nastavení působící gravitace, nebo počet snímků výsledné animace.

V jedné doméně může být více zdrojů tekutin, je ale důležité brát ohled na výpočetní čas takové simulace. Blender vyžaduje výpočet jednotlivých prvků simulace ještě před renderováním, jde o tzv. baking. Čím složitější je nastavení a čím vyšší je počet snímků, tím delší je čas výpočtů.

# 7 KONVERZNÍ FORMÁTY

Pro zjištění co nejlepšího formátu pro konverzi mezi programem Blender a Unreal Enginem se budou porovnávat tyto požadavky:

- Převedení statických objektů a částicových systémů
- Převedení materiálů a jejich navázání na objekty
- UV maping textur vybraných objektů
- Převod vytvořené postavy a její animace
- Rigid body animace
- Korektní sestavení Fluid simulace

Export a následný import probíhá v aktuálních verzích obou softwarů Blender 2.82 a Unreal Engine 4.25. V jiných verzích, případně při importu souborů vytvořených ve starších verzích, může docházet k nekorektnímu převodu, nebo výsledku, který se liší od testovaných souborů. Pro import do enginu budou testovány formáty FBX, Wavefront OBJ, Alembic, gITF 2.0, USD a také rozšíření „Blender for Unreal Engine“, které je volně dostupné na repozitáři GitHub. Blender dokáže exportovat soubor i do dalších formátů například Collada (*.dae) či Extensible 3D (*.x3d), ale Unreal Engine tyto formáty nedokáže žádný způsobem importovat do svého prostředí.

## 7.1 FBX (*.fbx)

Při importu formátem FBX se převedou všechny základní statické objekty se správným rozložením po scéně - tuto vlastnost lze ověřit především na modelu automobilu, který je korektně sestaven. Problém nastává při převodu materiálů. Ty se totiž v Unreal Enginu přepočítávají na parametr základní barvy. To znamená že se neprojeví nastavení materiálu typu průhlednost, tvrdost, vyzařování apod. Tyto vlastnosti je nutné v UE nastavit znovu. Výsledkem převodu materiálů bude tedy odstín konkrétní barvy.

Pokud se převádí textury navázané na objekty pomocí UV mapování, tyto textury se převedou korektně jak svým navázáním na objekt, tak i zobrazením. Ke správnému převedení z Blenderu dojde nejen u objektů s klasickou texturou, ale i např. s normálovou mapou v samotném souboru textury.

Při převodu částicových systémů pomocí FBX formátu je nutné brát ohled na to, že při importu se každý mesh objekt importuje samostatně. Při emitování tisíců objektů se tedy každý

objekt importuje zvlášť. Tím se rapidně zvyšuje doba importu objektů do Unreal Enginu, přičemž může dojít k prodloužení času importu i o několik hodin.

Formát FBX se hodí pro export postavy s následným rozpohybováním v UE, zároveň ale platí stejné materiálové omezení jako pro statické objekty.

Rigid animace a Fluid simulace se do UE převedou pouze staticky, bez animací. Statické rozložení je dáno snímkem animace ve chvíli, kdy dochází k exportování souboru, tedy například uprostřed probíhající animace.

## 7.2 Alembic (*.abc)

Formát Alembic se pro převod statických objektů příliš nehodí. Sice obsahuje možnost při importu vybrat, jaké konkrétní objekty se mají ze zdrojového souboru importovat, zároveň ovšem nepodporuje objekty, které nejsou sestaveny pomocí ploch (spojení bodů trojúhelníku nebo čtyřúhelníku). Může tak nastat situace, že se některé objekty vůbec nenaimportují, nebo se kvůli chybě import přeruší. Při převodu také dochází k nekorektnímu sestavení modelu. Dochází ke stonásobnému zmenšení a pootočení o 90° na ose x. Po opravení těchto chyb v nastavení importu se sice model korektně převede, může ale docházet k poměrovému zvětšení i v místech, kde byl v modelu použit prvek vytažení „Extrude“ programu Blender. Výsledný převedený model se tedy liší od zdrojového. Alembic se tak nehodí ani na převod postavy či rigid animace. Nepodporuje ani žádné funkce pro převod materiálů či textur a UV mapováním nevznikne požadovaná vizuální stránka modelu.

Jediné jeho využití tedy spočítá v tom, že je vhodným, nebo spíše jediným funkčním, formátem pro převod Fluid simulace. Je nutné však počítat s tím, že se nepřevedou žádné materiálové vlastnosti objektů v simulaci.

## 7.3 Wavefront (*.obj)

Wavefront (OBJ) má obecně stejné vlastnosti pro převod jako FBX formát, Unreal Engine se také chová, jako by došlo k importu souboru s příponou fbx – má stejné importovací menu i popis. Při převodu OBJ ovšem může nastávat, na rozdíl od již zmiňovaného FBX formátu, k nekorektnímu natočení o 90° a zmenšení modelu. Materiálové vlastnosti, včetně UV mapování, se také shodují s formátem FBX (materiál jako parametr odstínu barvy). I přesto

všechno lze Wavefront využít pro převedení jak statických objektů, tak i postavy. Animace a simulace tento formát nezvládá.

Ač jsou si formáty Wavefront a FBX velice podobné, exportovaný model z programu Blender s příponou „obj“ může mít až pětinásobně vyšší datovou velikost souboru než formát FBX.

## 7.4 gITF 2.0 (*.glb/*.gltf)

Před importem toho typu formátu je napřed v Unreal Engine nutné povolit zabudovaný plugin „Datasmith gITL Importer“, jenž umožní importovat 3D modely s přípononou „glb“ a „gltf“. Tyto přípony značí dva typy formátu – gITF Binary (*.glb) a gITF Embedded (*.gltf). První jmenovaný má svá data kódována binárně a je efektivnější a lepé přenositelný. Je však horší ho později editovat. Druhý jmenovaný využívá pro kódování dat JavaScriptovou knihovnu JSON. Je méně efektivní, ale je vhodnější pro případně úpravy. Blender dokáže oba tyto typy vyexportovat, v případě gITF Embedded dokonce včetně separovaných dat pro JSON, binární části a textur v několika souborech.

Pro samotné importování objektů má tento formát (v obou možných typech) omezené možnosti. Nedokáže soubor o několika objektech správně rozložit po scéně, všechny objekty jsou importovány do jednoho bodu přes sebe a je těžké s nimi pak pracovat. Hodí se tedy pouze na jednoduché modely pouze s jednotkami modelů, kde není nutné brát ohled na jejich rozložení. Zároveň není možné ani jeden typ tohoto formátu využít pro konverzi animací či simulací. Pro převod postavy ho použít lze, pouze ale jako statický objekt. Nejde na něj následně navázat charakter postavy z Unreal Enginu.

Formát gITF má ovšem vynikající materiálové vlastnosti. Dokáže převést téměř všechna nastavení korektně z programu Blenderu do Unreal Enginu. Výjimkou je průhlednost, ta zůstává zobrazená jako plná barva obdobně jako u FBX či Wavefront. Základní barva se tedy převedete obdobně jako u výše zmíněných formátů, gITF ale tuto barvu doplní dalšími faktory, které ovlivní výsledný materiál a dochází k opravdu velice přesnému přenesení materiálu a textur.

## 7.5 Blender for UnrealEngine 4

Blender for Unreal Engine je addon, rozšíření, vytvořené vývojářem a uživatelem Blenderu za účelem usnadnění exportu pomocí FBX formátu. Na rozdíl od klasického exportu umožňuje více možností, jak export celkově zpřehlednit. Ať už se jedná o připojení předpony k názvu souboru pro jejich identifikaci (například statické objekty mají předponu "SM_"), nebo o možnost exportu objektů podle kolekcí, ve kterých se nacházejí. Jinak je funkčnost tohoto rozšíření totožná s klasickým exportem FBX formátu a platí pro něj stejná, například materiálová, pravidla. Jeho použití je výhodné u modelů s vysokým počtem různých typů objektů. Nabízí totiž možnost exportovat kromě FBX formátu i Alembic simulace. Lze tedy pomocí jednoho rozšíření exportovat model ve více formátech najednou. Toto rozšíření zároveň umí automaticky nastavovat kolizní tvary pro exportované objekty.

## 7.6 USD

Formát USD (Universal Scene Description) je nejméně vhodný formát pro převod jakéhokoliv testovaného modelu. Neimportuje žádné materiálové ani UV mapovací textury a stejně jako gITF má problém se sestavením modelu na scéně. Statické objekty, včetně modelu postavy, se naimportují na jedno místo bez vzájemných vazeb. Totéž platí i pro rigid animaci a fluid simulaci, jejich následná animace tudíž pomocí tohoto formátu není možná.

## 7.7 Výsledky testování

Výsledky testování jsou zobrazeny na Obrázku 13, který porovnává jednotlivé formáty a jejich požadované vlastnosti. Z testování vyplývá, že pro převod statických objektů, včetně modelu kvadrokoptéry a postavy, je nejvhodnější formát FBX. Ten udrží požadované rozložení objektů po scéně. Nepřevede ale stoprocentně materiálové nastavení. Pro tento typ konverze se lepé hodí formát gITF, který v tomto ohledu vykazuje daleko lepší vlastnosti. Pro převod Fluid simulace je nejvhodnější formát Alembic, který je jako jediný schopný tento typ simulací převést.

Osvědčilo se i využití rozšíření programu Blender „Blender for Unreal Engine 4“, které usnadňuje export komplexních modelů s vysokým počtem objektů. Jedná se především o

ulehčení samotného procesu, kdy lze exportovat jak statické modely pomocí FBX formátu, tak i simulace pomocí formátu Alembic a následně je uložit s předponou podle druhu modelu.

Tabulka 3: Srovnání konverzních formátů

| | FBX | OBJ | Alembic | gITF | B for UE4 addon* | USD |
|---|---|---|---|---|---|---|
| Statické mesh objekty | ✓ | ⚠ | ⚠ | ⚠ | ✓ | ⚠ |
| Materiály a textury | ⚠ | ⚠ | ✗ | ✓ | ⚠ | ✗ |
| UV mapping | ✓ | ✓ | ⚠ | ✓ | ✓ | ✗ |
| Animace postavy | ✓ | ✓ | ⚠ | ⚠ | ✓ | ⚠ |
| Rigid body animace | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Fluid simulace | ✗ | ✗ | ✓ | ✗ | ⚠ | ✗ |

✓ Dokáže ✗ Nedokáže ⚠ S omezením

*Blender for UnrealEngine 4 addon s podporou FBX

Vývojáři z Epic Games (vydavatelé Unreal Enginu) momentálně pracují na rozšíření pro Blender s názvem „Send to Unreal“, které by mělo propojit prostředí těchto dvou programů. Počítá se s interakcí, při které se modely vytvořené v Blenderu nebudou muset pokaždé do enginu reimportovat, ale převod bude prováděn v reálném čase s využitím FBX formátu. Samozřejmostí bude také převod materiálů, částicových systémů i animací. Momentálně je toto rozšíření ve fázi rozpracování a dochází k implementaci požadavků uživatelů na funkčnost. Výhledové dokončení je do konce roku 2020.

# 8 IMPLEMENTACE MODELU V UNREAL ENGINU

Pro sestavení modelu prostředí byla využita předem vytvořená šablona Unreal Enginu z pohledu třetí osoby, kde je kamera umístěna za a lehce nad ovládanou postavou. Pro využití možnosti ovládání byla postava ze šablony nahrazena vytvořeným modelem kvadrokoptéry. Jak se model pohybuje pomocí klávesnice, kamera sleduje jeho pohyb. Do zmíněné šablony byly poté naimportovány zbylé vytvořené modely. Celý postup importu a nastavení v Unreal Enginu je popsán v následujících podkapitolách.

## 8.1 Kvadrokoptéra s PID regulátory

Převod modelu kvadrokoptéry vytvořeného v programu Blender do Unreal Enginu je jednoduchý, jedná se totiž o statický mesh objekt bez dynamických či animovaných prvků. Všechna nastavení budou v tomto případě probíhat až ve zvoleném enginu. Cílem je mít kompletně ovladatelný model kvadrokoptéry pohybující se ve vytvořeném prostředí. Pro přesnější simulaci jsou pak pohyby modelu stabilizovány pomocí PID regulátorů. K nastavení chování, ovládání a ke stabilizaci slouží systém vizuálních skriptů Blueprint integrovaný ve vývojovém prostředí UE4.

Nastavení kvadrokoptéry lze rozdělit do dvou částí:

1) **Nastavení modelu**

   Zde se nastavuje vizuální stránka modelu. K tomu slouží editor „Viewport“, což je v podstatě 3D modelovací program uvnitř Unreal Enginu. Slouží k přidávání nových objektů či komponent, nebo k úpravě importovaných částí. Pro kvadrokoptéru je zde například nastaveno ohraničení kolizí (neviditelná oblast definující, v jakém rozsahu může model interagovat s okolním prostředím při nárazu) nebo také kamera neustále sledující pohyb modelu, tzv. pohled třetí osoby.

2) **Nastavení chování**

   K nastavení chování slouží část Blueprint skriptu nazvaná „EventGraph“. Je to v podstatě zobrazení událostí, které se stanou s modelem ať už v závislosti na čase, nebo jiné vstupní veličině. V případě kvadrokoptéry je zde nastaveno, že od startu simulace se vizuálně roztočí rotory (otočení lopatek kolem svého středu nemá vliv na výsledné chování). Nejdůležitější je ovšem funkce „EventTick“, kdy se s každým snímkem simulace

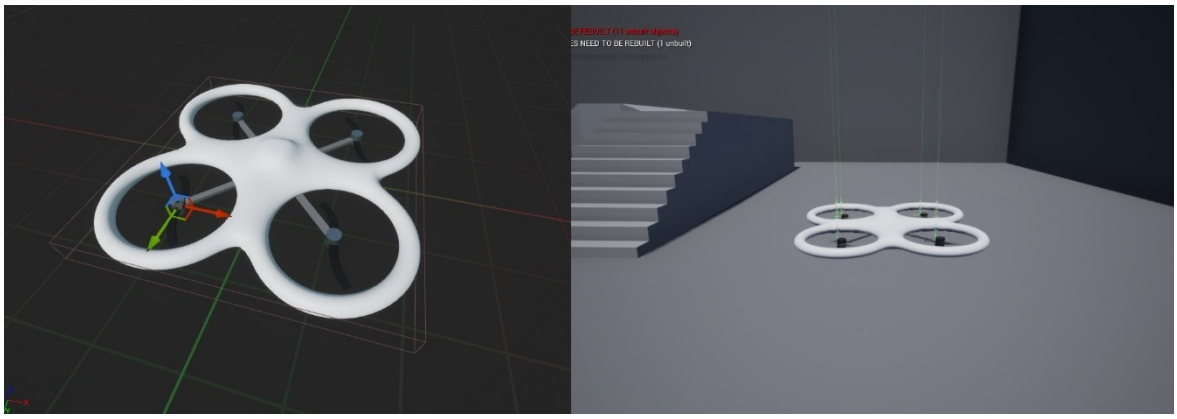
generuje čas „DeltaTime“ – de facto se generuje impuls, na který reagují další vytvořené funkce sloužící k ovládání modelu.

Pro korektní znázornění pohybu kvadrokoptéry se použijí Eulerovy kinematické rovnice popisující rotaci tuhého tělesa a konkrétně především trojice Eulerových úhlů udávající jednoznačné otočení v prostoru. Jsou to:

- rotační úhel φ (v angl. „Roll“) – rotace kolem osy X
- nutační úhel θ (v angl. „Pitch“) – rotace kolem osy Y
- precesní úhel ψ (v angl. „Yaw“) – rotace kolem osy Z

### 8.1.1 Integrace 3D modelu

K převodu modelu kvadrokoptéry z Blenderu do UE4 poslouží formát FBX, jakožto nejjednodušší způsob převodu pro statické objekty, které mají dané rozměry a textury. K převedenému modelu je nutné v enginu implementovat vlastní objekty, které mohou interagovat s vývojovým prostředím Blueprint. V tomto případě to budou čtyři jednoduché válce znázorňující rotory kvadrokoptéry, na které budou působit jednotlivé generované síly. Zde je důležité dbát ohled na rovnoměrné rozložení válců rotorů od středu modelu tak, aby nedocházelo k vychýlení kvadrokoptéry od své osy. Na každý tento válec je implementována konstanta „Engine“ virtuálního motoru ovládající model. Zde se tedy definuje, kde přesně (na jakých souřadnicích) mají působit vytvořené síly.



Obrázek 14: Model kvadrokoptéry v Unreal Enginu

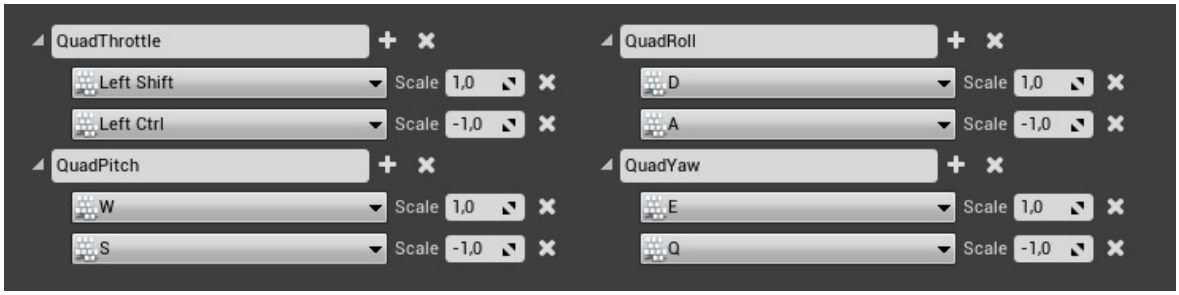
### 8.1.2 Ovládání kvadrokoptéry

V Unreal Enginu se používá pro ovládání pomocí vstupů z klávesnice tzv. mapování os a akcí. U mapování akcí (v angl. „Action Mapping“) se jedná o jednotkový stisk klávesy, zato u mapování os (v angl. „Axis Mapping“) může mít hodnota svůj rozsah v závislosti na délce

stisku dané klávesy. Pro ovládání kvadrokoptéry je proto výhodnější použití mapování os vstupních dat z klávesnice v této podobě:

- Nastavení výšky („Altitude“): Levý SHIFT (nahoru) + levý CTRL (dolů)
- Přímý pohyb („Pitch“): „W“ (pohyb dopředu) + „S“ (pohyb dozadu)
- Pohyb do boku („Roll“): „A“ (pohyb doleva) + „D“ (pohyb doprava)
- Otočení na místě („Yaw“): „Q“ (natočení doleva) + „E“ (natočení doprava)

Ke každému ze čtyř pohybů je v Blueprint vytvořena funkce pro jejich ovládání, která kontroluje vstupní parametry z klávesnice a následně aplikuje síly na konkrétní motory podle zvoleného pohybu. Zároveň má každý tento pohyb funkci pro jejich stabilizaci takovou, aby se po aplikovaní sil vrátil do rovnovážné polohy a nedošlo k překmitu, či nekonečnému pohybu.




Obrázek 15: Popis ovládání kvadrokoptéry pomocí klávesnice

**Handle Throttle**

Funkce pro nastavení zvolené výšky. Jedná se ve skutečnosti o pohyb z aktuální výšky modelu do požadované podle délky držení klávesy k tomu určené. Funkce získá aktuální polohu modelu, aplikuje virtuální síly na vytvořené pole motorů (tj. cyklicky po dobu stisku klávesy na všechny čtyři motory) a fyzicky přesune model do požadované polohy. Celý pohyb je pak stabilizován podle nastavení příslušného PID regulátoru.

**Handle Pitch**

Funkce pro ovládání přímého pohybu. Při stisku příslušné klávesy pro pohyb vpřed „W“ se začnou generovat síly na konkrétní dva motory, které uvedou model do pohybu v požadovaném směru, na zbylé dva se pak aplikuje síla inverzní. U kvadrokoptéry jsou tedy hlavními motory pro pohyb vpřed „Engine 1“ a „Engine 3“, to znamená, že „Engine 2“ a „Engine 4“ mají stejné, ale záporné hodnoty. Při stisku klávesy pro pohyb vzad „S“ jsou výstupní

hodnoty inverzní k hodnotě pro pohyb vpřed (jak je znázorněno na obrázku). To způsobí obrácení hodnot těchto dvojic motorů a model se pohybuje opačným směrem. Pokud není generován žádný signál stisku, funkce stabilizuje model v závislosti na aktuálních hodnotách natočení, to znamená, že si vyžádá od integrované funkce enginu hodnotu natočení od osy Y („Pitch“). Pokud jsou hodnoty odlišné od nuly (0 = stabilizovaná poloha), stabilizační funkce aktivuje pohyb v příslušné ose Y působící proti této získané hodnotě v závislosti na nastavení PID regulátoru. Tato funkce působí do té doby, než se model ustálí ve stabilizované poloze.

**Handle Roll**

Funkce pro ovládání pohybu do boku je obdobou funkce pro přímý pohyb. Rozdíl je v označení hlavních, tedy aktivních motorů, které budou udávat směr. U popsaného modelu se pro pohyb vpravo stiskem klávesy „D“ využijí motory „Engine 1“ a „Engine 4“, zbylé dva budou inverzní. Pro pohyb vlevo klávesou „A“ se hodnoty obrátí. Stabilizace také probíhá obdobně, jen se mohou lišit hodnoty nastavení PID regulátoru.

**Handle Yaw**

Funkce pro rotaci na místě. Svým způsobem se podobá funkci pro změnu aktuální výšky modelu. Tentokrát je ovšem potřeba změnit ne polohu, ale rotaci modelu v souřadnici Z. Podle stisku klávesy dojde k rotaci buď po směru hodinových ručiček, nebo proti. Stabilizace probíhá obdobně jako v předchozích případech, jen s odlišným PID regulátorem.

Funkce Handle Pitch a Handle Roll mají navíc každá ještě jeden PID regulátor, jenž omezuje posun modelu způsobený působícími silami. Jedná se jen o doplněk k přesnější simulaci a model by fungoval i bez těchto regulátorů.

V modelu vytvořeném v této práci je proto zabudováno celkem šest PID regulátorů (pro funkce Handle Throttle a Handle Yaw vždy jeden a pro Handle Pitch a Handle Roll dva).

### 8.1.3 Realizace PID regulátorů

Pro správné nastavení chování modelu je potřeba doplnit do návrhu řídící systémy, regulátory, jež budou stabilizovat pohyb kvadrokoptéry ve vytvořeném prostředí. Při návrhu se využije teoretických poznatků popsaných v kapitole č. 5.

Pro realizaci regulátoru v Unreal Enginu poslouží rovnice (2) ideálního PID regulátoru a její následná úprava na diskrétní tvar ve formě:

$$u(k) = K_P e(k) + \sum K_I e(k)\Delta t + K_D \frac{e(k) - e(k-1)}{\Delta t} \tag{7}$$

Zjednodušený přepis proměnných pro použití v Unreal Enginu vypadá následovně:

$$Vystup = (P \times Error) + \left(I \times (IntegralPrior + Error \times DeltaTime)\right) + \left(D \cdot \frac{Error - ErrorPrior}{DeltaTime}\right) + Const \tag{8}$$

Hodnoty označené jako „**Prior**“ jsou hodnoty předchozího stavu (kroku), proto jsou při inicializaci nastaveny na nulu:

$$\boldsymbol{ErrorPrior} = 0$$
$$\boldsymbol{IntegralPrior} = 0 \tag{9}$$

**P**, **I** a **D** jsou vstupní hodnoty regulátoru, nastavení hodnot je individuální pro každý pohyb.

Pomocná proměnná **„Const“** zabraňuje, aby byl výsledný výstup nulový.

Regulační odchylka **Error** je v tomto případě rozdíl požadované hodnoty od aktuální, konkrétně například u nastavení výšky je to rozdíl polohy, kam se má kvadrokoptéra přesunout a její aktuální polohy:

$$Error = DesiredValue - ActualValue \tag{10}$$

Pomocná proměnná **pIntegral** pro výpočet vnitřní části integračního členu:

$$pIntegral = IntegralPrior + Error \cdot DeltaTime \tag{11}$$

Pomocná proměnná **pDerivate** pro výpočet derivačního členu:

$$pDerivate = \frac{(Error - ErrorPrior)}{DeltaTime} \tag{12}$$

Výsledek rovnice PID regulátoru je pak návratovou hodnotou funkce **PID_Calculations:**

$$OutForce = P \cdot Error + I \cdot pIntegral + D \cdot pDerivate + Const \tag{13}$$

Na konci funkce je nutné doplnit další krok iterace:

$$ErrorPrior = Error$$
$$IntegralPrior = Integral \tag{14}$$

Podle tohoto přepisu se pak systém přepíše do prostředí vizuálních skriptů Blueprint v Unreal Enginu. Spojováním jednotlivých bloků vzniklých proměnných a jejich nastavením dojde k vytvoření kompletní funkce **PID_Calculations** pro výpočet PID regulátoru. Celostránkové zobrazení této funkce je k nalezení v příloze P II.

Na přiloženém DVD jsou graficky zdokumentovány všechny vytvořené funkce pro řízení kvadrokoptéry v prostředí Unreal Enginu s využitím Blueprint skriptování.

## 8.2 Import prostředí

Pro import do prostředí UE se využije několik různých formátů v závislosti na modelech, tak jak již bylo naznačeno v kapitole č. 6. Pro statické objekty prostředí, model kvadrokopotéry a postavy se použije FBX formát. Pro převod Fluid simulace se využije formát Alembic. V tomto případě by šlo využít i rozšíření programu Blender „Blender for Unreal Engine 4“, jež usnadňuje export komplexních modelů s vysokým počtem objektů.


Obrázek 16: Import modelu do prostředí Unreal Enginu

### 8.2.1 Statické objekty

Pro převod statických objektů se nejlépe hodí klasický FBX formát, případně rozšíření „Blender for Unreal Engine“. Tento formát je vhodný především proto, že dokáže korektně rozestavit modely po scéně podle zdrojového souboru. Je nutné však dávat pozor na měřítko, Blender totiž využívá jiný měřící systém než Unreal Engine. Samotný import je jednoduchý, v dialogovém okně lze pro modely nastavit jednoduché transformační nastavení (rotace, měřítko), případně zda se mají konverzí vytvořit nové materiály uvnitř prostředí enginu. Při importu většího počtu modelů je nutné počítat s menším časovým prodlením.


Obrázek 17: Převod modelu automobilu

### 8.2.2 UV mapa, materiály a textury

Podle zjištěných informací není formát FBX příliš vhodný pro převod klasických materiálových vlastností, při kterých se pracuje s nastavením barev, s jejich plastičností, emisí či průhledností. Pracuje ale efektně s UV mapováním a texturami. Je tedy nutné počítat s tím, že se klasické materiály nepřevedou a jejich následnou korekturu bude nutné poté provést rovnou v Unreal Enginu. Nebo se hodí využít formát gITF jen pro převod těchto materiálů a jejich následné navázání na konkrétní objekty (převod statických objektů pomocí gITF není vhodný). Docílí se tak přesnější podoby s původním modelem.

### 8.2.3 Částicové systémy

Jak již bylo zmíněno v kapitole 7.1, formát FBX lze využít pro převod částicových systémů. Je ovšem nutné brát ohled na počet emitovaných objektů, protože každý takový objekt se importuje do Unreal Enginu samostatně. Při větším množství by bylo vhodnější spojit tyto objekty do jednoho. V případě vytvořeného modelu není nutné brát ohled na toto omezení, protože počet emitovaných objektů je minimální. Trochu jiné pravidlo platí pro druhý typ částicových systémů.

Pro převedení typu Hair je totiž nutné napřed tyto systémy konvertovat pomocí modifikátorů v programu Blender na statické mesh objekty. Pokud by se tak nestalo, tento typ by se nepřevedl. Všechny částice vytvořeného objektu se spojí do jednoho celku a ten již dokáže formát FBX převést. Zároveň pro oba typy platí, že se konvertují pouze základní materiálové vlastnosti objektů.


Obrázek 18: Převod částicových systému a UV mapování

### 8.2.4 Rigging postavy

Pro převedení vytvořené postavy z Blenderu do Unreal Enginu je nutné při exportu z Blenderu nastavit správné souřadnice pro objekt Armature, a to tak, aby primární osa pro vytvořené kosti byla ve směru -X a sekundární ve směru osy -Y. V případě chybně nastavených os by došlo k nekorektnímu nasazení vytvořeného modelu na kostru v enginu. Při importu pomocí FBX formátu je nutné dát enginu na vědomí, že se jedná o model postavy. V dialogovém okně importu se tedy zaškrtne volba „Skeletal Mesh“ a importovací typ „Geometry a skinning weights“. Pro navázání na model je nutné v části skeleton vybrat „UE4_Mannequin_Skeleton“. Tento model se využívá v Unreal Enginu pro představení ovládání z pohledu třetí osoby.

Pro zajištění animace převedeného modelu postavy bylo využito předem vytvořeného balíčku animačních sekvencí „Animation Starter Pack“, který je k dispozici v obchodě Marketplace vydavatelů enginu firmy Epic games. Kromě jiného je v tomto marketu celá řada dalších modelů a animací od vývojářů, ať již zdarma nebo za příplatek. Použitý animační balíček stačí stáhnout a vložit do složky s vytvořeným modelem, v enginu konkrétní složku najít a jednoduše přetáhnout na model. Klasický import nemusí fungovat, protože importér nepodporuje import formátu Unreal Asset s příponou „*.uasset“, ve kterém jsou animace vytvořeny. Animace postav se spustí ihned po startu simulace a opakují se ve smyčce.


Obrázek 19: Převod modelu postavy

### 8.2.5 Rigid body animace

Rigid body animaci se bohužel žádným způsobem nepodařilo korektně převést do prostředí Unreal Enginu. Při použití FBX, OBJ a Alembic formátů se převedou jen statické prvky, přesněji první snímek animace vyexportované z Blenderu. Ostatní formáty se nedají použit ani pro statické přenesení kvůli jejich tendenci slučovat objekty na jedno místo.

Teoreticky je ovšem převod Rigid body animace možný. Pro jednoduší animace, například volný pád nějakého předmětu, lze využít prvek Armature obdobně jako v případě tvorby postavy. Na objekt, který tvoří aktivní prvek animace, se naváže v Pose editoru kost ("bone") a duplikují se na ni vlastnosti aktivního objektu. Po doplnění klíčových snímků na časové ose se může z toho aktivního objektu odstranit nastavení rigid body. Všechny prvky animace v podstatě převzal vytvořený prvek typu Armature a není již nutné toto nastavení dodržovat. Poté se může model vyexportovat ve formátu FBX s podobným nastavením jako v případě tvorby postavy. Při importu do Unreal Engine stačí zatrhnout, že se jedná o model postavy "Skeletal Mesh" a jeho animaci. Převedení vytvořeného modelu s bourací koulí se tímto způsobem však nepodařilo.

### 8.2.6 Fluid simulace

Pro převod simulace tekutiny poslouží exportní typ Alembic. Při importu do Unreal Enginu je možné vybrat ze dvou možností typu převodu (ve skutečnosti jsou tři, ale možnost Skeletal není pro simulaci vhodná). Pro převod vytvořeného modelu se využije těchto typů:

- Statické objekty (v angl. „Static Mesh“)
- Geometrie (v angl. „Geometry Cache (Experimental“))

Na základě těchto možností importu souboru je vhodné si simulaci, pro korektní převedení, rozdělit na dvě části. Část statickou, ve které budou objekty, které se v průběhu simulace žádným způsobem nemění (například vytvořený efektor, či podstavec). Druhou část pak bude tvořit simulovaná tekutina, v Blenderu zastoupena prvkem Doména. Vytvořený soubor simulace typu Alembic je vhodné importovat dvakrát, pro každou z těchto částí zvlášť. Pokud by se importovaly všechny objekty jedním způsobem, nedošlo by ke korektnímu nastavení simulace. Při převodu všech objektů jako statických by se neprovedla simulace. Objekt Domény by ztratil svou animaci a stal by se pevným objektem zastaveným na prvním snímku simulace. Naopak při převodu všech objektů pomocí typu Geometrie, by se sice simulace provedla, ale všechny vytvořené objekty by se spojily do jednoho. Nešlo by tak nastavit jednotlivé materiálové vlastnosti objektů.

K převodu statických částí modelu lze využít i formát FBX, při importu ale můžou mít jednotlivé části odlišné souřadnice a nemusely by na sebe navazovat. Rozdělením na dva importované soubory lze docílit korektního převodu simulace z Blenderu do Unreal Enginu. Je nutné ale počítat i s tím, že se formátem Alembic nepřevedou materiálové vlastnosti simulace. Buď se tedy materiál použitý v simulaci vytvoří nově v Unreal Enginu, nebo se naimportuje zvlášť FBX formátem, v lepším případě gITF formátem, a nasadí na model pro simulaci.

Při importu typu Alembic do Unreal Engine dochází ke zmenšení a k rotaci modelu. Důsledkem je nepřesná implementace. Pro napravení této chyby importu je nutné modelu změnit měřítko z hodnoty 1 na 100 na všech třech osách a poupravit rotaci modelu o –90° v ose X.


Obrázek 20: Převod Fluid simulace

# ZÁVĚR

Tato bakalářská práce se zabývá tématem 3D počítačové grafiky a jejího možného využití při návrhu vizualizačních simulací v informačních systémech řízení a regulace. Hlavní přidanou hodnotou práce je doporučení postupů a vhodných konverzních formátů při tvorbě 3D modelů v modelovacích programech a následná implementace 3D modelů při návrhu simulací v grafickém enginu. Vzhledem k tomu, že všech výsledků analýzy této práce bylo dosaženo pomocí volně přístupných programů, které nejsou zatíženy licenčními omezeními, může s nimi pracovat i běžný uživatel.

V praktické části práce byl vytvořen komplexní model prostředí v 3D modelovacím programu Blender, který byl následně převeden pomocí různých testovaných konverzních formátů do prostředí Unreal Enginu. Výsledkem testování je celkové srovnání, přehledně zpracované formou tabulky a doplněné o doporučení, jakými formáty lze nejvhodněji převádět různé typy vytvořených objektů a animací. Při testování bylo zjištěno, že pro převod statických objektů je nejvhodnější formát FBX díky jeho schopnosti správně udržet rozložení objektů po scéně. Pro převedení materiálových vlastností je zase nejvhodnější formát gITF. Pro fluid simulace je naopak vhodný formát Alembic. Zároveň lze pro složitější prostředí využít uživatelské rozšíření „Blender for Unreal Engine“, které usnadňuje konverzi vysokého počtu objektů s využitím FBX formátu. Zjištěním při testování také bylo, že formát USD není vyhovující pro konverzi ani jednoho z převáděných modelů.

Jednou z hlavních částí této práce je vytvoření řiditelného modelu kvadrokoptéry, který byl sestrojen s využitím teorie automatizace regulovaných soustav. Tato teorie byla aplikována při návrhu řízení pohybu pomocí PID regulátorů. Díky zabudovaným regulátorům bylo dosaženo řiditelné simulace chovaní kvadrokoptéry, která je umístěna do prostředí převedeného z externího modelovacího programu.

Vhodným rozšířením tohoto tématu by byla podrobnější analýza dalších typů vytvořených modelů a testování jejich konverze. Zvolený 3D modelovací program má nepřeberné možnosti tvorby různých i složitějších 3D objektů. Zároveň by bylo možné rozšířit i následné simulační modely v grafickém enginu o další řiditelné prostředky. Potenciálem do budoucna by mohlo být rozšíření pro Blender s názvem „Send to Unreal“ od vývojářů Unreal Enginu, které by výrazně usnadnilo konverzi mezi jednotlivými programy. Bohužel při tvorbě této práce bylo toto rozšíření stále ve formě testování a nebylo jej proto možné zařadit do testovaných formátů. Určitě ale bude stát za pozornost.

# SEZNAM POUŽITÉ LITERATURY

[1] Eck, David J. Introduction to Computer Graphics. Creative Commons Attribution 2.0, 2018. [online]. [cit. 2020-05-19]. Dostupné z: http://math.hws.edu/eck/cs424/downloads/graphicsbook-linked.pdf

[2] CARTER, Nathan. Introduction to the Mathematics of Computer Graphics. Mathematical Association of America, 2016. ISBN 1614441227.

[3] HEES, H., 3D Computer Graphics: 3Dc Z-fighting. Pediapress. 2006. BookID: uidrpqgpwhpwilvl

[4] VINCE, John. Essential Computer Animation fast: How to Understand the Techniques and Potential of Computer Animation. London: Springer Science & Business Media, 2000. ISBN 9781852331412.

[5] ŽÁRA, Jiří. Moderní počítačová grafika. 2., přeprac. a rozš. vyd. Brno: Computer Press, 2004. ISBN 80-251-0454-0

[6] POKORNÝ, Pavel. Blender: naučte se 3D grafiku. 2., aktualiz. a rozš. vyd. Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-244-2.

[7] SLICK, Justin. What Is 3D Rendering in the CG Pipeline? [online]. Lifewire, 2019 [cit. 2020-07-27]. Dostupné z: https://www.lifewire.com/what-is-rendering-1954

[8] BLAIN, John M. The complete guide to Blender graphics: computer modeling and animation. Fifth edition. Boca Raton; CRC Press, 2019. Blender. ISBN 978-0-367-18474-2.

[9] SIDDI, Francesco. Blender by the Numbers – 2019. Blender [online]. 2020 [cit. 2020-07-27]. Dostupné z: https://www.blender.org/press/blender-by-the-numbers-2019/

[10] What is Autodesk Maya? Edulearn [online]. 2016 [cit. 2020-04-13]. Dostupné z: http://www.edulearn.com/article/what_is_autodesk_maya.html

[11] BOJC, Ashley. Maya vs 3DS Max vs Cinema 4D. Triplet 3D [online]. 2014 [cit. 2020-07-27]. Dostupné z: http://www.triplet3d.com/maya-vs-3ds-max-vs-cinema-4d

[12] PETTY, Josh. What is 3ds Max & What is it Used For? Concept Art Empire [online]. [cit. 2020-07-27]. Dostupné z: https://conceptartempire.com/what-is-3ds-max/

[13] KUTZ, Sarah. Comparison: the Top 6 Most Popular 3D Modeling Software. Medium [online]. iMeshup, 2018 [cit. 2020-07-27]. Dostupné z: https://medium.com/imeshup/comparison-the-top-6-most-popular-3d-modeling-soft-1c6a9ed204a4

[14] SIMONDS, Ben. Blender Master Class: A Hands-on Guide to Modeling, Sculpting, Materials, and Rendering. No Starch Press, 2013. ISBN 9781593274771.

[15] Workspaces. Blender 2.82 Reference Manual [online]. Blender Documentation Team [cit. 2020-07-28]. Dostupné z: https://docs.blender.org/manual/en/latest/interface/window_system/workspaces.html

[16] Editors. Blender 2.82 Reference Manual [online]. Blender Documentation Team [cit. 2020-07-28]. Dostupné z: https://docs.blender.org/manual/en/latest/editors/index.html

[17] Primitives. Blender 2.82 Reference Manual [online]. Blender Documentation Team [cit. 2020-07-28]. Dostupné z: https://docs.blender.org/manual/en/2.82/modeling/meshes/primitives.html

[18] Meshes - Introduction. Blender 2.82 Reference Manual [online]. Blender Documentation Team [cit. 2020-07-28]. Dostupné z: https://docs.blender.org/manual/en/2.82/modeling/meshes/introduction.html

[19] Eevee - Introduction. Blender 2.82 Reference Manual [online]. Blender Documentation Team [cit. 2020-07-28]. Dostupné z: https://docs.blender.org/manual/en/2.82/render/eevee/introduction.html

[20] Cycles - Introduction. Blender 2.82 Reference Manual [online]. Blender Documentation Team [cit. 2020-07-28]. Dostupné z: https://docs.blender.org/manual/en/2.82/render/cycles/introduction.html

[21] Workbench - Introduction. Blender 2.82 Reference Manual [online]. Blender Documentation Team [cit. 2020-07-28]. Dostupné z: https://docs.blender.org/manual/en/2.82/render/workbench/introduction.html

[22] LEE, Joanna. Learning Unreal Engine Game Development. Birmingham: Packt Publishing, 2016. ISBN 9781784398156.

[23] Framework Class Relationships. Unreal Engine 4 Documentation [online]. Epic Games [cit. 2020-07-28]. Dostupné z: https://docs.unrealengine.com/en-US/Gameplay/Framework/QuickReference/index.html

[24] ŠMÍD, Antonín. Srovnání Unity a Unreal Enginu. Praha, 2017. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická. Vedoucí práce Doc. Ing. Jiří Bittner, Ph.D.

[25] THORN, Alan. Game Engine Design and Implementation. Jones & Bartlett Learning, 2011. ISBN 9781449666484.

[26] SHANNON, Tom. Unreal Engine 4 for design visualization: developing stunning interactive visualizations, animations, and renderings. Boston: Addison-Wesley, 2018. ISBN 0134680707.

[27] Frequently Asked Questions. Unreal Engine [online]. [cit. 2020-07-28]. Dostupné z: https://www.unrealengine.com/en-US/faq

[28] Unreal Editor Manual - Level Editor. Unreal Engine 4 Documentation [online]. Unreal Engine [cit. 2020-07-28]. Dostupné z: https://docs.unrealengine.com/en-US/Engine/UI/LevelEditor/index.html

[29] Get Started with UE4 - Tools and Editors. Unreal Engine 4 Documentation [online]. Unreal Engine [cit. 2020-07-28]. Dostupné z: https://docs.unrealengine.com/en-US/GettingStarted/SubEditors/index.html

[30] Blueprints Visual Scripting. Unreal Engine 4 Documentation [online]. Unreal Engine [cit. 2020-07-28]. Dostupné z: https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html

[31] FARRIS, Jeff. Forging new paths for filmmakers on "The Mandalorian." Unreal Engine [online]. 2020 [cit. 2020-07-19]. Dostupné z: https://www.unrealengine.com/en-US/blog/forging-new-paths-for-filmmakers-on-the-mandalorian

[32] A first look at Unreal Engine 5. Unreal Engine [online]. Unreal Engine [cit. 2020-07-28]. Dostupné z: (https://www.unrealengine.com/en-US/blog/a-first-look-at-unreal-engine-5)

[33] FBX. Blender 2.82 Reference Manual [online]. Blender Documentation Team [cit. 2020-07-27]. Dostupné z: https://docs.blender.org/manual/en/2.82/addons/import_export/scene_fbx.html

[34] Wavefront OBJ. Blender 2.82 Reference Manual [online]. Blender Documentation Team [cit. 2020-07-27]. Dostupné z: https://docs.blender.org/manual/en/2.82/addons/import_export/scene_obj.html

[35] Alembic 1.7.0 [online]. Sony Pictures Imageworks Inc. and Industrial Light & Magic, a division of Lucasfilm Entertainment Company, 2016 [cit. 2020-07-27]. Dostupné z: https://www.alembic.io

[36] glTF 2.0. Blender 2.82 Reference Manual [online]. Blender Documentation Team [cit. 2020-07-27]. Dostupné z: https://docs.blender.org/manual/en/2.82/addons/import_export/scene_gltf2.html

[37] Blender For Unreal Engine [online]. GitHub, 2019 [cit. 2020-07-27]. Dostupné z: https://github.com/xavier150/Blender-For-UnrealEngine-Addons

[38] Universal Scene Description. Blender 2.82 Reference Manual [online]. Blender Documentation Team [cit. 2020-07-27]. Dostupné z: https://docs.blender.org/manual/en/2.82/files/import_export/usd.html

[39] FBX Content Pipeline. Unreal Engine 4 Documentation [online]. Unreal Engine [cit. 2020-06-06]. Dostupné z: https://docs.unrealengine.com/en-US/Engine/Content/Importing/FBX/index.html

[40] NAVRÁTIL, Pavel. Automatizace: vybrané statě. Ve Zlíně: Univerzita Tomáše Bati ve Zlíně, 2011, 289 s. ISBN 9788073189358. Dostupné také z: http://hdl.handle.net/10563/18581

[41] ŠVARC, Ivan. Základy automatizace: Učební texty pro kombinovanou formu bakalářského studia [online]. Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2002 [cit. 2020-07-06]. Dostupné z: http://matlab.fei.tuke.sk/zar/subory/literatura/ZakladyAutomatizace_SVARC.pdf

[42] KOHANBASH, David. PID Control (with code), Verification, and Scheduling. Robots For Roboticists [online]. 2014 [cit. 2020-07-28]. Dostupné z: http://robotsforroboticists.com/pid-control/

# SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

| | |
|---|---|
| 2D | Dvoudimenzionální |
| 3D | Trojdimenzionální |
| AI | Umělá inteligence |
| BSDF | Obousměrná distribuční funkce odrazu světla (angl. bidirectional scattering distribution function) |
| CPU | Centrální procesorová jednotka |
| EEVEE | Extra Easy Visual Environment Engine |
| FBX | Filmbox – souborový formát |
| gITF | Graphics Library Transmission Format – grafický formát pro přenos 3D objektů |
| GPU | Grafická procesorová jednotka |
| HUD | Head-up display graficky zobrazující informace |
| LED | Elektroluminiscenční dioda |
| OBJ | Wavefront grafický formát |
| PBR | Fyzikálně založený rendering (angl. „Physically based rendering“) |
| PC | Osobní počítač |
| PSD | Proporcionálně-sumačně-diferenční regulátory |
| PID | Proporcionálně integračně derivační analogový regulátor |
| RGB | Barevný model Red-Green-Blue neboli červená-zelená-modrá |
| UE4/UE | Unreal Engine 4 |
| USD | Grafický formát Universal Scene Description |
| UV | "U" a "V" označují osy 2D textury při UV mapování |
| VFX | Zkratka pro vizuální efekty z anglického „Visual effects“ |
| VR | Virtuální realita |

# SEZNAM OBRÁZKŮ

Obrázek 1: Schéma klasického zobrazovacího řetězce [5] ..... 16
Obrázek 2: Základní pracovní plocha se čtyřmi editory ..... 22
Obrázek 3: Přehled editorů v Blenderu 2.82 [16] ..... 23
Obrázek 4: Přehled primitivních modelů [17] ..... 24
Obrázek 5: Přehled komponent 3D enginů ..... 26
Obrázek 6: Vztah herních tříd v herním rámci [23] (vlastní zpracování) ..... 27
Obrázek 7: Nejrozšířenější 3D enginy ..... 30
Obrázek 8: Základní rozložení Unreal Editoru [28] ..... 33
Obrázek 9: Využití vyrenderováného prostředí v UE4 místo zeleného plátna během natáčení seriálu The Mandalorian (Disney+, 2019) [31] ..... 35
Obrázek 10: Blokové znázornění regulátoru a regulované soustavy [40] ..... 38
Obrázek 11: Model automobilu vyrenderovaný pomocí Cycles ..... 45
Obrázek 12: Vytvořená postava v programu Blender ..... 46
Obrázek 13: Rigid body animace ..... 47
Obrázek 14: Model kvadrokoptéry v Unreal Enginu ..... 55
Obrázek 15: Popis ovládání kvadrokoptéry pomocí klávesnice ..... 56
Obrázek 16: Import modelu do prostředí Unreal Enginu ..... 60
Obrázek 17: Převod modelu automobilu ..... 61
Obrázek 18: Převod částicových systému a UV mapování ..... 62
Obrázek 19: Převod modelu postavy ..... 63
Obrázek 20: Převod Fluid simulace ..... 64

# SEZNAM TABULEK

Tabulka 1: Srovnání nejrozšířenějších 3D modelovacích programů [13] (vlastní zpracování) ..... 20
Tabulka 2: Srovnání nejrozšířenějších 3D enginů ..... 32
Tabulka 3: Srovnání konverzních formátů ..... 53

# SEZNAM PŘÍLOH

P I Obsah přiloženého DVD

P II Funkce výpočtu regulátoru v Blueprintu

P III Vytvořené prostředí v Blenderu

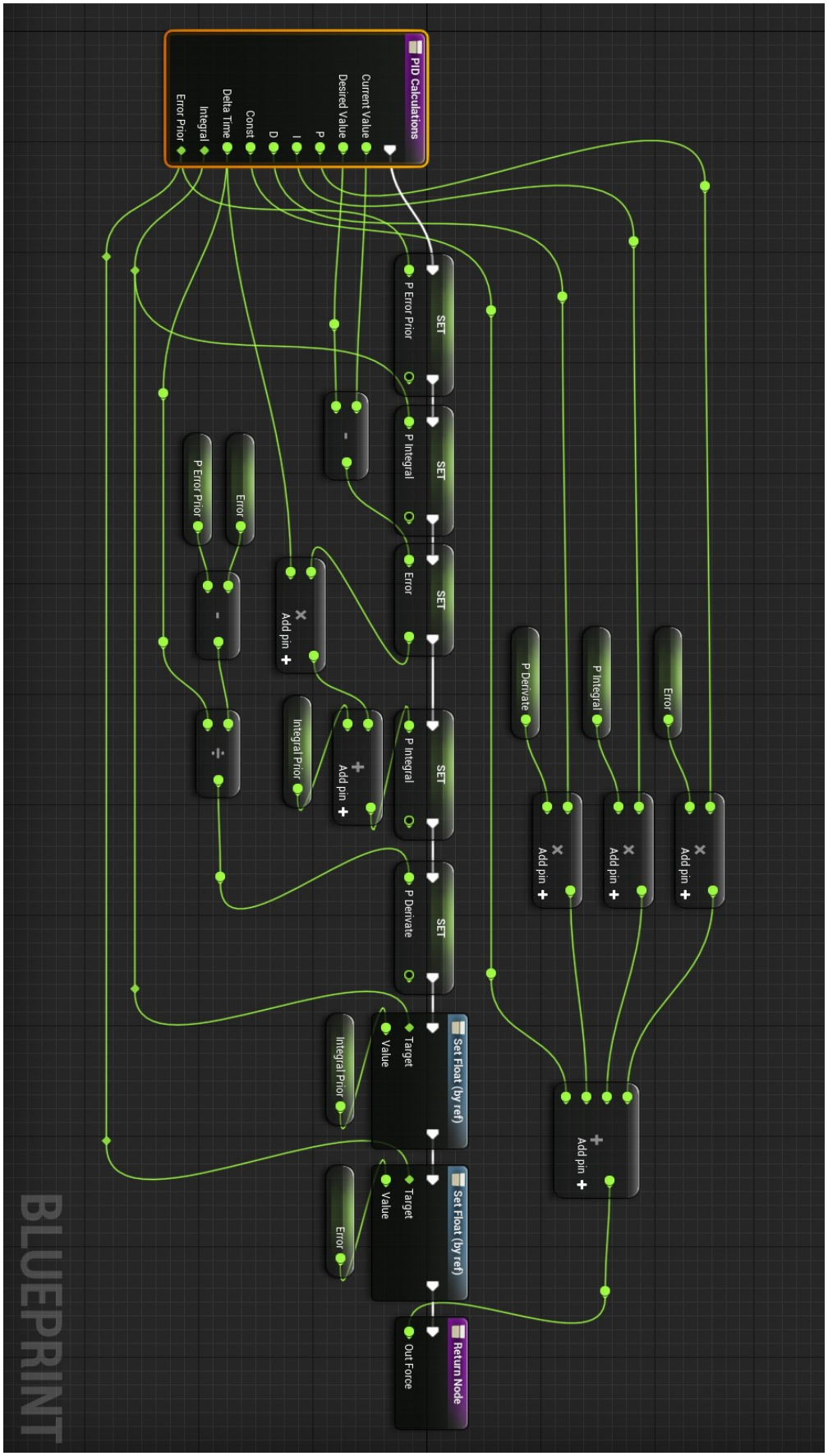
P IV Převedené prostředí do Unreal Enginu

# PŘÍLOHA P I: OBSAH PŘILOŽENÉHO DVD

Adresářová struktura přiloženého DVD je následující:

- Soubor **fulltext.pdf** obsahuje elektronickou verzi práce ve formátu PDF/A.
- Adresář **Blender_modely** obsahuje zdrojové soubory všech vytvořených modelů.
- Adresář **UE4_aplikace** obsahuje zdrojový soubor simulačního prostředí v Unreal Enginu s importovanými modely.
- Adresář **Obrazky** obsahuje grafickou dokumentaci bakalářské práce, včetně všech vytvořených funkcí v Blueprint

# PŘÍLOHA P II: FUNKCE VÝPOČTU REGULÁTORU V BLUEPRINT



# PŘÍLOHA P III: VYTVOŘENÉ PROSTŘEDÍ V BLENDERU

# PŘÍLOHA P IV: PŘEVEDENÉ PROSTŘEDÍ DO UNREAL ENGINU