Události současného Zlína v aplikaci virtuální reality

Bc. Tomáš Bukový

Diplomová práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

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

Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Tomáš Bukový**
Osobní číslo:	**A22291**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Události současného Zlína v aplikaci virtuální reality**
Téma práce anglicky:	**Events of Contemporary Zlín in a Virtual Reality Application**

Zásady pro vypracování

1. Vypracujte literární rešerši na programy, které budou použity v praktické části této práce.
2. Charakterizujte virtuální realitu a její implementaci v programech, které ji podporují.
3. Stávající aplikaci Zlína aktualizujte a rozšiřte o modely dalších důležitých staveb města.
4. Celý model optimalizujte pro použití ve virtuální realitě na headsetu Meta Quest.
5. Navrhněte a vytvořte interaktivní aplikaci virtuální reality.
6. Projekt rozšiřte o informace o důležitých stavbách a aktuálních událostech ve městě Zlín.

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

Seznam doporučené literatury:

1. *Unreal Engine documentation*. Online. Unrealengine.com. Dostupné z: https://docs.unrealengine.com/5.3/en-US/. [cit. 2023-11-17].
2. *Blender documentation*. Online. Blender.org. Dostupné z: https://docs.blender.org/. [cit. 2023-11-17].
3. *Developer Oculus – Documentation*. Online. Oculus.com. Dostupné z: https://developer.oculus.com/documentation/unreal/. [cit. 2023-11-17].
4. *Kalendář akci – Turistický informační portál města Zlína*. Online. Ic-zlin.cz. Dostupné z: http://www.ic-zlin.cz/25319--kalendar-akci. [cit. 2023-11-17].
5. PANGILINAN, Erin. *Creating Augmented and Virtual Realities: Theory and Practice for Next-Generation Spatial Computing*. United States: O'Reilly Media, 2019, 300 s. ISBN 978-149-2044-192.
6. GIANPIERO, Moioli. *Introduction to Blender 3.0: Learn Organic and Architectural Modeling, Lighting, Materials, Painting, Rendering, and Compositing with Blender*. United Kingdom: Apress, 2022, 466 s. ISBN 978-148-4279-533.
7. GONALO, Marques. *Elevating Game Experiences with Unreal Engine 5 – Second Edition: Bring your game ideas to life using the new Unreal Engine 5 and C++*. United Kingdom: Packt Publishing, 2022, 760 s. ISBN 978-180-3239-866
8. MCCAFFREY, Mitch. *Unreal Engine VR cookbook: developing virtual reality with UE4*. Boston: Addison-Wesley, 2017, 288 s. ISBN 01-346-4917-6.

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

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

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

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

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

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové 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 diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové 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 diplomovou 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 – diplomovou 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 diplomové 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 diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové 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 diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor;
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Tomáš Bukový, v. r.

Ve Zlíně, dne 13. 5. 2024 …….

podpis studenta

ABSTRAKT

Cílem této diplomové práce je převést již existující počítačovou aplikaci zabývající se vizualizací Zlína z pohledu třetí osoby, která byla vytvořena v rámci předcházející bakalářské práce, na interaktivní aplikaci virtuální reality. Aplikace bude umožňovat prohlížení aktuálních událostí a významných staveb dle jejich polohy zasazené do 3D modelu města. Program Blender byl použit pro úpravy modelu a výsledná aplikace byla vytvořena v programu Unreal Engine. Z Fakulty Aplikované informatiky byl pro testování aplikace zapůjčen headset pro virtuální realitu Meta Quest 1.

Klíčová slova: Zlín, Virtuální Realita, Unreal Engine, Meta Quest, Blender, Vizualizace, HMD

ABSTRACT

The aim of this thesis is to convert an existing computer application focused on the visualization of Zlín from a third-person perspective, which was created in the previous bachelor thesis, into an interactive virtual reality application. The application will allow viewing current events and important buildings according to their location embedded in a 3D model of the city. Blender was used to modify the model and the resulting application was created in Unreal Engine. The Meta Quest 1 virtual reality headset was borrowed from the Faculty of Applied Informatics for testing the application.

Keywords: Zlín, Virtual Reality, Unreal Engine, Meta Quest, Blender, Visualizaton, HMD

Tímto bych chtěl poděkovat Ing. Pavlu Pokornému, PhD, vedoucímu mé diplomové práce, za jeho cenné rady a zpětnou vazbu při vypracovávání této práce.

Dále bych chtěl poděkovat Fakultě aplikované informatiky Univerzity Tomáše Bati za zapůjčení headsetu Meta Quest pro testování aplikace.

Také bych rád poděkoval Městskému informačnímu a turistickému středisku města Zlín za poskytnutí dat událostí, které v práci využívám.

A nakonec bych chtěl poděkovat rodině za jejich podporu po celou dobu studia.

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

OBSAH

ÚVOD

Virtuální realita (VR) se stala jedním z nejdynamičtěji se rozvíjejících odvětví technologií v posledních letech, přinášející revoluční změny v různých oblastech lidského života. Její neustálý pokrok a široká škála aplikací ovlivňují způsob, jakým interagujeme se světem kolem sebe. Její implementace zasahuje jak do zábavního průmyslu v podobě her, tak také ke vzdělání, různými vizualizacemi a nebo třeba i lékařstvím.

Tato diplomová práce navazuje na předchozí bakalářskou práci, jenž měla za cíl vytvořit 3D model současného města Zlín a upravuje ji tak, aby šla spustit ve virtuální realitě. Aplikaci rozšířila a vytvořila v ní interaktivní prostředí, ve kterém je uživateli umožněno si prohlédnout zmenšený model města a prohlížet si jej. Dále bude uživateli umožněno prohlížet si události, které se ve městě Zlín konají s a ukázat jejich lokaci na samotném modelu a dále si například zobrazovat informace o budovách a dalších objektech důležitých pro historii města.

Teoretická část práce se zabývá programy, které byly použity. Pro modelování byl použit program Blender a pro tvorbu finální aplikace byl využit Unreal Engine 5. Dále zde byla popsána Virtuální realita, její historie a druhy headsetů, které ji podporuji. Nakonec zde je zmíněna implementace virtuální reality v programech, které ji podporují.

Praktická část se zabývá původní prací, modelem a její migrací na novější Unreal Engine. V prvních kapitolách je popsán stav předchozí aplikace a modelu a počátky tvorby aplikace virtuální reality. V dalších kapitolách je popsaná grafická aktualizace modelu následná optimalizace modelu pro použití ve Virtuální realitě. Poté se zabývá práci s daty o událostech a významných budovách. Následovala tvorba hlavní scény a nakonec byla popsána výsledná aplikace.

I. TEORETICKÁ ČÁST

1 UNREAL ENGINE

Unreal Engine (UE) je kompletní balíček nástrojů pro vývoj her, vizualizace automotive nebo architektury a další real-time aplikace. UE byl vytvořen a stále je vyvíjen společností EpicGames. V současnosti je ve své 5. verzi (5.3.2 v době psaní práce). Samotný engine je k dispozici ke stažení zdarma a veškerá jeho funkcionalita je v něm již obsažena. Pro stažení je ale nutné mít účet u společnosti EpicGames a nainstalovaný EpicGamesLauncher, oboje je také bez poplatku. [38]

1.1 Historie

V roce 1991 začal Tim Sneeney, zakladatel EpicGames, vytvářet editační nástroje pro svoji první hru. Všechno to začalo jednoduchou hříčkou ZZT, jednalo se o puzzle adventuru ve které bylo cílem probojovat se skrze různé monstra a vyřešit puzzle. V té době se nejednalo o revoluční hru ale programátorský přístup ke hře vytvářel podklady pro nejúchvatnější grafiku u her z dnešní doby. V devadesátkách praskal herní průmysl hrami naplněnými potenciálem, ale i ty největší hry byly tvořeny skupinkami lidí od samého základu v rámci přísného a složitého procesu. ZZT dělalo unikátním to, že v něm byl zabudovaný editor a skriptovací jazyk, díky kterému se mohl ZZT transformovat na vývojový kit pro jiné hry. Po prvotním úspěchu se hrou ZZT vytvořil s novým týmem o velikosti pěti členů novou hru, *Jill of the Jungle* a ta změnila vše. Tim věděl o významu herního enginu, který umožňoval výrazné uživatelské úpravy nad rámec tehdejších jednoduchých editorů map, ale co je důležitější, tento přístup položil koncepční rámec pro Unreal Engine, který způsobí revoluci v herním průmyslu. [10][11]

V roce 1995 začal Tim a jeho tým začali pracovat na první generaci Unreal Engine. Ten byl využit pro vývoj hry Unreal, jednalo se o střílečku z první osoby, které byly v té době velmi populární. Vývoj hry převzal filozofii, kterou Tim použil při vývoji ZZT a rozšířili ji o třetí dimenzi. Věděl, že flexibilita byla klíčová a trvalo to skoro tři a půl roku než byl hotový. Unreal obsahoval jako první polygonové postavy, detailní interiéry a exteriéry, tedy věci, které v dnešní době považujeme za standart. Zájem o Unreal Engine byl takový, že Epic ho začal nabízet a prodávat ostatním studiím dřív než vydal vlastní hru, která byla na něm založená. Tu až vydal 22. května roku 1998. Engine podporoval 16bitové barvy, volumetrickou mlhu, detailní texturing, a další techniky, které byly napřed před ostatními 3D hrami na trhu v té době. Studia Microprose a Legend Entertaiment byly mezi prvními, kteří si Unreal Engine licencovaly a hra *Duke Nukem Forever* v průběhu vývoje přešla od Queke enginu od společnosti ID na Unreal. [10]

V roce 1999 vyšla druhá hra, Unreal Tournament. I přes obrovský úspěch měla předchozí hra několik chyb spojených s enginem samotným, jako byly vysoké systémové nároky. Takže pro vývoj Unreal Tournamentu byl engine optimalizován pro méně výkonné počítače a byly opraveny i chyby spojené s Multiplayerem. Navíc byla přidaná podpora pro 24bit barvy, vyšší kvality textur, které neovlivňovaly negativně výkon. Tato snaha se vyplatila a Unreal Tournament patří mezi jednu z nejlepších her, který byla vytvořena a prodala přes 2 miliony kopii a na metacritics ji ocenily na 92 bodů.

Druhá verze Unreal enginu spatřila světlo světa v roce 2002 po boku hry American ´s army, jednalo se o multiplayerovou střílečku, která byla zdarma. Armáda Spojených států amerických a herní průmysl pracovali po dobu osmiletého kontraktu na vývoji této hry. Cílem bylo edukovat, informovat a naverbovat perspektivní vojáky. Zajímavostí je, že se jedná o první využití hry k naverbování do armády Spojených států amerických. Pokročilo také AI, jenž mimo pouhé stání a střílení se shlukovalo, pracovalo týmově a přišlo na pomoc, pokud bylo v nevýhodě. Novinky byly například přidání částicového systému, kostní animace, a po boku Unreal Tournament 2004 přišla i implementace řízení vozidel.

V roce 2006 vyšel Unreal Engine 3. První hra, která ho využila byla Gears of War, která u metacritic získala úctihodných 94 bodů. Byla chválena pro vizuální zpracování, prezentaci, ozvučení a koncept. UE3 přišel s řadou novinek, které těmto oceněním napomohli. I přes to, že byl založen na první generaci Unreal Enginu tak přišel s novým fyzikálním systémem, systémem pro ozvučení a nástroje byly samy o sobě více výkonné a schopnější. Největší změna oproti UE2 byla změna výpočtů, ty byly nyní prováděny po pixelech na místo vertexů. Tím se dalo docílit mnohem lepšího grafického zpravování, renderování se také vylepšeno a byla přidána podpora pro HDR, tedy vysoký dynamický rozsah. V roce 2010 přišla podpora pro iOS a Android. UE3 po celou dobu jeho cyklu dostával řadu aktualizací, které přidaly podporu pro ničitelné prostředí, simulaci davů a stereoscopické 3-D. Na UE3 bylo vytvořeno více než 300 her od velkých studií jako bylo EA, Ubisost nebo Activision. [11]

V roce 2014 vyšel Unreal Engine 4, jehož vývoj ale započal 11 let předtím. Největší novinkou bylo mnohem realističtější nasvícení ve 3D scénách a také přidání nového vizuálního skriptovacího jazyka *Bleurpints.* Celkově v UE4 vzniklo více než 400 her a nejvýznamnější přišla v roce 2017, při vzestupu žánru Battleroyal, a tou byla PUBG jenž trhala rekordy v počtu hráčů a byla následovaná hrou Fortnite stejného žánru.

V roce 2018 začal vývoj na pátou verzi Unreal Enginu. Ta vyšla v 5. dubna 2022 a jedná se zatím o poslední velkou verzi, která vyšla. Největší novinky byl Ninite, jenž

umožňoval importovat vysoce detailní fotografické zdrojové materiály do hry. Působivé je to, že zpracovává sám úroveň detailů odpovídající cílové platformě a vzdálenosti vykreslovaní, jenž ulehčí práci umělcům, jelikož se jedna o činnost, kterou museli jinak dělat ručně. Další novinou byl Lumin, ten se dá popsat jako plně dynamické řešení globálního osvětlení, které okamžitě reaguje na změny scény a světla Potenciálu UE5 bylo ukázáno v *The Matrix: Awakens Unreal Engine 5 experience*, jenž bylo demo, které posouvá hranice enginu na novou úroveň. Obsahuje biliony polygonů v obrovském otevřeném městě. [10] [11]

1.2 EpicGamesLauncher

Jedná se o platformu, která slouží k distribuci her a digitálního obsahu, obdobně jako další platformy jako je Steam, GOG nebo Origin. V sekci Store si můžou uživatelé kupovat hry, které jsou na této platformě nabízené. Na rozdíl od společnosti Steam požaduje společnost EpicGames nižší poplatky z prodeje her. Dalším výhodou, který si našel oblibu u herních fanoušků je, že každý měsíc nabízí nějakou z her zdarma, může se jednat o indie hru nebo také o AAA hru. Ta hra je pak spojená s vaším účtem a můžete ji hrát kdykoliv.

Platformu společnost EpicGames používá také pro šíření samotného UE, který má vlastní část aplikace vyhrazenou přímo pro sebe. Uživatelé si zde můžou stáhnout jednotlivé verze herního enginu (i starší), spravovat aktualizace či instalovat a zapínat různé rozšíření, které UE umožňuje. Jako v případě her se i zde můžou uživatelé dostat každý měsíc k několika assetům do UE zdarma, assety se můžou lišit obsahem a nebo cenou. Můžou to být assety v řádech korun nebo i tisíců korun. Assety máte poté spojeny s vaším účtem obdobně jako hry a můžete je kdykoliv využít ve svých projektech.

1.2.1 Popis uživatelského prostředí EpicGamesLauncheru

Prostředí EpicGames launcheru, se dá rozdělit do několika části, v této části se hlavně zaměřím na záložku Knihovna pro UE

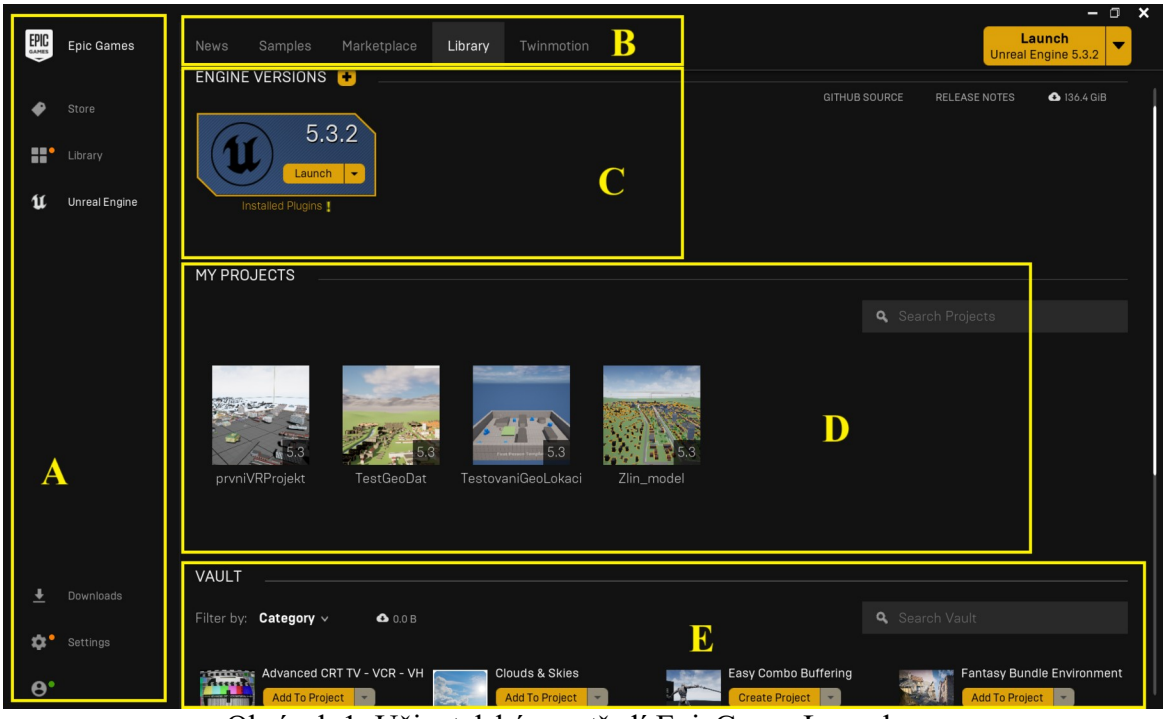
A) Záložky jednotlivých části programu, Obchod, Knihovna a záložka pro Unreal Engine, ve spodní části panelu se nachází zkratka pro nastavení, uživatelský profil a pro správu stahování

B) V této části můžeme přepínat mezi jednotlivými záložkami pro Novinky o UE, stahovat ukázkové mapy, prohlížet a nakupovat na marketplace, knihovna a odkaz pro twinmotion.

C) Přehled nainstalovaných verzí UE

D) jednotlivé projekty, které se nachází v PC

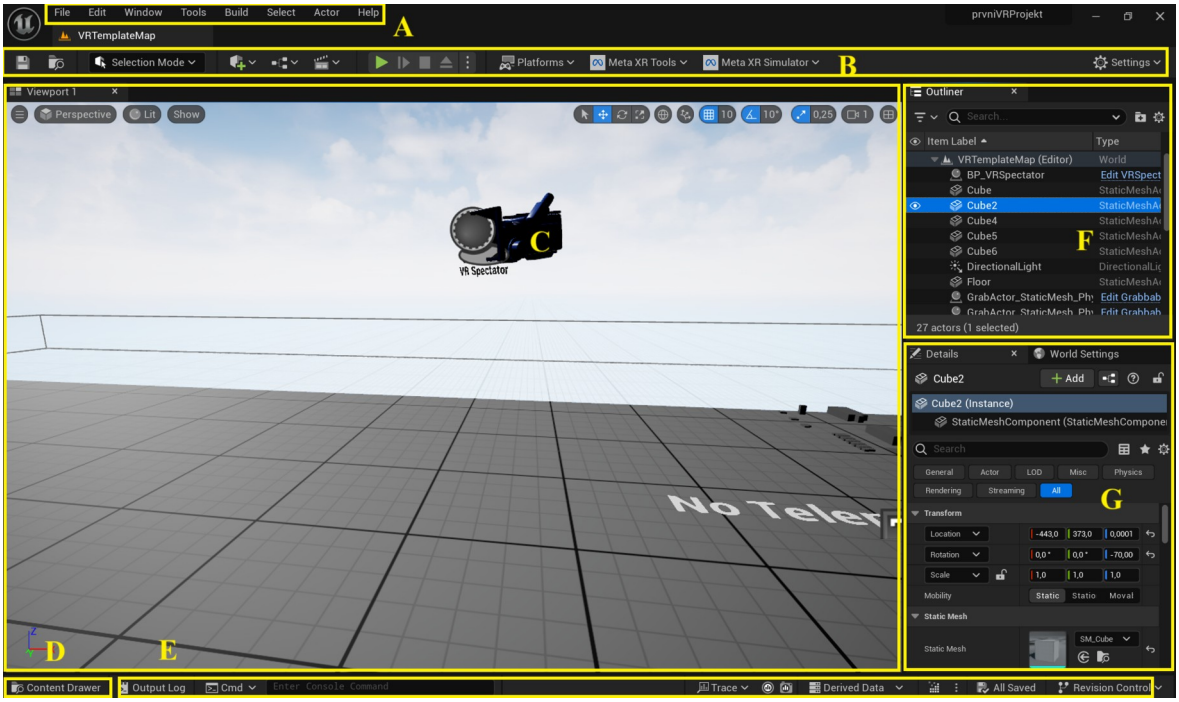
E) správa položek z marketplace



Obrázek 1: Uživatelské prostředí EpicGamesLauncheru

1.3 Uživatelské prostředí

Základní uživatelské prostředí se dá rozdělit do několika částí, jednotlivé body jsou popsány pod obrázkem a písmena korespondují s písmeny na obrázku. [3][40]



Obrázek 2: Uživatelské prostředí UE5

A) **Menu** – Zde se nachází položky pro práci se soubory, zobrazování nástrojů v prostředí editoru či položka pro nápovědu [3][40]

B) **Hlavní panel nástrojů** – Obsahuje nejčastěji používané nástroje pří práci v Unreal Editoru. Je to ukládání, změna Režimu práce, zkratka pro vytvoření Actora, nové Bleuprint třídy, nebo klávesa pro spuštění scény. Záložce Platforms se poté nachází veškeré nastavení pro Vytvoření spustitelných a zabalených souborů aplikace. Nástroje XR Tools a XR Simulátor jsou oproti výchozímu nastavená navíc a slouží pro práci s virtuální realitou.

C) **Náhled levelu** – Hlavní okno editoru, zde můžu pokládat objekty a manipulovat s nimi. Lze přepínat mezi 2D podle předem vybrané osy a 3D zobrazením.

D) **Content Drawer** – Po kliknutí na tlačítko se otevře okno ve kterém jsem schopen prohlížet veškeré assety, Blueprinty a ostatní soubory, které mám nahrané v projektu. Soubory jsem schopen migrovat mezi více projekty, importovat je a nebo pomocí přetažení je vložit do scény

E) **Spodní panel nástrojů –** Ve spodním panelu najdeme Výstup Konzole, ve které můžeme například sledovat výstup při buildu aplikace a hledat Errory, V položce CMD můžeme volat příkazy, které spouští specifické chování editoru. Poté se zde nachází položka pro verzování a Derived Data.

F) **Hiearchie prvků v levelu/scéně (Outliner)–** Zde vidíme veškerý obsah, který se nachází ve scéně. Jednotlivé prvky můžeme skrýt pomocí symbolu Oka a stiskem pravého tlačítka myši vyvolat kontextové menu pro vybraný předmět. Lze vytvářet složky, mazat prvky a přesouvat je

G) **Panel detailů** – Po vybrání prvku v *Outlineru* jej můžeme podrobně nastavovat. Od pozice a rotace, přes výběr materiálu a nastavení fyziky objektu.

1.4 Funkce

1.4.1 Tvorba virtuálního světa

Unreal Engine obsahuje řadu nástrojů sloužící pro vytváření velkých virtuálních světů. Mezi ten hlavní patří *Unreal Editor,* díky kterému jsem schopni editovat naši vytvořenou scénu, a to sami nebo ve více lidech. *Unreal Editor* umožňuje bezpečné sdílení modelů a programových částí mezi další tvůrce na stejném projektu. [38]

Mezi nástroje pro práci prostředím patří tyto nástroje:

- Landscape and terrain tools – s využitím těchto nástrojů jsme schopni přidat do modelu výškové mapy a díky ním vytvářet jeskyně, údolí a nebo hory.
- Water systém – Za pomocí křivek jsme schopni tvořit plně simulované realistické jezera, řeky a oceány, které reagují na změny v okolí a interagují s jinými objekty.
- Sky, cloud and enviroment lighting – jsme schopni tvořit oblohy, mraky a další atmosférické efekty s dynamickým nasvícením v závislosti na čase.
- World partition – umožňuje nám rozdělit náš svět do jednotlivých bloků, které nám umožní načítat a skrývat určité části krajiny, když hráč prochází krajinou
- Modeling, Uvs, and baking – tato sada nástrojů nám provádět jednoduché změny na našem objektu bez nutnosti model upravit v jiném programu zvlášť. Můžeme změnit geometrii, materiály a nebo stínování objektu.

- Scalable foliage – Použití nástroje nám umožňuje automaticky pokrývat části krajiny různými druhy travin, keřů či kamení

1.4.2 Simulace a efekty

Za pomocí zabudovaného *Niagara editoru* jsme schopni v reálném čase tvořit komplexní vizuální efekty kouře, ohně nebo vody a nechat je ovlivňovat nasvícení naší scény. Můžeme vytvořit komplexní efekty, které mezi sebou komunikují a díky tomu je nechat společně reagovat na hudební stopu. [38]

Pomocí fyzikálního systému *Chaos* jsme schopni simulovat chování různých typů látek, měnit jejich parametry a v reálném čase sledovat výsledek. Umožňuje také simulovat chování vlasů, vousů u lidí nebo chlupů u zvířat. Ochlupení bude při růstu respektovat přirozené nedokonalosti tváře aby docílilo co nejvíc realistického efektu. *Chaos* nám také umožňují simulovat destrukci prostředí a v kombinaci s *Niagara* jsme schopni přidat sekundární efekty jako kouř nebo prach.

1.4.3 Materiály, nasvícení a rendering

Material editor nám umožní modifikovat vzhled a působení postav nebo různých objektů. Umožňuje nám vrstvit materiály a do pixelové přesnosti nám umožní nastavit veškeré hodnoty abychom mohli dosáhnout cíleného vzhledu. *Unreal engine* nám také umožní pomocí *Virtual Texturing* rozdělit velké a komplexní textury na menší a načítat tak pouze viditelné části, čímž zlepší výkon vykreslování textur. [38]

Pro nasvícení se využívá *Lumen,* jedná se o systém tvorby dynamického nasvícení a odrazů, pomocí kterého jsem schopni tvořit realistické nasvícení, které reaguje na změny prostředí. Pomocí Lumenu nemusí tvůrce čekat, než světelná UV mapa „zapeče“ nebo pokládat odrazové prvky do scény. K dispozici je také hybridní sledování paprsků v reálném čase (raytracing), díky kterým můžou světelné efekty nabít Hollywoodské kvality.

Ve výchozím nastavení využívá Unreal Engine tzv. *Deferred Renderer*, jedná se o nejvíc univerzální způsob renderování, který má více možností pro renderování samotné. Méně výkonné zařízení jako jsou standalone headsety Quest s ním ale můžou bojovat a proto je zde možnost přepnout na tzv. *Forward Rendering*. [38]

1.4.4 Blueprinty

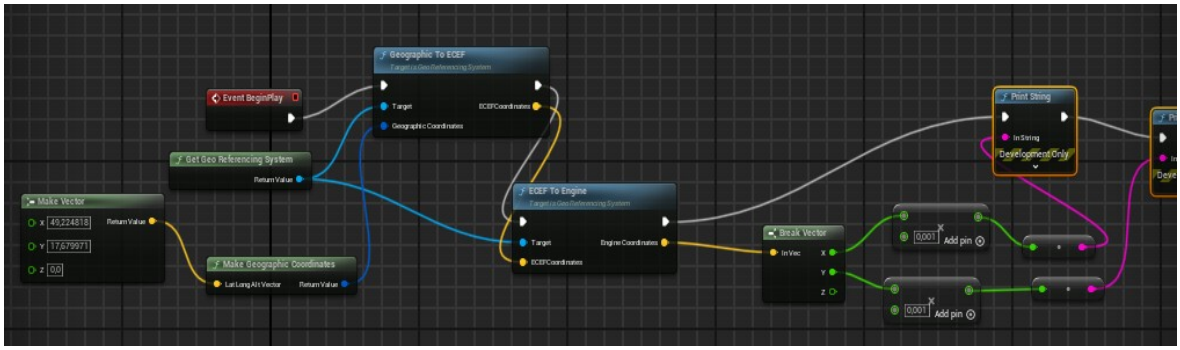
Oproti konkrečním herním enginům obsahuje UE visuální skriptovací jazyk nazývaný Blueprints. Pracuje s jednotlivými uzly, které vykonávají definovanou funkci a pomocí řetězení uzlů pomocí tzv. *Wire* jsme schopni docílit požadované logiky. Blueprints otevírají možnost tvůrcům bez širší nebo žádné znalosti programování tvořit funkční části aplikace a nebo i aplikaci celou. Každý uzel má svůj vstup a výstup do kterého se napojuje *Wire*. Tyto vstupy a výstupy, nazývané *Pins*, se rozlišují barevně podle jednotlivých datových typu. [26]

Datové typy a jednotlivé barvy

- Bool – tmavě červená
- Integer – zeleno-modrá
- Float – světle zelená
- String – neonově fialová
- Transform – oranžová
- Vector – žlutá
- Linear Color – tmavě modrá
- Rotator – světle fialová

Jednotlivé *Wires* poté nabývají stejné barvy jako *Pin* pro jednodušší identifikaci v editoru. Pokud chceme propojit mezi sebou *Integer* a *Float* tak UE automaticky vloží Uzel, který se stará to tzv. *AutoCasting*. [2]

Při běhu naší scény vývojového prostředí UE jsme schopní v Blueprint editoru vidět vizuálně co se zrovna provádí a při přiblížení kurzoru k Pinu nahlédnout jaké data vrací nebo přebírá z předchozího Uzlu. [26]



Obrázek 3: Vizualní programovací jazyk - Blueprints

1.4.5 Postavy a animace

Pomocí speciálních Antimation Blueprints jsem v UE schopni rozpohybovat mesh kostru postavy. Přímo v Animation Blueprintech jsme schopni ovládat jednotlivé části kostry, provádět směšování animací a nastavovat logiku, která bude definovat finální animaci Meshe. [38]

Live Link data streaming nám umožňuje v reálném čase nahrávat data z externích DDC nástrojů jako je Maya, Motionbuilder nebo z externích zařízení jako je například motion capture. Podporuje také ARKit od společnosti Apple, díky kterému jsme schopni zaznamenávat obličeje pomocí mobilního telefonu iPhone. V kombinaci s Take Recorder jsem zaznamenávat animace pomocí externího monitor capture v reálném čase a ukládat je do projektu.

1.4.6 Platformy

UE je multiplatformní a tvůrci můžou vydávat své projekty na řadu různých operačních systémy jako MacOS, Windows a nebo Linux, tak i pro herní konzole jako jsou ty od společnosti Microsoft, Sony či Nintendo. UE umožňuje také tvořit aplikace pro Virtuální nebo Mixed Realitu na platformách Meta Quest, SteamVR, Playstation VR a Hololens 2 a nebo za použití ARKit a ARCore vytvářet aplikace augmentované reality na mobilní telefony. [38]

2 BLENDER

Blender je volně dostupný open-source nástroj určený k tvorbě vizuálního obsahu. Lze zde tvořit 3D modely a řešit jeho texturování, vizualizace a nebo tvořit animace.

Program je určen jak pro profesionály tak začátečníky. Pokročilý uživatelé jsou schopni využit Blender API pomocí kterého jsou schopni psát vlastní skripty a tím přidávat novou funkčnost a nebo si upravit program k obrazu svému. [43]

2.1 Historie

Blender byl vytvořen Nizozemským art directorem a samoučeným programátorem Tonem Roosendalem. Lákalo ho vše co bylo technické nebo kreativního rázu a proto se i pokusil studovat technický design. Školu opustil a v roce 1989 založil vlastní 3D animační studio s názvem NeoGeo. společnost, která v začátcích sídlila na půdě se rázem začala rychle růst a brzy se stala jednou z největších firem v Nizozemí. [35]

První zdrojové soubory, které nesly název Blender napsal Ton 2. ledna 1994 a tento den se také považuje za den zrození Blenderu. Původně byl ale Blender zamýšlel jako firemní nástroj, který bude využívám zaměstnanci v prostředí společnosti NeoGeo. V začátcích se jednalo o balíček existujících nástrojů pro Amigu. Tato prvotní verze Blenderu měla za cíl vyřešit problémy mezi kreativci, jak například upravit projekt pokud chce klient více změn a jak je provádět bezbolestně. A díky vysoce konfigurovatelnému přístupu byl Blender odpověď na tento problém.

Blender 1.0 vyšel v lednu roku 1955, byla to první iterace, která obsahovala správně začleněné pak inovativní myšlenky, mezi které například patřilo jedno velké okno, které si uživatelé mohli rozdělit podle vlastních potřeb.

V této době bylo 3D komerčně považováno za neatraktivní ale Ton Roosendaal si ho zamiloval a jeho slovy ho popsal jako „magickou věc pro tvorbu celých světů v počítači“. Po konci GeoGeo si spolu s jeho partnerem Frank van Beek založili novou společnost, jejímž cílem bylo nadále vyvíjet a propagovat Blender. V roce 1998 tedy vznikla společnost Not a Number, zkráceně NaN, která distribuovala Blender pod freemium strategií. To znamenalo, že si Blender mohl kdokoliv stáhnout zadarmo ale pokročilejší funkce se odemkli pouze po koupi klíče, který NaN prodával.

Tato strategie fungovala a NaN si mohl koupit stánek SIGGRAPH v Los Angeles, jednalo se o konferenci počítačové grafiky. Výsledkem byly 2 kola financování v celkové

výši pět a půl milionů dolarů. Ale i přes tento úspěch v kombinaci s drsným ekonomickým klimatem, nerozvážným utrácením a problém mezi NaN a investory se v roce 2002 společnost NaN rozpadá a vývoj Blenderu se ocitl na mrtvém bodě.

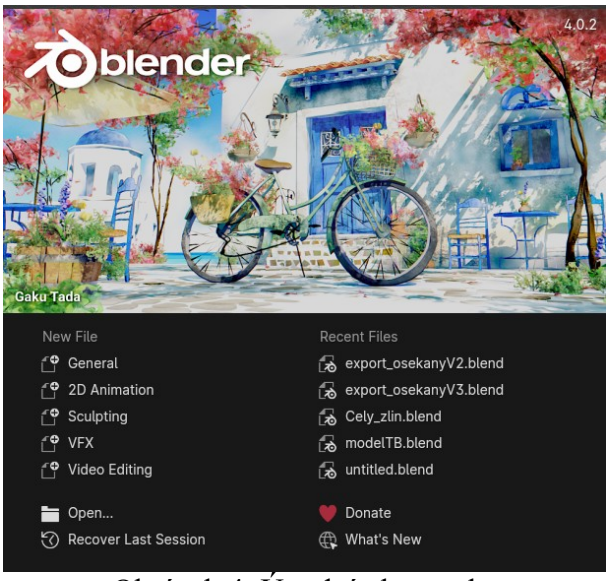
Bohužel Ton nebyl schopný koupit práva na Blender zpět od sponzorů NaN přišel s novým plánem. V Květnu roku 2002 založil neziskovou organizaci Blender Foundation a jeho cílem bylo udělat Blender open-source. Doufal, že se mu povede vytvořit veřejnou památku Blenderu a dát všem, kdo na něm pracovali, možnost si ho vložit do svého portfolia. Července téhož roku vytvořil první crowdfounding kampaň s názvem Free Blender. V horizotnu 7 týdnů se mu povedlo sehnat díky komunitě tvořenou 250 000 uživateli dostatek peněz aby jej mohl odkoupit. Dne 13. října 2002 byl Blender vydán pod licencí GNU General Public Licence, pod nejpřísnějším možným kontraktem a díky tomu byl nejen Blender zdarma ale i celý zdrojový kod by byl zdarma, navždy, pro každého s jakýkoliv účelem. V současnosti má Blender 28 zaměstnanců pracující z Amsterdamu, vzdáleně a na základě grantu a jedná se pouze o zlomek lidí, pracující a vyvíjející Blender. [35]

2.2 Uživatelské prostředí

Popis prostředí se bude skládat ze dvou částí, první část bude popisovat Spash screen aplikace Blender, který se zobrazí vždy po spuštění aplikace. Ve druhé části budou popsané jednotlivé sekce samotné aplikace s jejich funkčností.

2.2.1 Úvodní obrazovka

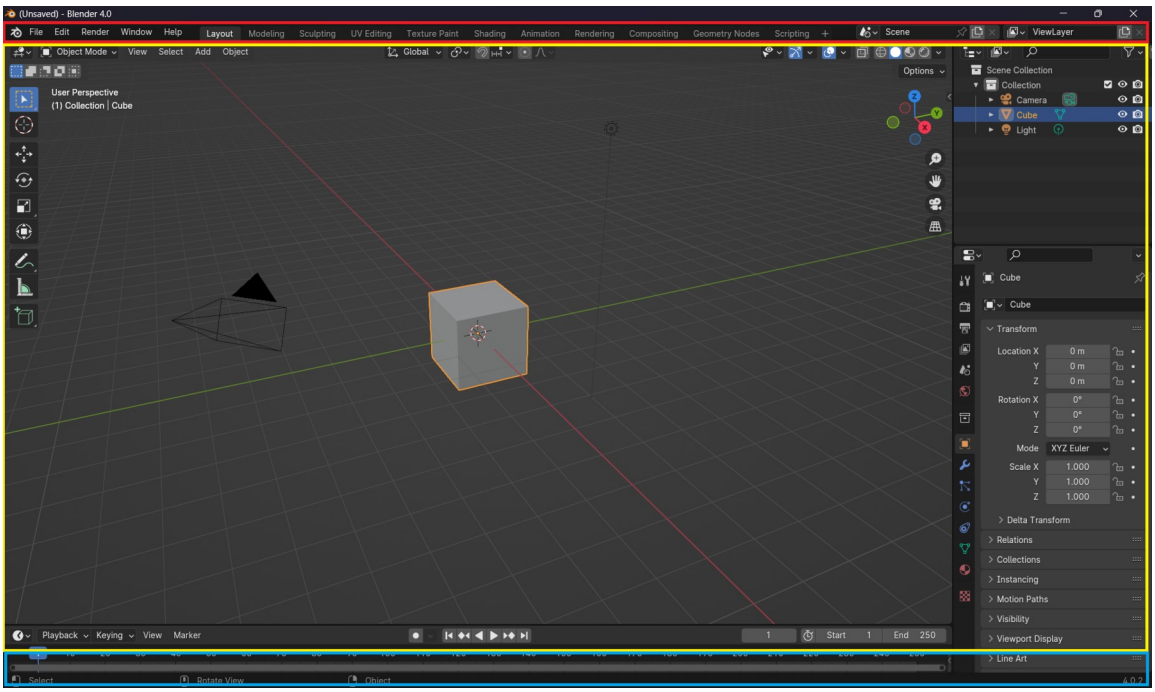
Slouží k rychlému vytvoření předdefinovaných typů projektů, otevírání posledně upravovaných souborů. Nachází se zde i klávesa pro obnovení při pádu aplikace. V horní části okna se nachází obrázek, který je unikátní pro každou verzi programu Blender. V neposlední řadě se tu nachází odkaz k přečtení novinek a také odkaz, kde můžeme darovat peníze a tím podpořit vývoj aplikace. [4]



Obrázek 4: Úvodní obrazovka

2.2.2 Výchozí prostředí aplikace

V následujícím obrázku se nachází zvýrazněné sekce výchozího rozložení aplikace Blender, jejichž popis se bude nacházet pod obrázkem. Aplikace se skládá ze třech větších částí. Jmenovitě se jedná o *Panel Nástrojů, Hlavní pracovní plochu a tzv. Status Bar*



Obrázek 5: Výchozí uživatelské prostředí

Panel Nástrojů *(obrázek 5. červený rámeček)* – Nachází se na samotném vrchu okna aplikace a jednolitých záložkách a nalezneme tu tlačítka pro načítání, ukládání importování, cestu k nastavení nebo nastavování renderování. Součástí Panelu nástrojů je také

přepínání mezi jednotlivými „Workspace“, což jsou různé předdefinované rozložení pracovní plochy. [5]

Druhý pracovních režimů

- Modeling – nástroje k modifikacím geometrických modelů
- Sculpting – k modifikaci *meshes*
- UV Editing – k mapování texturových souřadnic na 2D model
- Texture paint – k vybarvování textur ve *3D Viewportu*
- Shading – k přidání a úpravě vlastností materiálu k renderování
- Animation – pro tvorbu specifických úprav závislých na čase
- Rendering – k zobrazení a analýze výstupu renderu
- Compositing – pro tvorbu post-processingu obrázků a renderingu
- Geometry Nodes – procedurální modelování pomocí Uzlů
- Scripting – Pro interakci s Python API a psaní skriptů

Hlavní pracovní plocha (*obrázek 5. žlutý rámeček)* **–** Jedná se o největší část celé z celé aplikace. Samotná hlavní pracování plocha se dá rozdělit do čtyř částí. Každá část umožňuje změnit svoji velikost a nebo ji skrýt. [5]

- *3D Viewport* – Jedná se o část, se kterou uživatel nejčastěji interaguje. V této části se manipuluje s objekty a slouží i jako *náhled* pro prováděné operace
- *Outliner* – Nachází se zde list, který seřazuje prvky obsažené v projektu, můžeme zde jednotlivé prvky řadit, shlukovat do kolekcí, mazat a nebo je skrýt před zobrazením v 3D Viewportu
- *Parametry* – Dělí se do jednotlivých Záložek, ve kterých jsme schopni editovat vybraný objekt ze scény ale i scénu samotnou nebo parametry renderování
- *Timeline* – Slouží uživateli k práci s časovými událostmi ve scéně, na *timeline* se schopný vytvářet jednotlivé keyframes a poté události spouštět v reálném čase.

Status Bar (o*brázek 5. modrý rámeček)* – Nachází se na samotném spodku aplikace a ukazuje statistické informace jako je počet hran, ploch nebo kolik projekt zabírá v paměti RAM, nápovědu pro klávesové zkratky a v případě Renderingu ukazuje i jeho průběh v procentech. [5]

2.3 Funkce

2.3.1 Renderování

Jedná se o proces, který z 3D scény vytváří 2D obraz či video. Blender nabízí 2 různé způsoby jak toho dosáhnout. Jedná se méně náročný *Eevee* a realističtější ale také náročnější *Cycles*. [6][7]

Cycles – je fyzicky založený renderovací engine, který využívá tzv. Path-tracing, tedy sledování paprsků v reálném čase. Využívá fyzikálně založených materiálů a osvětlení. Tento engine podporuje různé techniky renderování, včetně globálního osvětlení (*global illumination*), stínů, odrazů, refrakcí a mnoha dalších efektů. Cycles umožňuje renderování pomocí CPU, tak i GPU a uživatelům umožňuje využít jejich hardware k urychlení renderování. Urychlení renderování pomocí GPU je umožněno vlastníkům grafických karet od společnosti Nvidia využívající CUDA a karty od společnosti AMD s OPENCL [6] [7]

Eevee – Jedná se real-time renderovací engine, který využívá technologie podobné těm, které se využívají ve hrách. Je optimalizován pro interaktivní náhledy a animace, ačkoliv jeho výstup může mít mírně odlišný vzhled od Cycles renderu. Podporuje různé real-time efekty, včetně stínů, odrazů, ambient occlusion, bloom efektů a mnoha dalších, což umožňuje uživatelům vidět výsledky své práce okamžitě. [6][7]

Obrázek 6: Rozdíl mezi Eevee a Cycles [28]

2.3.2 Modelování

Editační režimy v programu Blender umožňují uživatelům provádět různé úpravy na geometrických objektech ve 3D scéně. Jedná se o:[8]

- **Object Mode** – jedná se o výchozí režim pro všechny objekty a umožňuje editovat jeho pozici, velikost, rotaci a duplikovat objekt
- **Edit Mode –** jedná se o nástroje, které umožní upravovat geometrické vlastnosti objektu (tvary hra, přidávání či odebírání stěn,...)
- **Sculpt Mode** – neboli sochařství je alternativní způsob jak upravovat *meshe*. Pracuje se štětci a křivkami, díky kterým jsem schopni dosáhnout detailů, které by byly za použití obyčejných nástrojů obtížné nebo dokonce nereálné. U štětce jsme schopni nastavit tvar, šířku, druh nebo sílu štětce a tím ovlivníme výstup na *meshi.* Samotný proces se dá přirovnat k práci s modelářskou hlínou a je využívá se například pro tvorbu obličejů lidí.
- **Vertex Paint Mode** – Vertex Paint Mode v Blenderu umožňuje uživatelům přímo malovat na vrcholy jejich 3D modelů, což umožňuje vytvářet textury, detaily a barevné efekty přímo na geometrii modelu. Tento režim poskytuje kontrolu nad vzhledem jednotlivých vrcholů, což umožňuje vytvářet detailní vizuální efekty a speciální úpravy, jako jsou masky, texturování a barevné změny.
- **Weight Paint Mode** – umožňuje uživatelům přidělit váhy (weights) jednotlivým částem (vertexům) 3D modelů. Tato váha určuje, jak moc ovlivňuje určitý bod deformace při animaci. Při použití *Weight Paint* můžete snadno označit různé části modelu barvami, kde každá barva reprezentuje určitou váhu. Čím světlejší je barva, tím více váhy je přiděleno dané části modelu. Tato funkce je často využívána při modelování, riggingu (tvorbě kostře) a animaci postav v Blenderu.
- **Texture Paint Mode –** umožňuje nanášet textury přímo ve 3D Viewportu
- **Particle Edit Mode –** část sloužící k nanášení proměnlivých prvků, například chlupy či vlasy
- **Pose Mode –** Slouží k nastavení pohybů pro animaci

- **Draw Mode** - díky těmto nástrojům lze kreslit přímo do 3D scény za pomocí *Grease Pencil*. Ta slouží k vytváření 2D grafiky ve 3D prostředí, umožňuje nám dělat různé náčrtky pro požadované umístění modelu.

2.3.3 Animace a Simulace

Animování je pohyb nebo změna tvaru statického objektu v časovém průběhu. Lze pracovat s jedním objektem a nebo nebo jen s některými z jeho částí. Lze propojit animace více objektů, tak aby na sebe navazovaly. K animacím jdou přidat také zvukové efekty. Animace tvoříme z jednotlivých *keyframes,* které přidáváme na časovou osu, výslednou animaci poté vytvoříme z rozdílů těchto *keyframes* v čase. [34]

Pomocí simulací můžeme docílit určitého stupně realističnosti ve scéně. Skládá se jak ze statických tak dynamických prvků a umožňuje nám přidat na objekt nebo do scény efekty kouře, látky.

2.3.4 Textury a Materiály

Materiály se starají o to jak bude objekt reagovat na světlo. Jestli je matný nebo lesklý, průhledný nebo neprůhledný, jakou má barevnou strukturu a barevnost. [34]

Textura je obrázek s detaily, který můžeme nanášet na jednotlivé objekty. Slouží jako napodobenina reálných materiálů jako jsou například prasklina zdi, spáry mezi cihlami a nebo kůra stromu s detaily, které by byly velmi obtížné vymodelování a snižuje náročnost objektu. Pomocí Nodes, neboli uzlů, jsem schopni provádět pokročilejší nastavování materiálů a umožnit například průhlednému sklu propouštět světlo, což za pomocí pouze materiálů není možné.

3 VIRTUÁLNÍ REALITA

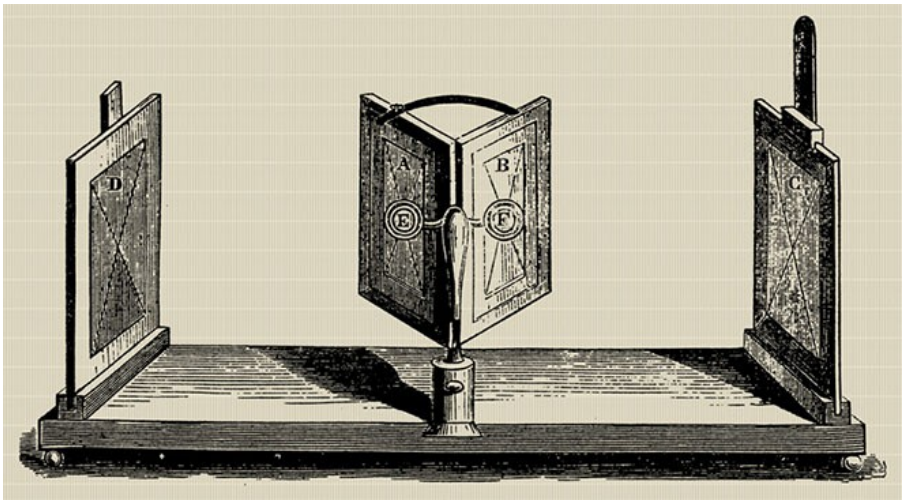
Virtuální realita (zkráceně VR) je softwarově vytvořený umělý 3D svět, jenž má člověk možnost prozkoumávat a pokud je mu to umožněno, tak s ním i interagovat. [16]

Pro to aby bylo někomu umožněno interagovat, tak musí využít speciálně vytvořené zařízení k tomuto určené. Jedná se o takzvané VR Headsety. Tyto speciálně vytvořené zařízení jsou vlastně brýle, které přenáší jak obraz tak zvuk, který byl vygenerován buďto připojeným počítačem a nebo v případě standalone headsetů samotnými brýlemi. Uživatel vidí a slyší pouze to co bylo vytvořeno a nemá možnost vidět reálný svět okolo sebe. Způsobů jak je uživatel schopný interagovat s virtuálním světem je několik, buď bude využívat speciálně vytvořené ovladače, různými gesty a nebo jen pohybem hlavy, čímž vytváří iluzi, že se uživatel v tom uměle vytvořeném světe opravu nachází. [17] [27][16]

V této kapitole mé diplomové práce se budu zabývat historií, vlastnostmi virtuální reality a samotnými zařízeními, které toho umožňují. V neposlední řadě popíši jak se Virtuální realita implementuje v herním enginu Unreal Engine.

3.1 Historie

Počátky vývoje virtuální reality se spojují s vědeckým výzkumem Sira Charlese Wheatstona, který v roce 1838 popsal mozkový proces, kdy byl mozek schopný spojit 2 obrázky, pro každé oko jeden, do jednoho celistvého obrazu ve třech dimenzích. Tento jev poté Wheatstone demonstroval na zařízení s názvem *Stereoscope*. Zařízení se skládalo ze dvou zrcadel, které byly v 45 stupňové pozici k očím člověka, a odrážely obrazy, které byly umístěny po stranách. Předpovězení konceptu Virtuální reality se také objevilo v díle amerického Sci-fi spisovatele Stanley Weinbauma v roce 1935 v jeho krátkém příběhu s názvem „Pygmalion's Spectales". V tomto díle se hlavní postava potkala s profesorem, který vymyslet brýle, které umožňovali nositeli sledovat film, jako kdyby byl součástí něj samotného.[14]



Obrázek 7: Wheatstonův zrcadlový stereoscope [14]


PYGMALION'S SPECTACLES

By STANLEY G. WEINBAUM

Author of "The Black Flame," "A Martian Odyssey," etc.

Unbelieving, still gripping the arms of that unseen chair, Dan was staring at a forest

Obrázek 8: Poutač na Pygmalionův Spectacle [14]

V polovině roku 1950 kameraman Morton Heilig vytvořil *Sensoramu* (patentováno v roce 1962), což byla „filmová kabinka/buňka“, která měla za cíl simulovat veškeré lidské smysly. Zařízení obsahovalo stereo reproduktory, stereoskopický 3D displej, větráky, generátor zápachů a vibrační křeslo. Cílem *Sensoramu* bylo diváka kompletně vtáhnout do děje. Pro tento účel vytvořil šest krátkých filmů, které sám režíroval, natočil a nakonec i sestříhal. Názvy filmů byly , Motorcycle, Belly Dancer, Dune Buggy, helicopter, A date with Sabina and I’m a coca cola bottle! [14] [15]

Obrázek 9: Sensorama [15]

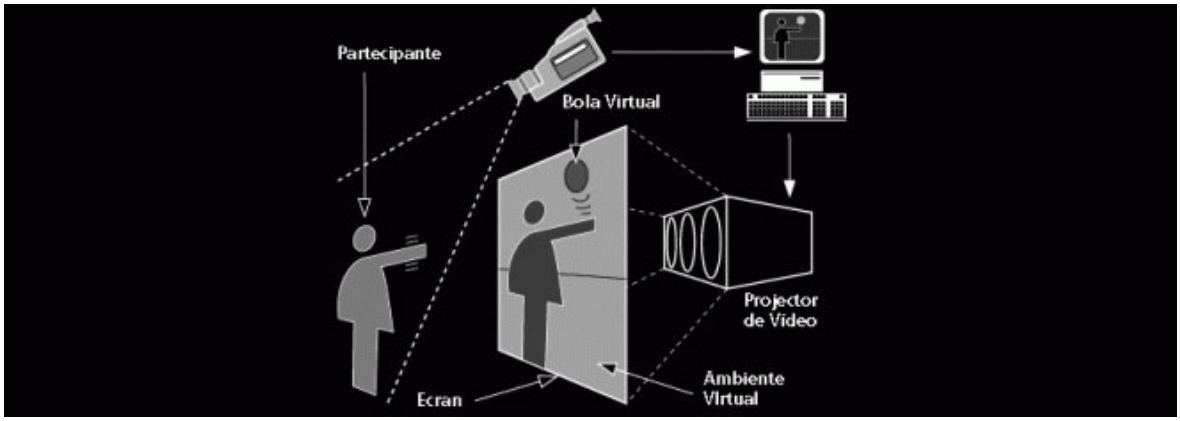
V roce 1960 vznikla první varianta tzv. Head-mounted display (HMD), jednalo se o displeje, které uživatel nosil pomocí různých popruhů na hlavě. Samotné zařízení disponovalo stereoskopickým 3D displejem a stereo ozvučením. Sloužilo pouze k filmům, bez jakékoliv interakce a tudíž ani neobsahovalo žádnou formu motion trackingu, tedy sledování pohybu. Rok poté dva inženýři ze společnosti Philco Corporation, Comeau a Bryan, vyvinuli prvního předchůdce HMD, jak je známe dnes – Headsight. Obsahovalo displej, pro každé oko zvlášť a motion tracking pomocí magnetů, který byl propojen obvodem ke kameře. Samotný Headsight nesloužil pro aplikace virtuální reality, jelikož tento pojem ještě neexistoval. Jeho využití bylo například v armádě, umožňovalo totiž pohlcující sledování nebezpečných situací na dálku. Pohyby hlavy by pohybovaly vzdálenou kamerou, což by uživateli umožnilo přirozeně se rozhlížet po prostředí. Headsight byl prvním krokem v evoluci VR formou headsetů ale zatím mu chyběla integrace s počítači. [14] [15]

O pár let později, v roce 1965, představil počítačový vědec Ivan Sutherland jeho vizi tzv. Ultimate Display. Jednalo se o koncept virtuálních světů do kterého bylo nahlíženo přes HMD s cílem replikovat svět na takové úrovni aby uživatel nebyl schopný jej rozeznat od reality. Obsahovalo to možnost interagovat s objekty virtuálního světa. Jeho koncept obsahoval počítačový hardware, který sloužil k vytvoření virtuálních světů a udržování ho v reálném čase. Jeho práce je považována za základní koncept pojmu Virtuální realita. Neuběhl ani rok a v roce 1966 představil Thomas Furness první letecký simulátor, pro americkou armádu. To napomohlo rozvoji VR, protože armáda následně poskytla velké množství finančních prostředků na výrobu lepších leteckých simulátorů.

Před koncem šedesátých let minulého století vytvořil Ivan Sutherland s jeho studentem Bobem Sproullem první HMD virtuální realitu. Zařízení bylo pojmenováno The Sword of Damocles. Narozdíl od předchozích HMD, které byly vždy připojeny ke kameře bylo Sword of Damocles připojeno k počítači. Připojení bylo hodně primitivní a díky tomu zvládal zobrazovat pouze jednoduché drátěné tvary. Tyto tvary měnily svoji perspektivu v moment co uživatel pohnul svojí hlavou. Samotného zařízení se ale nikdy nedostalo z laboratoře, jelikož bylo velmi těžké a navíc muselo být ukotvené ke stropu.

Obrázek 10: Sword of Damocles [15]

V roce 1969 přišel Miron Krugere se sérií zážitků, které nazval artificial reality (AR), neboli umělá realita. Vytvořil počítačově generované prostředí, které reagovalo na lidi v něm. Projekty nesly jména GLOWFLOW, METAPLAY, PSYCHIC SPACE a byly pokrokem v jeho výzkumu, které nakonec vedly k vývoji technologie VIDEOPLACE. Videoplace spatřil světlo svět v roce 1975 a skládal se z temné místnosti, která obsahovala velké obrazovky, které byly kolem účastníka a díky ním se ponořil do „virtuální reality“. Účastník viděl jeho vlastní počítačově generovanou siluetu, která imitovala jeho pohyby. Kamera snímala jeho pohyby a zobrazovala je jako siluetu. Účastníci měli možnost komunikovat s ostatními účastníky v jiných místnostech ale stejným virtuálním světě. To podpořilo myšlenku, že lidé mohou komunikovat ve virtuálním světě, i když si nejsou fyzicky blízko. [14][15]



Obrázek 11: Koncept Videospace [15]

O 2 roky později v roce 1977 vznikl Aspen Movie Map a mělo ho na starost MIT. Výstupem byla poté virtuální prohlídka města Aspen v Colorádu a dalo by se to přirovnat ke Google StreetView. Vzniklo to za použití kamer, které snímaly město a vytvářely tak hrubý model města. I když zde nebylo využito žádného HMD a jednalo se o využití interaktivity z první osoby a naznačovalo to, že VR může přenést lidi na jiná uměle vytvořená místa. [14] [15]

Začátkem osmdesátých let vznikly první rukavice, které mohly sloužit pro sledování pohybů prstů. O tento vynález se v roce 1982 postarali Daniel Sandin a Thomas DeFanti. Rukavice byly připojeny do počítače a pohyb byl zaznamenáván pomocí světelných zářičů a fotobuněk v těle rukavic. Fungovalo to tak, že při pohybu prstu se množství světla, které dopadlo na fotobuňku změnilo a to se pak převedlo na eletrický signál. Tento vynález se dá také považovat za počátek rozpoznávání gest. V roce 1985 založil Jaron Lanier a Thomas Zimmerman společnost VPL Research, Inc. Společnost je známá tím, že byla první, která prodávala brýle pro VR a rukavice. Vyvinula několik VR zařízení mezi které patří například DataGlove, EyePhone HMD a Audio Sphere. Mezi lety 1986 a 1989 vyvinula společnost Furness letecký simulátor zvaný Super Cockpit. Simulátor obsahoval počítačově generované 3D mapy, pokročilé infračervené a radarové snímky a pilot měl možnost slyšet a vidět v reálném čase. Trackovací systém v helmě umožňoval pilotovi ovládat letadlo pomocí pohybu očí, hlasu a gest. Rok poté, v roce 1987, vymyslel Jaron Lanier, zakladatelem společnosti VPL, termín Virtuální realita. Od této doby mělo toto pole výzkumu i název. [14]

Na konci desetiletí se pole vývoje virtuální reality připojila NASA, ta s pomocí společnosti Crystal River Engineering vytvořila projekt VIEW. Jednalo se o simulátor, který měl sloužit ke trénovaní astronautů. Zařízení připomínalo současná zařízení a

obsahovalo rukavice, které sloužili k simulaci doteku. Technologie použitá v těchto rukavicích později pomohla k vytvoření Nintendo Power Glove. [15]

Obrázek 12: project View [14]

V devadesátých letech začala být virtuální realita dostupná veřejnosti, více než tomu bylo doposud. I když vlastnictví virtuální reality bylo pro obyčejné domácnosti stále pouhým snem, tak v roce 1991 společnost Virtuality Group vydala řadu arkádových her a herní automaty virtuální reality s názvem Virtuality. Hráči nosili VR brýle a mohli hrát hry se skoro realtime stereoskopickými 3D vizuály a odezvou menší než 50ms. Některé automaty zvládali také multiplayer. Ve stejném roce oznámila společnost SEGA, že vyvíjí SEGA VR headset, který měl být dostupný veřejnosti ke koupi. Mělo se jednat o zařízení, které by sloužilo pro hraní arkádových her a ke konzoli Mega Drive (Sega Genesis pro Severní Ameriku). Vzhledově to připomínalo štít, jelikož byla hodně ovlivněna filmy jako byl RoboCop. Ve štítu se také nacházely LCD displeje, stereo reproduktory a také senzory pro sledování pohybu hlavy. Tento headset ale nikdy nespatřil světlo světa a nikdy nebyl prodáván, i přes to, že na něj existovaly 4 hry. Jeden z důvodu byl například strach společnosti, z možný zranění způsobený věrohodným VR efektem. O pár let později, v roce 1994, vydala společnost Sega VR-1. Jednalo se o arkádový simulátor pohybu, který se pohybuje podle toho, co se děje na obrazovce. Tím se podobá modelu AS-1, ale VR-1 používá displej na hlavě, zatímco AS-1 využíval běžnou projekční plochu. [13] [14] [15]

V roce 1995 vyšel Virtual Boy (původní název VR-32) od společnosti Nintendo. Jednalo se o první přenosnou herní konzoli, která zvládala 3D grafiku, byť v monochromatickém režimu (červená a černá). Virtual Boy byl vydán v Japonsku a Severní Americe s

cenovkou 180usd ale i přes mnohonásobné zlevňování byl selháním. Jedním z důvodu bylo samotné monochromatické zobrazování ale zmiňovalo se také jeho nepohodlnost a také absence softwarové podpory. Rok po vydání byl prodej a výroba ukončena. [14] [15]

V druhé polovině devadesátých let přišli vědci z Georgijského technického institutu a Emoryho univerzity s kolaborací pro využití virtuální reality pro léčbu post traumatického syndromu, zkráceně PTSD, válečných veteránů. Nazývalo se to jako Virtual-Vietnam. Tento přístup léčby PTSD se používá do dnes, jelikož terapeuti mají přístup k tomu co pacient vidí.

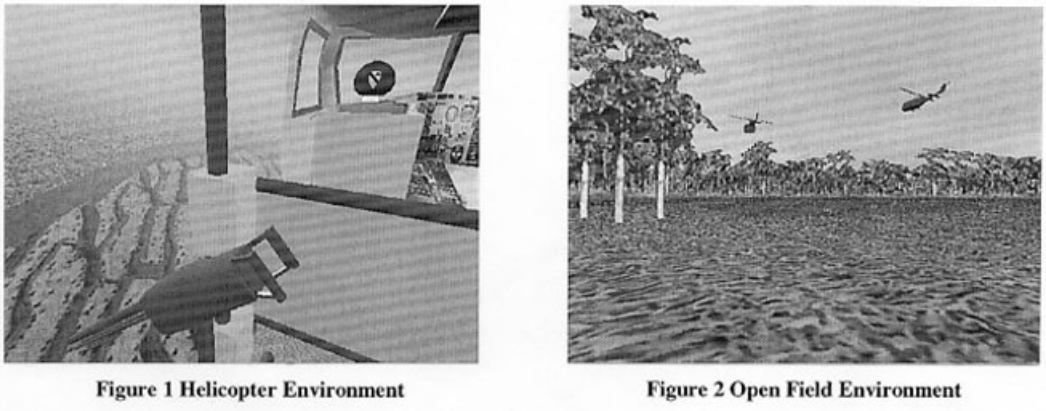


Figure 1 Helicopter Environment

Figure 2 Open Field Environment

Obrázek 13: Virtual Vietnam [15]

V roce 2007 společnost Google představila Google SteetView a společnost Immersive Media byla označena za dodavatele, který pořídil snímky pro čtyři z pěti měst, jež StreetView původně mapoval, a to pomocí patentované dvanáctistěnné soustavy kamer na jedoucím autě. Tři roky poté Google představil 3D stereoskopický režim pro Google StreetView. [14] [15]

Ve stejném roce představil teprve 18ti letý podnikatel první prototyp headsetu Oculus Rift. Měl 90stupňové zorné pole, což dosud nikdo nedokázal, a při zpracování obrazu se spoléhal na výpočetní výkon počítače. Tato novinka zvýšila a oživila zájem o VR. O 2 roky později, v roce 2012, vyšla Kickstarter kampaň pro Rift na které bylo vybráno více než 2,4 milionu dolarů. Tento úspěch nedal spát technologickým gigantům a v roce 2014 byla společnost Oculus odkoupena společností Facebook (v současné době Meta) a jednalo se o velký milník v historii virtuální jako takové. Po této události nabírala virtuální realita velmi rychle na obrátkách. Společnost Sony oznámila, že pracují na Project Morpheus a jednalo se o headset Virtální reality pro konzoli Playstation 4. Google představil Cardboard, jednalo se o velmi levný držák pro mobilní telefony z kartonu.

Umístěním telefonu a za použít speciálních aplikací pomáhal vytvořit stereoskopický efekt. Stejné zařízení představil Samsung pro názvem GearVR, to využívalo mobil od značky Samsung a bylo celkově z kvalitnějších materiálů, než je karton. [14] [15]

V roce 2016 byly stovky společností snažící se vyvinout headsety virtuální reality. Většina headsetů využívala dynamické binaurální audio. Haptické rozhraní pro ovládání bylo nedostatečně rozvinuté a proto většina headsetů využívala tlačítka na ovládání. Prvním headsetem, který měl sledování založené na senzorech a umožňoval tak uživatelům volný pohyb v prostoru byl HTC VIVE. V roce 2018, na konferenci Facebook F8, byl představen nový prototyp tzv HalfDome headsetu. Tento headset využíval varifokální čočky a extrémně široké zorné pole 140 stupňů. V těchto letech virtuální realita výrazně pokročila a začala se využívat různými způsoby, od poskytování pohlcujících herních zážitků přes pomoc při léčbě psychických poruch až po výuku nových dovedností a dokonce i virtuální cesty nevyléčitelně nemocných lidí. [14]

Obrázek 14: HTC Vive [15]

Pokrok ve virtuální realitě se nezastavil a roce 2019 byl představen Oculus Quest, na rozdíl od předchozích headsetů se jednalo o plně nezávislé zařízení. Samotný Quest oproti předchozím headsetům totiž obsahoval jak baterii, tak i SoC. Díky tomuto nebylo nutné využívat počítače a veškerý výpočetní výkon se prováděj na zařízení. Quest obsahoval operační, obchod s aplikacemi a i díky tomuto byl jednodušší pro obyčejného člověk na používání. Vznik standalone headsetů také znamenal konec headsetů využívající telefon. Rok poté vyšel Quest 2 a jednalo se inkrementální update, který přinášel hlavně

vyšší výkon. V těchto letech také začali leaky a teorie o tom, že společnost Apple pracuje na headsetu pro Virtuální nebo Mixed Realitu. V roce 2021 společnost Pico vydala Pico Neo 3 a mělo se jednat o konkurenci ke Quest 2. Společnost Pico se poté stala součástí Čínské společnosti ByteDance, jenž jej odkoupila při rozšiřování pole své působnosti. [14]

Obrázek 15: Oculus Quest 1 [15]

V roce 2023 byl na Worldwide Developers Conference (WWDC) představen mixed-reality headset od společnosti Apple. Jednalo se o prémiový headset od této značky s cenovou stanovenou na 3499usd a jeho vydání naplánováno až na únor roku 2024. V roce 2023 přišel také Quest 3, ten oproti Quest 2 byl již mixed-reality headset a oproti vyššímu výkonu obsahoval také nový a kvalitnější displej. Rozlišení displeje vzrostlo z 1832 x 1920 na 2160 x 2160 pro každé oko. [14] [15]

3.2 Headsety pro virtuální realitu

Jedním z nejdůležitějších a stěžejním příslušenství pro VR jsou stereoskopické brýle s binaurálním prostorovým zvukem. Využití tohoto zařízení plynule přesune uživatele do virtuálního světa, aniž by vnímal okolní reálný svět. Těchto zařízení je v dnešní době plná řada a liší se cenově, podle platformy využití a funkcemi. Můžeme se tak setkat s VR brýlemi pro konzole Sony Playstation, standalone headsety řady Quest od značky Meta, headsety jako je HTC Vive Pro nebo Meta Quest Pro, které ke své funkci vyžadují PC a nebo také headsety pro využití dronů.

3.2.1 Playstation VR

Narozdíl od svého konkurenta ve světě herních konzolí, má společnost Sony vlastní headset pro virtuální realitu. Jedná se již o druhou generaci PS VR2. Jedná se o headset dělaný přímo a pouze pro konzoli Playstation 5, to tedy znamená, že není zpětně kompatibilní s konzolí Playstation 4. Pro tyto účely existuje 1. generace PS VR. Předchozí generace PS VR je ale kompatibilní i s novějším systémem PS5. Pro tuto funkčnost je ale nutné zakoupit speciální adaptér. Nejedná se tak o standalone headset jako je například Meta Quest 3 a pro jeho funkčnost je nutné jej mít připojený ke konzoli po celou dobu používání. Součástí balení, které stojí cca 14 tisíc korun dostanete obdobně jako u jiných VR headsetů headset samotný, tak ovladače. Rozdílem může být to, že nemá v sobě reproduktory a v balení dostanete špuntová sluchátka. Headset je dostupný pouze v bílé barvě a na rozdíl například od dříve zmíněného Quest 3 nemá pás přes hlavu. Celý headset drží obepnutý kolem hlavy, což nemusí každému vyhovovat. Na headsetu se nachází veškeré kamery, sloužící k *passthrough* a sledování okolí. Ze předu se také nachází zabudovaný mikrofon, tlačítko pro zapnutí a speciální tlačítko, které po stisknutí zapne režim *passthrough,* pomocí kterého jste schopni zapnout „průhlednost“ a vidět okolní svět pomocí dříve zmíněných kamer bez nutnosti headset sundávat. Jednotlivé čočky se dají posouvat pomocí kolečka, a díky ním ovládat IPD (Interpupilary Distance). Samotné čočky mají také IR LED diody, které slouží sledování pohybu očí. Pomocí sledování pohybu očí můžeme poté renderovat ostřeji scény například ve hrách, podle toho kam se uživatel zrovna dívá. [25] [22]

Tabulka 1: Specifikace headsetu PS VR2 [22]

Cena	550usd
Váha	560g
Zorné pole	110°
Typ displeje	OLED
Rozlišení displeje	2000x2040 px
Obnovovací frekvence	120Hz (možnost snižit na 90Hz)

Každý přibalený ovladač má na sobě analogovou páčku, dvě akční tlačítka, tlačítko *Možnosti* pro pravý, *Vytvořit* pro levý ovladač a klávesu s logem Playstation. Ze spodku ovladačů se poté nachází tzv. Adaptivní Trigger neboli adaptivní spouště. Tyto spouště umožňují herním vývojářům naimplementovat „odpor“, který například bude napodobovat

natahování luku. Bude tedy těžší tuto spoušť zmáčknout. Možností je také zkrácení „dojezdové“ doby těchto spouští. [25] [22]

3.2.2 Standalone VR

Jedná se o typ headsetů, který je určen pro samostatný provoz. Tyto stereoskopické brýle mimo displej obsahují také procesor a baterii. Nevyžadují tedy připojení externího zařízení jako je počítač nebo konzole, které by obstarávalo výpočetní výkon. Jako procesory se zde využívají vlajkové procesory z mobilních telefonů. Tyto procesory jsou také důvod, proč na těchto brýlích nelze spouštět ty nejnáročnější VR aplikace. [27]

Mimo brýle, které obsahují veškeré potřebné senzory se k nim dostává ve většině případů i pár bezdrátových ovladačů. Tyto ovladače slouží poté k snímání pohybu rukou a interakci s aplikacemi. V některých případech lze využívat gesta na úkor zmíněných ovladačů.

Oculus Quest

Jedná se o řadu standalone headsetů od společnosti Meta. Mimo využití při hraní her podporujících virtuální realitu se můžou také využít pro prozkoumávání realných či smyšlených světů a nebo také k profesionálním činnostem a nebo vzdělání. V současné době je k dispozici hned několik variant headsetů od společnosti Mety. Mezi ně patří Quest 3 a Quest Pro. V historickém kontextu také existoval Quest, Quest Go a Quest 2. Poslední zmíněný se dá stále zakoupit se sníženou cenou na stránkách společnosti meta. [21]

Quest 3 jako nejnovější přírůstek do řady Quest nezapadá do klasické kategorie VR headsetů ale jedná se již o Mixed Reality headset a přináší oproti svému předchůdci řadu vylepšení. Mimo generační zvýšení výkonu díky novému procesoru XR2 Gen 2 z dílny společnosti Qualcomm a vyšší kapacitě paměti RAM z 6GB na 8GB se zde nachází nový displej, a i když se stále jedná o LCD displeje tak vzrostlo rozlišní z 1832x1920 pro oko na 2064x2208 na oko. Zlepšení se také dotklo samotného zorného pole, které vzrostlo z 97° na 110°. Novinkou je také inkluze dvou 4Mpx kamerek a nového Time-of-flight senzoru, pomocí kterých jste schopni zobrazit si okolí pomocí passthrough v barevném zobrazení a díky nízké odezvě se v něm pohybovat. Tato možnost z headsetu dělá MR headset a jedná se o největší rozlišovací faktor mezi předchůdcem a současným headsetem. Proti svému předchůdci zvládá Quest 3 také handtracking, tedy sledování pohybu rukou bez použití ovladačů či jiných nástrojů. Pomocí gest je možné ovládat. První 2 generace Questu umístili vždy uživatele do virtuálního světa, které šlo změnit podle libosti. Quest 3 v tomto

případě hned využije svých nových schopností a po nasazení brýlí zůstanete v reálném světě.

Quest Pro vyšel po boku headsetu Quest 2 v roce 2022, jedná o vyšší řadu cílenou na profesionální využití. Což odůvodňovalo skoro 3x vyšší cenu než Quest 2. Quest Pro byl cílen pro práci s VR a AR aplikacemi. To, že headset má více profesionální využíti bylo zohledněno i ve specifikacích samotného headsetu. Místo procesoru Snapdragon XR2 se zde nachází Snapdragon XR2+, který dle slov společnosti Qualcomm nabízí až o 30% více výkonu a dvojnásobek paměti RAM, takže Quest pro měl k dispozici 12GB RAM. A ikdyž rozlišení oproti Quest 2 bylo nižší než u levnějšího Questu 2, tak se zde změnila technologie displeje, na místo klasického LCD displej se zde nacházel QLED displej. Největší výhodou QLED v tomto případě bylo 500 lokálních stmívacích zon, které pomáhali vytvářet lepší kontrast než klasické LCD. Další novinkou bylo sledování pohybu očí a celé tváře, sledování tváře bylo využíváno v Metaverse hrách jako Horizon, a vaši mimiku obličeje headset převáděl do hry na vašeho avatara. Sledování pohybu očí poté umožňovalo využít tzv Foviated rendering, kdy se ostře vykreslovala pouze část, na kterou se aktivně díváte.

Značného vylepšení dostaly ovladače, u levnějšího Questu se o sledování pozice ovladače staral samotný headset. Na ovladači k tomu sloužil „půlkruh“, který byl nad tlačitky. Ovladače Questu pro mají totiž vlastní 3 kamery na sledování prostorové pozice. Z tohoto důvodu má každý ovladač procesor Snapdragon 662, který se stará o zpracování dat těchto ovladačů. Tento způsob přinášel řadu výhod, například pokud u klasického Quest 2 nebo 3 se dostal ovladač mimo úhel kamery headsetu, například při natahování luku ve hře, tak to mohlo způsobit potíže, jelikož by ovladač ztratil ovládání. Ovladače také ztratily možnost využívat AA baterie a místo nich se ovladač nabíjel pomocí speciálního kontektoru. [21] [20]

Pro potřeby diplomové práce se využívá Quest 1 z roku 2019.

Tabulka 2: Porovnání specifikací řady Quest [41][42]

	Quest 1	***Quest 2***	***Quest 3***
Váha	571	503	515
Zorné pole	93º	97º	110º
Typ displeje	OLED	LCD	LCD
Rozlišení displeje	1440 × 1600	1,832 × 1,920	2,064 × 2,208
Obnovovací frekvence	72 Hz	72, 90 Hz	90,120 Hz
Procesor	Snapdragon 835	Snapdragon XR2	Snapdragon XR2 Gen 2
RAM	4 GB	6 GB	8 GB

3.2.3 PC VR

Podobně jako konzolové VR headsety tak i počítačové vyžadují externí zařízení, které se stará o výpočetní výkon a spouští samotné VR aplikace, které poté do headsetu přenáší pomocí kabelu. Headset má v sobě displej, stereoskopické brýle a senzory pro snímání pohybu. Oproti standalone headsetům má limitaci v pohybu, jelikož se musí dávat pozor, může totiž dojít ke zranění a nebo například se strhnutí headsetu z hlavu či zalomení kabelu. Zatím co u standalone headsetů se spoléhalo na výkon samotného SoC tak v případě PC VR je nutné brát i v potaz výkon počítače.

Bigscreen Beyond

Patří mezi nejmenší a nejlehčí VR headsety, které se dají sehnat a i přes svojí vyšší cenu si najde svou cílovou skupinu. Triumf v rukávu má hned při objednávání ze stránek. Za použití novějších iPhonu, které mají výřez v displeji a podporují FaceID, si jste schopni vytvořit 3D sken vašeho obličeje. Bigscreen poté Váš sken využije a vytvoří vám polštářek mezi čočky brýlí a oči speciálně navržený pro váš obličej. Na rozdíl ale od konkurence se zde nenachází upravitelná vzdálenost IPD čoček. Bigscreen místo toho nabízí 18 různých variant, které mají rozsah IPD mezi 55mm a 72mm. Absenci modulárního IPD společnost argumentovala tak, že jeho absence jim umožňuje snížit váhu. Kombinace těchto dvou věcí, znamená, že brýle budou sedět perfektně na člověka, který si je koupí. Nevýhoda ale nastane v moment, kdy chcete brýle sdílet s přáteli nebo rodinou. Vaše nastavení IPD a tvar polštářku jim nemusí vyhovovat a zážitek nebude dokonalý. Další limitací, která přichází s tak malou velikostí je podpora dioptrických brýlí. Zatím co headsety jako Quest 3 mají dostatek místa a to za použití speciálního distančního rámečku, tak u Bigscreen Beyond tato možnost není. Místo toho si při koupi musíte nechat vyrobit čočky přímo pro vaše specifické potřeby. Tohle je další limitace, která hodně ovlivní možnost používat brýle mezi více lidmi. Čočky jsou poté drženy v těle brýlí pomocí magnetů a dají se lehce vyčistit a popřípadě vyměnit za nové. Brýle od pohledu vypadá úplně jinak než ostatní headsety na trhu a spíše připomínají futuristické „sluneční“ brýle z budoucnosti nebo cyberpunk filmu. Ze předu brýlí se nachází průhledný plast a za ním pár diod, které ukazují jestli jsou brýle zapnuté či ne. Samotný headset váží pouze 127g, což je o trošku více než krabička hracích karet. V balení jsou k dispozici hlavové popruhy, ty se obepínají hlavu podobně jako lyžařské brýle a díky nízké hmotnosti není potřeba využít horní popruh. Ten je také součástí balení. Díky své nízké váze se ze také nenachází reproduktory, a proto je nutné si je obstarat samostatně formou sluchátek. Další eso v rukávu se nachází v disple-

jích. Nabízí 2 Micro-OLED displeje jejichž kontrast a rozlišení (2560 x 2560) překonává displeje v brýlích jako je Quest 3, i když za cenu nižšího maximálního jasu. Obnovovací frekvence brýlí je poté 72hz nebo 90hz. Podrobné nastavení brýlí poté probíhá v aplikaci, kterou si můžete stáhnout. V ní si můžete vybrat mezi obnovovací frekvencí a nastavit jas. Je možné poté nastavit režim „overdrive“, který nastaví jas na 150%. Lze také ovládat rychlost ventilátoru, který chladí zařízení. Při zapnutí režimu *overdrive* pro jas bude ventilátor nastaven na 100% a nelze to změnit. [23]

Obrázek 16: Bigscreen Beyond HMD [37]

Narozdíl od jiných headsetů, ale nedostanete v balení ovladače a samotná společnosti je ani neprodává. Headset je kompatibilní se všemi SteamVR kompatibilními ovladači. Dále potřebuje pro funkčnost tzv. SteamVR Base Station. Jedná se o externí zařízení pro sledování pozice ovladačů a headsetu v prostoru. Oboje se dá zakoupit přímo od společnosti Steam. [23]

Tabulka 3: Specifikace headsetu Bigscreen Beyond [27]

Váha	127g
Zorné pole	102º
Typ displeje	microOLED
Rozlišení displeje	2560 x 2560
Obnovovací frekvence	72, 90

3.3 Implementace v programech, které ji podporují

Vývoj aplikací pro virtuální realitu je možný v celé řadě herních enginů ale mezi enginy, které toto pole dominují, patří Unity a Unreal Engine. Oba herní enginy jsou schopni vyvíjet aplikace pro Virtuální realitu. Mají širokou uživatelskou základnu a kvalitní dokumentaci. Aby engine fungoval ale správně například s brýlemi Oculus Quest je nutné stáhnout rozšíření, které tuto komunikaci budou umožňovat. Pokud chceme vyvíjet v Unreal Engine pro Oculus Quest musíme si stáhnout rozšíření Meta XR (dříve Oculus VR). Jedná se o klíčový nástroj pro vývoj a umožňuje nám například: [12]

- Obsahuje XR API, díky kterému je aplikace schopna komunikovat s ovladači a headsetem samotným
- Nastavovat *Fixed Foveated Rendering*
- Nastavovat a pracovat se sledováním pohybu ovladačů
- Upravovat na jaký výkon poběží Procesor či grafika.
- Jestli aplikace bude umožňovat passthrough

Stejný nástroj pod stejným jménem poté existuje i pro herní Engine Unity a přidává totožnou funkčnost. O tento nástroj pro obě platformy se stará společnost Meta. Pro ostatní headsety, například od společnosti HTC nebo Microsoft existuje rozšíření OpenXR, samotné rozšíření plní stejnou funkcionalitu jako MetaXR. Headsetu od společnosti Meta jej můžou také využít a nejsou vázané pouze na MetaXR. OpenXR je na rozdíl od nástroje od společnosti Meta opensource. [12]

II. PRAKTICKÁ ČÁST

4 FUNKCIONALITA A CÍL PRÁCE

Cílem práce je vytvořit interaktivní aplikaci virtuální reality. Aplikace je vyvíjena v programu Unreal Engine ve verzi 5.3.2. Unreal Engine nabízí vývoj jak v programovacím jazyku C++, tak možnost využití vizuálního skriptovacího jazyka Blueprints. Tento skriptovací jazyk jsem využil pro tvorbu aplikace a její funkcionality.

Aplikace je přímo vyvíjena pro headset Meta Quest 1 z roku 2019, který mi byl zapůjčen Fakultou aplikované informatiky. Z tohoto důvodu je nutné samotnou aplikaci optimalizovat tak, aby byla spustitelná na tomto headsetu a to včetně optimalizace modelu města. Úpravy modelu a jeho rozšíření jsou tvořeny v programu Blender. Výsledná aplikace umožňuje:

- Prohlížet si vymodelovaný model města ve virtuálním prostředí památníku Tomáše Bati
- Získávání informací o budovách a událostech ze souboru
- Možnost interagovat s widgetem a zobrazit si tak aktuální informace o událostech, které se ve městě konají
- Zobrazit si informační texty o významných stavbách města Zlín
- Převod reálných geografických dat tak, aby se daly využívat ve zmenšeném modelu
- Zobrazení ukazatele na zmenšeném modelu města v místě události nebo stavby

5 VÝCHOZÍ STAV MODELU A APLIKACE

Tato práce navazuje na bakalářskou práci s názvem „3D Vizualizace současného Zlína s pomocí modulu BlenderGIS„ [33] ze školního roku 2020/2021. Tato práce se zabývala tvorbou aplikace jenž využívala vygenerovaný model v prostředí programu Blender za využití rozšíření BlenderGIS. Pomocí tohoto rozšíření bylo možné vygenerovat základní strukturu modelu města Zlín a poté byly ručně domodelovány jednotlivé detaily budov.

Předchozí práce využívala modely pro centrum města, které vznikly již v roce 2014 jako součást práce s názvem „3D vizualizace centra současného Zlína„. [32] Tato práce byla poté ukazována na akci STOČ pořádanou fakultou Aplikované informatiky Univerzity Tomáše Bati ve Zlíně.

Mimo grafické úpravy modelu byla vytvořena také počítačová aplikace, ve které bylo možné si tento vytvořený model projet vozidlem, které jelo po předem definované cestě. Bylo možné zrychlovat čas pro zrychlení průjezdu modelem a také se rozhlížet po okolí.

5.1 Počítačová aplikace

Po spuštění se uživateli objevila úvodní obrazovka, na které byla možnost si spustit aplikaci, zobrazení informací o autoru a tlačítko pro ukončení. Po zapnutí hlavní scény se automaticky spustila jízda po vytvořené trati. Pomocí mezerníku bylo možné vždy vozidlo zastavit, při zmáčknutí prostředního tlačítka myši a pohybu myší bylo možné se volně rozhlížet z auta. Klávesa *ESC* poté vyvolala menu pro pauzu, ze které se aplikace dala zastavit

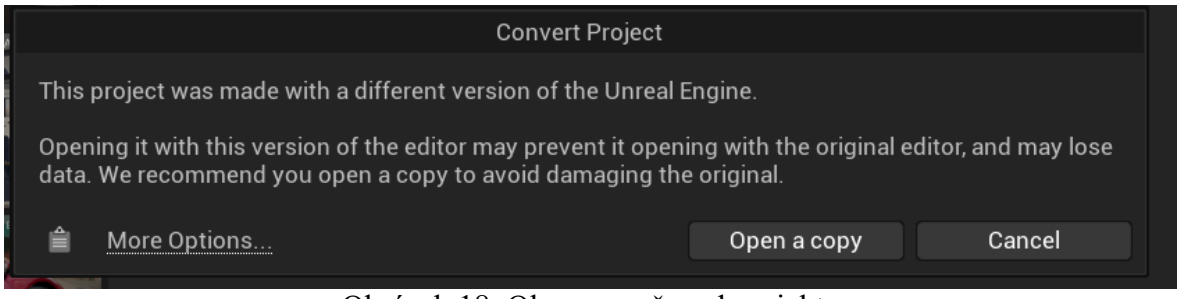
Obrázek 17: Původní aplikace

5.2 Model aplikace

Aplikace zahrnuje model města s okolím kolem Zámku včetně „jednadvacítky“ a části Svitu až po část Zlín-Podvesná. směrem na Vizovice. Aplikace obsahovala vymodelované stromy, zastávky pro městskou hromadnou dopravu a semafory na světelných křižovatkách. Vymodelované části byly mezi sebou propojeny silnicí a chodníky. Části, které nebyly v předchozí práci vymodelovány se v modelu nachází formou tmavých geometrických objektů, které svým tvarem mají připomínat tvar budovy na místě na kterém by se nacházely v reálném světě. Modely byly řešeny pouze za použití materiálů a nenachází se zde žádné textury, včetně terénu.

5.3 Migrace projektu na novější Unreal engine

Původní aplikace byla tvořena v herním enginu Unreal Engine ve verzi 4 a v průběhu vyšla již verze 5. Současná aplikace pro VR bude pracovat ve verzi 5.3.2 a bylo nutné samotnou aplikaci migrovat na novější verzi. Samotný UE dělá proces migrace velmi snadný. Při pokusu otevřít projekt Vám oznámí, že byl projekt vytvořen ve starší verzi UE, než máte momentálně nainstalovanou a nabídne možnost udělat kopii. Tato kopie se vytvoří vedle adresáře s původním projektem a dostane dovětek 5.3 , jelikož se jedná o verzi, kterou využívám. Při kliknutí na možnost „*More options…“* dostanete možnost pro ignorování konverzi (může negativně ovlivnit funkčnost) a nebo udělat konverzi v místě původního projektu a vyměnit staré soubory za nové. Obě možnosti jsou skryté, jelikož se to nedoporučuje dělat. Pokud je v počítači k dispozici verze UE4, ve které byl projekt k vytvořen, bude možnost ho otevřít v ní a nebo si vybrat konverzi na novější.



Obrázek 18: Okno pro převod projektu

6 PRVNÍ SPUŠTĚNÍ QUESTU

Prvotní spuštění Headsetu Meta Quest vyžaduje proces nastavování. Pro ten je je nutné míti připojení k internetu. V obchodě aplikací, ať už Google Play Store či AppStore na platformách Android nebo iOS si vyhledáme aplikaci s názvem Oculus, po jejím stažení budeme vyzvání buď k přihlášení se pomocí Emailu nebo Facebook účtu. Pokud účet nemáme, bude nutný vytvořit, jelikož bez něj se Quest nedá nastavit. Po přihlášení nás aplikace automaticky přesměruje na stránku, kde si vybereme headset, který vlastníme. V mém případě jsem vybral „*Oculus Quest*“. V tento moment zapneme samotný headset pomocí tlačítka na pravé straně a počkáme až naběhne. Po načtení v headsetu uvidíme 5ti místný kod, který zadáme do aplikace. Po připojení budeme odkázání abychom zadali údaje k wifi, kterou bude headset využívat. Jelikož headset byl zapůjčen školou tak párování ovladačů v následujícím kroku proběhlo automaticky.

Jako telefon s operačním systém android umožňuje Meta Quest instalaci aplikací pouze z ověřeného obchodu, který se v něm nachází. Pro potřeby diplomové práce budeme muset aktivovat vývojářský režim. V horním části mobilní aplikace uvidíme headset Meta Quest 1 a klikneme na něj. Vybereme možnost *Nastavení headsetu* a *Nastavení pro vývojáře*. Při prvotním vytvoření účtu se zde bude odkaz na stránky, kde budeme muset založit Organizaci. Je nutné se přihlásit pod stejným účtem, pod kterým spravujeme headset. Organizaci jsem vytvořil s názvem *Bukovy_FAI_UTB*. Poté je nutné ověřit účet pomocí telefonního čísla nebo kreditní karty. Vybral jsem možnost telefonního čísla. Při dokončení procesu se v záložce *Nastavení pro vývojáře* objeví toggle, a pokud už není zapnutý tak jej zapneme. Teď bude možné doinstalovávat aplikace, které vytvoříme v prostředí UE.

Obrázek 19: Možnost pro režim Ladění

7 APLIKACE PRO VR

Po převedení projektu na aktuální verzi Unreal Engine 5.3.2 (v průběhu práce na DP vyšla verze 5.4 ale z důvodu nekompatibility s pluginy se aktualizace neprovedla).

Funkčnost aplikace bylo nutné změnit, jelikož aplikace nebyla v předchozí práci koncipovaná tak, aby fungovala na virtuální realitě. Nebylo tam možné se volně pohybovat po scéně a interagovat s okolím. Celá aplikace musela být optimalizována pro zaručení plynulosti samotné aplikace na brýlích Meta/Oculus Quest 1.

7.1 Závislosti

Jelikož je Meta Quest 1 standalone headset a běží na operačním systému Android, je nutné mít v PC doinstalované nástroje, který kompilaci pro toto zařízení umožní. V případě vývoje v UE5.3.2 je nutné mít v počítači Java SE Development Kit ve specifické verzi 17.0.10 a také Android Studio Flamingo v této specifické verzi *2022.2.1 Patch 2 May 24, 2023.* Dále je potřeba mít nainstalovanou aplikaci Meta Quest Developer Hub. Infromace o potřebných verzích je mění podle toho jaká verze UE je nyní dostupná.

7.1.1 Java Development Kit

Pro instalaci Javy je nutné si ji stáhnout v ZIPu a ten zip extrahovat do adresáře *C:/Program Files.* Tato cesta byla zvolena pro lehčí hledání při nastavování přímo v herní enginu Unreal Engine. A také při složitější cestě je také možné, že může Unreal Engine mít problém. Pokud se v počítači nějaká verze Javy před instalací nacházela, je nutné opravit tzv. Java_HOME. To se provede tak, že do vyhledávání Windows zadáme „Environment Var... „, to nám otevře okno s systémovýma proměnnýma a klikneme na tlačítko „Environment Variables„. Zde vyhledáme JAVA_HOME a smažeme ho.

7.1.2 Instalace Android SDK a NDK

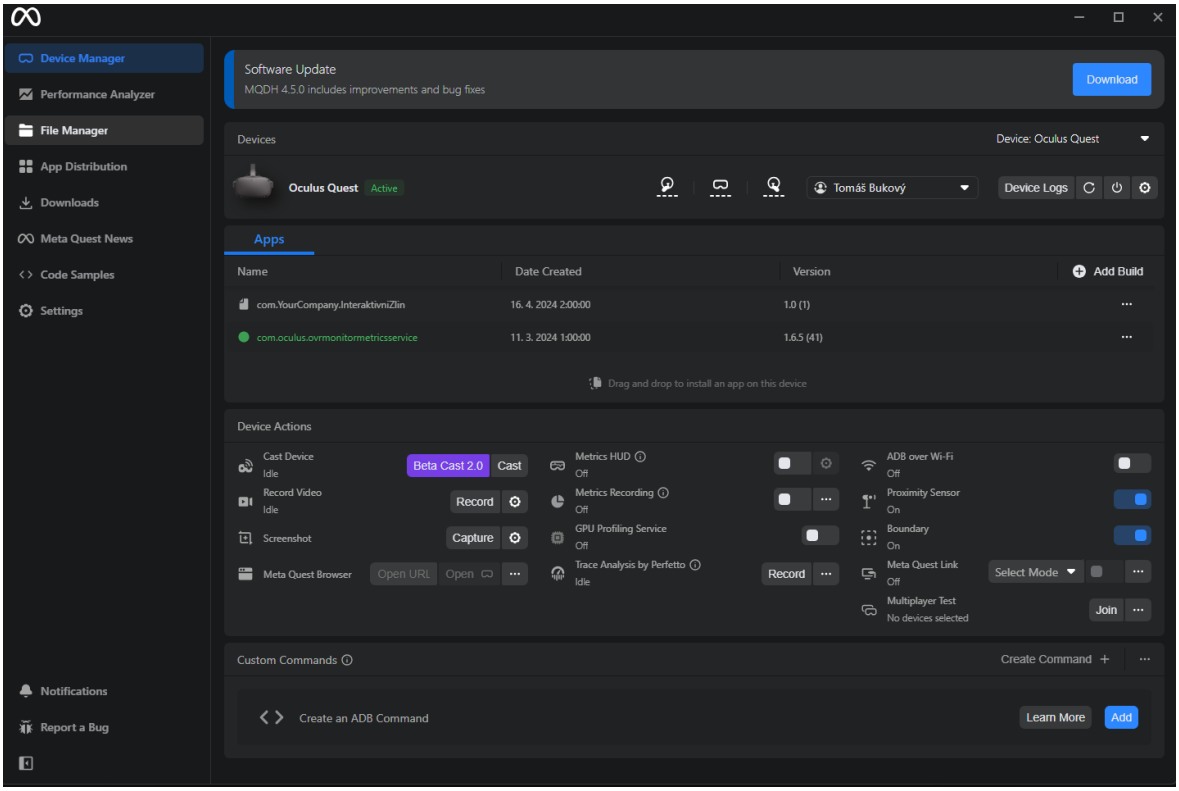
Mimo Javu je nutné mít v počítači specifická SDK a NDK, z tohoto důvodu se instalovalo Android studio, které mimo získání těchto souborů nebudeme využívat nikterak pro vývoj. Po nainstalování výchozím nastavení Android Studia jej znovu zapneme, klikneme na položku „*More Options*“ a vybereme „*SDK Manager*“. Na pravém dolním rohu aplikace zaškrtneme checkbox „*Hide Obselete Packages*“ a pod „*SDK Platfrom*“ Zaškrneme Android API 34, Android 12L (Sv2). Pod položkou „*SDK Tools*“ rozklikneme tab s názvem Android SDK Build-Tools 35-rc2" a v něm vybereme verze 34.0.0 a 33.0.1.

Poté pod NDK vybereme verzi 25.1.8937393. Poté budeme scrollovat až narazíme na „Android SDK Command-line Tools", v něm zaškrtneme checkbox pro verzi 11.0. Poté budeme hledat CMake a po rozkliknutí vybereme verzi 3.10.2. Nakonec akorát ověříme, že je zaškrtnuté také Android Emulator, Android Emulator hypervisor driver a Android SDK Platform-Tools. Poté stiskneme tlačítko Apply a počkáme až se vše potřebné nainstaluje, načež restartujeme počítač. Od této chvíle s Android Studiem už nebudeme pracovat. Tyto verze jsou doporučeny na základě dokumentace Meta, a i když je možné, že novější nebo například starší verze bude fungovat tak se nedoporučuje používat jinou, než je specifikována

7.1.3 Meta Quest Developer HUB

Dalším z stěžejních aplikací, které nám umožní pracovat s headsetem Meta Quest je aplikace Meta Quest Developer Hub. Při nainstalování nám aplikace sama detekuje 2 nainstalované *ADB*. My vybereme tu, která nemá v názvu „Meta“ ale využijime tu, která se doinstalovala po boku Android Studia. Po úspěšném nastavení můžeme připojit zapnutý headset k počítači pomocí USB-C kabelu. Po připojení kabelu si nasadíme headset a potvrdíme, aby headset důvěřoval počítači. Jakmile to provedeme tak klikneme na „*Device Manager*„ v levé části aplikace a uvidíme náš Quest. Uvidíme jeho úroveň baterie, že se nabíjí a stav baterií v ovladačích.

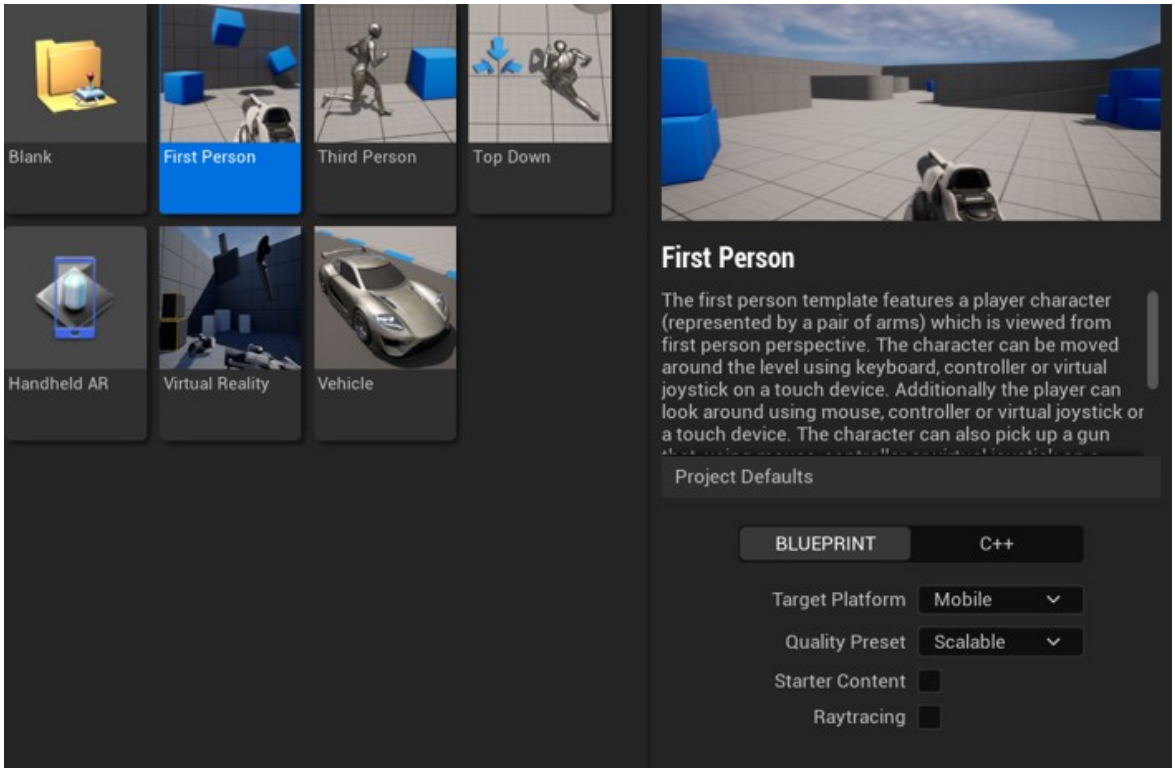
Aplikace nám bude sloužit k nahrávání sestavených aplikací do headsetu formou .apk souboru, dále slouží k jednodušší správě screenshotů a záznamů obrazovky. Aplikace nám také umožní sledovat vypínat zaseknuté aplikace, pokud samotný headset neodpovídá a nelze to provést na něm. Další funkcí aplikace je možnost sledovat realtime snímkovou frekvenci, využití CPU a GPU a další ukazatele. Vše funguje pouze pokud je ale headset připojení zmíněným kabelem.



Obrázek 20: Prostředí Meta Quest Developer Hubu

7.2 Nový projekt a prvotní nastavení

Jelikož z předchozí práce potřebuji pouze model samotného města, tak bylo nejvhodnější vytvořit nový projekt. Od verze 4.2.7 se v předvytvořeném VRTemplate ale nachází chyba jenž i přes mé snahy optimalizace znemožní aplikaci spustit plynule. Proto se vytvoří *First Person* projekt a začít znovu. Vhodné by bylo použít například Blank projekt, ale ten od verze 5 má poměrně velký terén. A po vytvoření projektu tento velký terén ovlivní načítání. Po vybrání First Person projektu je nastavit ještě tzv. *Project Defauls*, v těch vybereme položku *Blueprint* pro využívání vizuálního skriptovacího jazyka, druhou možností by bylo vybrání jazyka C++. Jako target plaftorm vybereme *Mobile* a Quality preset vybereme jako *Scalable*, Projekt pojmenujeme *InteraktivniZlin* a projekt necháme vytvořit.



Obrázek 21: Tvorba nového projektu

7.2.1 MetaXR plugin

Po načtení projektu je nutné stáhnout a nainstalovat Pluginy pro práci s VR headsetem od společnosti Meta. Jeden plugin se jmenuje *Meta XR Plugin* a druhý *Meta XR Platform.* Dle dokumentace společnosti meta je vhodné tyto pluginy nainstalovat přímo do adresáře Enginu samotného na místo adresáře jednotlivých projektů. V mém případě byla cesta *C:\Program Files\Epic Games\UE_5.3\Engine\Plugins\Marketplace.* Stažené zipy jsem extrahoval do této složky a poté je nutné tyto pluginy aktivovat v samotném projektu.

V otevřeném okně UE Editoru vybereme v liště záložek položku *Edit*, v ní vybereme položku *Plugins* a do vyhledávajícího okna napíšeme XR, pokud bylo nakopírování do adresáře enginu provedeno úspěšně objeví se tam dvě položky s názvy pluginů, který jsem instalovali. U obou v levé straně vybereme checkbox a zaškrtneme. UE bude poté vyžadovat restartování projektu. Po znovu zapnutí uvidíme v panelu nástrojů vedle dropdown menu pro *Platfrom* dvě nové okna, tyto okna se váží právě na zmíněné pluginy.

7.2.2 Nastavení projektu pro android a Quest

Po nastavení těchto pluginů jsme schopni nastavit projekt tak aby fungoval pro Meta Quest. Vybereme záložku *Edit* a tentokrát vybereme položku *Project Settings*, nyní nám vytvoří nové okno, ve kterém budeme nastavovat jednotlivé prvky:

1. V levé části budeme scrollovat dokud nenalezneme část s názvem „*Plaftforms*“ a v ní rozklikneme „*Android*“
2. V políčku pro *Minimum SDK Version* napíšeme hodnotu 29
3. Další hodnotou, kterou změníme je *Target SDK Version*, tam nastavíme hodnotu na 32
4. Poté stiskneme oblast, který nad tím svítí červeně a pokud jsme vše dobře nastavili, bude svítit zeleně.
5. Posledním krokem je zaškrtnutí checkboxu s názvem „*Package game data inside .apk*“, bez tohoto nastavení bychom nebyly schopni získat .apk, které poté budeme využívat pro instalaci aplikace.

V následujícím kroku budeme nastavovat samotné SDK a NKD, tedy ty, které jsem si ručně nainstalovali pomocí Android Studia. Jelikož jsem využil výchozí cestu uložení tak se mé soubory budou nacházet v tomto adresáři *C:/Users/tomas/AppData/Local/Android/Sdk*, tu vložíme do políčka pro Location of Android SDK a v Location of Android NDK vložíme *C:/Users/tomas/AppData/Local/Android/Sdk/ndk/25.1.8937393*. Dále je zde nutné vložit cestu k instalaci JDK, ta byla v mém případě *C:\Program Files\jdk-17.0.10*. Posledním krokem je nastavení SDK API Level a NDK API Level. První zmiňovaný bude mít hodnotu „*android-32*“ a druhý „*android-29*“.

Pokud bychom teď aplikaci sestavili, tak by sestavení proběhlo úspěšně a po spuštění aplikace by se nedala na headsetu ovládat. Chybí nám totiž specifická nastavení přímo pro Meta Quest. V levém sloupci pod nadpisem Plugins nalezneme MetaXR a ten otevřeme.

1. Vybereme XR API a zvolíme *Oculus OVRPlugin + OpenXR backend* – tato volba je v současné době doporučována společností Meta pro vývoj v UE pro platformu Quest
2. Dalším nastavením je *Color Space* na *Oculus*, jinak může dojít ke špatnému vykreslování bílé barvy, která by bez tohoto nastavení byla žlutá
3. Dalším nastavením je *Controller Pose Aligment*, to nastavíme na *Default*, pokud tomu tak již není.
4. Dále je nutné nastavit „*Advanced APK Packaging*“. Ten nalezneme pod záložkou Android a musíme zde přidat nový element „*Quest*“ Tohle docílí, že aplikace, kterou sestavíme bude schopná pracovat i s ovladači.

7.3 Instalace a spouštění

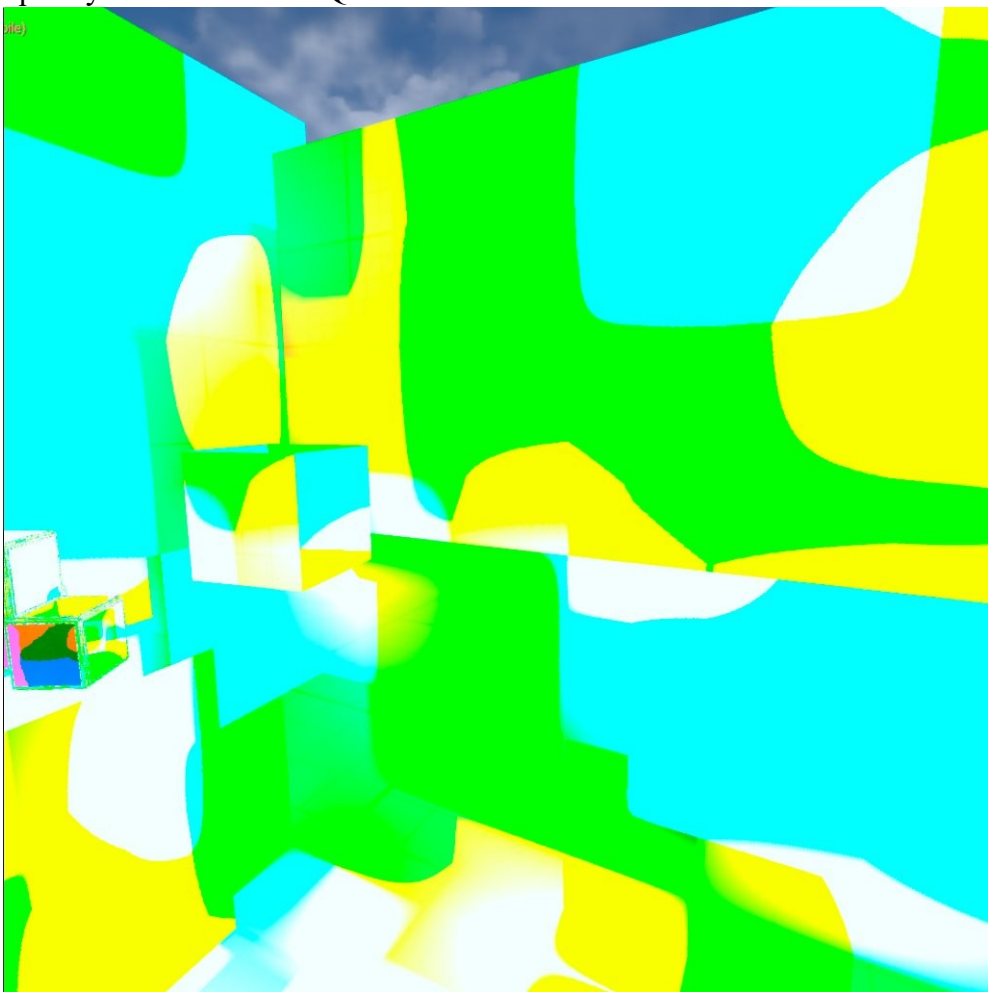
V současném stavu jsme schopni aplikaci sestavit a spustil na headsetu. Nutné je mít vždy připojený headset k počítači a v headsetu potvrzenou důvěru v toto zařízení. Spustit aplikaci lze několika způsoby. Nejrychlejším způsobem, který ale slouží primárně pro rychlé testování funkčnosti a využíval jsem jej při tvorbě grafického rozhraní je V*R Preview*. Jedná se režim, který nám umožní si naši scénu pustit v počítači bez nutnosti kompilace a stále ji ovládat pomocí ovladačů. Tento způsob nereprezentuje to, jak aplikace poběží na headsetu samotném, jelikož se o vykreslování scény stará hostující počítač, na kterém běží Unreal Editor. *VR Preview* se spustí tak, že v panelu nástrojů vedle zelené šipky „*Play*“ vybereme 3 tečky, které značí více možností a zde vybereme *VR Preview*. Pokud chci spustit aplikaci na headsetu musím ji prvně sestavit. Obě tyto možnosti sestavení najdeme na stejném panelu nástrojů jako *Play* ale nachází se pod tlačítkem Platforms. Jedním způsobem je tzv. *QuickLaunch*, jedná se o sestavení aplikace, která automaticky aplikaci nainstaluje do brýlí a rovnou spustí. Výhoda toto způsobu je možnost debugovat výstup aplikace v konzoli Unreal Editoru. V mém případě samotný build proběhl úspěšně ale aplikace se na headsetu odmítla spustit a bylo nutné vždy udělat *Quicklaunch* znovu, na podruhé se většinou zvládla spustit správně. Proto jsem musel zvolit druhý způsob. Tím je sestavení a poté manuální převedení aplikace do headsetu pomocí Meta Quest Developer Hubu pomocí vytvořeného *.apk* souboru.

Ve stejném menu pod tlačítkem *Plaftorm* najdeme Položku *Android* a v ní bude *Package project*. Délka sestavení závisí na velikosti výsledné aplikace, počtu modelů a na výkonu samotného hostujícího počítače. V mém případě to bylo pod 10 minut od spuštění sestavení. Po sestavení otevřu *Developer Hub* a vyberu záložku *Device Manager* a pod ikonkou *Oculus Quest* uvidím seznam aplikací. Za tlačítka *Add Build* vyberu *.apk* a nebo samotné .apk přetáhnu do prostředí Developer Hubu. Poté bude aplikace nainstalována na zařízení. V prostředí headsetu si v dolním panelu vybereme seznam aplikací. V pravé části nového okna klikneme na dropdown menu a vybereme *Unknown Sources*. V této části seznamu aplikací se bude nacházet spustitelná aplikace

7.4 Prví spuštění

Při prvním sestavení, pro otestování správnosti nastavení parametrů sestavení se aplikace spustila, ale ve stavu, kdy nebyla ovladatelná ani použitelná. Aplikace tvořila určitou formu barevných artefaktů po celé scéně. Samotný headset sice zvládal snímat pohyb hlavy ale pohyb nebyl plynulý a aplikace běžela svoji snímkovou frekvenci okolo

25FPS, což je velmi nenáročný projekt ještě bez modelu města velmi nízké. Dále nebylo možné ve scéně vyvolat ovladače a s ničím nešlo interagovat. To samé platí pro pohyb po celé scéně, ten nebyl možný. Takže bylo nutné dodělat nastavení pro VR a optimalizovat aplikaci pro využití na headsetu Quest 1.

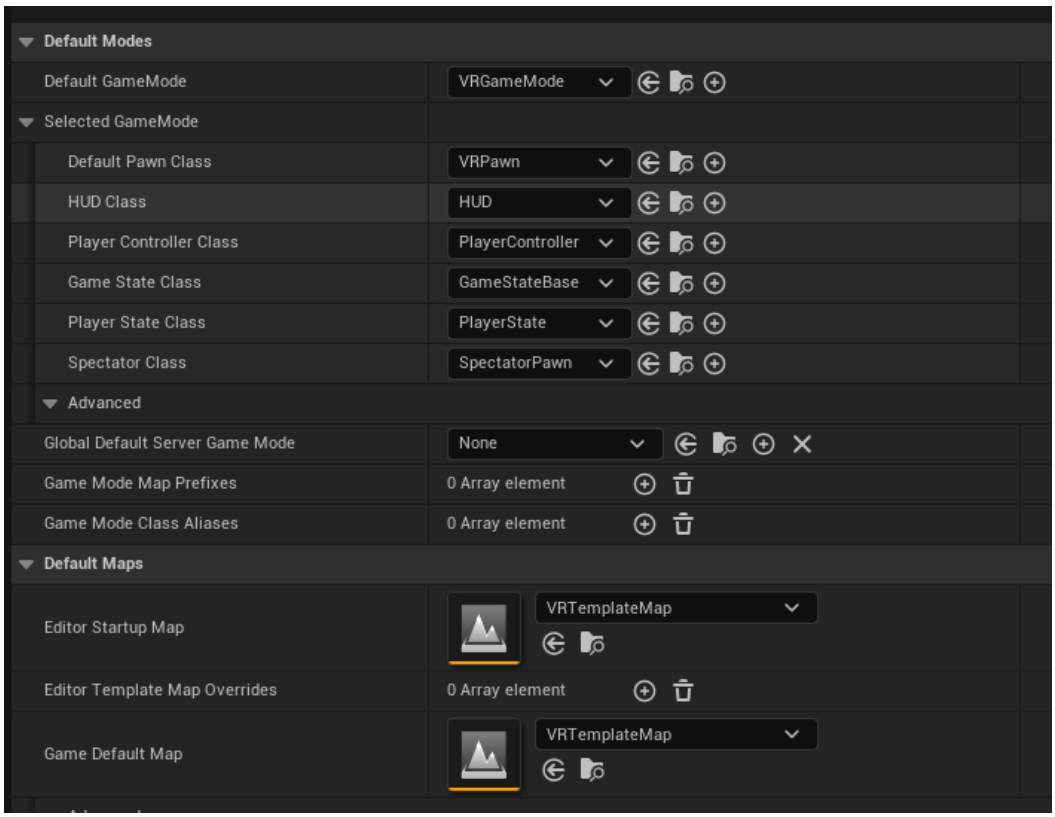


Obrázek 22: První spuštění

7.5 Nastavení VR funkcionalit

I přes nefunkčnost samotného *VRTemplate* projektu můžeme importovat jeho prvky a tím stále využít možnosti předkonfigurovaných *Actorů* pro *VRPawn* a ovládání například teleportace. Importovaní těchto nástrojů provedeme tak, že otevřeme *Content Browser* a v levém horním rohu okna uvidíme tlačítko se zeleným symbolem plus a nápisem *Add*, po kliknutí se nám objeví kontextové menu a v něm vybereme 2 položku od vrchu s názvem „*Add Feature or Content Pack*“. Po kliknutí se nám otevře podobné okno, jako to, které jsem využívali při vytváření projektu. Vybereme záložku *Blueprint* a v ní *Virtual Reality*. Po kliknutí na modré tlačítko *add to project* se nám jeho nástroje přidají do projektu. Po

importování, které zabere pár sekund se v adresáři projektu objeví několik nových složek. Tyto složky obsahují nastavené prvky pro VR. Nyní budeme schopni přepnout projekt z výchozího projektu 1. osoby na projekt pro Virtuální realitu.



Obrázek 23: Nastavení World Settings pro VR

Unreal Engine pracuje na základě tzv. *Herního režimu*. Jedná se o sadu pravidel, které určují přechody mezi scénami, jak hra interaguje s hráči a jak se zobrazuje HUD a nebo například to, čím se bude hra ovládat. Tyto hodnoty budeme muset změnit tak abychom docílili herních pravidel pro VR. Otevřeme tedy *project settings*, který se nachází pod záložkou *Edit* v horním panelu, otevření okna klikneme na lupu a vyhledáme „*Default GameModes*“. Poté se nám zobrazí okno a v *Default Gamemode* vybereme možnost *VRGameMode*. Část pod *Selected GameMode* se změní sama podle *Defaultního režimu*. Ale je lepší zkontrolovat jestli se správně nastavil. Nejdůležitějšími body jsou *Default Pawn Class*, v něm musí být *VRPawn*, *Player Controller* musí být *PlayerController* a posledním důležitým je *Specator Class*, kde musíme nastavit *SpectatorPawn*. Tato nastavení říkají aplikaci, že se má spustit ve VR a má očekávat ovládání pomocí pohybových ovladačů. Posledním krokem bylo aktivování „*Start in VR*“

7.5.1 Teleportování

Teleportace slouží k pohybu po scéně ve Virtuální Realitě. I když některé aplikace samotnou teleportaci nevyžadují a uživatel je ve statické podobně umístěn na jednom bodě tak v případě diplomové práce je vhodné aby se mohl uživatel pohybovat například na okolo celého modelu města. V tomto případě je nutné implementovat teleportaci do scény. Samotná funkcionalita teleportace se provádí za použití tzv. Navigačních bodů, které umístíme do scény, navigační plocha se poté vytvoří sama a my s ní můžeme manipulovat a ovlivňovat kam bude možné a kam se nebude dát teleportovat.

K přidání navigačních bodů do scény použijeme pomocí Objektu *Nav Mesh Bounds Volume* a *Nav Modifier Volume*

1. Otevřeme si Okno, které nám umožní umisťovat *Actory* do scény. V záložkách vybereme možnost *Windows* a v ní *Place Actors*, na levé straně editoru se otevře okno s *Actory*
2. Vybereme záložku *Volumes*
3. Do scény přetáhneme poté objekt *Nav Mesh Bounds Volume*, ten roztáhneme tak aby pokrýval celou plochu scény.
4. Poté do scény přetáhneme *Nav Modifer Volume*, ten slouží k nastavení kolizí ve scéně a ten roztáhneme tak aby pokryl zmenšený model města a tím zamezíme možnosti teleportovat se na něj.

Po přetažení bude UE vytvářet navigační body. Jelikož model není rozsáhlý tak to zabere chvíli. V tento moment jsme schopni se po scéně volně teleportovat.

Jelikož jsme si do projektu naimportovaly nástroje z *VRTemplate*, tak ovládání teleportace je již nastaveno a nemusíme manuálně vytvářet nastavení jednotlivých tlačítek akcí.

7.6 Optimalizace aplikace pro VR

Pro zaručení největší možné plynulosti aplikace na headsetu Meta Quest muselo být provedeno hned několik kroků. Pro sledování jednotlivých prvků aplikace byl do zařízení nainstalován nástroj *OVR Metrics*. Jedná se o nástroj, který lze stáhnout ze stránek společnosti Meta a umožňuje nám sledovat jednotlivé metriky a stavy běhu aplikace. Mezi nejdůležitější ukazatele patřili snímky za sekundu. Dle doporučení by aplikace na Virtuální realitu měla dosahovat na stabilních 72FPS, při nižších hodnotách u delšího používání může totiž docházet u některých jedinců k bolestem hlavy a nebo závratím. Snaha byla aplikaci optimalizovat natolik, aby běžela nejlépe, jak jen to jde. A to i přes stáří samotného zařízení a náročnosti modelu, jehož optimalizace bude popsána v kapitole 9.

Pro tvorbu jednotlivých úprav otevřeme v záložkách Edit a v ní položku *Project Settings*

1. Mezi nejdůležitější body bylo zapnutí *Forward Shading* místo *Deferred Rendering*. Deferred Rendering vyžaduje více paměti a výpočetního výkonu, protože ukládá velké množství informací o scéně pro každý pixel, což je náročnější pro zařízení s omezenými zdroji jako Quest 1. Zatímco Forward Shading jsou světla a stíny počítány přímo během průchodu scénou (render pass) pro každý viditelný pixel. To znamená, že každý pixel na obrazovce projde výpočtem osvětlení a stínů na základě světel a materiálů v scéně. [39]
2. Dalším krokem bylo zapnutí *Instanced Stereo*. *Instanced Stereo* funguje tím, že místo vykreslování scény pro každé oko zvlášť, což může být velmi náročné na výpočetní výkon, se scéna vykresluje pouze jednou a poté se výsledek použije pro obě oči. Tím se snižuje zátěž na procesor a grafický čip, což je vhodné pro zařízení s limitovaným výkonem jako je Quest 1
3. Poté bylo nutní vypnutí *Mobile HDR*, to je pro zařízení Quest 1 až moc náročné a pro tuto práci nepřináší žádné výhody
4. Dalším krokem na zlepšení výkonu a plynulosti aplikace bylo zapnutí Dynamického rozlišení a snížení minimální hustoty pixelu na 0,6. Maximální hodnota byla ponechána na hodnotě 1,0. Díky tomu při náročnější části scény bude automaticky sníženo rozlišení aby nedocházelo k drastickému propadu snímků za sekundu.

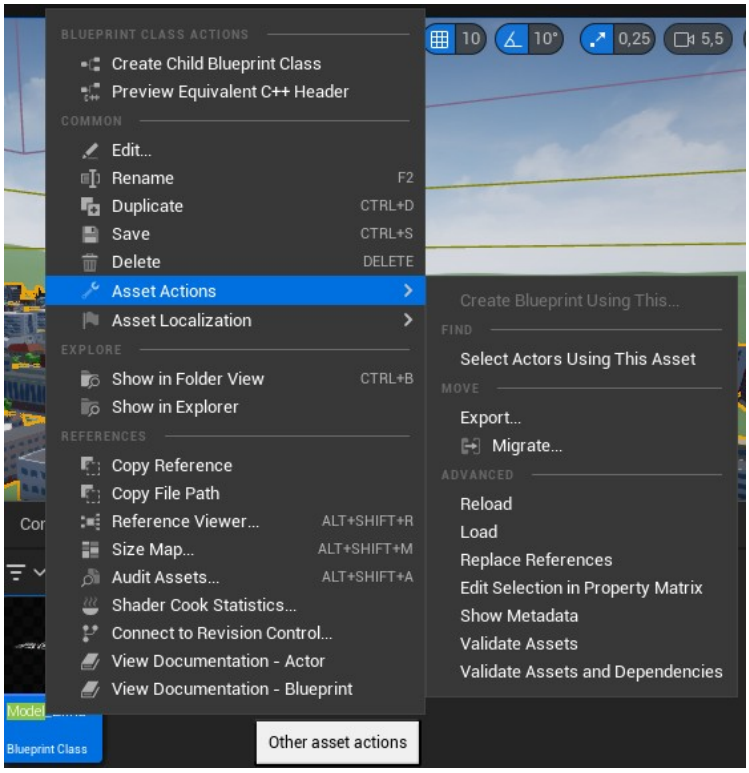
5. Dále bylo nastaven *Suggested Cpu Perf Level* a *Suggested Gpu Perf Level* na *SustainHigh*, to za cenu snížení výdrže baterie zvyšuje takt zmiňovaných komponent a dokáže je při vyšším taktu držet delší dobu
6. Krokem, který zvýší obrazovou kvalitu za minimální ztrátu výkonu bylo zapnutí Mobilního Anti-Aliasingu. Tady jsem zvolil možnost Multisample Anti-Aliasing neboli zkráceně MSAA a jako hodnotu pro sample count jsem zvolil 4x
7. Následovalo zapnutí *Round Robin Occulusion Queries*, zvyšuje výkon tím, že střídá rendering occlusion mezi jedním okem každý snímek na místo obou.
8. Posledním krokem optimalizace bylo deaktivování post process efektů a dalších efektů jako byl *LensFlare* nebo vypnutí všech dynamických světel. Jelikož headset oculus Quest 1 v novějším Unreal Enginu 5 s nimi nepracuje nejlépe.

Po provedení optimalizace se mi povedlo projekt bez modelu z původních 25FPS při prvotním sestavení rozchodit na 72 snímků za sekundu.

Obrázek 24: ORV overlay v aplikaci

8 IMPORT MODELU DO PROJEKTU

Pro přidání modelu do scény je jen nutné exportovat z předchozího projektu. Unreal Engine naštěstí obsahuje, který tento export a následní import umožní bez nutnosti kopírovat manuálně soubory mezi projekty. Otevřeme původní projekt a vybereme si v *Content Browseru Actor Class* s názvem *Model_Zlín*. Kliknu na něj pravým tlačítkem myši vyberu možnost *Asset Actions* → *Migrate*. Poté se mi otevře okno, které obsahuje seznam veškerých assetů, který *Model_Zlín* obsahuje a poté kliknu na Ok. V otevřeném průzkumníku windows poté vyberu cestu k novému projektu a vyberu složku *Content* a potvrdím. Nyní bude v novém projektu naimportovaný samotný model Zlína i se všemi budovami, terénem a materiály.



Obrázek 25: Migrace projektu

8.1 Aktualizace a rozšíření modelu

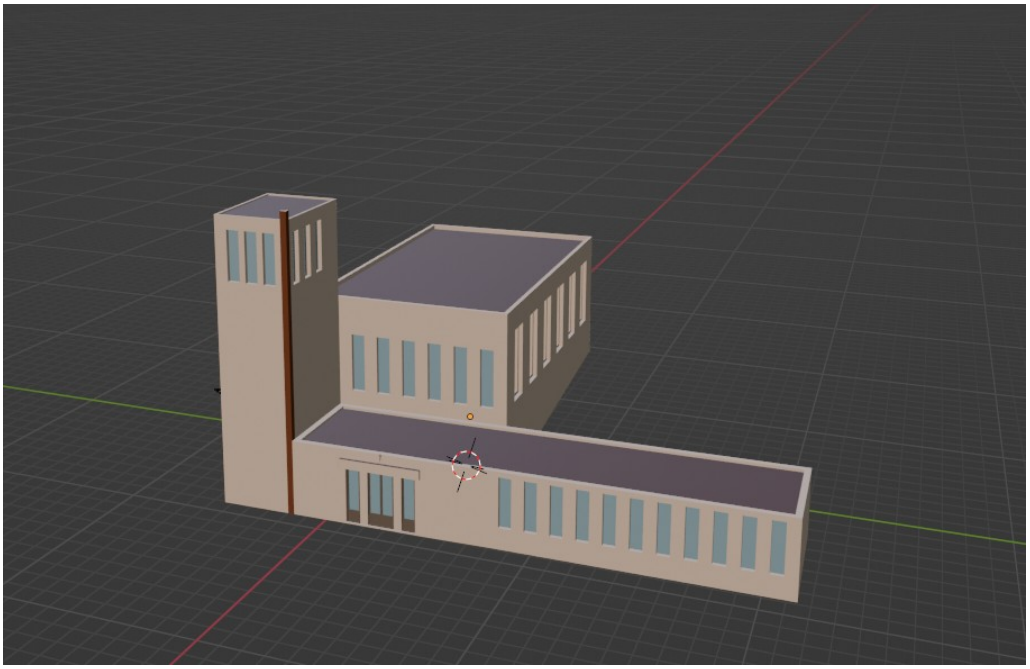
Nové modely budov do modelu Zlína se drželi obdobného stylu jako veškeré předchozí budovy. To znamená, že nebyly používány žádné komplexní textury a cílem budovy bylo připomínat jeho reálnou podobu. Jelikož model samotného Zlína bude ve výsledku práce zmenšen, tak je v hodné vynechat některé detaily, které by při tak malé velikosti nebyly ani vidět. Dále se tím ušetří počty trojúhelníku, a tím se nebude zbytečně zvyšovat náročnost poměrně komplexního modelu města jako celku.

Původní model neobsahoval například model Evangelického kostela na ulici Štefánikova. Z původní aplikace jsem proto vyexportoval jeho soubor *.fbx*. Export provedu tak, že v *Content Browseru* naleznu požadovaný objekt s názvem „Evangelický kostel“ a kliknu na něj pravým tlačítkem myši. V zobrazeném kontextovém menu *vyberu Asset actions* a poté položku *Export*. V průzkumníku Windows vyberu místo kam si chci exportovaný .fbx soubor uložit a stisknu klávesu „Uložit“. Tím práce v Unreal Enginu s modelem končí a nyní ho importuji do prostředí aplikace Blender.

V Blenderu si vytvořím nový projekt, ten bude typu *General*. Projekt poté pomocí klávesy *CTRL-S* uložím a pojmenuji. Nyní do vytvořeného projektu jsem schopný naimportovat vyexportovaný *FBX* soubor z modelu. Importovaní provedu tak, že v listě záložek vyberu záložku *File → Import → FBX*. Poté v integrovaném prohlížeči souboru najdu a vyberu požadované *fbx*. Pro referenci na vzhled budovy si nainstaluji rozšíření „*Image as Planes*“. Jedná se o rozšíření, které umožní a ulehčí importování obrázků jakožto textury na plane, bez nutnosti ručního mapování Textury na objekt. V záložce Edit vyberu položku *Add-Ons* a vyhledám „*image*“, Poté zaškrtnu položku „*Import-Export: Import Images as Planes*“ a zavřu okno Addonů.

Nyní za použití Google Map a StreetView najdu Evangelický kostel a udělám screenshot. Ten poté za použití tohoto addonu vložím do scény. Vložení provedu tak, že zmáčknu klávesovou zkratku *Shift + A,* v ní vyberu *Images* a v ní *Images as Planes* a vyberu uložený screenshot z map. Za pomocí klávesy S poté roztáhnu referenční obrázek tak aby pokrýval celou šířku objektu Kostela. Poté začnu upravovat rozměry modelu, tak aby připomínal samotný kostel. Pomocí nástroje *Loopcut* udělám 2 řezy, tak abych mohl oddělit zadní část kostela a věž. Poté použiji nástroj *Extrude* vytáhnu věž a zadní část kostela do požadované výšky.

Jakmile jsem měl vytvořenou kostru kostela, vytvořil jsem materiál, který bude pokrývat většinu modelu a barvou bude připomínat fasádu kostela. Pro výběr barvy sloužilo kapátko a právě Plane s texturou na kterých byla fotka z Google map. Samotný odstín není nejpřesnější a musel být upraven a jemně zesvětlen. Pro materiál oken byl použit stejný materiál jako u ostatních budov v původním modelu. Obdobně byl vytvořen model vily Tomáše Bati, Park Tower a památník Tomáše Bati.



Obrázek 26: Model Evangelického kostela

Model jsem následně exportoval z programu Blender a importoval do prostředí Unreal Engine

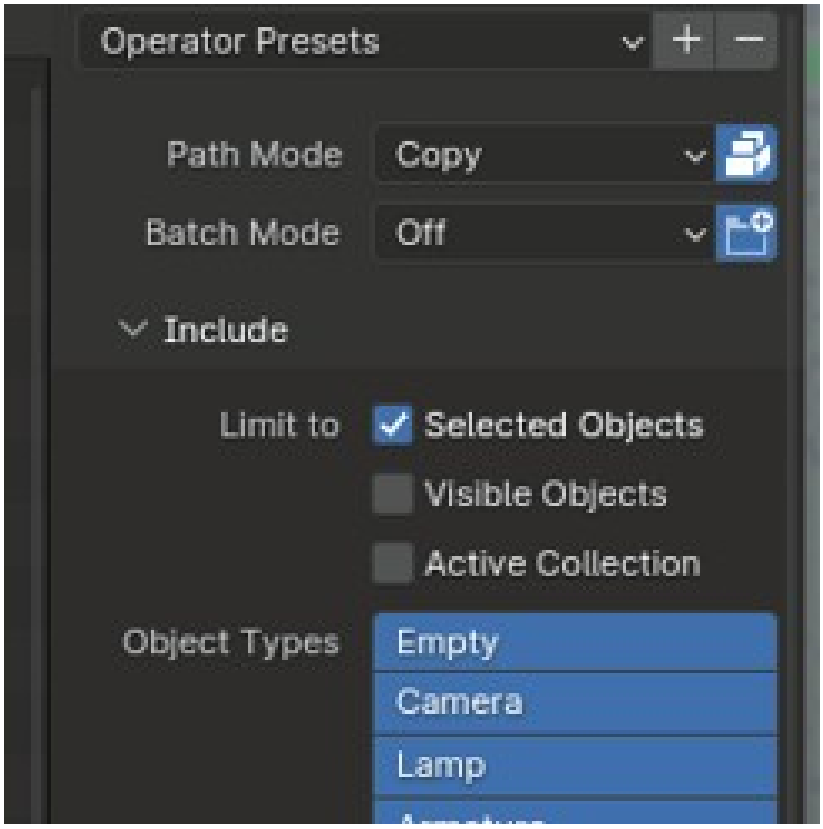
1. Vybral jsem objekt a kliknul na *File → Export → FBX (.fbx)*
2. Poté zaškrtnul možnost *Selected Objects*, nastavil si cestu a kliknul na *Export*
3. V prostředí Unreal engine jsem otevřel *Content Browser*, kliknul na tlačítko *Import*
4. Naleznu vytvořený *.fbx* soubor a kliknu na *Open*
5. V Unreal Engine vyskočí okénko *FBX Import Options*, v něm nic neměním a kliknu na *Import All*

Od doby kdy vznikla původní práce došlo ve Zlíně ke zbourání nejstarší budovy fakulty technologické Univerzity Tomáše Bati. Z tohoto důvodu bylo nutné i samotnou budovu z modelu odstranit. Pro odstranění budovy se otevře *Actor* ve kterém se nachází model města. V novém okně se předpnu do *Viewportu* a poté se pohybem kamerou dostanu na místo, kde se tato budova nachází a kliknu na ni. Na pravé straně se mi otevře panel s detaily a zkopíruji si jméno budovy. To poté vložím do vyhledávání na levé straně v okně pro komponenty a jakmile se mi zobrazí tak jej smažu. Po změnách jej uložím a kliknu na tlačítko *Compile*. Aplikace chvilku zamrzne, jelikož je model poměrně veliký. Budova bude smazána z modelu Zlína ale v projektu stále zůstane a mazat jsem se ji nerozhodl.

8.2 Úprava terénu

V původní aplikaci se nacházel jednobarevný terén, pro potřeby aplikace z první osoby to nedělalo žádný problém. Při zmenšení modelu, kdy se nahlíží na celý model z vrchu ta barva nevypadá nejpřirozeněji. Je tedy nutné terén upravit. Úprav bude docíleno v prostředí aplikace Blender. Z prostředí Unreal Engine jsem si exportoval samotný Mesh Terénu ve formátu *.fbx*. Ten poté otevřel v programu Blender. Dále jsem vytvořil screenshot modelu z výšky, který jsem umístil nad terén a nastavil na něm průhlednost. Takto sloužil jako referenční obrázek abych věděl, kde se nachází budovy. Postup k vytvoření jednoduché textury byl následující

1. Provedl jsem *UV Unwrap* na samotný terén, přepnul jsem se do Editačního režimu modelu pomocí klávesy *TAB* a zmáčkl klávesu *A*, pro výběr všech částí a zmákl klávesu *U* a vybral možnost *Unwrap*
2. Poté jsem se v panelu nástrojů do režimu *Texture paint*
3. Poté jsem skrze *Texture Slot* vytvořil novou texturu, její výšku a šířku nastavil na *2048 px x 2048 px* a vypnul *Alpha* kanál. Jako barvu jsem vybral stejnou barvu, jakou mají v modelu chodníky v hex hodně: *bababa*
4. Poté jsem za použití nástroje štětec kreslil po terénu a vytvářel jednotlivé barevné části
5. Jakmile jsem byl hotový, tak jsem terén exportoval pomocí *File → Export → FBX (.fbx)* a v parametrech jsem u *Path mode* z dropdown menu vybral možnost *Copy* a zatrhneme tlačítko vedle něj.
6. Poté jsem v sekci *Include* vybral *Selected Objects*, aby importoval pouze terén
7. Následně vybral složku a kliknul na tlačítko *Export*
8. Model jsem následně importoval do programu Unreal Engine



Obrázek 27: Parametry exportu terénu

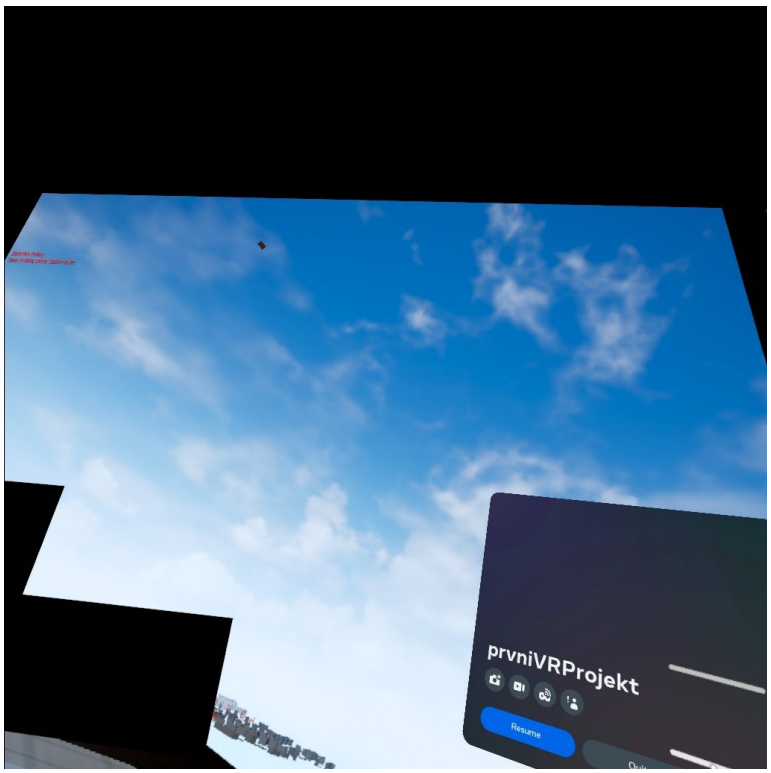
8.3 Oprava Normál

Jedním z nedostatků byly rozbité normály u některých budov. Naštěstí od verze UE5 jsme schopen opravit normály bez nutnosti samotný mesh exportovat do programu jako je například Blender a nebo instalovat plugin jako tomu bylo v původní práci. V přepínači režimů UE vybereme záložku Modeling Mode a seznamu seznamu režimů klikneme na dvě šipky na jeho konci. Poté vybereme možnost *Attribs*. Po kliknutí se nám otevře záložka jenž mi možní vybrat položku *Normals*. V novém okně poté zaškrtneme položku *Fix Normals*. Poté jen kliknu na potvrdit a objekt bude opraven.

9 OPTIMALIZACE MODELU PRO VR

Původní aplikace měla celý model vytvořený a vložený do třídy *Actor*, tohle řešení přinášelo řadu výhod pro práci s modelem města. V prohlížeči objektů scény se totiž nacházel pouze jeden prvek Model_Zlína. Tento objekt v sobě obsahoval veškeré budovy a vymodelovanou silnici a tvářil se jako kdyby byl celý model pouze jeden velký objekt. Výhoda byla taková, že při manipulaci s městem se dalo posunout pouze s jedním prvek na místo jednotlivých budov a to platilo i pro zvětšování či zmenšování. Pokud bych chtěl upravovat jednotlivé budovy, mohl jsem otevřít editaci samotného *Actora*. To se dá provést tak, že v prohlížeči objektů ve scény najedu na model a kliknu na možnost „*Edit Model_Zlína*“. Ve skutečnosti obsahoval *Actor class* reference na veškeré modely budov města, takže i při změně v *Content Browseru* se změna provedla v *Actor Class* modelu města, při mazání modelu z *Content Browseru* zobrazí UE upozornění, že existuje v *Actor Class* modelu a jestli jej chci nahradit.

Přímé vložení modelu do projektu a provedení testovacího sestavení způsobilo to, že aplikace zamrzla, samotný Quest přestal reagovat jak na pohyby hlavou, tak zmizeli i ovladače. 3D scéna, kterou sám Quest vytváří byla rozbitá a nacházeli se v ní černé čtverce. Aplikace musela být ukončena pomocí Meta Developer Hubu skrze D*evice Managment.* Z tohoto důvodu je nutné samotný model optimalizovat.



Obrázek 28: Spuštění s neupraveným modelem

Tento způsob používání *Actor Class* pro město byl vhodný spíše pro počítačovou aplikaci, jelikož s ním moderní grafická karta neměla problém, ale u zařízení Meta Quest jsem narazil na limitaci a to v počtu *DrawCalls*. Jedná se o základní mechanismus pro vykreslování scény a všech vizuálních prvků, včetně objektů, textur, stínů, efektů a dalších. Každý objekt vyžaduje jeden *drawcall*, to platí ale i pro materiály, který se v modelu nachází. Například vymodelovaný Evangelický Kostel má na sobě 5 materiálů a každý je použitý na jinou část. Takže Kostel pro své vykreslení vyžaduje celkově 6 Drawcallů. Předchozí práce měla vymodelovaných okolo 800 budov, kdy každá budova měla minimálně 3 materiály (Okna, Střecha, Fasáda) a dle oficiální dokumentace pro Meta Quest je optimální počet *Drawcallů* pro toto zařízení pod hranicí 500 Drawcallu. Pro zjištění počtu *DrawCallu* existuje v Unreal Engine nástroj, který se ale schovává pod příkazem „*stat rhi*“ ve spodním panelu nástrojů. Tento nástroj přes celou obrazovku zobrazí tabulku, která se v reálném čase mění a ukazuje mimo další statistiky počet Drawcallů pod názvem *DrawPrimitive Calls*.

Po neúspěšném sestavení aplikace, jsem při použití tohoto nástroje zjistil, že model Zlína obsahuje přes 15 000 těchto primitivních *Drawcallů*. Což je mnohonásobně vyšší než doporučení pro zařízení Quest a vysvětluje to i problém, proč aplikace nebyla schopná ani na zařízení běžet, například i zasekaně s nižší snímkovou frekvencí. Řešením tohoto problému je mergování všech objektů do jednoho velkého objektu a to platí i pro materiály. Tímto způsobem dokážeme snížit *Drawcally* na pouhé 2. Tedy jeden „slepený“ model města a jeden velký mix materiálů. Tento způsob ale ztíží veškeré další úpravy modelu, jekož se nyní vše nachází dohromady. Pro tyto potřeby byl v projektu ponechán stále povodní *Actor Class* s modelem města se všemi jednotlivými budovami a texturami, tak jak to bylo v původní práci. Při pozdních úpravách bude nutné provést mergování znovu. Mergování se provede následovně:

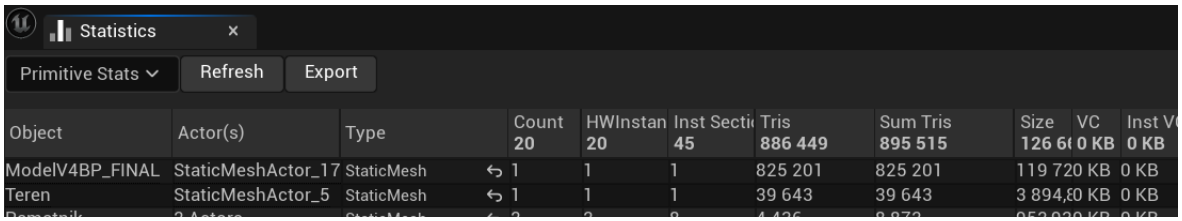
1. Vezmu Actor class s modelem a přetáhnu jej do scény
2. V panelu záložek vyberu možnost *Actor* → *Merge Actors Settings*, Unreal Engine se mě zaptá kam chci ukládat a otevře se mi okno ve kterém budou zaškrnuty veškeré budovy.
3. Na pravé straně okna je Nastavení pro samotné Mergování. Položka pro meteriály je ale zašedlá a pro její aktivaci jsem musel přepnout *LOD Selection* type na *Use Specific LOD Level*
4. Poté bude možné zatrhnout *Merge Materials*

Další limitací modelu byl velmi vysoký počet trojúhelníků. Trojúhelník nebo tris je základním stavebním blokem pro vytváření všeho co je ve hře viditelné. Počet a kvalita samotných trojúhelníků ovlivňuje výkon samotné aplikace. U původní aplikace se nejednalo o žádný problém a moderní počítač s novějšími komponenty neměl problém utáhnout vyšší počty trojúhelníků v modelu. Tak jak existovala limitace na počet *Draw-Callů* pro headset Meta Quest, tak existuje stejná limitace pro počet trojúhelníků ve scéně. Dle dokumentace od společnosti Meta je optimální hodnota pro trojúhelníky ve scéně 300 000 až 500 000. Původní model má těchto trojúhelníků něco málo přes 1 600 000. Takže tato hodnota je velmi vysoká oproti optimálnímu počtu. Tohle je mimo jiné způsobené velkou rozlohou modelu a také celkový počtem jednotlivých objektů ve scéně. Od verze Unreal Engine 5 lze tyto hodnoty upravovat i bez nutnosti externích programů jako je například Blender. Tato operace lze provádět pouze pro Statické Meshe a jelikož jsem z důvodu optimalizace Drawcall musel spojit veškeré modely do jednoho, tak můžu tohle zjednodušení modelu použít přímo na celé město na jednou. Bez předchozího mergnutí celého modelu bych musel upravovat všechny budovy zvlášť. Samotné zjednodušení provedu následovně.

1. Vyberu si model města a přepnu se do Modeling Mode v přepínači režimů
2. Poté vyberu záložku s nástroji pro práci s *Mesh*
3. Následně vyberu funkci *Simplify*
4. U té nastavím *TargetPercentage* na hodnotu *51%*. V Případě nižších hodnot již docházelo k deformaci některých částí modelu.
5. Jakmile Unreal Engine zpracuje uvidím na spodní části hlavního okna tlačítko pro potvrdit.

Kombinace mergování objektů, materiálů a následné zjednodušení počtu trojúhelníků umožnilo to, aby se model dal na headsetu spustit aniž by zasekl celý headset. Ale i přes veškerou snahu model optimalizovat co nejlépe tak se mi nepovedlo snížit počet trojúhelníků co nejblíže k hodnotám, které jsou doporučené společností Meta. I když se stále nabízela možnost smazat veškeré budovy, které nejsou vymodelovány, tak jejich smazání zapříčinilo mimo mírného zvýšení snímků za sekundu i velký negativní efekt na celkový vzhled modelu. Bez těchto budov model působil velmi prázdně. Proto jsem i za cenu nižšího výkonu celé aplikace se rozhodl, že ponechání bude vhodné. Při pohledu na model snímková frekvence klesne na přibližně 30FPS, tato hodnota je nízká ale i se zmíněným smazáním nevymodelovaných částí jsem se nedostal ani přes hranici 40FPS. Novější verze Meta

Quest 2 nebo 3 zvládají mnohem vyšší počty trojúhelníků, každá nová generace ten počet zvedla skoro dvojnásobně oproti předchůdci a zde se ukazuje i samotné stáří headsetu. Výsledný model města měl něco málo přes 825 000 trojúhelníků.



Obrázek 29: Primitivní statistiky pro model města

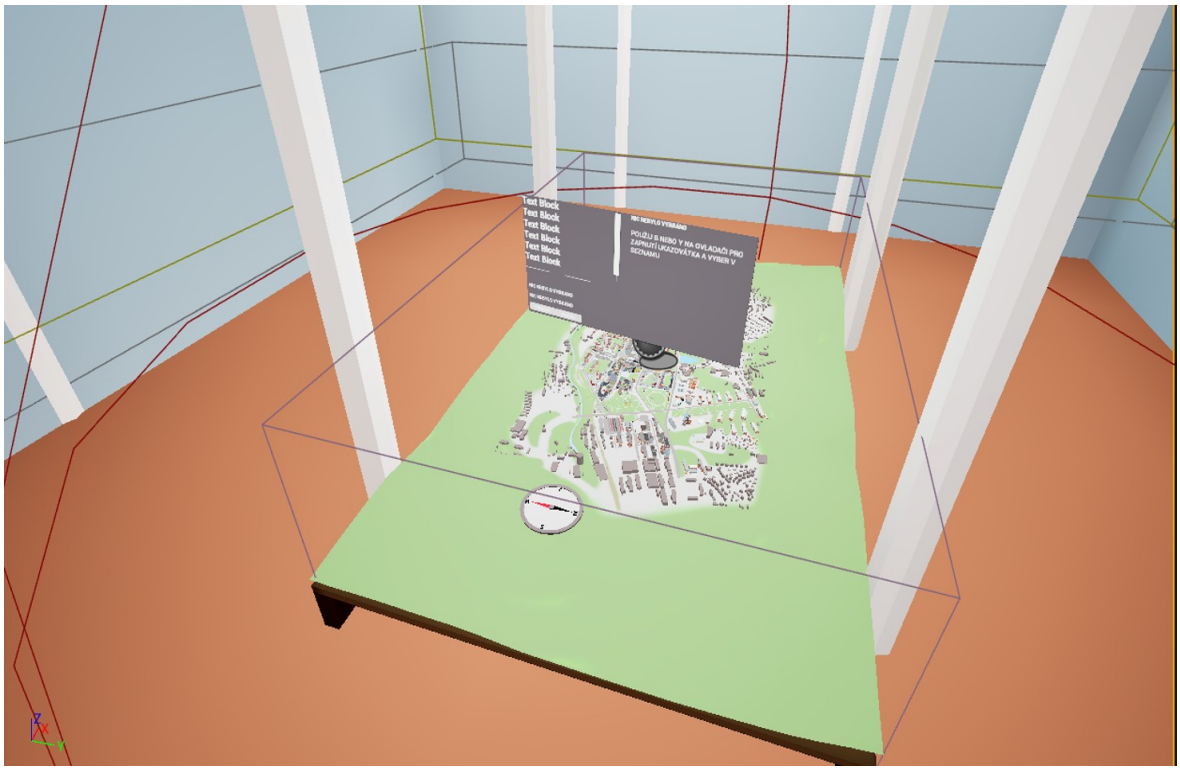
9.1 Tvorba scény

Jakmile byl model připravený byl vložen do scény. Výchozí šedý sandbox, který se nacházel v předvytvořeném projektu vymažu a nahradím modelem památníku Tomáše Bati, ten byl modelován stejným způsobem jako Evangelický Kostel. I zde jsem dbal na používání jednoduchých materiálů. Na rozdíl ale od Evangelického kostela se zde musel vytvořit interiér. A to stejným způsobem tak, aby model památníku zapadal do scény se zmenšeným modelem města. Zmenšenina byla poté vložena do modelu města na místo původního starého modelu. Mezi sloupy vložím zmenšený model města. Jeho velikost byla nastavená v sekci Scale pro všechny osy na 2,5 jelikož výchozí velikost 1,0 byla příliš malá.

Model památníků nemá žádné okna, proto bylo nutné vložit do scény osvětlení. Unreal Engine má k dispozici řadu různých typů světel, jak pro nasvícení interiéru tak svě-ta. V mém případě jsem vyžil *PointLights*. Ty do scény přidám tak, že v prohlíženi Actorů vyberu záložku *Lighting* a světlo přetáhnu do scény. Z důvodu velikosti bylo nutné světlo nastavit a také rozkopírovat vícekrát po scéně. Nejdůležitější nastavení, které mělo i vliv na výkon celé scény byl typ světla. Unreal Engine umožňuje aby světla byly *statická*, *movable* a nebo *stationary*. Poslední dvě vyjmenované možnosti ale mají negativní dopad na výkon, jelikož se výpočet stínů a odrazy tvoří *realtime* za běhu aplikace. Proto jsem zvolil statické světla. Tyto mají výhodu, že se dají tvz. *Zapéct*. Tento proces provedu tak, že v panelu záložek vyberu *Build → Build Lighting Only*. Po provedení budu mít jednot-livé světla zapečené.

Aby model města ve scéně nelétal, byl pod něj vložen jednoduchý model stolu. Samotný stůl byl vytvořen v programu Blender a byla na něj nanesená hnědá barva. Stůl byl poté importován do projektu do složky Stůl a přetažen do scény. Pro snazší orientaci v

modelu byl do rohu modelu umístěn model kompas. Samotný kompas byl také vymodelován v programu Blender.



Obrázek 30: Scéna

10 PŘEVOD GEOGRAFICKÝCH SOUŘADNIC

Jelikož cílem aplikace je zobrazit nejen informace ale i lokaci na zmenšeném modelu města, bylo nutné provádět přepočet geografických souřadnic formátu *WGS84* na souřadnice, se kterými si Unreal Engine rozumí. Pro tyto potřeby obsahuje Unreal Engine rozšíření *Georeferencing* [36]. I když se jedná o součást herního enginu, tak je výchozím nastavení vypnutá. Zapnutí probíhá stejně jako aktivace Pluginů. V panelu záložek vybereme *Edit* → *Plugins* a v něm vyhledáme *Georeferencing* a zaškrtneme checkbox. Nyní je možné pracovat s převodem souřadnic.

V projektu si otevřeme Prohlížeč *Actorů* pomocí záložky *Windows* → *Place Actor* → a vyhledáme *Georeferencing*, ten přetáhneme poté do scény. Po té co bude *actor* ve scéně jej musíme nastavit.

1. Jako *Planet Shape* vybereme *Round Planet*
2. Poté zrušíme označení *Origin at Planet Center* a *Origin Location in Projected*..
3. Aby se lépe pracovat s geografickými daty a jejich převodem v prostředí Unreal Engine nastavíme *Origin Latitude* a *Origin Longitude* na hodnoty, na kterých se nachází Zámek Zlín. Původní střed souřadnic je totiž nastaven a lokaci *Unreal HQ* a hodnoty, které by při práci se Zlínem vracel by byly moc vysoké. Nyní máme nastavený georeferencing tak, že střed světa je na Zlínském zámku.
4. Nyní musíme mít celou scénu včetně zmenšeného modelu města posunutou na tuto pozici. Vybereme všechny prvky ve scéně a budeme je posouvat tak, aby bod X a Y byly rovné 0 a nad ním se nacházel Zámek.

   V prostředí Unreal Editoru a samotných blueprintech je limitace v počtu desetinných míst. Tudíž při zadání správných geografických souřadnic se v případě *Latitude* zkrátí o jedno desetinné místo a předchozí číslo bude zaokrouhleno. To může způsobit menší nepřesnost v modelu samotném.

11 UDÁLOSTI A INFORMACE O BUDOVÁCH

Události a informace o budovách jsou nahrány v Json souborech zvlášť na platformě GitHubu. Takto se může docílit aktualizování obou souborů bez nutnosti kompilovat aplikaci znova. Bohužel neexistuje žádné veřejné API, které by se dalo využít a proto informace o událostech mi byly poskytnuty Městským informačním a turistické střediskem města Zlín souborem ve formátu XML. [31] Tento soubor byl poté převeden pomocí online nástroje na formát Json. Informace o budovách jsem tvořil sám a to ve formátu Json. Unreal Engine ve svém základě pomocí Blueprintů není schopen tvořit HTTP Requesty, pro tyto potřeby byl stažen plugin s názvem *VaRest*. [28] Tento plugin mi umožní pomocí blueprintů získat samotný Json objekt v projektu a pracovat s ním. V následující kapitole se budu zabývat tvorbou widgetu, který se bude zobrazovat nad modelem a získáváním dat ze samotného API.

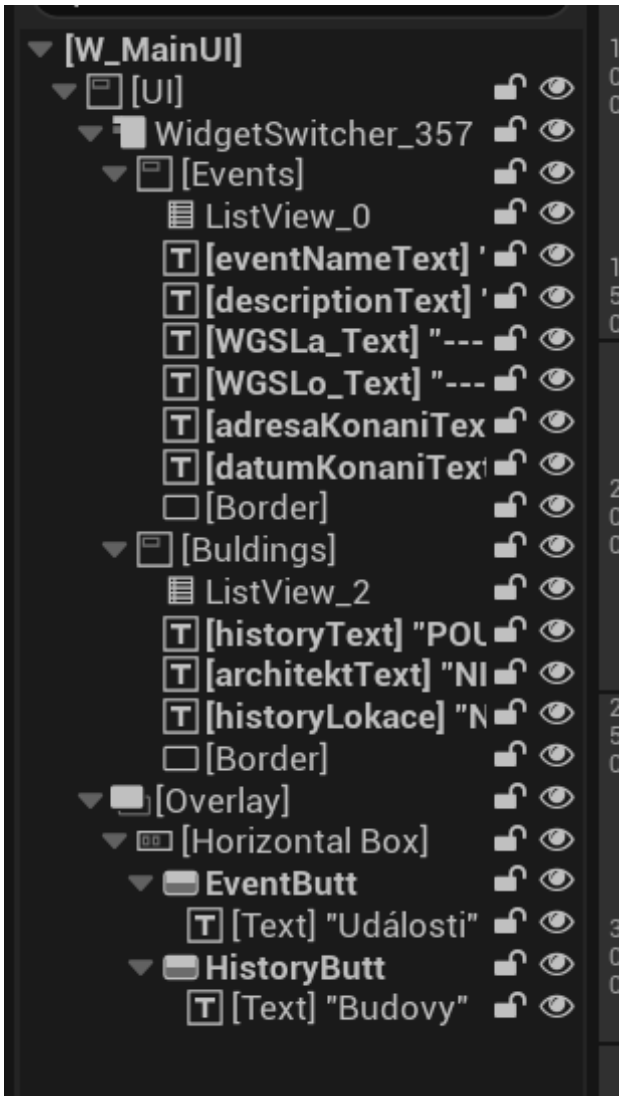
11.1 Tvorba widgetu

Pro zobrazování informací jsem vytvořil nový Widget Blueprit, to jsem provedl tak, že v *Content Browseru* kliknu pravým tlačítkem myši na volné místo a vyberu možnost *User Interface → User Widget*. Samotný widget jsem pojmenoval jako *W_MainUI.* Po přidání jsem následující prvky

1. První jsem přidal *Canvas Panel*, ten jsem postupně při přidávání další prvků roztahoval aby se do něj všechny vlezly
2. Widget bude zobrazovat jak informace o událostech a budovách, proto jsem do něj přidal *WidgetSwitcher*, ten mi umožňuje přidat více widgetu do jednoho *Canvas* panelu a přepínat mezi nimi pomocí jejich indexu
3. Na přepínání mezi widgety byl na spodek přidán *overlay*, ten obsahuje *horizontal box* a dvě tlačítka, které po stisknu změní index mezi 0 a 1. Samotné tlačítko poté dostalo text, aby bylo možné poznat, co které tlačítko dělat
4. Do *WidgetSwitcheru* jsem přidal dva *Canvas* Panely, jeden bude sloužit pro zobrazování Událostí a druhý pro zobrazování informací o budovách.
5. Do obou *Canvasů* jsem poté přidal *ListView* a *textboxy* pro zobrazování samotných informací.

ListView slouží pro zobrazení seznamu všech událostí nebo budov, které mají dostupné data. Pro to abych mohl zobrazovat text v *listView* musím vytvořit nový widget,

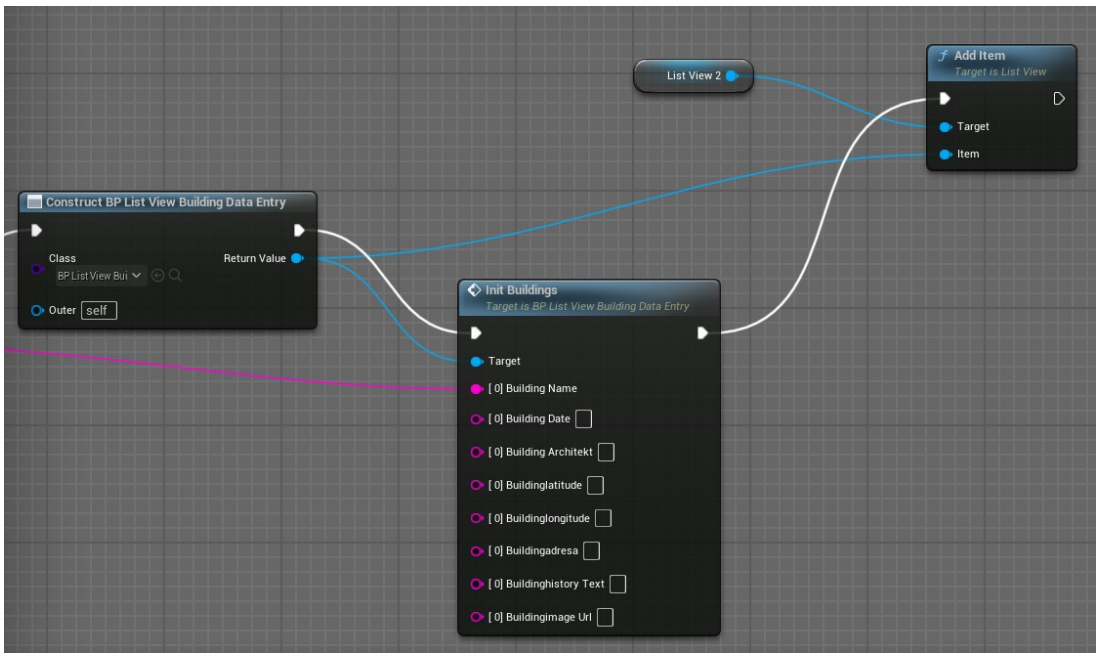
který bude určovat jak bude vypadat jeden prvek tohoto *listView*. Zobrazování událostí a budov dostane tento prvek zvlášť. Jednotlivé widgety jsem pojmenoval *W_ListViewEntry* pro události a *W_ListViewBuildingEntry* pro budovy. Abych je mohl využít v *ListView* tak jsem musel implementovat Interface. Otevřu oba *W_ListViewEntry* a vyberu možnost *Class Settings*, v levé části okna se mi zobrazí v panelu Details položka *Interfaces*, poté jsem kliknu na možnost *Add* a vybral *UserObjectListEntry*. Nyní můžu oba *ListViewEntry* uložit a zkompilovat. Poté jsem se přepnul do *W_MainUI* a kliknul na vytvořený *ListView*, který patří pod *Události* a v *Details* panelu poté naleznu sekci List Entries. Kliknu na dropdown menu a vyberu *W_ListViewEntry*. Poté se přepnu do části widgetu pro budovy a vyberu *W_ListViewBuildingEntry*.



Obrázek 31: Hiearchie prvků ve Widgetu

Poté jsem vytvořil *Struct* abych mohl jednodušeji přistupovat k datům, které spolu souvisí. Jak pro budovy, tak pro události jsem vytvořil nový struct, to jsem provedl tak, že jsem na volné místo klikl pravým tlačítkem myši a vybral možnost *Bleuprints* → *Structure*.

Strukturu pro události jsem pojmenoval *S_ListViewDataFull* a *S_ListViewDataBuildings* pro budovy. Do něj poté vložil jednotlivé proměnné formou řetězců, které chci používat. Pro načítání dat do *ListView* si vytvořím nový *blueprint class* typu *object*. Zase budou 2 pro každou část widgetu, pro události se bude jmenovat *BP_ListViewDataEntry* a *BP_ListViewBuildingData.* Jednotlivé Blueprinty otevřu a v levém panelu *My Blueprint* vytvořím novou proměnnou jenž bude mít typ struktury, kterou jsem si vytvořil. Poté přepnu typ na *Array*. Následně jsem vytvořil *Custom Event*, který bude přidávat data do struktury.



Obrázek 32: Přídávání prvků do ListView

Tento *Event* budu poté pomocí Castingu volat v *W_MainUI* a pomocí naplněného *Struct Arraye* naplním samotný *ListView* daty. Jelikož v seznamu využívám pouze název budov a událostí, tak mi stačí napojit pouze název. Ostatní piny v uzlech můžu nechat nevyužité. Naplnění probíhá uzlem pro Smyčku, která prvek po prvku přidá pomocí uzlu append jednotlivé události do *Struct Arraye*, po jeho dokončení poté proběhne samotný *Casting* a naplnění *ListView* těmito daty.

11.2 Zobrazení widgetu ve scéně

Abych si mohl zobrazit widget ve scéně podobně jako jakýkoliv jiný objekt, musím vytvořit novou *Actor Class*, ten nazvu *SpawnUIElement*. Do *SpawnUIElement* přidám poté Widget komponentu a v panelu *Details* poté vyberu *Widget* Class a v ní vyberu vytvoření *W_MainUI*. Nyní můžu upravit velikost widgetu a umístit ho na střed modelu.

Aby widget rotoval a část s textem byl vždy viditelná, přidám do *SpawnUIElement* nový komponent s názvem *Sphere Collsion*. Poté zvětším její *Scale* aby přesahoval přes hrany modelu. V tomto okruhu bude hledat hráče a točit se vůči němu. Pro samotné natáčení vytvořím *Bleuprint Script*. V *SpawnUIElement* vyberu samotnou Sphere collision a kliknu na ním pravým tlačítkem myši. Vyberu *Add Event → Add onComponentBeginOverlap* a *onComponentEndOverlap*. Samotnou implementaci provedu následovně

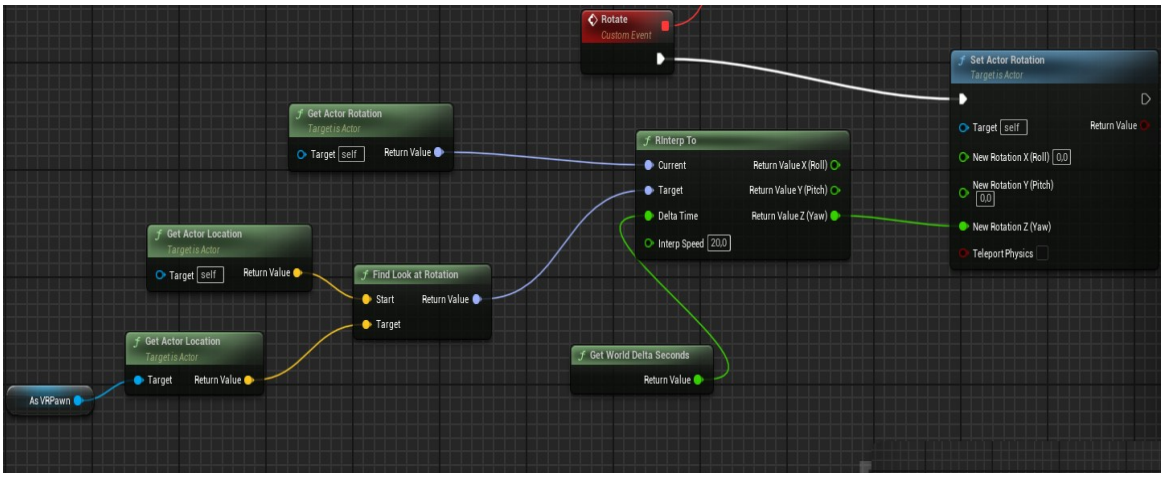
1. Z uzlu *onComponentBeginOverlap* vytvořím uzel *Cast To VRPawn*, jeho pin *Object* napojím na *OtherActor* u *onComponentBeginOverlap*
2. Poté kliknu pravým tlačítkem myši u *Cast To VRPawn* na pin As *VRPawn* a vyberu *Promote to Variable*.
3. Z vytvořeného *SET* uzlu vytvořím *Set Timer by Event* a jeho výstupní Pin *Return Value* tako pomocí *Promote to Variable* přetvořím na proměnnou.
4. Dále z uzlu *Set Timer by Event* vytvořím *Custom Event* pomocí pinu Event, který bude nastavovat rotaci sebe samotného, tedy *SpawnUIElement*.
5. Vstupní pin *Target* bude napojení na uzel *Rinterp*, který bude plynulo zvyšovat hodnoty mezi svou vlastní rotací a místem, kde se nachází můj *VRPawn*

11.2.1 Kolize VRPawn

Aby rotace proti hráči fungovala správně je nutné upravit *VRPawn*. Od verze UE4 chybí s implementaci *VRPawn* funkčnost generování kolizí s ostatními objekty. Proto jsem musel tuto kolizi přidat sám. V *Content Browseru* si otevřu složku *VRTemplate* a v ní složku Blueprints, poté otevřu samotný blueprint pro *VRPawn* a provedu následující

1. V seznamu *Components* na levé straně klikneme na tlačítko *Add* a vyhledáme *box collision*
2. Tento *collision box* poté zmenšíme na *0,155* po všech osách
3. Poté jej přesuneme tak aby byl součástí RightHand, který patří pod *MotionControllerRight*.
4. V nastavení samotného *collision* boxu na pravé straně okna poté nalezneme sekci „*Collision*“ a zaškrneme možnost *Generate Overlap Events* a změníme *CollisionPreset* na *OverlapAllDynamic*
5. Poté klikneme na *Compile* a *uložíme*

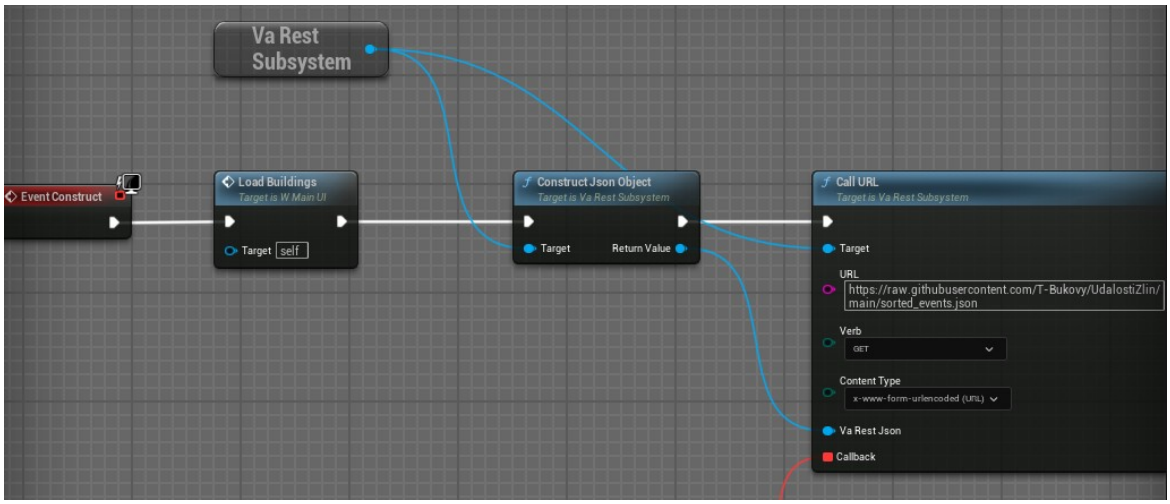
Nyní bude Widget správně reagovat na pozici hráče, a rotovat vůči hráči samotnému.



Obrázek 33: Rotace vůči hráči

11.3 Získávání informací z API

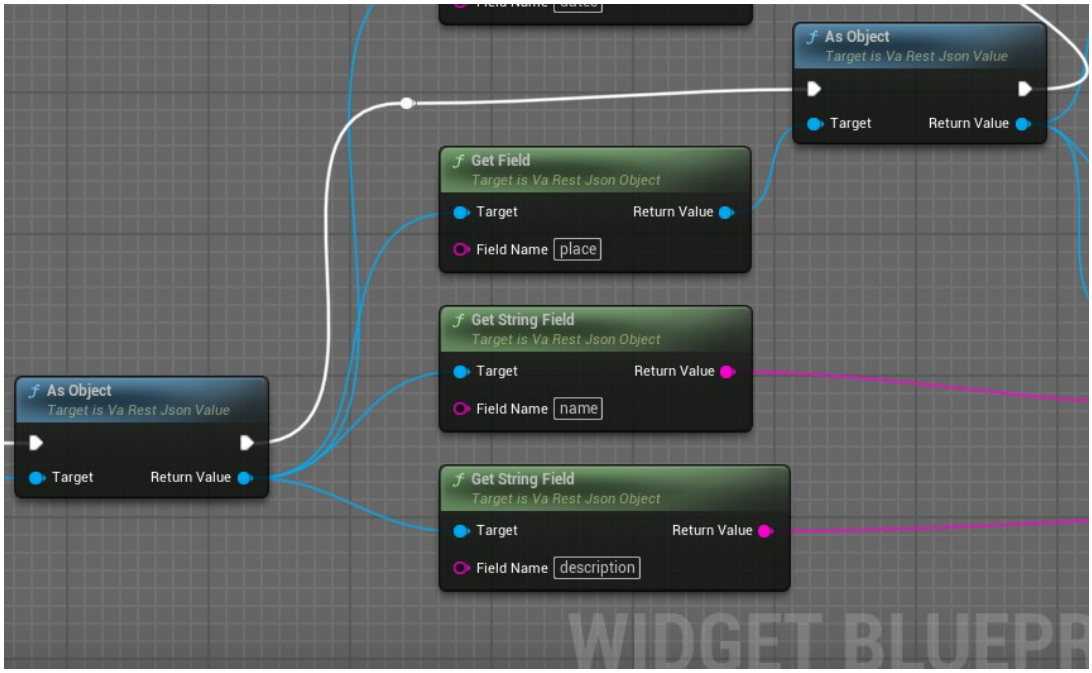
Abych mohl v blueprintech volat funkce VaRest pluginu tak musím do pracovní plochy přetáhnu nový uzel *VaRestSubsystem*. Z něj poté vytvořím uzel s názvem *Construct Json Object*, ten říká, že to co bude API vracet bude formátu json. Poté si vytvořím uzel *CallURL*, jeho vstupní pin napojím na *VaRestSubsystem*. Do políčka URL vložím adresu na svůj github účet. Je nutné aby se jednalo o raw odkaz na json soubor.



Obrázek 34: Volání URL

Poté si z pinu *Event* vytvořím nový *custom Event*, ze kterého přidám 2 uzly za sebe kdy vrácený json převedu na objekt typu *Array*, ten budu poté moc cyklovat pomocí uzlu *For Each loop*. K jednotlivým datům v každém cyklu poté budu přistupovat pomocí uzlů *Get*

Field, který mi umožní přístup k dalším objektů, které json obsahuje. Takto byly například uloženy informace o souřadnicích. Pro přístup k jednotlivým prvků, ze kterých chci získat hodnotu využiji uzel *Get Field String*.

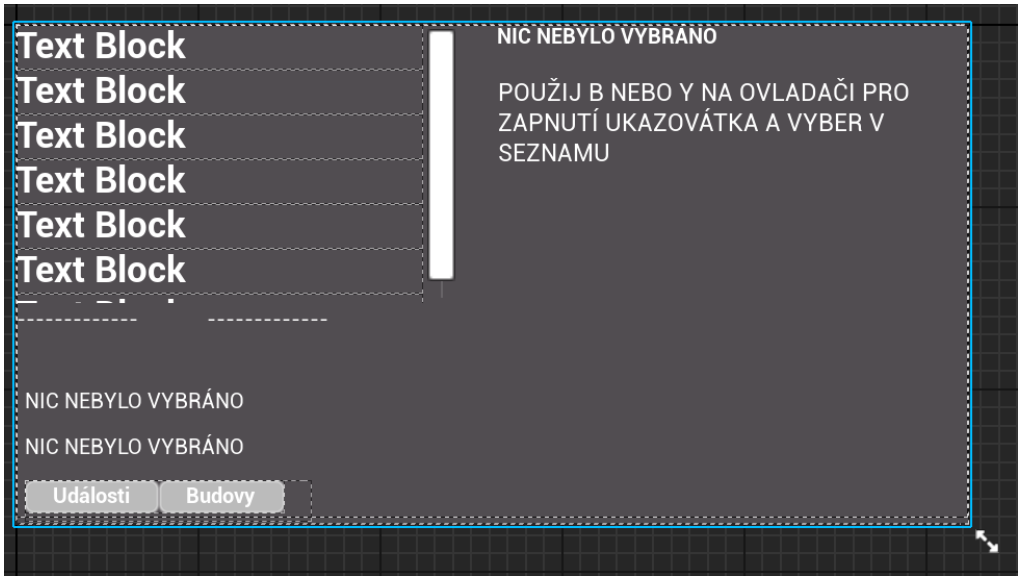


Obrázek 35: Získávání informací pomocí uzlů

Aby se dala data aktualizovat, je nutné upravit samotný .json soubor v repozitáři na GitHubu. Oba soubory mají svoji určenou strukturu a tu je nutné dodržovat, protože bez toho by nefungovalo samotné získávání dat. Jedná se o *json array* a každý objekt v něm tvoří jednu událost nebo významnou budovu. Ukázka objektu události a významných budovách je uvedená v přílohách P2 a P3 diplomové práce. Při změně dat za běhu aplikace je nutné aplikaci restartovat aby proběhlo obnovení dat.

11.4 Informace o událostech

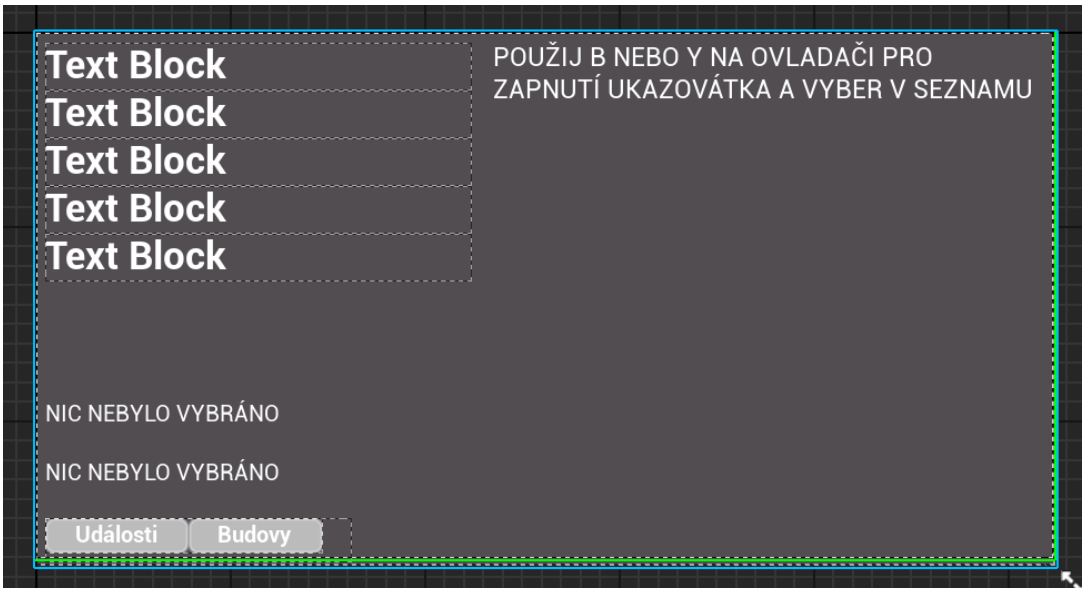
Jelikož se jedná o data využívaná na stránkách Informačního centra tak obsahují více informací než sám potřebuji [31]. Nejdůležitější informace pro potřeby diplomové práce jsem vybral jako název události, její datum, ulici konání, popis události a samotné geografické souřadnice, kde se nachází. Datum je tvořeno dvěma objekty *start_data a end_date*, tyto hodnoty jsem pomocí uzlů *append* spojil a vytvořil tak jeden řetězec pro datum.



Obrázek 36: Widget pro události

11.5 Informace o důležitých budovách

Informace o jednotlivých budovách jsem získával ze stránky ZAM [30] a brožurky, která je volně dostupná na infocentru města Zlín s názvem Zlínská architektura. Z těchto informací jsem poté tvořil jednotlivé objekty v souboru json, který je nahraný na GitHubu, ze kterého je poté v aplikaci získávám. Informace, které jsem získával byly název samotné budovy, její adresa, kdo byl jejím architektem a kdy byla postavena. Poté se zde nacházel samotný informační text o budově a geografické souřadnice, které využívám pro zobrazení ukazatele na mapě.



Obrázek 37: Widget pro budovy

11.6 Převod souřadnic

Jelikož samotný výpočet budu volat jak pro budovy tak pro události, je vhodné vytvořit funkci, kterou budu poté volat. V otevřeném blueprintu W_*MainUI* v levé části okna vybereme možnost *Graph*. Po přepnutí v levé části, v seznamu objektů v Blueprintu kliknu na symbol + u *Functions* a funkci pojmenuji *Prevod souradnic.* Dé pracovní plochy si přidám prvek *Get Georeferenecing system*. Jeho návratová hodnota bude směřovat to uzlů *Geografic to ECEF* a *ECEF to Engine*. Tyto dva uzly také propojím mezi sebou pomocí pinů ECEFCoordinates. V případě *Geografic to ECEF* musím vytvořit vstupní souřadnice. Z pinu *Geografic Coordinates* vytáhnu *wire* a vytvořím uzel *Make Geografic Coordinates*. Jako vstupní piny budu využívat pouze *X* a *Y* a tyto piny pomocí *wire* napojím do Vstupního uzlu. Výstup souřadnice, které se dají použít v prostředí herního Enginu získám z pinu *Engine Coordinates* z Uzlu *ECEF to Engine*. Jelikož nepoužívám geografickou výšku, tak budu tento výstup rozdělit na jednotlivé prvky. K tomu využiji uzel *Break Vector*. Poté jednotlivé Piny vynásobím konstantou 0,0019 a jejich výstup napojím pomocí *wire* do Výstupního uzlu. Konstanta 0,0019 byla vypočítána rozdílem souřadnic v modelu (tam kde se nachází objekt) a souřadnice, kterou vrátila funkce, kdy byla konstanta nastavena na hodnotu 1. Tedy hodnoty, kde by se po převodu budova nacházela, být model 1:1 s reálným světem.

11.7 Zobrazení lokace v modelu

Pro tyto potřeby byl vytvořen v aplikaci Blender jednoduchý ukazatel. Ukazatel byl poté stejným způsobem do modelu importován jako například model kostela. Poté jsem vytvořil novou třídu *Actor*, který budu zobrazovat samotný ukazatel na místě události a budovy. Do této třídy jsem mimo samotného vymodelovaného ukazatele také přidal *Widget Component*. Widget Component poté přebíral vytvořený Widget Class nově vytvořeného Widgetu pro zobrazování Textu události. Stejně jako widget se seznamem událostí se bude natáčet vůči hráči. Toho bylo docíleno také za použití *Sphere collision*, která detekovala *overlap event*, který byl vyvolávaný hráčem.

Funkce přebírá převedené souřadnice a využívá je pro zobrazení ukazatele ve scéně. Po zobrazení se nastaví *Bool* na *True*. Pokud poté vyberu novou událost nebo budovu bude původní smazán a zobrazí se ukazatel na místě nově vybrané události nebo budovy.

11.7.1 Statické body lokací

Pro lepší orientaci v modelu jsem také přidal malé ukazatele na permanentní budovy jako je například Zámek nebo budova radnice. Tyto ukazatele jsem vytvářel za použití stejného statického meshe, jako ukazatele události ale byl zmenšen a dostal jiný materiál aby byl lépe rozeznatelný od ukazatele pro události a budovy. Stejně jako tyto ukazatel také dostal text, který byl navíc doplněn o pozadí aby šel lépe rozeznat. Samotný text poté stejně jako hlavní widget rotuje proti hráči, aby bez ohledu kde u modelu stojí mohl vždy přečíst název.

12 VÝSLEDKY PRÁCE

Ve finální verzi aplikace si může uživatel ve vymodelovaném památníků Tomáše Bati prohlédnout zmenšený model města Zlín a pomocí interaktivního widgetu nad městem si zobrazit informace o historických stavbách města Zlín včetně jejich polohy, která je zobrazena formou ukazatele na samotném modelu. Widget může přepínat mezi dvěma stavy a může si tak také zobrazit informace o událostech, které se ve městě konají včetně jejich lokace.

Aplikace je optimalizovaná pro využití ve virtuální realitě a reaguje na pohyb hlavou a ovladačů. Ovladače je možné využít pro teleportaci ve spodním patře památníku a poté pro interakci s widgetem. Aplikace při pohledu na model ztrácí snímkovou frekvenci z důvodů náročnosti samotného modelu a samotným stářím headsetu Quest 1.

Obrázek 38: Výsledná aplikace

Samotný model byl aktualizován o nové budovy, které v předchozí práci chyběli a byl také graficky upraven terén původní aplikace. Aplikace neobsahuje hlavní menu a uživatel se zobrazí v předem definovaném prostoru před schodištěm v modelu památníku Tomáše Bati. Před sebou nalezne samotný zmenšený model města Zlín. Nad ním se nachází zmíněný Widget.

Aplikace se ovládá pomocí ovladačů, které jsou součástí zařízení Oculus Quest. Pomocí levé analogové páčky je možné se ve scéně rozhlížet, bez nutnosti se otáčet celým tělem. Aplikace reaguje na pohyb hlavou a je možné se rozhlížet i takto. Pravou analogovou páčkou se vyvolá teleportace. Při pohybu například dopředu se zobrazí ukazatel lokace, kam bude uživatel teleportován. Po uvolnění se na tuto lokaci teleportuje. Uživatelské menu se v aplikaci nenachází. Vypínání aplikace se provádí pomocí systémového tlačítka na pravém ovladači s logem *Oculus*.

Pro interakci s widgetem je nutné zobrazit si ukazovátka. Tyto ukazovátka umožňují interagovat s widgetem jako kdyby se používala myš. Tlačítkem B na pravém ovladači se vyvolá ukazatel a widget se ovládá pravou rukou. Při zmáčknutí pravého zadního triggeru je možné vybrat prvek v seznamu událostí a historických budovách a nebo ovládat tlačítka. Scrollování v seznamu je poté umožněno držením zadního triggeru a pohybem ovladače dolů či nahoru. Widget jde ovládat i levou rukou za použití klávesy Y pro vyvolání ukazovátka a levým triggerem.

Celková velikost aplikace ve formátu .apk je 235 MB. Aplikace se po nainstalování na zařízení Meta Quest objeví v sekci *Unknow Sources* v prohlížeči aplikací pod názvem *InteraktivníZlín*. Aplikace se načte po pár sekundách po načítací obrazovce tvořenou třemi tečkami.

ZÁVĚR

Tato práce měla za cíl vytvořit interaktivní aplikaci virtuální reality se zmenšenou vizualizací Zlína kterou si je možné prohlédnout v prostředí památníku Tomáše Bati ve Zlíně. Uživatel má možnost si zobrazit historické texty k významným budovám s jejich lokací ve zmenšeném modelu města Zlín. Dále si může prohlížet události, které se ve městě konají a zobrazit si lokaci, kde se nachází.

V teoretické části práce byly popsány jednotlivé programy které zde byly využívány. Pro modelování zde byl využit program Blender a pro tvorbu scény a výsledné aplikace byl poté použit Unreal Engine. Poté byla rozepsána historie virtuální reality a implementace v programech které ji využívají.

Praktická část se zabývá stavem předchozí práce a modelem, který v ní byl vytvořen. V následujících kapitolách byla popsána tvorba aplikace virtuální reality a její funkcionalita s prostředím. Poté byla popsána aktualizace a grafické úpravy původního modelu a následná optimalizace modelu aby mohl být využít pro potřeby virtuální reality. Následovala kapitola popisující práci s daty o událostech a významných budovách a tvorba jejich vizuálního zobrazení v aplikaci. V poslední kapitole byla poté popsána samotná výsledná aplikace a její funkčnost a omezení.

Aplikace by se mohla rozšířit s využitím více podkladových modelů, uživatel by mohl například přepínat mezi současným zobrazením města Zlín a například historickým. Dále by bylo možné samotný model rozšířit aby pokrýval větší část města Zlín a využít pro to i výkonnější hardware. Dalším možností by bylo předělat model to rozšířené reality a místo zobrazení modelu v prostředí památníku Tomáše Bati by se samotný Zlín zobrazil přímo v reálném světě.

SEZNAM POUŽITÉ LITERATURY

[1] *Unreal Engine documentation*. Online. Unrealengine.com. Dostupné z: https://docs.unrealengine.com/5.3/en-US/. [cit. 2023-11-17].

[2] *Nodes | Unreal Engine 4.27 Documentation*. Online. Docs.unrealengine.com. 2022. Dostupné z: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Nodes/#datapins. [cit. 2024-01-31].

[3] *Unreal Engine Interface*. Online. 2023. Dostupné z: https://docs.unrealengine.com/5.3/en-US/unreal-editor-interface/. [cit. 2024-02-20].

[4] *Category:Blender splash screens*. Online. 2023, 2018. Dostupné z: https://commons.wikimedia.org/wiki/Category:Blender_splash_screens#. [cit. 2024-02-21].

[5] *Workspaces - Blender Manual*. Online. 2023. Dostupné z: https://docs.blender.org/manual/en/latest/interface/window_system/workspaces.html [cit. 2024-02-22].

[6] *Cycles and Eevee: which renderer should we choose?* Online. 2023. Dostupné z: https://irendering.net/cycles-and-eevee-which-renderer-should-we-choose/. [cit. 2024-02-22].

[7] *Blender Eevee vs Cycles [2024]*. Online. 2024. Dostupné z: https://renderguide.com/blender-eevee-vs-cycles-tutorial/. [cit. 2024-02-23].

[8] *Object Modes - Blender Manual*. Online. Blender. 2023. Dostupné z: https://docs.blender.org/manual/en/latest/editors/3dview/modes.html. [cit. 2024-02-25].

[9] *Blender.org*. Online. 2023. Dostupné z: https://www.blender.org. [cit. 2024-02-26].

[10] *25 years Late: The History of Unreal and an Epic Dynasty*. Online. PCMag. 2023. Dostupné z: https://www.pcmag.com/news/25-years-later-the-history-of-unreal-and-an-epic-dynasty. [cit. 2024-02-26].

[11] *Unreal Engine and its Evolution*. Online. Externlabs. 2022. Dostupné z: https://externlabs.com/blogs/unreal-engine-and-its-evolution/#The_Beginning_Reply_1991. [cit. 2024-02-26].

[12] *Unity vs Unreal Engine for XR Development: Which One Is Better? [2022 Updated]*. Online. Circuit Stream. 2022. Dostupné z: https://circuitstream.com/blog/unity-vs-unreal. [cit. 2024-03-12].

[13] *AS-1*. Online. Sega Retro. 2023. Dostupné z: https://segaretro.org/AS-1. [cit. 2024-03-15].

[14] *History Of Virtual Reality*. Online. Virtual Reality Society. 2020. Dostupné z: https://www.vrs.org.uk/virtual-reality/history.html. [cit. 2024-03-15].

[15] *History of VR – Timeline of Events and Tech Development*. Online. Virtual Speech. 2023. Dostupné z: https://virtualspeech.com/blog/history-of-vr. [cit. 2024-03-17].

[16] *What is Virtual Reality?* Online. Smlease Design. Dostupné z: https://www.smlease.-com/entries/technology/what-is-virtual-reality/. [cit. 2024-03-17].

[17] *What is virtual reality*. Online. TechGarden. 2022. Dostupné z: https://www.-techtarget.com/whatis/definition/virtual-reality. [cit. 2024-03-25].

[18] *Virtualní realita - historie a součastnost*. Online. VR Education. 2020. Dostupné z: https://vreducation.cz/virtualni-realita-historie-a-soucasnost/. [cit. 2024-03-25].

[19] *Blenders History*. Online. Huihoo. 2020. Dostupné z: https://docs.huihoo.com/blen-der/user-guide-2.3/ch01s02.html. [cit. 2024-03-25].

[20] *Meta Quest 3 Is an Excellent Portal to an Empty Metaverse*. Online. Blender. 2023. Dostupné z: https://www.wired.com/review/review-meta-quest-3/. [cit. 2024-03-25].

[21] *Meta Quest Pro review: get me out of here*. Online. The Verge. 2022. Dostupné z: https://www.wired.com/review/review-meta-quest-3/. [cit. 2024-03-25].

[22] *Technické údaje*. Online. Playstation. 2023. Dostupné z: https:// www.playstation.com/cs-cz/ps-vr/tech-specs/. [cit. 2024-04-01].

[23] *Bigscreen Beyond – The World's smallest VR Headset*. Online. Bigscreen. 2023. Dostupné z: https://www.bigscreenvr.com. [cit. 2024-03-25].

[24] *The new Snapdragon XR2 Plus Gen 1 offers 30% improved performance over its pre-decessor*. Online. XDA-Developers. 2022. Dostupné z: https://www.xda-developers.com/ snapdragon-xr2-plus-gen-1-launch/. [cit. 2024-04-03].

[25] *Review: Sony PSVR 2*. Online. Wired. 2023. Dostupné z: https://www.wired.com/ review/playstation-vr-2/. [cit. 2024-03-25].

[26] *Blueprints Visual Scripting for Unreal Engine 5*. 3rd. Packt, 2022. ISBN 9781801811583.

[27] PANGILINAN, Erin. *Creating Augmented and Virtual Realities: Theory and Practice for Next-Generation Spatial Computing.* United States: O'Reilly Media, 2019, 300 s. ISBN 978-149-2044-192.

[28] *Cycles vs Eevee rendering – speed comparison*. Online. Render-Street. 2019. Dostupné z: https://blog.render.st/cycles-vs-eevee-rendering-speed-comparison. [cit. 2024-04-23].

[29] *VaRest*. Online. Https://www.unrealengine.com/marketplace. 2024. Dostupné z: https://www.unrealengine.com/marketplace/en-US/product/varest-plugin. [cit. 2024-04-20].

[30] *Zlínský architektonický manál*. Online. 2023. Dostupné z: https://zam.zlin.eu/. [cit. 2024-04-21].

[31] *Kalendář akcí - Turistický informační portál města Zlín*. Online. 2022. Dostupné z: http://www.ic-zlin.cz/25319-kalendar-akci. [cit. 2024-04-22].

[32] DUREC Marián, Markéta MAZÁČOVÁ a Kamil STOKLÁSKA. *3D vizualizace centra současného Zlína*. Zlín: Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky, 2014

[33] BUKOVÝ, Tomáš. *3D Vizualizace současného Zlína s pomocí modulu BlenderGIS*. Bakalářská práce, vedoucí Ing. Pavel Pokorný, PhD. Zlín: Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky, 2022.

[34] *Blender Documentantion*. Online. Blender.ogr. 2024. Dostupné z: https://docs.blender.org/. [cit. 2024-04-24].

[35] *History*. Online. Blender.org. 2022. Dostupné z: https://www.blender.org/about/history/. [cit. 2024-04-24].

[36] *Georeferencing a Level | Epic Developer Community*. Online. Epic Developer Community. 2024. Dostupné z: https://dev.epicgames.com/documentation/en-us/unreal-engine/georeferencing-a-level-in-unreal-engine?application_version=5.3. [cit.2024-04-25].

[37] *Bigscreen Beyond – Promising but Incomplete, Just Like This Review*. Online. RoadToVR. 2023. Dostupné z: https://www.roadtovr.com/bigscreen-beyond-review-pc-vr-headset/. [cit. 2024-05-02].

[38] *Features - Unreal Engine*. Online. The most powerful real-time 3D creation tool - Unreal Engine. 2023. Dostupné z: https://www.unrealengine.com/en-US/features. [cit. 2024-05-01].

[39] MCCAFFREY, Mitch. *Unreal Engine VR cookbook: developing virtual reality with UE4*. Boston: Addison-Wesley, 2017, 288 s. ISBN 01-346-4917-6.

[40] GONALO, Marques. *Elevating Game Experiences with Unreal Engine 5 - Second Edition: Bring your game ideas to life using the new Unreal Engine 5 and C++*. United Kingdom: Packt Publishing, 2022, 760 s. ISBN 978-180-3239-866

[41] *Compare Quest VR Headsets: Quest 2 vs. Quest Pro vs. Quest 3 | Meta Store*. Online. Meta Store. 2023. Dostupné z: https://www.meta.com/quest/compare/. [cit. 2024-05-01].

[42] *Oculus Quest: Full Specification - VRcompare*. Online. VRcompare. 2020. Dostupné z: https://vr-compare.com/headset/oculusquest. [cit. 2024-05-01].

[43] *About - Blender.org*. Online. Blender.org - Home of the Blender project. Https:// www.blender.org. Dostupné z: https://www.blender.org/about/. [cit. 2024-05-01].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UE	Unreal Engine
SIGGRAPH	Special Interest Group on Computer Graphics and Interactive Techniques
CPU	Centrální procesorová jednotka
GPU	Grafický procesor
NaN	Not a Number
HMD	Head Mounted Display
SoC	System-on-chip
FPS	Snímky za sekundu
VR	Virtuální realita
NDK	Native Development Kit
SDK	Software Development Kit
AR	Augmentovaná Realita
GNU	GNU´s Not Unix!
LCD	Liquid Crystal Display
OLED	Organic Light-Emitting Diode
APK	Android Application Package

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P 1: OBSAH PŘILOŽENÉHO DVD .

Přiložené DVD obsahuje následující soubory a adresáře:

- Soubor **fulltext.pdf** obsahující digitální verzi práce ve formátu PDF/A
- Adresář **Aplikace_Quest** obsahuje soubor .apk pro spuštění na headsetu Meta Quest
- Adresář **ProjektoveSoubory.zip** obsahuje veškeré projektové soubory z prostředí Unreal Engine včetně modelů, soubory vytvořené z programu Blender a datové soubory událostí a budov.
- Soubor **Ukazka.mp4** obsahuje natočené video aplikace ze zařízení Oculus Quest

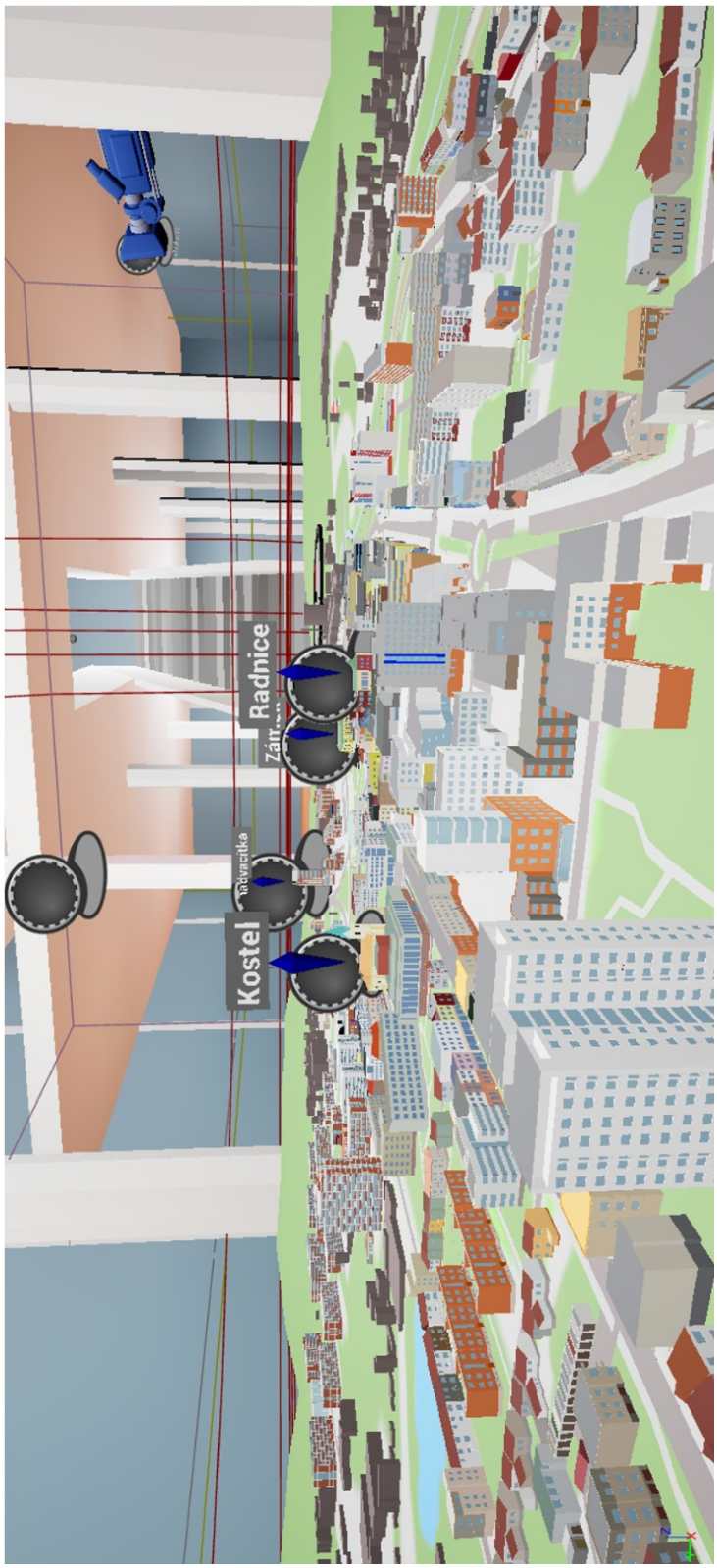
PŘÍLOHA P 2: STRUKTURA UDÁLOSTÍ.


```
},
{
    "id": "1952",
    "name": "Den české státnosti",
    "categories": [
        {
            "id": "3",
            "name": "-Hlavní rubrika"
        },
        {
            "id": "46",
            "name": "Ostatní"
        }
    ],
    "dates": {
        "date": {
            "start_date": "2024-09-27",
            "end_date": "2024-09-27",
            "start_time": []
        }
    },
    "description": "Statutární město Zlín uctí tichou vzpomínkou a květinovým darem Den české státnosti u sochy TGM před Kongresovým centrem, nám. TGM 5556, Zlín.",
    "place": {
        "name": "Kongresové centrum Zlín",
        "street": "nám. T. G. Masaryka 5556",
        "city": "Zlín",
        "zip": "760 01",
        "wgs84": {
            "latitude": "49.2231508",
            "longitude": "17.6643589"
        }
    },
    "organizer": {
        "name": "statutární město Zlín",
        "street": "náměstí Míru 12",
        "city": "Zlín",
        "zip": "761 40"
    },
    "details": {
        "url": "://www.ic-zlin.cz/1952a-den-ceske-statnosti"
    },
    "wh_image": {
        "@src": "http://www.ic-zlin.cz/wcd/events/t.g.m..jpg",
        "@alt": ""
    }
},
```

PŘÍLOHA P 3: STRUKTURA DŮLEŽITÝCH BUDOV.


```
{
    "name": "Zámek",
    "date": "1904-1905",
    "architect": "Leopold Bauer",
    "wgs84": {
        "latitude": "49.2257686",
        "longitude": "17.6631897"
    },
    "adresa":"Soudní 1",
    "historyText": "Důležitý mílník v budování moderního Zlína představuje proměna zámku a přilehlých pozemků ze soukromého majetku na veřejný. Od roku 1923 začíná František Lýdie Gahura projektovat regulační plány rychle se rozrůstajícího města. Zámecký areál nacházející se dříve na okraji historického centra se ocitl na spojnici mezi historickým a moderním centrem města. Majitelem rozlehlých pozemků panského statku byl Štefan Viktor Leopold Haubt von Buchenrode, potomek brněnských textilních podnikatelů. Ten v roce 1929 prodal zámek i přilehlý areál městu Zlín, v jehož čele stál starosta Tomáš Baťa. ",
    "imageUrl" : "https://zam.zlin.eu/data/photo/thumb/556_17cd43e00b.jpg"
},
{
    "name" : "Měšťanská škola - Zlínský klub 204",
    "date": "1897",
    "architect": "Dominik Fey",
    "wgs84": {
        "latitude": "49.2251697",
        "longitude": "17.6656944"
    },
    "adresa": "třída Tomáše Bati 204",
    "historyText":"O možnosti zřídit ve zlíně měšťanskou školu se mezi místními zastupiteli začíná diskutovat v návaznosti na změnu rakousko-uherského školského zákona už v roce 1872. Vzdělání, které doposud zajišťovaly obecné školy, měly o navazující tři ročníky rozšiřovat školy měšťanské. Než byla postavena nová neoklasicistní budova, která na dobových snímcích z počátku 20. století dominuje malému moravskému městu, uplynulo dlouhých 25 let. Zřízení měšťanské školy zamítla zemská školní rada poprvé roku 1875, nevyšel ani druhý a třetí pokus z let 1883 a 1891. ",
    "imageUrl" : "https://zam.zlin.eu/data/photo/thumb/564_17cd43e00b.jpg"
},
```

PŘÍLOHA P 4: STATICKÉ UKAZATELE LOKACÍ



PŘÍLOHA P 5: UKAZATEL UDÁLOSTÍ