VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

3D DEMO PRO HOLOGRAFICKÝ DISPLEJ

3D DEMO FOR HOLOGRAPHIC DISPLAY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ONDŘEJ MOTYČKA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ CHLUBNA**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Motyčka Ondřej**
Program: Informační technologie
Název: **3D demo pro holografický displej**
3D Demo for Holographic Display
Kategorie: Počítačová grafika

Zadání:

1. Seznamte se s vhodným API pro vývoj 3D aplikace (OpenGL, Unity, Unreal)
2. Navrhněte vhodnou scénu v omezeném prostoru využívající efektivně zaostřovací roviny holografického displeje
3. Implementujte aplikaci
4. Zhodnoťte zobrazení aplikace na displeji a proměřte efektivitu
5. Publikujte demo v oficiální knihovně
6. Vytvořte video reprezentující výsledky vaší práce

Literatura:

- Looking Glass Factory website
- Oficiální dokumentace k vybranému 3D enginu či API

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, experimenty vedoucí k bodu 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Chlubna Tomáš, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cílem této bakalářské práce je vytvořit herní demo aplikaci pro holografický displej s použitím Unreal Engine 4. Díky speciálnímu displeji se může hráč mnohem hlouběji a intenzivněji propojit se hrou díky jednotlivým interaktivním prvkům. Tyto herní prvky jsou vytvořeny na míru a jejich plného potenciálu lze využít pouze při práci s holografickým displejem.

Abstract

The goal of this bachelor thesis is to create a game demo application for holographic display with usage of Unreal Engine 4. Thanks for this unusual display, the player can immerse deeper in the game using its interactive components. All of the components are made directly for the holographic display and without it, they won't reach their full potential.

Klíčová slova

Unreal Engine 4, holografický displej, Looking glass, herní demo, hra, C++, modulární modelování, procedurální generování

Keywords

Unreal Engine 4, holographic display, Looking glass, game demo, game, C++, modular modeling, procedural generation

Citace

MOTYČKA, Ondřej. *3D demo pro holografický displej*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Chlubna

3D demo pro holografický displej

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Chlubny. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Ondřej Motyčka
4. května 2021

Poděkování

Chtěl bych poděkovat mému vedoucímu Ing. Tomáši Chlubnovi za jeho odborné rady a čas, který mi věnoval při této práci.

Obsah

Kapitola 1

Úvod

Hraní počítačových her se v dnešní době stalo součástí každodenního života moderního člověka[1], a právě proto se rodí každý rok nová a nová herní studia[2]. Díky extrémním obratům v herním průmyslu[3] se rozrůstá hned několik odvětví s ním úzce souvisejících. Ať už se jedná o rychlejší vývoj hardware u velkých společností, jako jsou například Nvidia, Intel nebo AMD, nebo o vývoj nového software pro vytváření počítačových her (např. Adobe, Blender, atd.).

Díky enormnímu zájmu se na trh dostávají nové technologie. Jednou z nich je právě Looking Glass, který nahrazuje klasický 2D displej a přináší nové možnosti. Displej umožňuje uživateli sledovat scénu z několika úhlů, a tím pádem navozuje pocit 3D efektu. To všechno je možné bez jakýchkoliv pomocných prostředků, jako jsou například brýle nebo různé doplňky. Tento fakt otevírá vývojářům nové možnosti, jak vyvíjet hry nebo aplikace které umožní uživatelům prožít vzrušující chvíle v již velmi známém prostředí. Uživatel má možnost využít další prvky, jako například Leap Motion senzor, který dokáže snímat pohyby ruky, a tak umožňuje provádět interakci se scénou.

Cílem této práce je navrhnout a implementovat herní demo, které plně využívá těchto interaktivních prvků. Hlavní předností by mělo být využití 3D displeje pro větší pohlcení hrou. Část práce tvoří procedurální generovaní, které je dnes součástí nejen herního průmyslu. Jeho využití sahá od generovaní generovaní malých rostlin až po generování celých měst. Mnohá herní studia díky tomu začala využívat programy jako je např. Substance Designer, který umožňuje rychle a nedestruktivně tvořit textury. Generování se v této práci využívá pro vytváření jednotlivých úrovní hry.

Kapitola 2 vysvětluje principy potřebné pro pochopení problematiky holografického displeje. Čtenář bude mimo jiné seznámen se základy procedurálního generování a herním enginem Unreal Engine 4. V kapitole 3 je čtenář seznámen s návrhem řešení. Zde se dozví, jaké bude hra obsahovat mechaniky a jak bude probíhat generování úrovní. V kapitole 4 jsou uvedeny zajímavé pasáže z implementace hry a v kapitole 5 je popsáno uživatelské testování. V poslední kapitole 6 jsou shrnuty dosažné výsledky této práce.

[1] https://www.limelight.com/resources/white-paper/state-of-online-gaming-2019

[2] https://techjury.net/blog/video-games-industry-statistics/#gref

[3] https://venturebeat.com/2018/04/30/newzoo-global-games-expected-to-hit-180-1-billion-in-revenues-2021/

Kapitola 2

Teorie

Následující kapitola se zabývá základními principy fungování Looking Glass displeje. Ty jsou popsány v sekci 2.1, kde je dopodrobna vysvětlen princip fungování displeje a zobrazování dat. V této sekci jsou také popsány formáty, které tento displej podporuje a nástroje, které s ním pracují. Následující sekce 2.2 se zabývá herním enginem Unreal Engine 4, který je využit jako hlavní vývojové prostředí této aplikace. Zde jsou popsány klíčové části, které jsou v aplikaci využity. V sekci 2.3 jsou popsány principy procedurálního generování a jeho využití. V poslední sekci 2.4 je částečně popsán pojem fotogrammetrie a je přiblížena role, kterou hraje v této práci.

2.1 Holografický displej Looking Glass

Tento výrobek poprvé spatřil světlo světa v roce 2014, kdy společnost Looking Glass Factory přišla na trh s novým typem displeje. Hlavní změnou je pro uživatele odpoutání od 2D displejů jako jsou např. široce rozšířené plazma a OLED displeje. Současně zážitek není omezující, jako virtuální helmy s velmi podobnou funkcionalitou. Hlavním cílem je zobrazit člověku data ve 3D tak, jak je na to zvyklý z reálného světa.

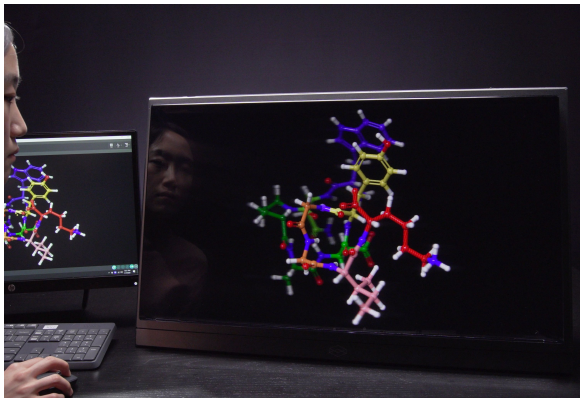
Displej se prozatím využívá převážně v lékařství, kde je na něm možné zobrazovat chemické vzorce a rentgeny. Chemický vzorec lze vidět na obrázku 2.1. Nicméně se rozšířil také do jiných odvětví, kde se používá například k zobrazení proudění vzduchu nebo tekutin. V neposlední řadě se využívá i jako marketingová pomůcka, s jejíž pomocí mohou prodejci lépe představovat svůj výrobek.

Společnost Looking Glass Factory momentálně nabízí tři varianty displeje. Nejmenší varianta má 8.9” displej. Druhá varianta má 15.6” displej a největší varianta disponuje 8K displejem s 33.2 miliony pixelů. Jedná se o největší holografický displej na světě [6].

Základní principy displeje

Princip holografického displeje je v mnoha ohledech úplně jiný než princip klasického 2D displeje. Největší rozdíl je v tom, že holografický displej pracuje se 45 pohledy na scénu, které následně zobrazuje pod zorným úhlem 50°, viz [6]. Všechny snímky jsou pořizovány pouze v horizontální rovině. Díky této vlastnosti může uživatel vidět třetí rozměr, kterým je hloubka. Výsledný obraz působí 3D efektem, kterého se docílí dvěma způsoby:

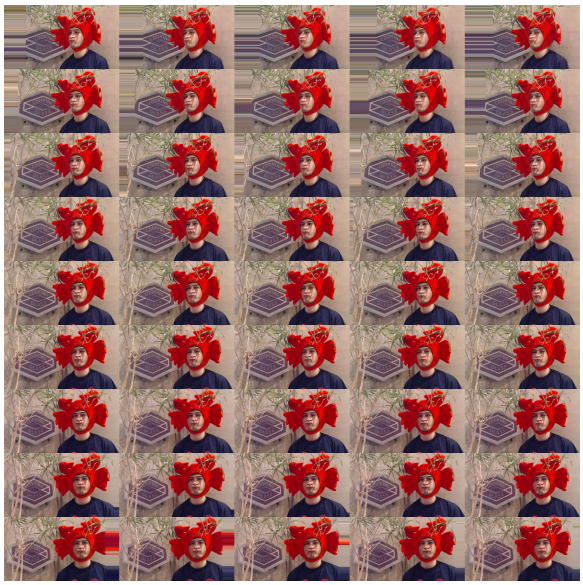
- Pozorovatel se pohybuje kolem displeje a sleduje scénu vždy z jiného úhlu.
- Každému oku pozorovatele je představena jiná perspektiva.



Obrázek 2.1: Vizualizace chemického vzorce na holografickém displeji, převzato[1]

Quilt

Quilt je zobrazovací standard, který je využíván tímto typem displejů. Je to formát, díky kterému se dosáhne požadovaného 3D efektu. Tento standard se používá jak pro popisování obrázků, tak pro videa [6]. Příklad quiltu lze vidět na obrázku 2.2.



Obrázek 2.2: Příklad quilt zobrazení statického obrázku, převzato[2]

Quilt formát slouží k několika účelům [6]:

- K ukládání a načítání jednotlivých snímků zobrazených na displeji.
- Jako jeden z kroků v renderovací pipeline (vykreslovací řetězec).
- Jako formát k práci s manuálním vytvářením světelných ploch za pomoci obrázků nebo videí.

[1] https://lookingglassfactory.com/tech
[2] https://docs.lookingglassfactory.com/Appendix/quilts/

Vývojáři mají možnost si quilt vytvořit sami, a tak mít nad celým procesem větší kontrolu. Nicméně obecně vzato si tuto funkcionalitu obstarává sám Looking glass software (plugin pro Unity, Unreal Engine 4, atd.), který je zodpovědný za vytváření výsledného quilt obrázku [6].

Formáty

Jak již bylo zmíněno, quilt je obrázek složený z jednotlivých 2D obrázků, které jsou ve formě matice. Obecně se užívají hlavně dva formáty [6]:

- 45 pohledů, 9 řad a 5 sloupců
- 32 pohledů, 8 řad a 4 sloupce

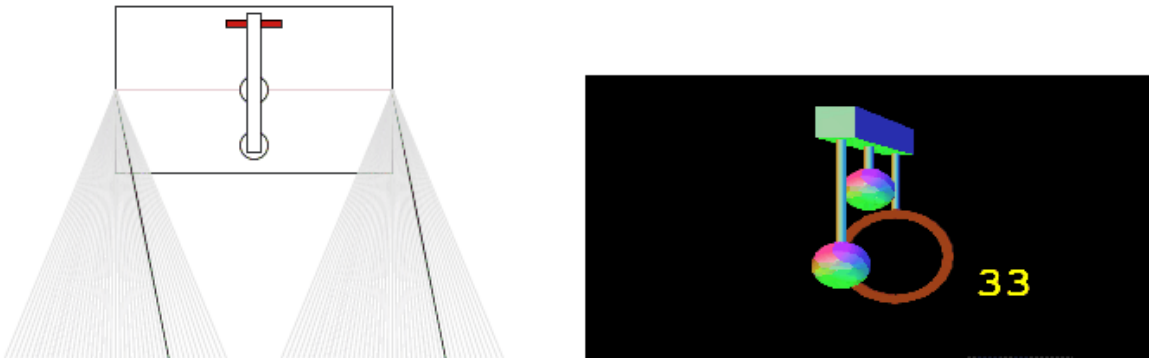
Existují samozřejmě i jiné formáty, ty ale nemusí být podporovány, a tak se mohou na displeji zobrazovat chybně. Tento standard lze uplatnit na jakýkoliv ze široce užívaných typů obrázků nebo videí. Jedná se například o jpg, png, gif, mp4, mov a jiné.

To, v jakém jsou data formátu určují metadata, která se využívají při komunikaci s Looking glass aplikacemi. Metadata jsou zapsána v JSON souboru a obsahují potřebné informace o formátu quiltu.

Na obrázku 2.2 lze vidět, že snímky začínají v levém dolním rohu a končí v pravém horním rohu. Levý dolní roh představuje pohled na objekt z levé strany a pravý horní roh naopak pohled ze strany pravé. Lze si to představit jako pole, kde obrázek na každém indexu odpovídá jinému úhlu zobrazení.

Rovina nulté paralaxy

Tato rovina je místo v displeji, které se z pozorovatelova pohledu nemění. Není důležité, z jakého úhlu se na toto místo pozorovatel dívá, vždy vidí to stejné. Tento jev vidíme na obrázku 2.3.



Obrázek 2.3: Na obrázku vlevo je pohled na model shora, který je zobrazen na pravém obrázku. Také zde lze vidět červenou čáru, která popisuje nultou rovinu, na které jsou všechny objekty zaostřené. Na pravé straně obrázku lze vidět tentýž model, již zobrazený ve 3D, převzato[3]

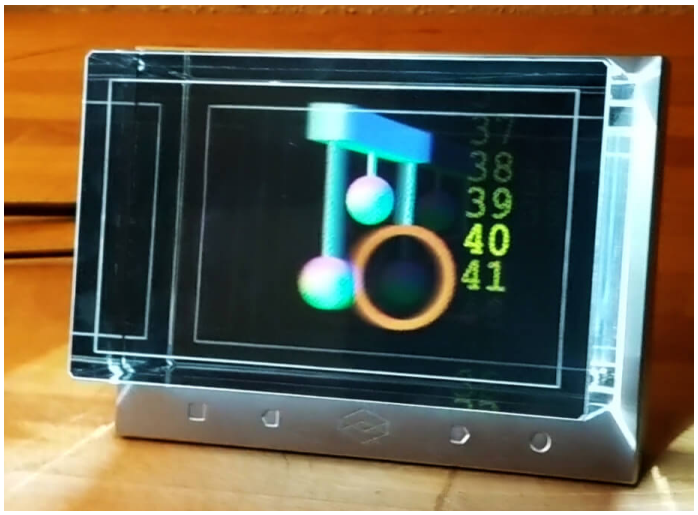
Koule v pozadí, na obrázku 2.3, je středem scény a ať se na ni pozorovatel podívá z jakéhokoliv ze 45 pohledů, nikdy se její pozice nezmění. Nicméně objekty, které se nachází

[3] https://docs.lookingglassfactory.com/Appendix/how-it-works/

blíž nebo dál této rovině podléhají paralaxu. V tomto případě se jedná o kouli v popředí a kruh v pozadí [6]. Také si lze všimnout dvou kuželů v levé části obrázku 2.3, ze kterých vychází dvě černé přímky. Ty reprezentují úhel, ze kterého uživatel aktuálně displej sleduje. Z tohoto úhlu vidí uživatel snímek 33.

Paralaxa a ostrost

Díky rovině paralaxy vzniká několik zaostřovacích rovin. Objekty, které se nachází ve větší hloubce jsou rozmazanější než objekty blíže nulté rovině. To stejné platí i pro objekty blíže pozorovateli. Na obrázku 2.4 lze vidět, že pozorovatel je v jednom momentě schopný vidět hned několik pohledů [6].



Obrázek 2.4: Pohled na displej z pravé strany, převzato[4]

Hlavní výhodou prolínání snímků je možnost se na displej dívat vždy z jiné strany a mít plynulý přechod mezi jednotlivými pohledy. Díky tomu nedochází k žádnému skokovému přechodu, ani jiným podobným problémům. Na druhou stranu má displej i nevýhodu – objekty, které nejsou zobrazené na nulté rovině (tj. uprostřed hloubky displeje), jsou rozmazané. Tuto vlastnost lze vidět na obrázku 2.4.

Problém se zaostřovacími rovinami se dá vyřešit tím, že se bude objekt tvořící hlavní prvek scény nacházet na nulté rovině. Jestliže je toto pravidlo dodrženo, do jisté míry se rozostření eliminuje. Objekty, které jsou v pozadí rozostřeny příliš, lze upravit pomocí použití hloubky ostrosti (angl. *depth of field*).

Nástroje pro práci s Looking Glass

Společnost Looking Glass Factory vytvořila několik základních nástrojů pro práci s jejich zařízením. Tyto nástroje lze rozdělit do dvou skupin. Nástroje pro zobrazování a úpravu dat a nástroje určené pouze k prohlížení různých typů dat. Hlavní aplikací pro multiplatformní vývoj je HoloPlay Core SDK [6]. Díky této aplikaci může vývojář propojit 3D software s displejem a prohlížet si výsledný produkt na druhém monitoru.

Mezi editovací a zobrazovací nástroje patří pluginy pro velmi rozšířené enginy jako jsou Unity a Unreal Engine 4. Také existují knihovny pro Javascript, C/C++ a Blender.

[4] https://docs.lookingglassfactory.com/Appendix/how-it-works/

Mezi aplikace, které jsou určené pouze k prohlížení dat patří např. Lightfield Photo App, ParaView nebo Depth Media Player. Každá aplikace se využívá pro zobrazování jiných typů dat [6].

2.2 Unreal Engine 4

Unreal Engine 4 je herní open source engine vytvořený společností Epic Games. Vývojář, který vydá jakoukoliv aplikaci, vytvořenou na této platformě, odvádí 5 % z výdělků, jestliže překročí celkový výdělek 1 milion dolarů. Jedná se o multiplatformní engine s podporou scriptování a velkým množstvím různých nástrojů pro tvoření her.

Unreal Engine 4 se nepoužívá pouze pro vývoj počítačových her, naopak, jeho využití je v dnešní době velice rozšířené. Používá se například v odvětví architektonických vizualizací, kde se díky jeho dobrému výkonu dají v reálném čase renderovat celá města. V posledních letech se začal aktivně využívat i ve filmovém průmyslu. Engine je spojený s LCD displeji, před kterými stojí herec, a promítá na ně scénu. Díky tomu lze scénu měnit přímo v produkci podle potřeb režiséra. Těchto možností bylo využito například při natáčení seriálu Mandalorian.[5]

Skriptování v Unreal Engine 4

Unreal Engine 4 umožňuje využití takzvaného Blueprint Visual Scripting systému. Jedná se o způsob, který využívá vizuální programování k tvoření herních elementů přímo v editoru enginu. K programování se využívají uzly, které se vzájemně propojují a reagují na sebe v závislosti na různých událostech. Uzly v tomto případě představují např. funkce, události nebo proměnné. Výsledkem těchto spojení je graf, který má velkou škálu využití. Jedná se například o vytváření objektů, individuálních funkcí nebo herních událostí, které jsou specifické pro každou instanci blueprintu, ve snaze o implementaci různého chování a funkcionalit. Velkou výhodou je možnost vytvářet vlastní makra a funkce, na které se lze odkazovat v jiných blueprintech. Blueprint je tedy možné popsat jako reprezentaci chování libovolného modelu, nebo objektu, pomocí objektově orientované třídy [2].

Tento systém je velmi flexibilní, jelikož umožňuje propojení dvou světů, a to toho programátorského a designerského. Jelikož je systém také do jisté míry uživatelsky přívětivý, umožňuje designerům využít prostředky jinak dostupné pouze programátorům. Programátoři mohou díky implementaci enginu psát blueprinty přímo v jazyce C++ a následně na ně navazovat v editoru [2].

Typy blueprintu

V Unreal Engine 4 existuje několik typů blueprintů, každý z nich má však jiné využití [2].

Level Blueprint

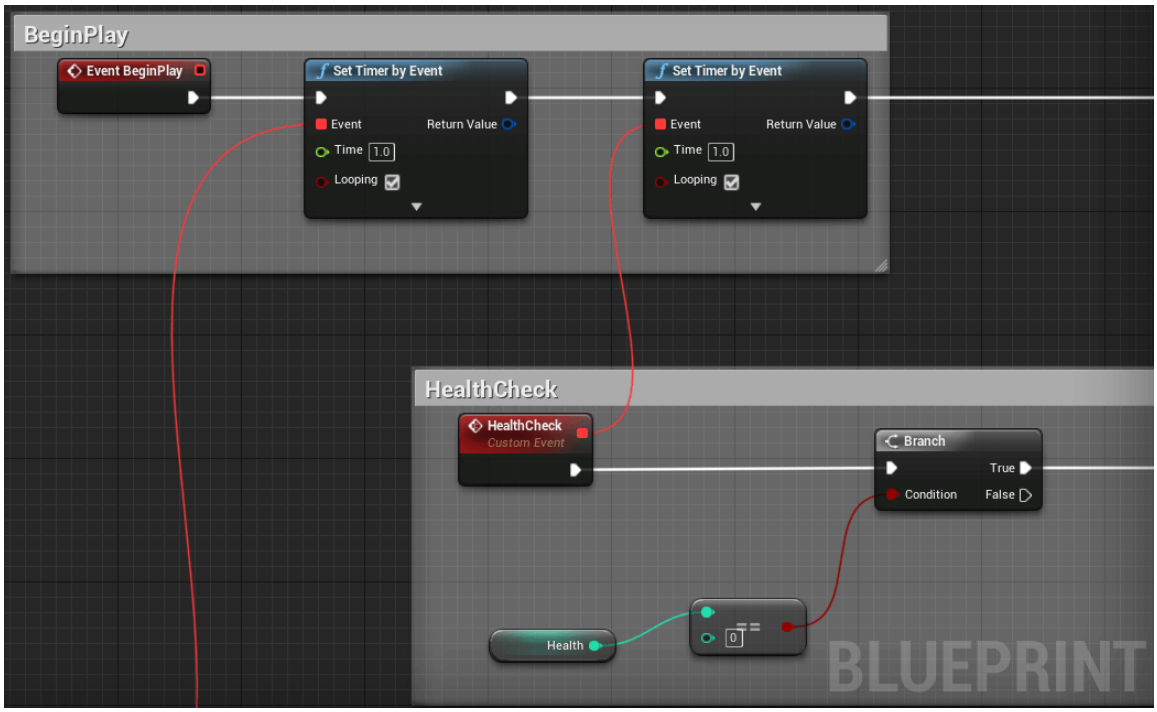
Jedná se o speciální blueprint, který funguje na úrovni jedné herní úrovně a obsahuje všechny její grafy událostí. Každá nově vytvořená úroveň obsahuje tento typ blueprintu, který je pro ni automaticky vytvořen a který spravuje všechny v ní obsažené prvky. Jednotlivé události, nebo specifické instance prvků, spadající pod nadřazenou úroveň, se používají k zahájení sekvencí událostí například v podobě volání funkcí. Výhodou tohoto typu

[5] https://www.unrealengine.com/en-US/blog/forging-new-paths-for-filmmakers-on-the-mandalorian

blueprintu je možnost odkazovat se na všechny objekty obsažené v úrovni, což umožňuje jednoduchou práci s nimi.

Blueprint Class

Je to nejzákladnější blueprint, který se používá převážně pro přidávání funkcionality třídám typu Actor. Actor je standardní objekt, který se dá vložit do scény a který podporuje 3D transformace. Actor může být v průběhu herní simulace vytvořen a zničen právě pomocí blueprintu.



Obrázek 2.5: Ukázka blueprint třídy

Na obrázku 2.5 lze vidět část blueprintu třídy `MyCharacter`, implementované v této práci. Funkce `EventBeginPlay` se spouští při každém startu hry, pokud je tedy instance třídy umístěna ve scéně. Následují dva časovače, které spouští události. Jedna z těchto událostí se jmenuje `HealthCheck` a v průběhu hraní sleduje stav hráčových životů.

Data-Only Blueprint

Tento Blueprint Class obsahuje pouze kód ve formě grafů, proměnné a komponenty zděděné od svého rodiče. Ty se využívají pouze ke změně zděděných vlastností.

Blueprint Interface

Jedná se o sbírku jedné nebo více funkcí (bez implementace), které mohou být přidány do jiných blueprintů, umožňující tak jejich využití. Tento typ obsahuje omezení, jako je zákaz přidávání nových komponent, změna grafů nebo přidávání proměnných.

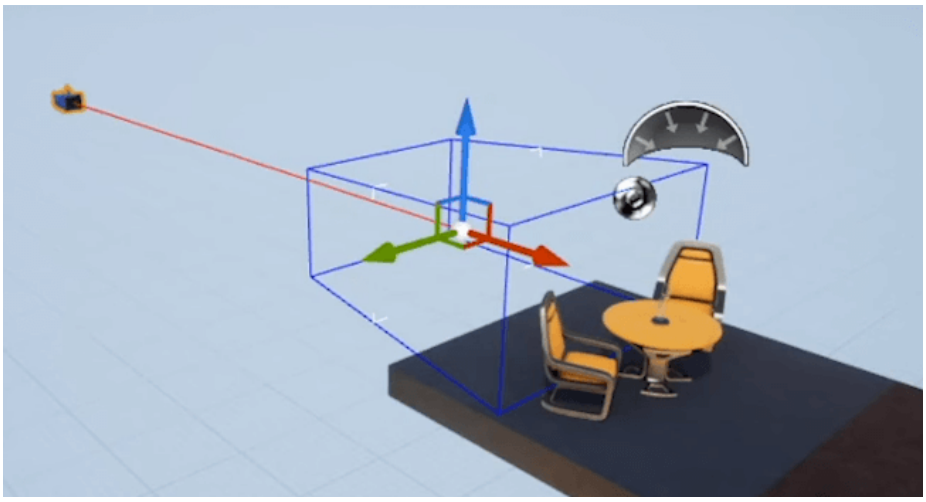
Blueprint Macro Library

Je to knihovna maker, které se dají použít v novém blueprintu, což umožní zefektivnění práce v enginu.

HoloPlay plugin pro Unreal Engine 4

HoloPlay plugin je rozšíření pro Unreal Engine 4. Jedná se o balíček, který přidává nové funkce do editoru a umožňuje vývojářům vyvíjet aplikace pro holografický displej.

Hlavní částí pluginu je HoloPlayCapture Actor, který lze vidět na obrázku 2.6. Jedná se o kameru, která ve scéně nesmí chybět. Tato kamera má plno svých funkcí, pomocí kterých lze měnit výstup na displej.



Obrázek 2.6: Kamera obsahuje prostor ohraničený modrou barvou. Všechny modely umístěny v tomto prostoru jsou zobrazeny na displeji. V tomto případě by se tedy židle, které lze vidět ve scéně, na displeji nezobrazily. Dále si lze povšimnout bílých čar, které jsou zhruba v polovině modrého prostoru. Tyto čáry udávají pozici již dříve zmiňované roviny nulté paralaxy. Podle nich lze vidět, co bude na displeji zaostřené a co naopak rozostřené.

Další technologie pro vývoj počítačových her

Dalšími technologiemi se myslí herní enginy, které se využívají k tvorbě her. Na trhu se jich pohybuje velké množství a každý je orientovaný na trochu jinou cílovou skupinu herních studií.

Unity

Unity je herní engine vytvořený firmou Unity Technologies. První verze byla vydána v roce 2005. Do dnešního dne se engine významně rozrostl a momentálně je hlavním konkurencí Unreal Engine 4. Podporuje až 25 platforem a disponuje širokou nabídkou využití nejenom v herním průmyslu [10]. Je také široce využívaný mezi indie vývojáři.[6] Právě díky své cenové dostupnosti a multiplatformnímu využití se využívá v malých herních studiích. Nicméně díky jednoduché distribuci na velké množství platforem Unity využilo také například světově

[6] https://blogs.unity3d.com/2018/08/03/the-way-small-independent-studios-create/

známé herní studio Blizzard Entertainment pro vývoj multiplatformní hry Heartstone.[7] Mezi další herní tituly, vytvořené pomocí tohoto enginu, patří například Ori and the Blind Forest nebo Fall Guys: Ultimate Knockout.

Amazon Lumberyard

Amazon Lumberyard je engine vytvořený firmou Amazon v roce 2016. Tento engine je postaven na základech CryEngine. Lumberyard je volně přístupný a multiplatformní. Umožňuje vývojářům používat Amazon servery jako hosting pro jejich hry [10]. Mezi herní tituly, vytvořené pomocí tohoto enginu, patří například Star Citizen, New World nebo Deadhaus Sonata.

CryEngine

CryEngine je engine vytvořený firmou CryTek v roce 2002. Nejnovější verzí je CryEngine V. Je to engine, který se dá využít napříč platformami. Využívá se především na hry v první osobě. Mezi herní tituly patří například Hunt: Showdown nebo Kingdom Come: Deliverance [10].

2.3 Procedurální generovaní

Jedná se o jakýkoliv typ automaticky generovaného prvku, který využívá pouze limitující sadu vstupních parametrů definovaných uživatelem [9]. Ke generování se využívá množina algoritmů a pravidel. Je to způsob, kterým se dá výrazně urychlit například vývoj počítačových her.

Jednou z forem procedurálního generování v herním průmyslu je generování krajiny pomocí různých šumů. V dnešní době se procedurální generování používá i ve filmovém průmyslu, např. při generování explozí. Hlavní výhodou, zároveň také nevýhodou, je malá moc nad generovaným obsahem. Například v grafické části herního průmyslu se považuje tento fakt za hlavní nevýhodu generovaní, jelikož je na grafických prvcích, jenž jsou vygenerovány procedurálně, stále poznat fakt, že nejsou vytvořené člověkem. Tento nedostatek byl ale za posledních pár let v podstatě eliminován do takové míry, že už hráč tento nedostatek vůbec nepozná.

Vlastnosti procedurálního generování

Hlavním omezením procedurálního generování je zajisté jeho náročnost. Při generování jsou důležité dvě informace. Jak precizní má generování být a kolik je dostupné paměti. Čím složitější modely se generují, tím více paměti zabírají a tím větší výpočetní sílu potřebují. Někdy naopak místo v paměti šetří, a to třeba při generování herního prostředí. Místo načtení celé herní mapy lze generovat pouze oblast kolem hráče.

Procedurální generování má několik výhod, mezi které patří například šetření časem, možnost rozšířit generátor nebo znovu využít kód.

Generování pseudonáhodných čísel

Generování pseudonáhodných čísel se v procedurálním generování používá právě proto, že se dokáže jejich generování do jisté míry ovládat. Pro jejich generování se využívá speci-

[7] https://unity3d.com/case-study/hearthstone

álních algoritmů a jejich výstup dosahuje podobných vlastností jako pravá náhodná čísla [7]. Mezi nejvyužívanější generátory patří například lineární kongruentní generátor (angl. *Linear Congruential Generator*, LCG), který je níže popsaný.

$$x_{i+1} = (ax_i + b) \mod m$$

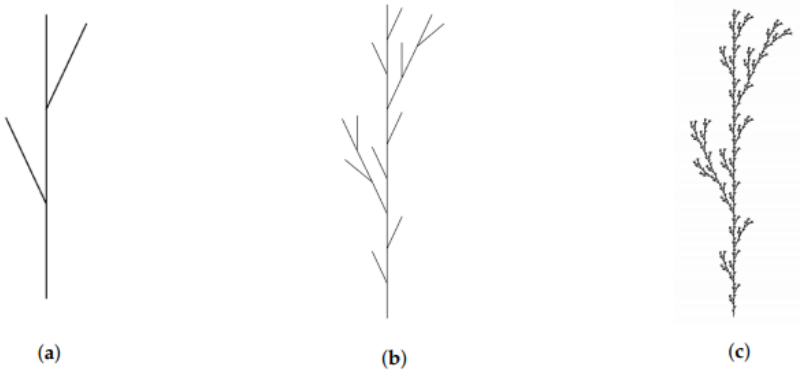
kde

- operace *mod* znamená zbytek po celočíselném dělení,
- a, b a m jsou vhodně zvolené konstanty [7].

L-systémy

L-systémy, též známy pod názvem Lindenmayerovy systémy, jsou formální gramatika, vytvořena v roce 1968 biologem Aristidem Lindenmayerem. Lindenmayer vytvořil tuto gramatiku pro popis růstu řas.

Hlavní koncept L-systémů je přepisování, které se provádí jako opakované nahrazování jednoduchých částí objektu pomocí množiny obsahující pravidla pro přepisování. Nejběžnější podobou L-systému je řetězec symbolů, tedy abeceda. Počátečnímu stavu se říká *axiom*. Na tyto symboly se v každé iteraci aplikují jednotlivá pravidla, čímž se generují další symboly [8]. Na obrázku 2.7 lze vidět generování jednoduché rostliny pomocí L-systému.



Obrázek 2.7: Rostlina vygenerována pomocí L-systému. (a) jedna iterace, (b) dvě iterace, (c) pět iterací, převzato a upraveno [9, str. 8]

DOL systémy

Základním typem L-systému jsou tzv. DOL systémy. Jedná se o deterministické a bezkontextové systémy.

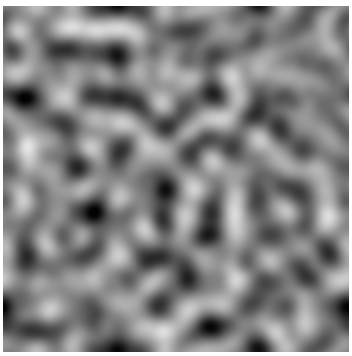
Mějme *axiom* b a pravidlo $a \to ab$, kde znak a nahradí řetězec ab a pravidlo $b \to a$, kde znak b nahradí znak a. V první iteraci se pomocí pravidla $b \to a$ nahradí znak b znakem a. V dalším kroku se využije pravidla $a \to ab$, díky kterému se nahradí znak a symboly ab. V této iteraci již aplikujeme obě pravidla zároveň, tudíž se nahradí znak a znaky ab a znak b znakem a. Vznikne řetězec aba a tento proces lze opakovat, dokud je to vyžadováno [8].

Šumy a Diagramy

V počítačové grafice jsou široce rozšířené generované šumy a diagramy. Jedná se o funkce, které se chovají jako generátory náhodných čísel, nicméně jejich využití je v počítačové grafice rozšířenější než kde jinde [5]. Šumy a diagramy se využívají při generování textur, celých úrovní nebo při využívání normálových map pro generování detailů modelů.

Perlinův šum

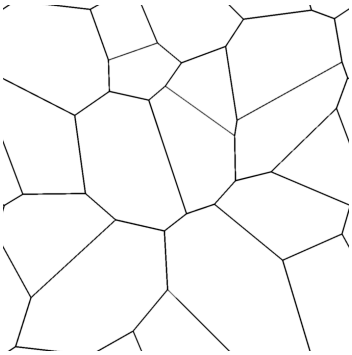
V roce 1983 Ken Perlin vytvořil Perlinův šum, který využil pro generování procedurálních textur ve filmu Tron. Do dnešní doby se jedná o šum, který je nejvíce využívaný v počítačové grafice. Využívá se například při generování krajiny, mraků nebo vzorkovaných materiálů jako je třeba mramor [5]. Na obrázku 2.8 lze vidět perlinův šum.



Obrázek 2.8: Perlinův šum vytvořený v aplikaci Substance Designer od firmy Adobe

Voroného diagram

Voroného diagramy mají využití v mnoha odvětvích. V počítačové grafice se používají především při texturování 3D modelů nebo při generování krajiny. Na obrázku 2.9 lze vidět voroného diagram.



Obrázek 2.9: Voroného diagram vytvořený v aplikaci Substance Designer od firmy Adobe

2.4 Fotogrammetrie

Fotogrammetrie je obor 3D grafiky, který se začíná poslední dobou velmi rozšiřovat [4]. Převážně tedy do herního a filmového průmyslu.[8] Jedná se o proces převodu 2D statického obrazu na 3D objekt [3]. Velkou výhodou tohoto procesu je především to, že lze zachytit detaily každodenního světa, které by bylo náročné vytvářet pomocí klasického 3D postupu. Hlavní předností fotogrammetrie je možnost v pozdější fázi práce promítnout na model texturu, a tím dosáhnout neuvěřitelné reálnosti objektu. Nejpoužívanější programy jsou RealityCapture, Agisoft Metashape nebo 3DF Zephyr.

U tohoto procesu se musí grafik řídit pevně stanovenými pravidly, jinak nebude výsledek jeho práce plnohodnotný. Fotky objektu nebo krajiny se pořizují, když je zataženo a slunce přímo nesvítí na focený objekt. Pokud by se objekt fotil při přímém slunci, měl by na své textuře znatelně zapečené světlo a tím by byla negativně ovlivněna kvalita výsledného produktu. Z toho důvodu vznikají nástroje pro odstranění stínů. Jedním z nich je například program Artomatix. Také by se nemělo fotit při dešti, a to hlavně kvůli fotoaparátu a jeho možnému poškození. Musí se vzít v potaz také to, že nelze skenovat objekty, které mají silné reflexe (např. sklo, kov).

Při využívání fotogrammetrického přístupu při vytváření modelů se postupně prochází několika kroky. Nejprve se musí upravit a přichystat pořízené fotografie objektu. Všechny fotografie musí mít stejnou velikost, stejné nastavení clony a objekt by se měl na dvou sousedních fotografiích vždy do jisté míry překrývat. V libovolném grafickém editoru lze fotografie upravit například zredukováním stínu nebo ztmavením světlých částí snímku.

Po úpravě se fotografie vloží do libovolného programu pro fotogrammetrii a provede se první krok, kterým je srovnání kamer. V této fázi program porovná všechny fotografie a seřadí si je ve 3D prostoru tak, jak byly rozmístěny při focení daného objektu. Tento krok se provádí pomocí vyhledávání podobných pixelů na jednotlivých obrázcích. Proto by se obrázky měly překrývat. Výsledkem této operace je řídký oblak bodů (angl. *point cloud*), ze kterého se v dalším kroku tvoří hustý oblak bodů (angl. *dense cloud*), který lze vidět na obrázku 2.10. V této chvíli lze některé body vymazat, a tak ušetřit výpočetní paměť při generování polygonového modelu. Z upraveného hustého oblaku bodů se vygeneruje polygonální model, který je možné následovně vyexportovat například ve formátu fbx nebo obj. Výsledný objekt může vypadat například jako je znázorněno na obrázku 2.11.

Objekt lze v libovolném programu zdecimovat – zredukovat hustotu jeho polygonální sítě, rozbalit a vytvořit tzv. UV mapu, která slouží pro 2D reprezentace 3D objektu. Následně se dá za použití vytvořené UV mapy vygenerovat textura, která je hlavní výhodou tohoto modelovacího postupu. Obsahuje totiž všechny možné detaily, které by se velmi těžko daly suplovat klasickým texturovacím postupem.

V této práci jsou využity některé modely a textury z knihovny Quixel Megascans[9], která je zdarma k použití při práci s Unreal Engine 4.

[8] https://3dscanexpert.com/3d-scanning-star-wars-battlefront/
[9] https://quixel.com/megascans/home

Obrázek 2.10: Na obrázku lze vidět kmen stromu, složený z hustého oblaku barevných bodů

Obrázek 2.11: Modely vytvořené pomocí fotogrammetrie

Kapitola 3

Návrh

Tato kapitola a její podkapitoly se zabývají návrhem řešení a přístupem k jednotlivým herním prvkům této práce. Popisuje rozhodnutí, které vedly ke zvolení určitého žánru. Dále popisuje podobu hry a její zasazení do kontextu. V další podkapitole se rozebírají jednotlivé herní mechaniky, které využívají novým způsobem vlastnosti holografického displeje. Také se rozebírají důvody výběru daného herního enginu. Mimo jiné je zde popsána problematika generování úrovní a způsoby, kterými k ní přistupovat. Hlavním cílem je vytvořit hru, která inovativně využívá prvky holografického displeje a objevuje nové možnosti jeho využití.

3.1 Žánr

Výběr herního žánru byl do jisté míry limitován vlastnostmi holografického displeje. Jelikož displej pracuje s rovinou nulté paralaxy, popsané v kapitole 2.1, nepřipadalo v úvahu vytvářet hru ve třetí osobě. Když by se hra odehrávala z pohledu třetí osoby, byla by zaostřena pouze postava hráče. Nedalo by se tedy využívat dalších implementovaných mechanik a hraní by nebylo komfortní. Díky tomu jsem se rozhodl pro hru v první osobě, která umožňuje využívání onoho zaostření na rovinu a pracovat tak s různými herními elementy.

3.2 Prostředí

Herní prostředí využívá jako hlavní motiv prostředí dolů, které jsou tvořené spletitými šachtami s množstvím tajemných zákoutí a slepých uliček. Cílem hry je dostat se zpět na zemský povrch za pomoci sestavení žebříku. Každá úroveň nabízí různé úkoly, za které je hráč odměňován v podobě kousků žebříku. Určitý počet kousků žebříku umožní postup do další úrovně. Pro ozvláštnění jsou ve hře zakomponovány prvky, znesnadňující dosažení cíle. Doly jsou tvořeny několika typy chodeb, což umožňuje eliminovat dlouhé rovné vzdálenosti, které by se na displeji z důvodu rozostření neměly zobrazovat.

3.3 Herní tempo

Vzhledem k využití holografického displeje je herní tempo pomalejší. Tento fakt je způsobený především dvěma důvody. Prvním z nich je náročnost pro vykreslování na displeji, který se většinou využívá ve spojení se statickou kamerou a pohybem scény kolem ní. Z tohoto důvodu je omezena plynulost hry. Druhým je nepřehlednost v kombinaci s ovládáním. Hráč

je nucen se často otáčet a v některých částech mapy být velice ostražitý. Hlavní myšlenkou tohoto spojení je rozvážné prozkoumávání a objevování tajů nových technologií.

3.4 Herní mechaniky

Ve hře se vyskytuje několik herních mechanik, které se odvíjí od vlastností displeje. V následujících podkapitolách jsou tyto mechaniky přiblíženy.

Zobrazování objektů

Modifikace objektů vývojáři umožňuje libovolně měnit jeho viditelnost. Je možné nastavit objektu viditelnost pouze z jednoho úhlu pohledu. Díky této funkcionalitě je hráč nucen se dívat na displej vždy z více úhlů, aby nalezl např. skrytý text nebo vyřešil mini hru.

Rychlé události

Jsou to události, při kterých musí hráč rychle zareagovat, zorientovat se v okolí a provést nějaký úkon. Tyto události jsou spojeny se zobrazováním objektů pod jinými úhly. Hráč je tedy nucen být v některých částech hry v pozoru a reagovat na rychlé změny.

Nahlédnutí za roh

Jelikož displej disponuje zobrazovacím standardem quilt, rozebraným v podkapitole 2.1, který funguje na principu zobrazování 45 pohledů v jeden okamžik, lze toho využít. Tato vlastnost displeje utváří tří dimenzionální vjem vznikající u pozorovatele. V našem případě se dá využít tak, že hráč procházející dolem musí nahlížet za rohy a kontrolovat tak, jestli někdo za rohem není. Kdyby hráč prošel bez nahlédnutí, viděli by ho nepřátelé a obdržel by penalizaci v podobě ztráty života. Nahlížet hráč ale nebude ve hře, nýbrž fyzicky před displejem. Fyzicky se tedy podívá, zdali za rohem není nepřítel a popřípadě počká, až nebezpečí pomine.

Perspektiva a světlo

V této hře se využívá perspektivy a různými způsoby, jak s ní pracovat. Také se při tom využívá quilt a více pohledů. Hráč si např. musí vybrat jednu ze tří krabic, které jsou k němu postaveny čelem. Na boku každé krabice je naznačeno, jestli je její obsah bezpečný nebo naopak hráče postihne ztrátou života. Hráč se dokáže díky fyzickému pohledu z jiného úhlu podívat na obsah krabice a zjistit, kterou možnost vybrat.

Dále jsou různě po jednotlivých úrovních umístěny skryté chodby, které uvidí hráč pouze při pohledu z určitého úhlu. V úrovních se také mohou nacházet skryté klíčové dírky, které otevřou dveře vedoucí do bonusových částí úrovně.

Světlo svítí na několik objektů vedle sebe a všechny budou pod určitým úhlem vypadat stejně. Hráč musí vybrat objekt, který bude mít jinou vlastnost než ostatní objekty.

Mini hry

V úrovních se také budou nacházet mini hry v podobě různých hlavolamů nebo pexes. Když hráč tuto mini hru úspěšně vyřeší, umožní mu to postup do další části úrovně nebo bude odměněn vylepšením.

Bod zájmu

Další mechanikou, kterou hra disponuje, je tzv. bod zájmu (angl. *point of interest*). Základní vlastností displeje je zaostřování na osu nulté paralaxy, takže vše za ní a před ní je rozmazané. Této „nevýhody“ se však dá využít. Po mapě budou rozmístěny body zájmu v podobě lesklých vykřičníků, které budou v dálce sice rozmazané, ale pro hráče velice zajímavé. Díky své lesklosti na displeji vyčnívají, i když jsou zrovna rozmazané. Tyto body zájmu upozorňují na úkol, nebo kousek žebříku, který je ukryt někde v blízkosti.

Kamera

Práce s kamerou hraje významnou roli. Jedním z prvních nápadů bylo vytvořit hru ve 2D, která by se inspirovala herním titulem Ori and the Blind Forest, ve kterém je postava hráče po celou dobu viditelná a kamera ji z profilu následuje. Nakonec se přistoupilo k myšlence vytvořit hru ve 3D.

Při rozhodováním mezi pohledem z první osoby nebo třetí osoby jsem musel zohlednit především chování displeje a výsledný vizuální dopad na hratelnost hry. Kamera disponuje vykreslovací oblastí, která by se dala rozdělit do tří vrstev. První vrstva nejblíže kameře je popředí, poté je zaostřovací rovina a pozadí. Při použití kamery ve třetí osobě, by postava hráče byla buď nepříjemně rozostřená, nebo by zabírala velké místo v zaostřovací rovině, což by omezilo další aspekty hry. Tím pádem kamera z pohledu třetí osoby nevyhovuje tomuto zobrazovacímu zařízení a musí se využít kamera z pohledu první osoby.

Při využití kamery z pohledu první osoby hráč bohužel nevidí svoji postavu, nicméně může využívat zaostřovacího spektra před sebou do mnohem větší míry.

Ovládání

Ovládání v této hře je velmi specifické, právě díky zobrazovacímu zařízení, které se používá pro interakci s hráčem. Zprvu se využívalo klasické ovládání pomocí klávesnice a myši. Nicméně tento typ ovládání znemožňuje využívat některé navržené herní mechaniky. Proto se přistoupilo k ovládání pouze z klávesnice. Hráč se může libovolně pohybovat dopředu, dozadu a do stran. Pro otáčení kolem osy Z se používá rotace o 90° doleva nebo doprava. Tato rotace umožňuje využívání některých herních mechanik, které vyžadují fixaci hráče v určité poloze a pod určitým úhlem.

Úrovně

Úrovně budou procedurálně generovány, i napříč tomu musí být vždy dokončeny vývojářem, ať už se jedná o jednotlivé rozmístění assetů nebo o přestavbu mapy z estetických důvodů. Aby byla úroveň kompletní, musí obsahovat následující položky [12]:

- Místnosti – jsou to rozměrově větší oblasti, ve kterých se zpravidla objevují pro hráče zajímavé předměty. Mohou se zde také nacházet mini hry. Mají pouze jeden vchod/-východ.
- Chodby – jsou to rozměrově menší oblasti, které se využívají ke spojení jednotlivých místností. Mají dva východy, resp. vchody.
- Křižovatky – jsou to rozměrově menší oblasti, které disponují alespoň třemi vchody, respektive východy.

- Předměty – jsou to herní elementy, které hráči umožňují posunout se v příběhu.
- Výzva – je to překážka, kterou musí hráč zdolat při cestě úrovní.
- Hlavolamy – jsou to překážky, které brzdí hráčův postup a jejich vyřešení slouží jako předpoklad ke splnění úrovně.

Ke generování se dá přistupovat mnoha různými technikami[1]. Lze využít například generování pomocí celulárních automatů, gramatik nebo pomocí omezení [11]. Vzhledem k typu hry se zde pracuje s generováním pomocí omezení. Tento způsob se využívá ke generování podzemních chodeb a sklepení.

Jednotlivé úrovně jsou koncipované jako labyrint. Celkově se dá tento labyrint popsat jako sada 3D modulů, které jsou vzájemně propojeny v závislosti na množině pravidel. Každý modul má speciální označení, který slouží ke zjištění typu modulu. Moduly také disponují označením vchodu, respektive východu. U těchto značek musí být možnost se odkazovat na jejich pozici ve scéně (angl. *world position*) a na jejich rotaci ve scéně. Z každé značky lze zjistit, jaký modul k ní může být přiřazen.

Základní pravidla pro spojení modulů jsou následující:

- Místnosti lze spojit s chodbami.
- Chodby lze spojit s místnostmi a křižovatkami.
- Křižovatky lze spojit s chodbami.

Jednotlivé kroky pro spojení dvou modulů jsou následující:

1. Vybrat nespojený vchod/východ starého modulu.
2. Vybrat nový modul, jehož specifikace odpovídají pravidlům pro připojení ke starému modulu.
3. Vytvořit novou instanci modulu.
4. Vybrat východ/vchod nového modulu.
5. Spojit moduly.
6. Odstranit odkazy na východy/vchody z množiny.
7. Opakovat, dokud stále existují nějaké nespojené moduly.

Nepřátelé

V jednotlivých úrovních se vyskytují nepřátelé, jejichž hlavní rolí je zpozorovat hráče. V každé úrovni je prozatím pouze jeden nepřítel, který hlídkuje na předem stanovené trase. Jakmile se hráč ocitne v jeho zorném poli, následuje hráčova penalizace v podobě ztráty života. Do budoucna by se dalo do úrovně přidat více nepřátel, a to i s možností pronásledování hráče po nějakou dobu.

[1] https://gamedevelopment.tutsplus.com/tutorials/bake-your-own-3d-dungeons-with-procedural-recipes--gamedev-14360

3.5 Herní engine

Jak je již zmíněno v sekci 2.2, v této práci se využívá vývojové prostředí Unreal Engine 4. Vzhledem k relativně omezenému výběru, způsobenému především faktem, že firma Looking Glass Factory Inc. vytvořila aplikace podporující pouze 2 herní enginy, byl vybrán ze zmíněné dvojice Unreal Engine 4. Druhým enginem je herní engine Unity.

Hlavním důvodem pro tento výběr je jazyk **C++**, který se v Unreal Engine 4 používá. Na rozdíl od Unity, který pracuje s jazykem **C#**. Další výhodou je práce s vizuálním skriptováním. To slouží k programování logické části herních mechanik, jako je například generování úrovní, programování logiky nepřátel anebo chování různých objektů.

Výhodou je beze všeho velké množství dostupných návodů, různých fór, discordových serverů a videí, které řeší problematiku herního vývoje v tomto vývojovém prostředí. Také stojí za zmínku výborné propojení s ostatními programy, jako jsou například Substance Painter a Substance Designer.

Všechny tyto důvody vedly právě k výběru **Unreal Engine 4**.

Kapitola 4

Implementace

Tato kapitola a její podkapitoly se zabývají řešením a implementací jednotlivých herních prvků této práce. Popisuje způsoby, kterými se řešily vzniklé problémy při tvorbě 3D dema. Dále jsou zde popsány některé postupy, které vedly k vytvoření herních mechanik. Jsou zde také popsány a vysvětleny skripty, které jsou v demu použity.

V průběhu celého vývoje byla použita verze 4.24.3-11590370+++UE4+Release-4.24 herního enginu. Vzhledem ke kompatibilitě s rozšířením HoloPlay od firmy Looking Glass Factory Inc., bylo nutné využít právě tuto starší verzi software. Jako vývojové prostředí pro psaní a úpravu kódu bylo využito Visual Studio 2019 od firmy Microsoft. Vzhledem k vzájemné podpoře se toto IDE hodí přesně pro práci s Unreal Engine 4. Umožňuje editovat a překládat kód v reálném čase a tím pádem se vyhnout zdlouhavému vypínání a zapínání herního enginu. Mimo jiné slouží jako příjemné prostředí pro ladění kódu.

Vývojáři je umožněno použít několik verzí quilt formátu. Každá verze má odlišné požadavky na výpočetní sílu, a tak se tímto způsobem dá zrychlit vykreslování výsledného obrazu. Kvůli tomu je v této práci využit quilt s výsledným počtem 32 snímků.

4.1 Použité prostředky

V této práci jsou použity modely a textury z knihovny **Quixel Megascans** od firmy Quixel. Dále byl použit materiál pro zvýrazňování objektů, dostupný zde[1].

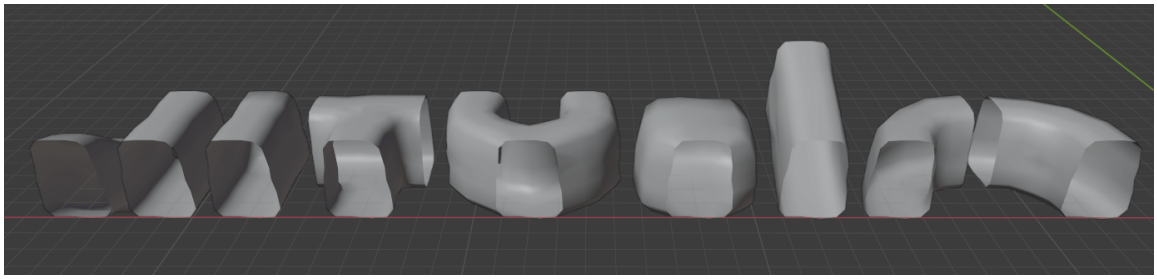
4.2 Modely

Kvůli procedurálnímu generování jednotlivých herních úrovní se muselo k vytváření modelů přistoupit určitým způsobem. Bylo třeba aby modely byly jednoduše použitelné a byla umožněna jejich návaznost. Díky tomu se dalo využít známého stylu modelování, tzv. modulární modelování (angl. *modular modeling*), které se používá v herním průmyslu. Na obrázku 4.1 lze vidět modulární modely, vytvořené pro tuto práci.

Při využití tohoto způsobu se pracuje s předem pevně danými rozměry jednotlivých assetů, se kterými se následně manipuluje na mřížce (angl. *grid*) 3D editoru. Objekty jeden na druhý přesně navazují, což umožňuje použití procedurálního generování.

Všechny modely z knihovny Megascans mají již připravené a aplikované materiály, a tak uživatel nemusí trávit čas nastavováním materiálů. Nicméně jsou zde použity taktéž materiály, které byly vytvořeny pro speciální účely nebo pro ručně dělané modely.

[1] http://www.michalorzelek.com/blog/tutorial-creating-outline-effect-around-objects/



Obrázek 4.1: Modululární modely

4.3 Světla a post processing

Jediným zdrojem světla jsou statická světla, která jsou umístěna v jednotlivých modulech. Kvůli problémům s plynulostí hry bohužel nebylo možné využít dynamické světlo a některé funkce, které se běžně využívají (např. *volumetric fog*). Mezi funkce post processingu, které byly použity, patří možnost vizuálně zvýrazňovat zvolené objekty a hloubku ostrosti. Ta se využila pro dodatečné rozostření objektů v pozadí.

4.4 Uživatelské rozhraní

Uživatelské rozhraní je v této hře vzhledem k demoverzi pouze základní. Rozhraní je vytvořené pomocí `Widget blueprint` třídy. Obsahuje informace o aktuálním zdraví hráče a také o počtu získaných předmětů. Tato třída se mimo jiné používá pro zobrazení informační zprávy pro hráče při spatření nepřítelem.

4.5 Procedurální generování úrovní

Jednotlivé moduly jsou v úrovni rozmístěny pomocí skriptu `ModularGenerator`. Jedná se o tzv. Utility Blueprint, který se na rozdíl od normálního blueprintu dá používat, i když není hra aktivní. Takové skripty slouží pro práci v editoru, jako je například generování herních úrovní nebo úprava textur.

Moduly

Jak již bylo řečeno v sekci 3.4, pro generování procedurální úrovně se používají moduly. Tyto moduly jsou v editoru definovaný jako blueprint obsahující statické objekty. Výhoda takového uspořádání objektů je rychlá práce s moduly a možnost flexibilní úpravy během tvorby.

Na obrázku 4.2 lze vidět jeden z modulů. Jedná se o typ chodby. Se všemi částmi modulu je možné pohybovat, což umožňuje rychle měnit kompozici modulu. Každý modul je opatřen statickým světelným zdrojem a boxy, které slouží k určení kolizí. Tyto kolizní boxy fungují jako zábrana pro hráče, aby se nezasekl v ostatních objektech, a tak byl jeho pohyb plynulý. Lze si také všimnout červené šipky. Tato šipka slouží jako označení východu z modulu.

U statických objektů obsažených v modulu se využívá tvoření automatických úrovní detailu modelu (ang. level of detail), které umožňují se zvětšující se vzdáleností od hráče decimovat zobrazovaný model. Tímto způsobem se šetří výpočetní paměť, a hra je plynulejší.

Obrázek 4.2: Předem připravený modul, sloužící k procedurálnímu generování úrovně.

Generátor

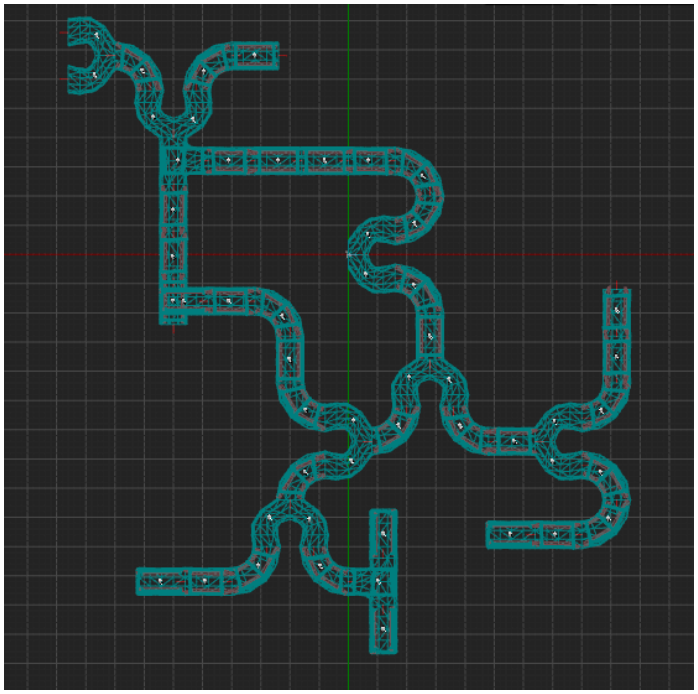
Generátor jako takový se skládá z několika částí. V první části se v aktivní úrovni nejdřív vymažou všechny objekty. Poté se provede inicializace, která obnáší vytvoření prvního modulu. Tento modul je vždy modul typu křižovatka. Vytvoří se jeho instance a umístí se na souřadnicích 0,0,0. Jelikož všechny moduly obsahují označení svých východů v podobě červené šipky, tak se dvě aktuální vloží do proměnné `ExitsActors`. Tato proměnná je typu pole a uchovává všechny ještě nespojené východy v podobě odkazu na objekt šipky. Tímto krokem končí inicializace.

Ve druhém kroku se provádí hlavní cyklus vytváření instancí modulů a jejich umísťování. Celkový počet umístěných modulů se dá řídit pomocí proměnné `Iterations` typu `integer`. Cyklus se opakuje do té doby, dokud je počet zbývajících modulů v poli větší než 0. V těle cyklu se volá funkce `GetMatches()`. Jako vstupní parametr přijímá odkaz na objekt šipky z pole `ExitsActors`. Funkce zjistí, o jaký typ modulu se jedná, aplikuje pravidla pro výběr a vybere správný modul. V této funkci se dají jednotlivá pravidla tvořit a modifikovat, což umožňuje omezit nebo rozšířit množinu vybíraných modulů. Po výběru správného modulu se vytvoří jeho instance, umístí se na pozici aktuálně vybrané šipky a současně se změní jeho rotace podle rotace šipky. Díky tomu, že se všechny hodnoty vybírají relativně ke scéně, všechno je přesné a rotace je vždy správná.

Ve třetím kroku generování se pomocí funkce `GetActorBounds()` vytvoří kvádr, který obalí hranice modulu. Pomocí funkce `BoxOverlapActors` a vytvořeného kvádru lze zjistit, s kolika dalšími moduly typu `WorldStatic` se modul překrývá. To zjistíme podle délky výstupního pole, které obsahuje všechny objekty, se kterými se aktivní modul překrývá. Jestliže je délka pole větší jak 2, modul se vymaže, ukončí se aktuální iterace a začíná se znovu. Jestliže se nenašly žádné překrývající se moduly, vymaže se aktuální šipka z pole `ExitsActors` a každá nová šipka v aktivním modulu se do pole přidá. Na konci iterace se dekrementuje proměnná `Iterations`.

Jakmile se ukončí hlavní cyklus, spustí se cyklus pro uzavření všech otevřených modulů. Tento cyklus se opakuje do té doby, dokud není pole s šipkami prázdné. Po ukončení tohoto

cyklu se vytvoří herní start na pozici 0,0,0. Na obrázku 4.3 lze vidět půdorys vygenerované úrovně z horního pohledu v editoru.



Obrázek 4.3: Vygenerovaná úroveň při pohledu shora

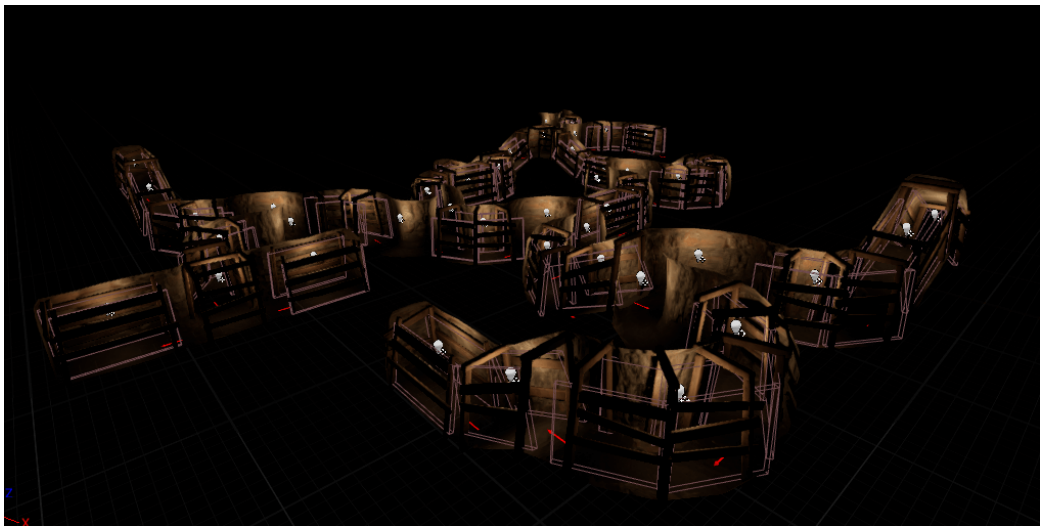
Nutno dodat, že tento generátor je převážně určen pro jednodušší, a v prvé řadě rychlejší tvoření velkého množství úrovní v editoru. Díky vstupním parametrům lze kontrolovat jeho velikost a vytvořením složitějších pravidel také jeho vzhled. Nejedná se ale o plně automatizovaný generátor, neboť po vygenerování úrovně stále musí zasahovat vývojář.[2] Algoritmus 1 zobrazuje pseudokód popisující vytváření úrovně.

Do budoucna lze rozšířit generátor o další funkce, které umožňují např. procedurální umístění assetů nebo tvoření komplexnějších struktur úrovní. Na obrázku 4.4 lze vidět finální vygenerovanou úroveň.

[2] https://gamedevelopment.tutsplus.com/tutorials/bake-your-own-3d-dungeons-with-procedural-recipes--gamedev-14360

Algorithm 1: Pseudokód popisuje postup procedurálního generování úrovní. Funkce se volá s několika parametry, mezi které patří startovací modul, pole modulů a počet iterací. Nejdříve se vytvoří instance startovacího modulu, která se následně umístí do scény a všechny jeho východy se přidají do pole východů. Tento modul slouží jako středobod scény a další kroky jsou na něm závislé. Následně se spustí cyklus, ve kterém se do scény přidávají další moduly. U každého modulu se zjišťuje zdali se překrývá s ostatními moduly – pokud ano, odstraní se modul ze scény a začíná nová iterace – v opačném případě se přidají všechny jeho východy do pole a pokračuje se v generování.

```
def generate_dungeon(starting_module, modules, iterations):
    starting_module = spawn(starting_module)
    exits_actors.append(starting_module.get_exits())
    while iterations > 0 do
        for exit in exits_actors do
            tag = get_tag(exit)
            new_module = spawn(choose_module(modules, tag))
            if new_module.overlap() != 0 then
                new_module.destroy()
                continue
            else
                exits_actors.remove(exit)
                exits_actors.append(new_module.get_exits())
        iterations -= 1
```



Obrázek 4.4: Finální vygenerovaná úroveň.

4.6 Třída hráče

Hráč je v editoru reprezentován jako třída typu `Character`. Každá hra musí obsahovat alespoň jeden objekt tohoto typu. Tato třída obsahuje mimo jiné ovládání pohybu, proměnné uchovávající informace o zbývajících životech, počtu posbíraných žebříků a další prvky. Aby byla třída kompletní, musí obsahovat také kameru, která slouží k vytvoření finálního výstupu pro uživatele.

Kamera

Během procesu tvorby se vyskytlo množství problémů. HoloPlay rozšíření implementuje pouze statickou kameru. Tím se rozumí kamera, se kterou se nedá dynamicky pohybovat. Dalším problémem bylo, že tento objekt nebyl typu `Camera`, nýbrž se jednalo o C++ třídu, kterou nelze využít při práci s blueprinty. Z toho důvodu bylo nutné vytvořit blueprint třídu, která zapouzdří tento kód a umožní jeho využití v ostatních blueprintech.

Nově vzniklá blueprint třída `HoloPlayCapture_Blueprint` je využita ve třídě hráče. Tímto způsobem je možné kameru z rozšíření používat dynamicky, takže není problém se s postavou bez problému pohybovat.

Kamera je nastavená na pozorovací úhel 65°. Tato hodnota je zvolena hlavně kvůli pohybu v uzavřených prostorách.

Funkce

Hlavními funkcemi v grafu je mimo zřejmých (pohyb, rotace), také obstarávání interakce s ostatními objekty. K tomu se používá například funkce `MultiSphereTraceForObjects()`, která v libovolné vzdálenosti před hráčem snímá objekty specifikovaného typu ve sféře s libovolným rádiem. Tato funkce se používá v mini hrách pro vyhledání objektů a následnou interakci s nimi.

Ovládání

Z důvodu využití některých herních mechanik se musel omezit způsob pohybu postavy. Aby byl hráč nucen využívat vlastnosti displeje (nahlížení ze stran atd.), jeho pohled je vždy fixován na jednotlivé osy X a Y. Pomocí kláves Q a E je hráč schopen se otáčet o 90° po směru a proti směru hodinových ručiček. Při vývoji byly vyzkoušeny dva způsoby ovládání. Výše zmíněný způsob a druhý způsob, který byl kombinací otáčení o 90° a otáčení pomocí myši. Když hráč prozkoumával tunely a zrovna neměl v zorném poli nějakou mini hru, která by využívala fixní kamery, mohl se jednoduše otáčet pomocí myši. Jakmile však narazil na herní mechaniku spjatou s vlastnostmi displeje, kamera se zafixovala a nebyla možnost využívat myš. Po uživatelském testování se od této myšlenky upustilo a přešlo se zpět na fixní pohledy, jelikož takové ovládání narušovalo herní zážitek.

4.7 Třída nepřátel

K tvorbě třídy nepřítele se přistupuje podobným způsobem jako při tvorbě třídy hráče. Jedná se o blueprint třídu typu `Character`. V této třídě se obstarává všechna komunikace s ostatními objekty a je zde implementováno chování nepřítele. Model, viditelný na obrázku 4.5, je vymodelovaný v programu Blender a textury jsou vytvořeny v programu Substance Painter. Pro rigging a animace byla využita aplikace Mixamo od firmy Adobe.

(a) Model nepřítele v tzv. T-pose (b) Model obsahující kostru a animace

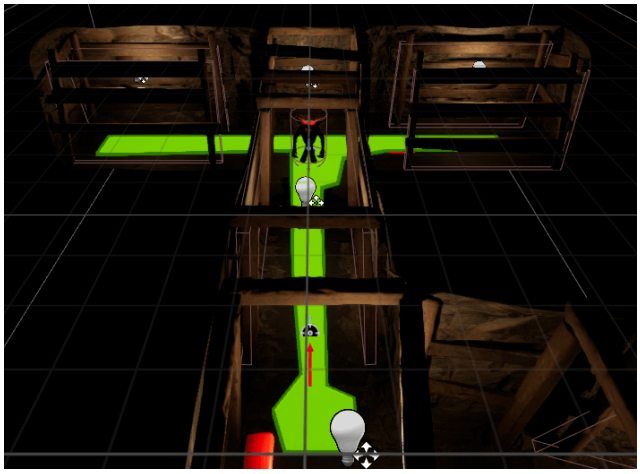
Obrázek 4.5: Model Crashe Bandicoota, který ve hře představuje nepřítele

Ve třídě se používají dvě hlavní metody. Obě využívají funkce z knihovny `Pawn Sensing Component`, která implementuje umělou inteligenci pro postavy. Třída obsahuje dvě veřejné proměnné, které slouží jako ukazatele na objekt typu `TargetPoint` [1].

Chování

Při spuštění hry se zavolá hlavní metoda `Patrol`. Tato metoda je cyklická a opakuje se až do ukončení hry. Využívá funkci `AIMoveTo`, která přijímá odkaz na model, který bude přemisťovat, a ukazatel na objekt `TargetPoint`. Ten v tomto případě slouží jako značka ve scéně, ke které má nepřítel dojít. Jelikož jsou ve třídě definovány tyto značky dvě, dají se využít jako body, mezi kterými nepřítel hlídkuje. V jednoduchém cyklu se tedy pohybuje mezi těmito dvěma body, přičemž na každém z nich čeká 3 sekundy. Na konci jednoho tohoto cyklu se volá znovu metoda `Patrol`. Nicméně, aby se mohl nepřítel pohybovat mezi body, musí být ve scéně umístěn NavMeshBoundsVolume. NavMesh je systém mapující povrch, který je vhodný pro pohyb umělé inteligence. Bez tohoto systému by se tedy nepřítel nemohl pohybovat a stál by na místě. Na obrázku 4.6 je vidět, jak vypadá tento systém ve scéně. Zelená oblast na zemi stanovuje povrch, po kterém se může umělá inteligence pohybovat.

Druhou využitou metodou je metoda `OnSeePawn` z knihovny `Pawn Sensing Compoment`. Tato metoda se zavolá ve chvíli, kdy má v zorném poli nepřítel námi hledanou třídu. Zorné pole nepřítele je vizuálně zobrazeno jako kužel, ve kterém se vrhají paprsky. Jestliže paprsek narazí do hráče, který je v přímém zorném poli, ubere se hráči život a přemístí se na start. Pokaždé, když je tímto způsobem hráč spatřen, provede se posloupnost úkonů. Vzhledem ke skutečnosti, že použitá kamera není typu `Camera`, nelze na ni aplikovat s ní související metody (např. `StartCameraFade` a `StopCameraFade`). V blueprintu hráče je v zorném poli umístěna jednoduchá průhledná rovina. Tato rovina je poté využita ke ztmavení obrazovky v případě spatření nepřítelem. Na rovinu je aplikovaný materiál, z něhož se vytvoří instance a změní se jeho parametr, čímž se rovina stane viditelnou. V pozadí se mezitím hráč přemístí na začátek úrovně a odečte se mu jeden život. Po celou dobu je hráči odebrána možnost pohybu. Jakmile jsou všechny procesy v pozadí dokončeny, rovina se stejným způsobem zprůhlední a hráči se zpět přiřadí možnost se pohybovat.



Obrázek 4.6: NavMesh mapující povrch pro pohyb umělé inteligence

Nahlížení za roh

Chování displeje se využívá ve spojení s nepřáteli v podobě nahlížení za roh. Hráč je ve hře upozorněn na možné nebezpečí např. nepřítel za rohem, v podobě červeného vznášejícího se vykřičníku. Když hráč vběhne do chodby a nepřítel si ho všimne, přijde o jeden život. Toho se dá však vyvarovat. Když se hráč postaví ke konci chodby, dokáže se fyzicky podívat na displeji za roh. Tím se mu podaří spatřit nepřítele, a to aniž by byl on sám spatřen. Jakmile je k němu nepřítel zády, může proběhnout chodbou bez ztráty života. Aby tato mechanika fungovala, bylo třeba upravit parametry komponentu `Pawn Sensing Compoment`. Úhel periferního vidění je nastaven na 50° a vzdálenost, při které nepřítel spatří hráče je 1200 jednotek. Na obrázku 4.7 lze vidět realizaci této herní mechaniky již na holografickém displeji.

(a) Hráč je schovaný za zdí, nepřítel ho nevidí

(b) Hráč fyzicky nahlíží za roh chodby, nepřítel ho stále nevidí

Obrázek 4.7: Herní ukázka mechaniky nahlížení za roh

4.8 Hra s krabicemi

V této mini hře se využívá práce s perspektivou a všímavostí hráče. Hráč musí vybrat ze tří krabic tu správnou. Jestliže se ve výběru zmýlí, přijde o jeden život. Háček je v tom, že každá krabice má na svých stranách jiný ornament. Strany s ornamentem však nesměřují

směrem k hráči ale do boku. Aby mohl hráč vybrat správnou krabici, musí se fyzicky před displejem podívat na krabice ze všech stran a zjistit, která z nich je ta správná. Na obrázku 4.8 lze vidět, jak vypadá mini hra v editoru.

Obrázek 4.8: Výbušné boxy umístěny ve scéně

Krabice jsou ve scéně implementovány pomocí blueprintu, a ne přímo statickým objektem. Tento přístup byl zvolen proto, že blueprint třídy mají vlastnost zvanou `Tag`. Tag umožňuje označit blueprint pro jednodušší zacházení s ním. V tomto případě se využívá pro zjištění, kterou krabici hráč vybral, s čímž je spojená následující akce. Krabice se zvýrazňují, aby hráč věděl, kterou má momentálně označenou.

Zvýraznění se implementuje pomocí použití funkce `MultiSphereTraceForObjects()`, která detekuje objekty, jenž využívají předem vytvořený typ kolizí `Interactable`. Funkce vrátí pole, které obsahuje objekty krabic a následně se mezi nimi dá přepínat pomocí šipek nebo tlačítek na displeji. Pro zvýrazňování objektů se používá renderování hloubkové mapy objektu ve spojení s `PostProcessVolume`, ve kterém je specifikovaný materiál, jenž se aplikuje na vybraný objekt.

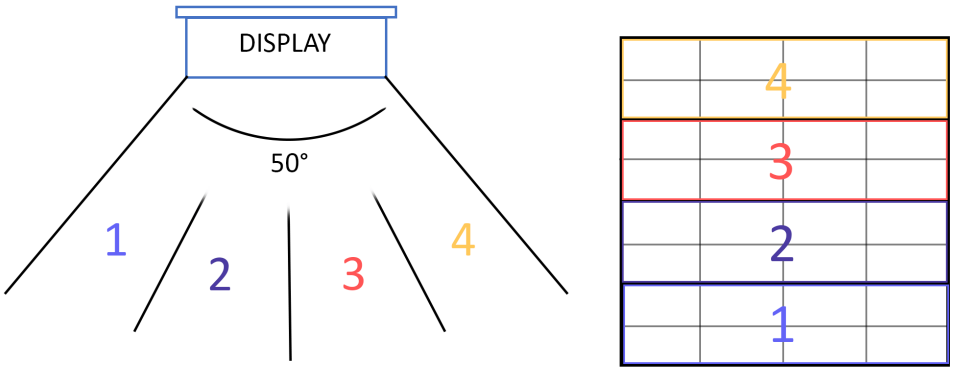
4.9 Viditelnost objektů

Jako další herní mechanika je ve hře implementována práce s objekty a jejich zobrazení na displeji. Tato mechanika umožňuje vývojáři určit, který objekt bude vidět z kterého úhlu. Tohoto efektu lze dosáhnout pomocí práce s quilt formátem, podrobně vysvětleným v kapitole 2.1.

HoloPlay plugin obsahuje komponent kamery, která se skládá ze dvou kamer – hlavní a sekundární. Sekundární je pomocí `Spring Arm` komponenty závislá na kameře hlavní. Sekundární kamera stále mění svoji pozici relativně k hlavní kameře, přičemž pro každý výsledný snímek pokryje všech 32 obrázků zleva doprava. Pro docílení toho, aby byl objekt vidět jenom tam, kde chce vývojář, se musí ovládat viditelnost prvku v jednotlivých obrázcích (částech quilt formátu).

Objekty

Všechny objekty využívající tuto mechaniku musejí být blueprint třída. Podle tagu, kterým jsou označeny, se určuje z jakého úhlu budou vidět. Lze vybírat mezi čtyřmi skupinami: first, second, third a fourth. Tyto skupiny jsou záměrně pouze čtyři, protože rozdělují zorné pole o 50° na čtyři části po 12.5°. Na obrázku 4.9 je tato mechanika přiblížena.



(a) Pohled na displej shora

(b) Quilt formát rozdělený na čtyři skupiny

Obrázek 4.9: Na obrázku 4.9a je načrtnuto zorné pole, které určuje prostor, ve kterém je zřetelně vidět obsah displeje (50°). Tento prostor je rozdělen na čtyři části, každá je označena patřičným číslem. Tato čísla reprezentují skupiny, které se dají přiřadit objektu v editoru. Například objekt, který je označen tagem first (číslo 1), je viditelný pouze při pohledu zleva (tudíž od 0° do 12.5°). Na obrázku 4.9b je formát quilt se 32 snímky, který je rozdělený na 4 skupiny po 8 snímcích. Když je objekt označený tagem first, je viditelný pouze v 8 obrázcích v rámečku 1.

Implementace

Mechanika je implementována přímo v kódu HoloPlay pluginu. Metoda `RenderViews()`, obsažena ve třídě `AHoloPlayCapture`, obstarává renderování quilt formátu. Sekundární kamera se pomocí výpočtu offsetu pohybuje zleva doprava a informace o její pozici se ukládají do pole. Toto pole se předá až do funkce `SetupViewVamilyForSceneCapture()`, kde se zpracuje.

Editovaný kód se nachází v souboru `HoloPlaySceneCaptureRendering.cpp`. Zde je definována globální proměnná `counter` typu integer, která se používá pro určení aktuálních osmi snímků. Kód jako takový je připsaný ve funkci `SetupViewVamilyForSceneCapture()`, která připravuje aktuální scénu pro vykreslení.

Pro iterování mezi objekty se využívá `TObjectIterator`, který iteruje přes objekty typu `AActor`. Iterátor prochází všechny objekty ve scéně. Pomocí výpočtu vzdálenosti aktivního prvku a kamery se zjistí vzdálenost mezi těmito dvěma objekty. Jestliže je vzdálenost menší nebo rovna 600 jednotkám, začne se zjišťovat, zdali je objekt označen jedním z výše definovaných tagů. Podle toho, o jaký tag se jedná a jaká osmice snímků je momentálně zpracovávána, se objekt buď ve scéně skryje nebo naopak zobrazí.

Ve výsledku si sám vývojář může určit, který objekt bude viditelný z kterého úhlu. Tímto způsobem se dají vytvářet zajímavé herní prvky.

Využití

Ve hře se tato mechanika používá např. pro zobrazení textu. Hráč je takto zábavným a inovativním způsobem poučen například o cíli hry. Dále se tato mechanika dá použít pro případné skryté rady a nápovědy.

4.10 Hra s čísly

V této mini hře se hráč musí rychle zorientovat a přiřadit správná čísla na správná místa. V prostoru se pohybuje čtveřice různě barevných číslic, které se v intervalu 25 sekund náhodně mění. Změnou intervalu se dá docílit zvýšení či snížení náročnosti. Tyto číslice využívají vlastnost popsanou v kapitole 4.9. Každá ze čtyř číslic je tedy na displeji vidět z jiného úhlu. Hráč musí zapsat všechny čtyři vznášející se číslice správně, načež je odměněn jedním kouskem žebříku. Na obrázku 4.10 je ukázka této mini hry v editoru.

Obrázek 4.10: Záběr mini hry s čísly v editoru

Pohyb čísel je zajištěn pomocí funkce `Timeline()`, která se opakuje v závislosti na předepsané funkci. Hned potom se využívá funkce `MoveComponentTo()`, která posouvá číslice po osách X,Y a Z. Vzdálenost, o kterou se číslice posune, se náhodně vybírá z předem stanoveného intervalu. Číslice se také otáčí kolem své osy, aby byly pro hráče hůře čitelné. Tento cyklus se stále opakuje, díky čemuž se číslice nikdy nezastaví, a tak je těžké zjistit, jakou hodnotu zrovna zobrazuje.

Kapitola 5

Uživatelské testování

Tato kapitola se zabývá statistikami a výsledky, pramenícími z testování uživateli. K získání důležitých dat a zpětné vazby byl vytvořen dotazník. Dotazník byl vytvořen pomocí platformy Google Forms.

5.1 Testeři

Testování se celkově zúčastnili 4 testeři. Žádný z nich neměl doposud žádné zkušenosti s holografickým displejem. Těmto testerům bylo v rozmezí od 14 do 30 let.

5.2 Průběh testování

Při začátku testování byl každý tester pouze seznámen s ovládáním hry a cílem hry, samotná hra následně proběhla bez dalšího zasahování vývojářem. Cílem bylo vytvořit herní prostředí podobající se starším herním titulům, kdy je hráč nucen použít metodu pokus-omyl.

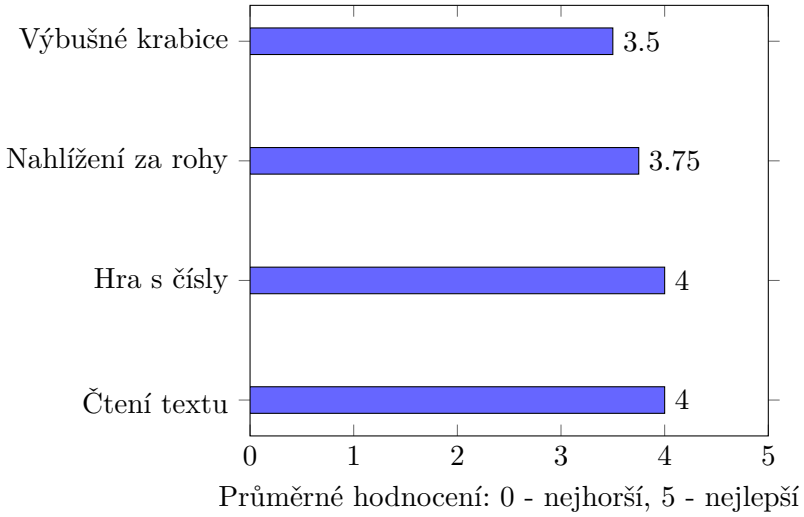
První problém, který se při testování vyskytl, bylo clippování kamery. Někteří testeři se dostali do úhlů, při kterých dokázali vidět skrze zdi. Tento problém byl vyřešen pomocí přidání kolizních boxů do modulů, což umožnilo eliminovat podobné situace. Následně byl změněn pozorovací úhel hráče, čímž se vyřešily skoro všechny problémy tohoto typu.

Další problém, který se pomocí testování odhalil byla chyba s nepřáteli. V určitých situacích se stalo, že nepřítel spatřil hráče i přes zeď. Tato chyba se opravila pomocí změny parametrů pro hledání hráče v třídě nepřítele.

Někteří testeři měli v určitých částech hry problém, protože nevěděli, co se po nich ve hře zrovna vyžaduje. Nicméně se přizpůsobili tomuto stylu a začali přemýšlet jinak. Po dohrání hry byl každým testerem vyplněný dotazník. V dotazníku byly otázky mířeny na provedení jednotlivých mechanik a samotný koncept vytvoření hry na tento kus hardware.

Na grafu 5.1 lze vidět, jak se liší hodnocení jednotlivých mechanik. Nejhorší hodnocení má mini hra s výbušnými krabicemi. To je do jisté míry zapříčiněno chybějícími pokyny pro interakci s touto mini hrou. Naopak nejlepší hodnocení má mechanika čtení textu a hra s čísly.

Jedna z otázek v dotazníku se zabývala konceptem hraní her na holografickém displeji – zájem a styl provedení. V první otázce „Jak zajímavý je koncept hraní her na holografickém displeji?“ odpověděli tři respondenti bodovým ohodnocením 5 a jeden hodnocením 3 (5 bodů je nejvyšší hranice). Naopak u druhé otázky „Jak se Vám líbila částečná realizace této myšlenky?“ se odpovědi respondentů rozcházely. Nejhorší hodnocení byly 3 body, dále



Obrázek 5.1: Průzkum herních mechanik

dvakrát 4 body a jednou 5 bodů. Pro některé testery bylo tempo hry moc pomalé, ale pro některé naopak adekvátní, vzhledem k povaze displeje. Jednomu testerovi bylo po tomto kratším demu nevolno, které přirovnal k podobnému stavu při využívání virtuální reality. Zajímavé je, že nejmladší respondent hned v několika otázkách hodnotil nejmenším počtem. Tento nápad na hru respondentovi připadal zajímavý, nicméně herní provedení mělo pomalé tempo a postrádalo prvek akce.

Výsledky testování

Z testování vyplývá, že tento kus hardware není pro každého a někteří hráči stále preferují hraní her na klasickém monitoru. Pomocí testování se podařilo objevit několik situací, ve kterých docházelo k chybnému chování aplikace, a které se následně podařilo eliminovat. Herní mechaniky vytvořeny v rámci této práce jsou funkční a uživatelům se líbí. Holografický displej rovněž otevírá nové možnosti hraní počítačových her a přidává do hraní nový aspekt.

Kapitola 6

Závěr

Cílem práce bylo vytvořit herní demo aplikaci, která inovativně využívá chování holografického displeje Looking Glass. Nejdůležitější částí bylo vytvořit a implementovat herní mechaniky, které fungují pouze na holografickém displeji. Jelikož se úrovně generují, bylo nutné se seznámit se základy procedurálního generování. Bylo také potřeba si nastudovat postupy a pravidla, které se používají při generování podzemních chodeb. V neposlední řadě bylo nutné se seznámit s vývojovým prostředím Unreal Engine 4 a jeho nástroji.

Výsledkem této práce je demo aplikace, obsahující jednu herní úroveň, ve které jsou obsaženy různé herní mechaniky spojené s holografickým displejem. Součástí výsledku je také procedurální generátor, který umožňuje generování dalších úrovní. Velkou roli hrálo při vývoji uživatelské testování, díky kterému se odhalilo nemalé množství chyb. V budoucnosti by se daly do hry přidat nové herní mechaniky a také by se mohl rozšířit generátor úrovní. Ten by například mohl generovat úrovně bez jakéhokoliv zásahu vývojáře, rovnou připravené ke hraní. Mezi novými mechanikami by mohlo být například hledání schovaných objektů v úrovni (viditelných pod různými úhly) a následná interakce s nimi.

Holografický displej je v mnoha směrech úplně jiný než klasické displeje, což je zřejmé z tohoto herního dema. Hlavní nevýhodou je rozostřenost a špatná plynulost herního zážitku. Překonání těchto dvou nedostatků je nezbytné pro úspěch v dnešním velice konkurenčním herním průmyslu.

V rámci této práce jsem získal spoustu nových zkušeností při práci s herním enginem Unreal Engine 4 a vytvářením modulárních modelů. Dozvěděl jsem se nové informace o fungování blueprintů a možnostech skriptování v UE4. Díky této práci jsem měl také jedinečnou možnost se seznámit a pracovat s holografickým displejem Looking Glass, který bych jinak neměl možnost otestovat. Jedná se o zajímavý kus hardware, který má potenciál sehrát velkou roli v budoucím herním průmyslu.

Literatura

[1] Drozina, A. a Orehovacki, T. Creating a tabletop game prototype in unreal engine 4. In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2018, s. 1568–1573. DOI: 10.23919/MIPRO.2018.8400282.

[2] Epic Games, I. *Introduction to Blueprints* [online]. 2020 [cit. 2020-11-11]. Dostupné z: https://docs.unrealengine.com/en-US/Engine/Blueprints/GettingStarted/index.html.

[3] Guery, J., Hess, M. a Mathys, A. PHOTOGRAMMETRY. In: *Digital Techniques for Documenting and Preserving Cultural Heritage*. Kalamazoo; Bradford: Arc Humanities Press, 2017, s. 229. ISBN 9781942401346.

[4] Kontogianni, G. a Georgopoulos, A. EXPLOITING TEXTURED 3D MODELS FOR DEVELOPING SERIOUS GAMES. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*. 2015, sv. 40.

[5] Lagae, A., Lefebvre, S., Cook, R., Derose, T., Drettakis, G. et al. A Survey of Procedural Noise Functions. *Computer Graphics Forum*. Prosinec 2010, sv. 29. DOI: 10.1111/j.1467-8659.2010.01827.x.

[6] Looking Glass Factory, I. *The Looking Glass User Guide* [online]. 2020 [cit. 2020-11-11]. Dostupné z: https://docs.lookingglassfactory.com/.

[7] Perigner, P. *Modelování a simulace: IMS Studijní opora* [online]. Fakulta informačních technologií VUT v Brně, 2012 [cit. 2020-12-26]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIMS-IT%2Ftexts%2Fopora-ims.pdf.

[8] Prusinkiewicz, A. L. P., Lindenmayer, A., Hanan, J. S., Fracchia, F. D. a Fowler, D. *The Algorithmic Beauty of Plants With*. 1990 [cit. 2020-12-26].

[9] Smelik, R., Tutenel, T., Bidarra, R. a Benes, B. A Survey on Procedural Modeling for Virtual Worlds. *Computer Graphics Forum*. Leden 2014, sv. 33. DOI: 10.1111/cgf.12276.

[10] Software, P. *What Are the Most Popular Game Engines?* [online]. Perforce Software, 2020 [cit. 2020-12-26]. Dostupné z: https://www.perforce.com/blog/vcs/most-popular-game-engines.

[11] van der Linden, R., Lopes, R. a Bidarra, R. Procedural Generation of Dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*. 2014, sv. 6, č. 1, s. 78–89. DOI: 10.1109/TCIAIG.2013.2290371.

[12] Viana, B. a Dos Santos, S. A Survey of Procedural Dungeon Generation. In:. IEEE Computer Society, 2019, 2019-, s. 29–38. ISBN 9781728146379.