UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2017 Jan Houžvička

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Tvorba assetů pro hru pro herní engine Unity 3D

Jan Houžvička

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Houžvička**
Osobní číslo:	**I14100**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Tvorba assetů pro hru pro herní engine Unity 3D**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Téma této práce je zaměřeno na vytváření tzv. assetů (3D modelů, textur, zvuků a dalších) pro herní engine Unity 3D. Tyto assety budou poté dále použity pro tvorbu prostředí a grafického rozhraní v multiplayerové počítačové hře ve stylu známé hry Bomberman.

V teoretické části práce bude uveden návrh vzhledu modelů, textur a grafického uživatelského rozhraní.
V praktické části práce bude popsána tvorba assetů za pomoci profesionálních programů z oblasti počítačového 3D modelování a animací, 3D sochařství nebo tvorby textur a zvuků, které jsou často používány ve velkých herních studiích.

Výsledkem budou vytvořené assety připravené pro okamžité použití v enginu Unity 3D.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

***DILLE, Flint. The ultimate guide to video game writing and design. New York: Watson-Guptill Publications, 2007. ISBN 158065066X.**
***PECINOVSKÝ, Rudolf. OOP: Naučte se myslet a programovat objektově. Brno: Computer Press, a.s., 2010. ISBN 978-80-251-2126-9.**
***KNUTH, D. E.: Umění programování - Základní algoritmy, Brno, Computer Press 2008, ISBN: 978-80-251-2025-5.**
***WRÓBLEWSKI, Piotr. Algoritmy: datové struktury a programovací techniky. Vyd. 1. Překlad Marek Michalek, Bogdan Kiszka. Brno: Computer Press, 2004, 351 s. ISBN 80-251-0343-9.**
***KEOGH, Jim; DAVIDSON, Ken. Datové struktury bez předchozích znalostí : průvodce pro samouky. Vyd 1. Brno : Computer Press, 2006. 223 s. ISBN 80-251-0689-6.**

Vedoucí bakalářské práce: **Ing. Josef Brožek**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2016**
Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 4. 5. 2017

Jan Houžvička

Poděkování

Na tomto místě bych rád poděkoval celé své rodině a přítelkyni za podporu dodávanou nejen při tvorbě této práce, ale během celého bakalářského studia. V neposlední řadě také děkuji vedoucímu práce, panu Ing. Josefu Brožkovi.

Anotace

Tato bakalářská práce se zabývá tvorbou assetů především vizuálního charakteru, které lze využít pro tvorbu počítačové hry v herním enginu Unity. Práce nejdříve představí použitý profesionální software a dále čtenáři vysvětlí principy a techniky tvorby obsahu pro počítačové hry a s nimi spojené pojmy, přičemž důraz je kladen také na šetření výpočetního výkonu. Nakonec je práce věnována praktické tvorbě herních assetů.

Klíčová slova

asset, prefab, herní engine, počítačová grafika, optimalizace

Title

Creating of assets for game created in Unity 3D engine

Annotation

This bachelor thesis deals with the creation of assets mainly of visual character, which can be used to create a computer game in the game engine Unity. First, professional software used in this thesis will be introduced and next, the basic principles and techniques of computer game content creation and related concepts will be explained, with emphasis also being placed on computing performance. Finally, the thesis is dedicated to the practical creation of game assets.

Keywords

asset, prefab, game engine, computer graphics, optimization

Obsah

Seznam ilustrací

Úvod

Počítačové hry zná v dnešní době téměř každý. Způsob jejich vývoje je již ale pro většinu lidí záhadou a počet publikací, které se tímto tématem zabývají je ještě menší. Tato práce si proto dává za úkol čtenáři přiblížit tvorbu assetů, atomických prvků potřebných pro vývoj počítačových her, a jejich následné nasazení v herním enginu Unity.

Nejdříve je vysvětlen pojem asset v kontextu vývoje počítačových her a v souvislosti s ním je také představen pojem prefab. Jelikož oba pojmy ve své obecnosti zahrnují velké množství herního obsahu, který zpravidla není dílem pouze jednoho vývojáře, nýbrž je prací většího kolektivu (herního studia), je tato práce zaměřena především na tvorbu assetů audiovizuálního charakteru.

Dále práce čtenáře seznámí se zástupci profesionálních softwarů, které dnes pro tvorbu assetů používají velká herní studia a stejně tak jsou použity během tvorby assetů pro tuto práci. Uvedeny jsou navíc také výhody a nevýhody jejich použití společně s možnou alternativou.

Cílem následující rozsáhlé části je podrobné teoretické představení nejrůznějších principů, technik, a s nimi spojených pojmů především z oblasti počítačové grafiky, které se při tvorbě assetů pro herní enginy používají. V souvislosti s nimi jsou zde popsány také způsoby a metody optimalizace výpočetního výkonu, na které je kladen velký důraz. Pochopení problematiky probrané v této části čtenáře vybaví základními znalostmi potřebnými pro praktické zvládnutí tvorby herního obsahu.

V poslední rozsáhlejší části je představena stylizace assetů hry pro více hráčů (multiplayerové) na motivy známé hry Bomberman. Následuje jejich výčet a dále ukázky jejich praktické tvorby s uplatněním znalostí získaných v teoretické části této bakalářské práce.

1 Asset

Assety jsou základní kameny herních projektů a jejich použití umožňuje vývojářům doslova poskládat v herním enginu to, jak hra a herní svět vypadá a jak funguje. Jejich charakteristickým znakem je možnost znovupoužitelnosti také v jiných projektech nebo scénách. První, převážně větší část assetů je vytvořena ve speciálních softwarech k tomu určených, ale stejně tak je možné některé assety zhotovit přímo v herním enginu.

Zdroj [1].

Grafické assety

Velké množství assetů je výsledkem práce grafiků a tvoří vizuální složku hry. Patří sem tedy vše, co má hráč možnost zahlédnout na obrazovce.

Prvním a četným zástupcem jsou 2D a 3D modely ztvárňující objekty herního světa jako jsou např. postavy, dopravní prostředky, budovy nebo vegetace, přičemž jsou také často doplňovány animacemi.

Neméně důležité jsou také textury. Jedním z mnoha příkladů jejich použití je tvorba materiálů aplikovaných na modely, čímž slouží k určení toho, jak je model „oblečen“, jak je barevně zpracován, jakými detaily jeho povrch oplývá nebo jak vypadá při dopadu světla na jeho povrch.

Textury své místo najdou také při vytváření tzv. cubemaps, což jsou kolekce šesti textur, z nichž každá odpovídá jednomu z pohledů směrem nahoru, dolů, dopředu, dozadu, vlevo a vpravo. Na jejich základě je poté možné vytvořit např. skybox, což je krychle, jejíž každá vnitřní strana obsahuje jednu z šesti textur. Celá herní scéna se nachází uvnitř této krychle a tímto způsobem je možné ve hře vytvořit např. oblohu.

Dalším příkladem použití textur je tvorba částicových efektů, s jejichž pomocí lze ve hře docílit požadované atmosféry. Jedná se např. o nejrůznější typy kouřů, ohňů, explozí, mlhy atd.

V neposlední řadě je nutná zmínka také o množství ikon, tlačítek a dalších prvků uživatelského rozhraní.

Zdroj [2].

Scripty

Velmi důležitou roli mají ve hře assety ve formě nejrůznějších scriptů, bez nichž by byl herní svět pouze statický a bez života. Tyto scripty jsou například používány pro vytvoření pravidel herního světa a herních mechanismů v podobě např. obchodních nebo soubojových systémů.

Stejně tak je scriptem řízen pohyb hráče na základě vstupů z klávesnice, myši a dalších periferií určených k ovládání počítačových her.

Další rozsáhlou oblastí řízenou scripty je chování umělé inteligence, tedy schopnost dalších postav reagovat na chování hráče nebo na jiné události v herním světě.

Zvukové assety

V neposlední řadě existují také assety zvukové, které jsou velmi důležité z hlediska výsledné uvěřitelnosti herního světa a jeho atmosféry. Své uplatnění tedy najdou ve formě dabingu postav, hudby nebo audio efektů.

1.1 Assety pro tuto práci

V dnešních velkých herních studiích již bývá pravidlem práci na různých typech assetů pevně oddělovat. Např. asset ve formě hlavní postavy může projít rukou hned několika grafiků a animátorů, než je kompletně dokončen. Výjimkou jsou pouze menší nezávislá studia, která z důvodu rozpočtu a menšího počtu zaměstnanců toto rozdělení práce do značné míry neumožňují. Stále to však znamená, že neexistuje člověk, který by celou hru a všechny assety vytvořil sám, a proto je i tato práce zaměřena na tvorbu především grafického herního obsahu.

1.2 Prefab

Vytvořené assety se v počítačových hrách zpravidla neobjevují samostatně, ale ve spojení s dalšími assety. Např. pro vytvoření plně funkční postavy, která bude hezky vypadat, hýbat se a reagovat na chování hráče, je nutné zhotovit několik různých assetů, které jsou následně spojeny dohromady. Je to např. materiál obsahující potřebné textury, který se dále aplikuje na 3D model postavy. Následně se připojí a nastaví animace a potřebné scripty určující jeho chování.

Takto spojené skupiny ale disponují nedostatkem, který se projeví ve chvíli potřeby vytvoření jejich duplikátu. Pouhé zkopírování by postačilo jen v situaci, kdy by v budoucnu již nedocházelo k žádným úpravám původní nebo zkopírované skupiny assetů. Ve skutečnosti je ale potřeba upravovat různé detaily častá. Pro tyto potřeby herní enginy umožňují vytvořit

prefab, což je speciální asset, do kterého je možné pod novým názvem uložit již složenou skupinu assetů. Prefab poté již jen stačí umístit do scény a případně vytvořit jeho duplikáty.

Výhodou používání prefabu a jeho kopií je možnost nastavovat a upravovat vlastnosti jednoho prefabu, přičemž konkrétní změna se ihned projeví na všech jeho kopiích umístěných ve scéně. Další výhodou, kterou přebírá od assetu je možnost znovupoužitelnosti v rámci více scén nebo projektů.

Zdroj [3].

2 Použitý software

Tato část představuje software použitý pro tvorbu assetů uvedených v této práci, uvádí také důvod jejich použití a možné alternativy.

2.1 Platforma Unity

Unity je nástroj používaný k tvorbě 2D a 3D her pro dnes již velké množství platforem. Nejvíce se však používá pro vývoj her pro mobilní telefony, osobní počítače, herní konzole nebo web.

Pro začátečníka je nejvhodnější volbou, protože má obrovskou a ochotnou komunitu, výbornou dokumentaci, a i ve své bezplatné verzi vývojáři zpřístupňuje všechny své funkce.

Důvodem použití je výuka práce s tímto enginem na této fakultě a již zmíněná silná komunita. Alternativou však může být také známý CRYENGINE nebo Unreal Engine.

Zdroj [4].

2.2 Autodesk 3ds Max

Autodesk 3ds Max je profesionální 3D software pro tvorbu modelů, animací, aplikaci textur a případně i následné renderování.

V oblasti počítačové grafiky a tvorby počítačových her je dnes velmi rozšířený. S každou novou verzí se rozrůstá jeho již bohatá nabídka modelovacích a animačních nástrojů a pod studentskou licencí ho je možné používat po 3 roky zdarma.

Tento software je použit z důvodu jeho znalosti ze střední školy, studentské licence a možnosti jednoduchého exportu hotového modelu do dalších aplikací od společnosti Autodesk. Podobným často používaným softwarem je Autodesk Maya nebo bezplatně dostupný Blender.

Zdroj [5].

2.3 Autodesk Mudbox

Autodesk Mudbox je dalším zástupcem grafického softwaru. V tomto případě se ale jedná o nástroj pro tzv. digitální malbu a sochaření. Umožňuje pohodlně a jednoduše přímo na 3D model rýt nebo obtisknout detaily, které se špatně modelují jako jsou např. vrásky nebo škrábance. Rovněž také nabízí možnost přímo na model kreslit a následně vygenerovat příslušnou texturu.

Mudbox je často používaný pro vývoj velkých počítačových her a pro studenty je jeho používání zdarma po dobu 3 let, což je jedním z důvodů jeho použití. Tím dalším je skutečnost,

že se jedná o velice efektivní nástroj pro tvorbu detailních objektů a bez problémů pracuje s ostatními produkty od společnosti Autodesk.

Případnou náhradou může být software ZBrush, který ale nenabízí žádnou bezplatnou licenci.

Zdroj [6].

2.4 Adobe Photoshop CC

Photoshop je nejznámější profesionální grafický editor určený pro práci s rastrovou grafikou.

V rámci tvorby assetů je používán k tvorbě a úpravě textur nebo pro design prvků uživatelského rozhraní. Díky své oblíbenosti, funkcím a komunitě, která tvoří velké množství návodů, již není osvojení základních postupů složité, a tak jediným důvodem, pro použití jiného editoru může být pouze cena licence.

Důvodem použití je výuka práce s tímto editorem v rámci předmětu vyučovaného na této fakultě a rovněž fakt, že je ve svém oboru nejlepší. Alternativou ale může být např. volně dostupný editor Gimp.

Zdroj [7].

2.5 Microsoft Visual Studio

Microsoft Visual Studio je vývojové prostředí od společnosti Microsoft pro vývoj aplikací v programovacích jazycích, kterými jsou například C++ nebo C#.

V této práci je použito pro psaní scriptů v jazyce C#. Důvodem použití je jeho znalost z výuky na této fakultě a s ním spojená vlastnost studentské licence. Výhodou je jeho časté použití při profesionálním vývoji počítačových her a integrované nástroje pro herní engine Unity.

Možnou alternativou je vývojové prostředí MonoDevelop, které je základně spolu enginem Unity dodáváno.

Zdroj [8].

3 Modelování

Modelováním se z hlediska počítačové grafiky rozumí vytváření a formování 3D modelů ve speciálních aplikacích poskytujících množství modifikátorů, které umožňují toto modelování urychlit a zjednodušit. Modelování tedy umožňuje přenést objekt z reálného světa nebo grafikovy fantazie do počítačové podoby, a to často za uplatnění abstrakce, jelikož ne každý detail předlohy modelu je nutné modelováním zpracovat.

Jedním z hlavních hledisek, podle kterých dnes hráči posuzují kvalitu počítačových her je bez pochyby grafické ztvárnění herního světa. Jasným důkazem může být fakt, který ukazuje, jakým způsobem odsuzují hráči v poslední době Ubisoft, jednu z největších herních společností za to, že grafická úroveň vydaných her v některých případech ani zdaleka nedosahuje kvalit v ukázkách, které firma prezentuje měsíce před samotným vydáním hry.

Je tedy patrné, že příprava a kvalita grafických modelů herního světa je nesmírně důležitou kapitolou vývoje celé počítačové hry a tato část práce se věnuje poznatkům potřebným pro správné vytvoření 3D modelů, které mohou být dále zpracovány pro tvorbu assetů pro herní engine Unity.

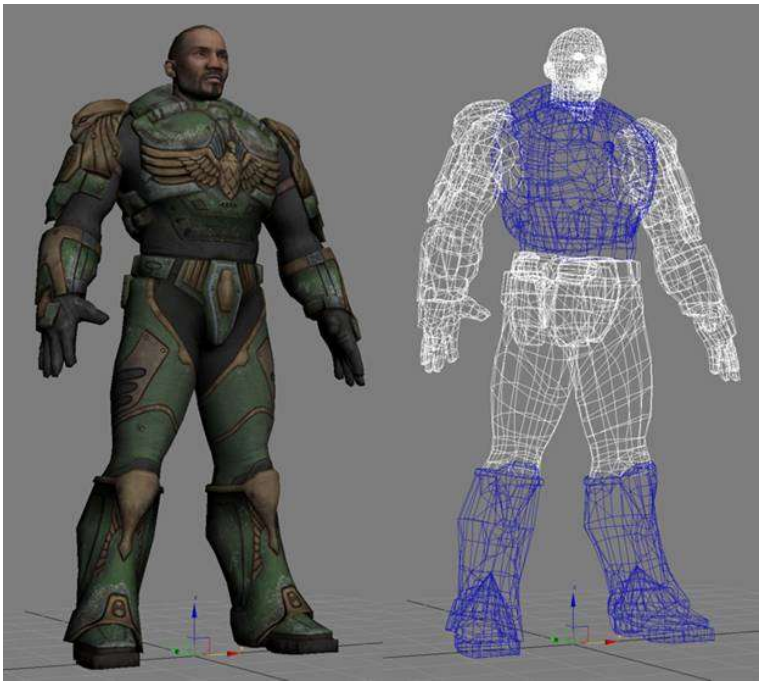
3.1 Mesh

Každý, pro hru zhotovený 3D model, si je možné představit jako síť, kterou grafik vytvaruje tak, aby svým vzhledem co nejvěrněji odpovídala požadavkům. Tato síť se nazývá mesh a je složena z následujících základních částí:

- Vertex, což je pouhý bod, který je umístěn na konkrétních souřadnicích v prostoru,
- hrana, kterou se rozumí spojnice dvou vertexů,
- polygon, který reprezentuje plochu ohraničenou příslušnými hranami (více o polygonech pojednává část 3.2 této práce).

Obecně platí, že jeden vertex může být společný pro více polygonů, jednu hranu můžou sdílet maximálně dva polygony a počet vertexů a hran tvořících polygon si je roven. Stejně tak platí, že čím větší je počet částí (tedy vertexů, polygonů a hran), ze kterých se mesh skládá, tím více místa zabírá v paměti. Pro lepší představu je model a jeho mesh zobrazen na obrázku číslo jedna.

Zdroje [9], [10].



Obrázek 1: Model a jeho mesh[1]

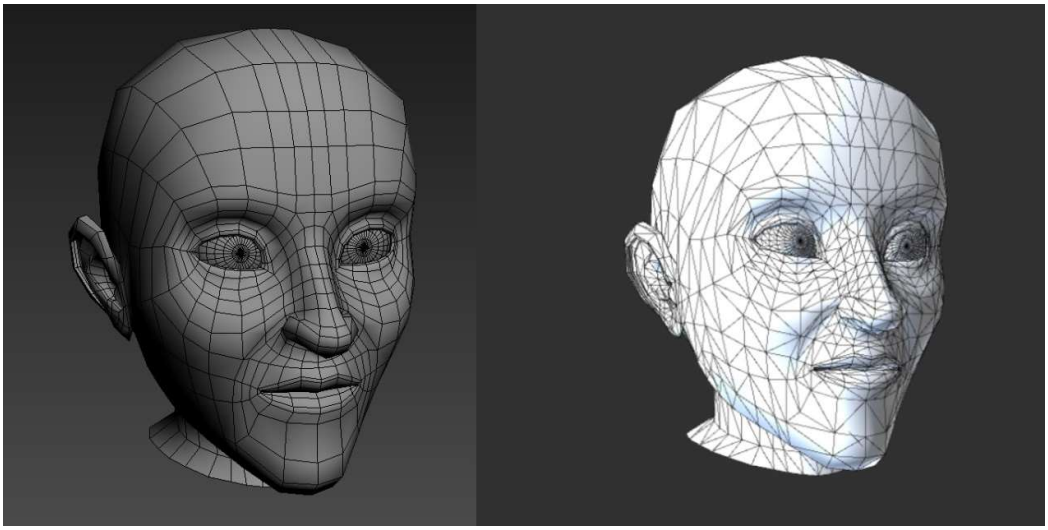
3.2 Polygon

Polygony tvořící model se od sebe odlišují tím, z kolika hran a vertexů jsou utvořeny. Pro modelování se ale nejčastěji používají buď tzv. triangly neboli trojúhelníky nebo čtyřúhelníky, kterým se také říká quady.

Polygon v Unity

Unity pracuje v základu s trojúhelníkovými polygony. Způsob modelování se čtvercovými polygony ale není problémem, jelikož po exportu modelu z 3D modelovacího softwaru do enginu je každý čtverec rozdělen na dva trojúhelníky (vizte obrázek číslo 2, kde část vlevo znázorňuje model vytvořený v softwaru 3ds Max a vpravo se nachází stejný model použitý v enginu Unity). Tento převod umí zajistit sám engine i aplikace jako 3ds Max nebo Maya.

[1] Obrázek převzat z [11].



Obrázek 2: Model vytvořený v softwaru 3ds Max a použitý v enginu Unity

Poměrně důležitým pravidlem je dodržování použití stejného typu polygonů na celém modelu což také zaručuje správnou funkci nástrojů, které grafikovi pomáhají zjednodušit práci (např. nástroj swift loop v softwaru 3ds Max při nedodržování čtvercových polygonů přestává fungovat správně).

Trojúhelníkové polygony jsou na rozdíl od všech ostatních také planární, což znamená, že jakkoli umístěné 3 vertexy tvoří v prostoru vždy rovinu. Takovéto polygony se grafickým kartám dobře a rychle vykreslují.

Zdroj [9].

3.3 Techniky modelování

Techniky modelování určují, jakým způsobem je model vytvářen. Její výběr záleží na složitosti a tvaru požadovaného modelu, ale možné samozřejmě jejich kombinování. Správný výběr dokáže grafikovi ulehčit velké množství práce.

Modifikátory zmíněné v této části jsou výbavou softwaru Autodesk 3ds Max.

Vytažení 2D předlohy do 3D

Technik, kterými lze rychle a téměř bez práce získat 3D model z 2D předlohy je více. Tou první je technika vytažení 2D tvaru do 3D. Je velmi jednoduchá a spočívá ve vytvoření dvourozměrného půdorysu, který je jednoduše vytažen do určité výšky, čímž vzniká 3D model. Její použití je efektivní především ve chvílích, kdy je 2D půdorys již dostupný např. v podobě importovaného souboru z aplikace Autodesk AutoCAD a jeho vytažení je docíleno pomocí

modifikátoru Extrude. Velmi rychle a bez práce lze tedy pomocí této techniky základní modely staveb připravených pro další úpravy.

V dnešní době je rovněž možné přímo převést předlohu v podobě obrázku na 3D model. Stačí pouze označit jednotlivé části obrázku, které je třeba vymodelovat a aplikace se již o vše postará sama. Kvalita takového modelu však téměř nikdy nebude dostatečná pro přímé použití ve hře a je tedy nutné takový model dále ručně upravit.

Zdroj [12].

3D scan

3D scan je technikou vytvoření modelu skenováním jeho skutečné předlohy z reálného světa. Kromě použití drahých zařízení je pro skenování možné předlohu vyfotit z různých úhlů a aplikace jako např. Autodesk 123D Catch na jejich základě vytvoří model. Na vytvořený model je rovněž aplikována textura, která je na základě těchto fotografií vygenerována. Tato aplikace však dnem 31. 3. 2017 přestává být dostupná a bude nahrazena aplikací ReMake.

Zdroj [13].

Skládání primitivních objektů

Modelování pomocí skládání primitivních objektů další jednoduchou technikou. Dostupné 3D modelovací aplikace obsahují základní objekty (také označované jako primitiva) jako jsou krychle, koule, jehlan atd. již předpřipravené a některé modely se tak s využitím této techniky dají pouze poskládat. Např. stůl nebo židle lze takto vytvořit během minuty.

Zdroj [5].

Modelování sítě mesh

Tato technika již pracuje přímo se sítí mesh a s prvky, ze kterých se skládá (vizte kapitolu 3.1). K vytvoření požadovaného modelu dochází pomocí přesouvání a umisťování prvků sítě a používání modifikátorů jako např. Tesselate, který každý polygon modelu rozdělí na několik menších, čímž dojde ke zjemnění sítě (tím pádem ke zvýšení počtu vertexů, hran a polygonů), což umožňuje její prvky umístit přesněji a model zpracovat detailněji.

Původní mesh, ze které se pomocí této techniky model vytváří může být buď primitivní objekt nebo výsledek jiných modelovacích technik. Např. stolu vytvořenému z primitiv je možné srazit hrany kvůli lepším odleskům světla nebo přidat nerovnosti po odštípnutí kousků dřeva.

Zdroj [5].

Modelování z křivek

Modelovací aplikace umožňují také vytvářet modely, jejichž tvar je popsán křivkami. Takové modely je před použitím v enginu nejprve nutné převést na mesh.

Touto technikou lze některé typy modelů vytvořit ze dvou křivek a použitím modifikátoru Lathe. Jedna z křivek tvoří daný profil, který je následně otočený kolem křivky druhé, která tvoří osu rotace. Vytvoření lahví, váz a jiných válcovitých objektů je tak rychlé a jednoduché.

Zdroj [5].

3.4 Optimalizace výpočetního výkonu při modelování

Již při tvorbě modelu musí grafik myslet na to, kde bude výsledný objekt v herní scéně umístěn. Pokud je možné s jistotou říci, že se např. hráč ve hře ke stromu ani nepřiblíží, protože bude umístěn na vysoké skále, je naprosto zbytečným a špatným krokem vytvářet model detailně propracovaný, protože stejně jako v reálnem světě, ani v tom počítačovém si hráč nevšimne toho, že grafik skvěle vypracoval každou větev. Hlavním problémem by ale v takovém případě bylo příliš velké množství polygonů a tím zbytečně vysoký dopad na výkon daného hardwaru.

Dalším důležitým kritériem je tedy také zařízení, pro které je hra vyvíjena. Samozřejmě platí, že s rostoucím výkonem roste také možnost, do jaké míry detailu je možné modely vytvářet. I s dnešními počítači a herními konzolemi mají grafici bohužel ruce poměrně svázané a existuje mnoho způsobů, kterými hry a počítačová grafika obecně podvádí oči hráče proto, aby jen sebemenší kousek výkonu mohl být využit jinde. Dále je uvedeno několik obecných zásad pro tvorbu modelů, které umožňují snížit výsledný dopad na výpočetní výkon.

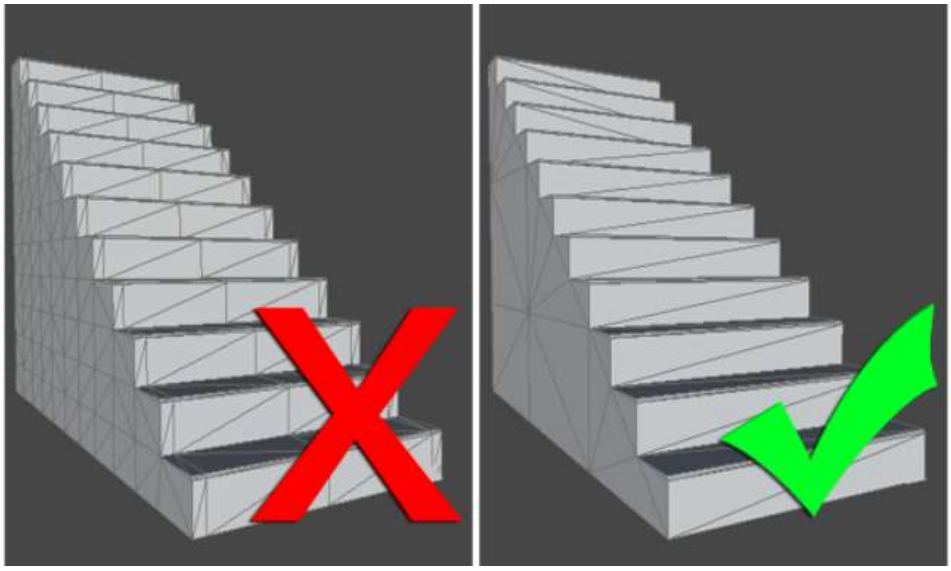
Šetření polygonů

Jednou ze zásad pro šetření výkonu už při modelování je vytváření jednoduchého modelu a přidávání detailů (polygonů) jen tam, kde je to potřebné a kde si toho hráč všimne. Při dodržování této zásady dochází k výskytu zbytečných polygonů, které jsou na modelu navíc, jen minimálně.

V opačném případě může dojít např. k situaci, kterou znázorňuje obrázek číslo 3. Z hlediska hráče se jedná o dva totožně vypadající modely, protože spojnice znázorňující hrany mezi vertexy pochopitelně ve hře zobrazeny nejsou. Z hlediska enginu a především výkonu, který je potřebný pro jejich zobrazení je však patrný velký rozdíl, který je způsoben právě použitím velkého počtu zbytečných polygonů. Oproti pouhým 156 polygonům vpravo, je model vlevo tvořen 726 polygony. 570 polygonů navíc je u modelu vlevo tedy naprosto zbytečných

a v situaci, kdy dochází k použití stejného modelu ve hře hned několikrát, by mohl být negativní dopad na výkon již patrný.

Zdroj [14].



Obrázek 3: Ukázka modelu se zbytečnými polygony a bez nich[2]

Level of Detail

Obecně platí zásada, že objekty, které jsou od hráče vzdáleny by neměly oplývat detaily, kterých si nelze všimnout. Problém ovšem nastává v situaci, kdy se hráč dostane k objektu tak blízko, že se nízká kvalita modelu stává patrnou. Řešení tohoto problému přináší optimalizační technika nazývaná Level of Detail. Tento termín nejčastěji označován zkratkou LOD a do češtiny přeložen jako úroveň detailu, představuje jedno z nejpoužívanějších paradigmat pro optimalizaci a dnes zřejmě neexistuje hra, která by ho nevyužívala.

Princip, na kterém tato technika stojí je velmi prostý. Jako první grafik vytvoří nejdetailnější model, který hráč vidí ve chvíli, kdy stojí přímo před ním. Následně dojde k vytvoření duplikátu tohoto modelu a odstranění hran nebo celých částí a tím pádem i snížení počtu polygonů. Stejný postup je možné zopakovat ještě jednou nebo dvakrát. Nepsaným pravidlem je snižování počtu polygonů přibližně o 50 procent vůči předchozí úrovni, pokud je to možné. Výsledkem je tedy několik variant stejného modelu, z nichž každá má jinou úroveň detailu a tím pádem i jiný dopad na výkon. Herní engine poté na základě vzdálenosti kamery jednotlivé verze modelu přepíná a tím je výkon ušetřen.

[2] Obrázek převzat z [14].

Důležité je, aby si hráč přepínání modelů všímal co nejméně, což je opět prací grafika. Při snižování detailů je nutné dbát na zachování toho, co je hlavní pro objekty nejen herní, ale i skutečné, a tím je silueta. Např. při vzdalování se od domu si hráč jistě všimne toho, že zmizel celý komín, ale naopak úbytek polygonů na střešních taškách nemusí být vůbec patrný.

Pro větší přehled je taktéž pravidlem pojmenovávat jednotlivé modely s koncovkou _LOD0, _LOD1, _LOD2 atd. pro příslušný počet úrovní. Úroveň je v názvu zastoupena číslem a platí, že čím nižší číslo, tím vyšší úroveň detailů. Při dodržování tohoto postupu také uživateli herní engine Unity usnadňuje práci, protože za něj sám vytváří a nastavuje tzv. LOD Group, což je komponenta zodpovědná za již zmíněné přepínaní.

Zdroje [15], [16].

Tessellation

Tessellation (dále jen česky teselace) je proces, pomocí kterého grafické karty vylepšují vzhled objektů ve scéně, které je potřeba vykreslit na obrazovku.

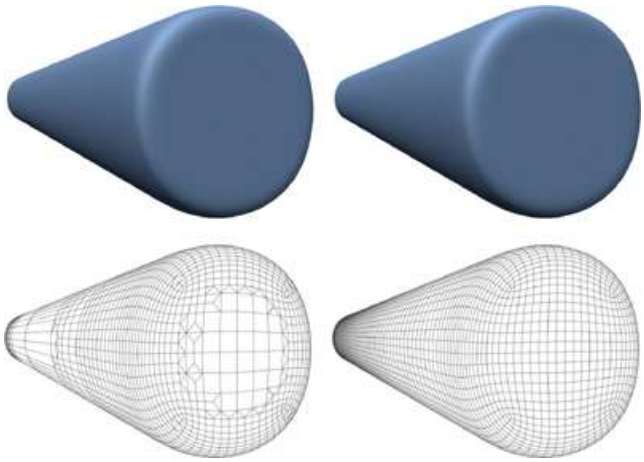
Kvalita modelů pro počítačové hry spočívající v míře detailů bývá nižší kvůli dopadu na výkon, hlavně na velikost, kterou model zabírá v paměti. Z tohoto důvodu pak takový model v některých místech působí hranatě. Dnešní grafické karty však právě pomocí teselace tuto hranatost odstraňují tak, že mesh samy zjemňují rozdělením polygonů na menší. Až poté dochází k vykreslení modelu, jehož povrch je kulatější. Následující obrázek číslo 4 pro lepší představu znázorňuje vzhled modelu před použitím teselace (vlevo) a po něm (vpravo).

Obrázek 4: Porovnání modelu před teselací a po ní[3]

Kromě teselace, která zjemňuje všechny polygony stejně, existuje také adaptivní teselace, která více upravuje ty oblasti modelu, u kterých je to v danou chvíli potřeba. Na obrázku číslo 5 je

[3] Obrázek převzat z [17].

na válec vlevo použita adaptivní teselace a na válec vpravo teselace klasická. Po vykreslení oba vypadají stejně, ale rozdílu si lze všimnout v úpravě sítě mesh, kdy v případě levého válce jsou více rozdělovány polygony tvořící siluetu nebo ležící v místech hrany od které se odráží světlo.



Obrázek 5: Porovnání adaptivní a klasické teselace[4]

Samozřejmé je, že nároky na výkon se s použitím teselace zvyšují. Dopad na využití paměti a dobu vykreslování modelu je však stále mnohem menší než kdyby model, který svou kvalitou odpovídá výsledku po aplikování teselace, byl použit ve hře přímo.

Své uplatnění teselace najde např. při vykreslování členitého zemského povrchu nebo obličejů postav.

Výhodou použití teselace je skutečnost, že se o vše stará samotná grafická karta a grafik má s modelem méně práce. Další výhodou, kterou teselace přináší je možnost používat tzv. displacement mapping, který je popsán v souvislosti s displacement mapou v části 5.3.

Zdroje [17], [18].

3.5 Verze modelu s vysokým a nízkým počtem polygonů

Přestože se při modelování s polygony šetří všude kde je to možné, není pravdou, že by se vývoj her bez do detailu provedených modelů obešel úplně. V každém větším herním studiu je možné nalézt zaměstnance, kteří sedí u počítače vybaveni grafickým tabletem a na modelech důkladně dotváří každou vrásku a další drobnosti za pomoci tzv. softwaru pro digitální sochařinu. Výstupem jejich práce je high poly model, tedy model s vysokým počtem polygonů, na kterém je možné jednotlivé detaily dobře zachytit, ale je naprosto nevhodný pro použití ve hře.

[4] Obrázek převzat z [18].

Jako low poly je možné naopak označit model, který lze ve hře použít a svou kvalitou odpovídá objektu s nejvyšší úrovní detailu, již popsanému v kapitole o LOD.

Důvod nutné koexistence obou typů bude dále probrán v rámci části o normal mapách (vizte 5.3).

Zdroj [5].

3.6 Nastavení jednotek

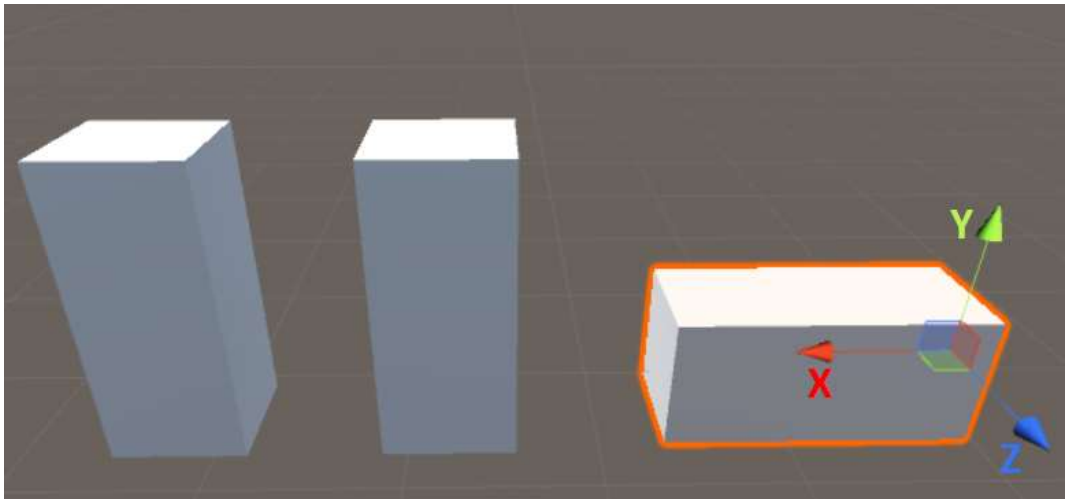
Jednotky souřadnicového systému v aplikaci 3ds Max jsou při jeho prvním spuštění nastaveny jako bezparametrické a v Unity jedna jednotka odpovídá jednomu metru. Před započetím jakékoliv práce je tedy užitečné jednotky obou souřadnicových systémů nastavit tak, aby výsledný model v enginu nebylo potřeba žádným způsobem zvětšovat nebo zmenšovat.

Např. krychle, které jsou v aplikaci 3ds Max vytvořeny s délkou hrany 100 cm a v Unity jsou v některé z os od sebe vzdáleny o jednu jednotku, se přesně dotýkají příslušnými stranami.

3.7 Nastavení pivot pointu

Každý vytvořený model mimo jiné obsahuje pivot point, což je bod určující místo, vůči kterému se provádějí transformace posun, rotace a změna měřítka daného objektu. Rovněž s pivot pointem samotným je v rámci modelovacího softwaru možné tyto transformace provádět, což může např. v případě nastavení jeho rotace vést k problémům. Rotace pivot pointu totiž na pohled nijak nemění rotaci objektu, ke kterému patří a tato změna je patrná pouze v rámci souřadnicového systému.

Tento problém nejlépe znázorňuje následující obrázek z enginu Unity, na kterém jsou umístěny tři kvádry. Pivot point kvádru vlevo má rotaci správnou. V případě prostředního je rotace pivot pointu v ose Z změněna o 90 stupňů, ale engine v případě ručního umisťování modelu do scény o této rotaci ví a tak objekt sám správně otočí. Poslední kvádr má pivot point nastaven stejně jako v předchozím případě, ale tentokrát byl do scény umístěn scriptem. Programátor tohoto scriptu ovšem předpokládá, že grafik udělal svoji práci dobře a že rotace pivot pointu je v pořádku, a tudíž ji nijak neupravuje (ve všech osách ji nastaví na 0 stupňů). Chyba se ale projeví ihned po spuštění hry a kvádr ve scéně nebude správně natočen.



Obrázek 6: Ukázka modelů s různým nastavením pivot pointu

Také poloha pivot pointu je důležitá a při špatném nastavení může při umístění objektu do hry způsobit problémy. Oprava takových chyb sice není složitá, ale vývoj zbytečně zpomaluje.

3.8 Export modelu do herního enginu

Nejen na konci, nýbrž také v průběhu celé práce je důležité mesh z modelovacího softwaru exportovat do herního enginu za účelem kontroly výsledného vzhledu. Engine Unity podporuje modely v souborových formátech dvojího typu. Prvním typem jsou exportované 3D souborové formáty a tím druhým jsou formáty samotných aplikací, ve kterých byly modely zhotoveny.

Nejpoužívanější souborovým formátem spadajícím do první skupiny má příponu .fbx. Jeho výhodou je malá velikost, protože exportována jsou pouze nejdůležitější data a také skutečnost, že soubor s tímto formátem lze otevřít a upravit i v modelovacím softwaru, ve kterém původně nebyl vytvořen. Naopak nevýhodou je nutnost uchovávání dvou souborů (původní soubor a exportovaný soubor).

Mezi souborové formáty ze druhé kategorie patří např. formát .max (software 3ds Max) nebo .blend (software Blender). Výhodou těchto formátů je možnost uchovávat pouze jeden soubor. Nevýhodou pak je, že takový soubor je silně vázán na aplikaci, ve které byl vytvořen a v jiných ho nelze otevřít. Navíc soubory s těmito formáty zabírají více místa.

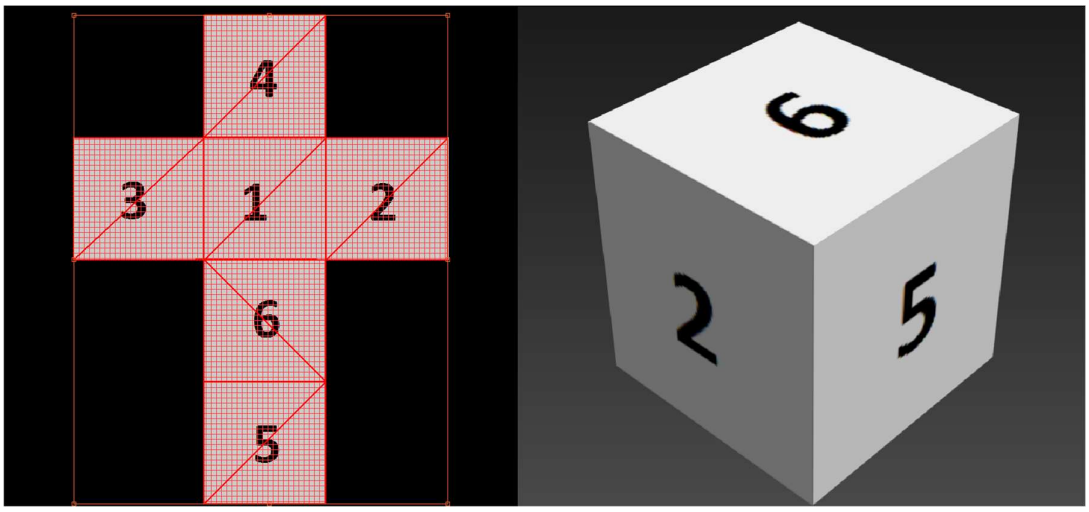
V obou případech je nutné model exportovat nebo uložit do složky Assets (nebo do složek v ní vnořených) v konkrétním projektu. Unity poté sám nově přidané soubory do projektu naimportuje.

Zdroj [19].

4 UV mapování

UV mapování je další fází vývoje grafického assetu, která připravuje model na aplikování textur (textury jsou probrány v rámci kapitoly číslo 5). Jednoduše řečeno se jedná o proces, během kterého je nutné trojrozměrnou síť mesh daného modelu rozbalit do dvourozměrné podoby tak, aby se žádné části sítě nepřekrývaly. Každý model si své mapování uchovává a plocha uvnitř takto rozbalené sítě odpovídá místům, která budou na modelu pokryta texturou. Písmena U a V v názvu mají význam os dvourozměrného prostoru, použitých místo X a Y, již spojených s prostorem trojrozměrným.

Pro lepší představu lze uvést příklad s jednoduchou krychlí, který také znázorňuje obrázek číslo sedm. Část obrázku vlevo znázorňuje mapu s rozbalenou sítí a část vpravo namapovanou krychli s aplikovanou jednoduchou texturou.



Obrázek 7: Jednoduchý příklad namapovaného modelu

4.1 UV Seam

Modely většinou nelze namapovat tak, aby byla rozložená síť tvořena pouze jedním kusem. V některém místě se tedy musí síť „rozříznout“ (ne doopravdy, skutečná síť modelu zůstává beze změny) a v tomto místě vzniká UV seam neboli šev. Tyto švy je nutné vhodně umisťovat na taková místa, která jsou na modelu nejméně vidět. Příkladem z reálného světa může být tričko. Neexistuje způsob, jak ho rozložit na stůl, tak aby žádný kus látky neležel na jiném. Za tímto účelem je nutné tričko rozstříhat. Vést střih bude v tomto případě nejlepší v místech skutečných švů než uprostřed trička od krku dolů.

Zdroj [20].

4.2 Projekce

Projekce je způsob, jakým se 3D model promítá na UV mapu. Způsob, kterým je možné projekci provést je manuální nebo automatický. Oba dva způsoby projekce zvládají 3D modelovací aplikace, ale lze využít také speciální nástroje jako např. UV Master.

Zdroj [21].

Manuální projekce

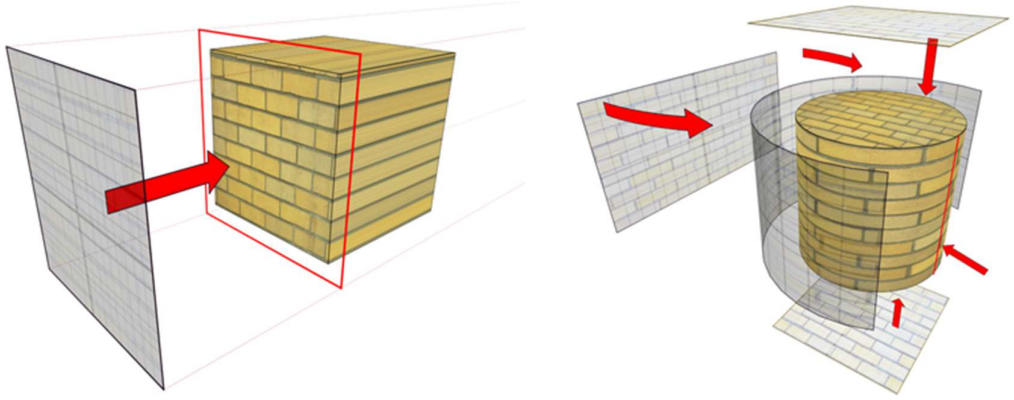
Během manuální projekce grafik musí model mapovat po částech. Nejprve vybere příslušné polygony modelu a poté ne ně aplikuje některý z typů projekce. Jednotlivé kusy namapované sítě je poté vhodné co nejlépe spojit, tak aby pokud je to možné byla tvořena jednou jedinou částí (aby počet švů byl co nejnižší). Proto je tato metoda poměrně náročná na práci i čas.

Typy manuální projekce umožňují promítnout označenou část modelu do mapy pomocí jednoduchých tvarů jako např. plocha, koule nebo válec. Výběr jednoho z nich pak záleží na tom, kterému tvaru se označený výběr modelu podobá nejvíce.

Nejjednodušším příkladem je plošná (nebo také planární) manuální projekce, zobrazená na obrázku číslo osm vlevo, kdy jsou vybrané polygony modelu promítnuty na rovnou plochu.

Dalším příkladem je válcovitá manuální projekce znázorněná na obrázku číslo osm vpravo (na podstavy je použita plošná projekce). V tomto případě jsou polygony pláště promítnuty na stěny válce, které je obklopují a následně je tento válec v jednom místě přerušen švem, aby mohlo dojít k jeho zobrazení na dvourozměrné mapě.

Zdroj [22].



Obrázek 8: Princip planární a válcovité projekce[5]

[5] Obrázek převzat z [22].

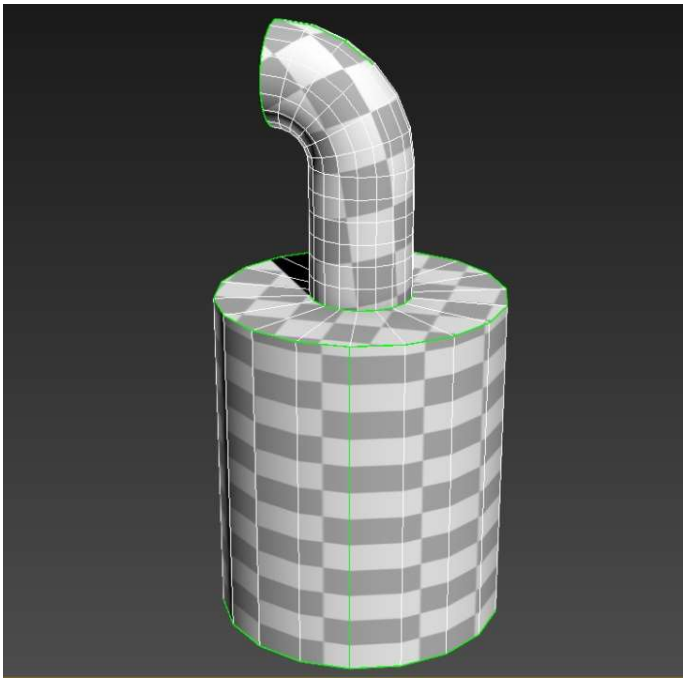
Automatická projekce

Automatická projekce umožňuje grafikovi nechat mapování modelu na aplikaci. Na rozdíl od manuální projekce je v tomto případě mapa vytvořena ihned. Bez zásahu grafika se ovšem ani tento proces neobejde, protože aplikací vygenerovaná mapa může být tvořena až příliš velkým množstvím oddělených částí a pozice UV švů se mohu nacházet na nevhodných místech. Stále se ale jedná o rychlejší variantu.

4.3 Kontrola roztažení namapované sítě

Během mapování je důležité dávat pozor na to, aby namapovaná síť nebyla v jednom z rozměrů roztažena více než je to nutné. Lze tak předejít stejnému roztažení textury aplikované na namapovaný objekt. Za účelem kontroly se na model nejdříve aplikuje speciální textura. Ta má nejčastěji podobu šachovnice nebo vedle se umístěných kruhů, protože je na nich případné roztažení možné pouhým okem odhalit velmi brzy. Pokud je na mapě nebo některé její síť roztažení velmi zřetelné je nutné ji v příslušném směru roztáhnout nebo naopak stáhnout.

Následující obrázek číslo devět znázorňuje objekt s mapou, která je pro spodní válcovitou část v ose U příliš roztažená. Po aplikování kontrolní textury s motivem čtvercové šachovnice se tedy ve výsledku v těchto místech modelu nezobrazují čtverce, ale obdélníky, což je chyba.



Obrázek 9: Příklad modelu s roztaženou UV mapou

5 Texturování

Texturování je další důležitý proces, kterým model během svého vývoje prochází. V aplikacích mimo herní engine dochází k vytváření textur a tyto jsou následně již v enginu uspořádány ve formě materiálu, který je aplikován na konkrétní model.

5.1 Textura

Textura by se dala přirovnat k oblečení, které se aplikuje na model. Jedná se o statický 2D obrázek, který určuje to, jak model vypadá, tedy jakou má barvu, z jakého materiálu je vytvořen, jak jeho povrch reaguje na světlo atd. Všechny tyto informace ale nelze zpracovat v podobě jedné textury, a proto je nutné vytvořit jich hned několik. Kromě svého účelu se od sebe jednotlivé typy textur liší také tím, které barvy používají (např. pouze odstíny šedi). Některé typy textur budou dále probrány.

Z hlediska enginu Unity lze za texturu považovat také video, obrázky prvků uživatelského rozhraní nebo tzv. cookies, které se používají v souvislosti se světly.

Z důvodu použití v herním enginu pro textury platí absence míst osvícených jakýmkoliv zdrojem světla nebo znázornění stínů, protože o osvětlení a stínování se stará sám engine. Textura, která by přímo na sobě nesla osvícení např. od slunce a byla navíc nasvícená také herním světlem, by ve hře nevypadala přirozeně.

Zdroj [23].

5.2 Metody tvorby textur

Textury lze připravovat dvěma různými metodami, které se od sebe liší použitou aplikací a především způsobem tvorby.

2D kreslení textury (2D texture painting)

Metoda 2D kreslení textury spočívá v její přípravě v některém z editorů pro rastrovou grafiku. Jako příklad lze uvést velmi známý Photoshop.

Z příslušné aplikace je nejdříve ve formě obrázku exportována vytvořená UV mapa modelu, na které je znázorněn alespoň obrys jeho namapované sítě. Do míst, která jsou tímto obrysem ohraničena poté grafik ručně umisťuje (např. část fotografie) nebo kreslí to, co se na příslušných místech nakonec zobrazí na modelu.

Zdroj [24].

3D kreslení textury (3D texture painting)

3D kreslení textury je metoda tvorby textury v trojrozměrném prostoru. V softwaru jako např. Mudbox nebo ZBrush stačí otevřít namapovaný model a pomocí různých štětců nebo obtisků kreslit přímo na něj, což se ihned promítne do jeho textury.

V případě softwaru Substance Painter se tato metoda navíc rozšiřuje o další velice užitečnou funkci, kterou zatím konkurenční nástroje nenabízejí. Umožňuje totiž automaticky na model a tím pádem i do textury, přidávat detaily jako např. škrábance jen do míst, na kterých by se na příslušném objektu nacházely i v reálném světě. Tato funkce je založena na principu použití pomyslných částic, které jsou vyslány na objekt. Ty po kontaktu s jeho povrchem zjistí jeho nerovnosti a detaily v příslušných místech dotvoří.

Zdroj [24].

5.3 Typy textur

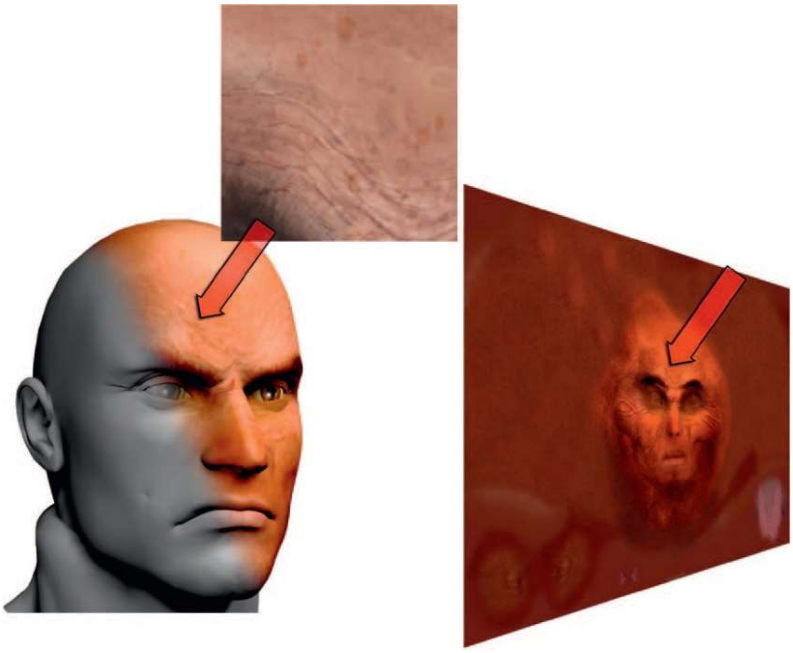
V následující části jsou představeny některé typy textur, které se při tvorbě grafického obsahu počítačových her často používají.

Diffuse map

Diffuse map je jednoduchá textura, která obsahuje informace o barvě povrchu modelu, na který je aplikována. Dále může také ztvárňovat detaily, které není možné jednoduše zpracovat modelováním jako jsou např. škrábance nebo jednotlivé vrásky a póry na kůži (vizte příklad zachycený na obrázku číslo 10). Povrch modelu tedy může být hladký (na výpočetní výkon méně náročný) a některé detaily jsou zpracovány pomocí této textury (tyto detaily je možné vytvořit více uvěřitelné za použití dalších textur jako např. normal map).

Hodnoty difúzních barev, které se pro správné zpracování některých typů materiálů v textuře používají, jsou zpravidla k nalezení v dokumentacích herních enginů (např. kovové materiály mají difúzní barvu velmi tmavou nebo dokonce černou).

Zdroj [25].



Obrázek 10: Příklad diffuse mapy a její aplikace na model[6]

Specular map

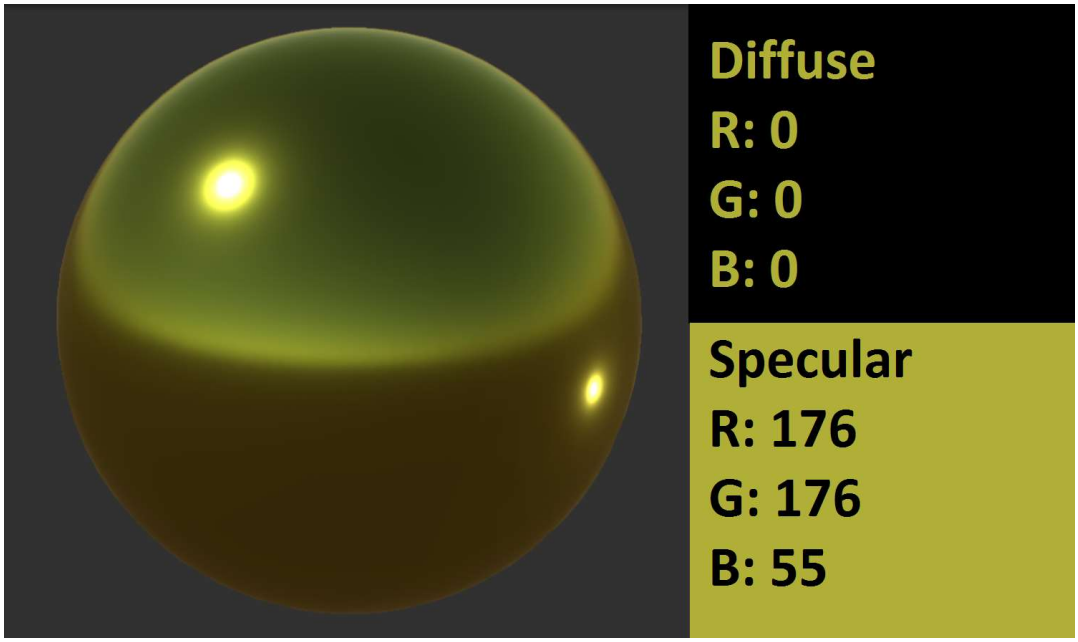
Povrch modelu může být tvořen několika druhy materiálů, z nichž každý odráží rozdílné množství světla, které na něj dopadá. Za účelem dosažení rozdílné odrazivosti povrchů a barvy světla, které se od nich odráží, se v počítačových hrách používá specular mapa.

Při její tvorbě platí pravidlo, že čím tmavší barva se na této textuře nachází, tím méně bude povrch příslušné části modelu odrážet světlo a naopak. Rovněž u většiny materiálů platí, že čím tmavší je barva materiálu v diffuse mapě, tím světlejší je jeho barva ve specular mapě a opačně. Hodnoty barev pro často používané materiály jsou k nalezení v dokumentacích herních enginů (pro kovové materiály jsou to světlé spekulární barvy a tmavší pro materiály nekovové).

Následující obrázek číslo jedenáct znázorňuje imitaci zlatého povrchu. Tento byl vytvořen jednoduše pomocí černé diffuse mapy a specular mapy se „zlatou“ barvou (s hodnotou 176 v červeném kanálu, hodnotou 176 v zeleném kanálu a hodnotou 55 v modrém kanálu) a následně aplikován na osvětlenou kouli. Bez použití této textury by v tomto případě byl objekt pouze černý.

Zdroj [26].

[6] Obrázek převzat z [25].



Obrázek 11: Vliv specular mapy na vzhled objektu

Normal map

Typ textury zvaný normal map slouží v počítačových hrách k vytvoření detailnějších modelů. Tyto detaily jsou ale pouze „podvodem“ pro oko hráče a na síti mesh skutečného modelu k žádným změnám nedochází.

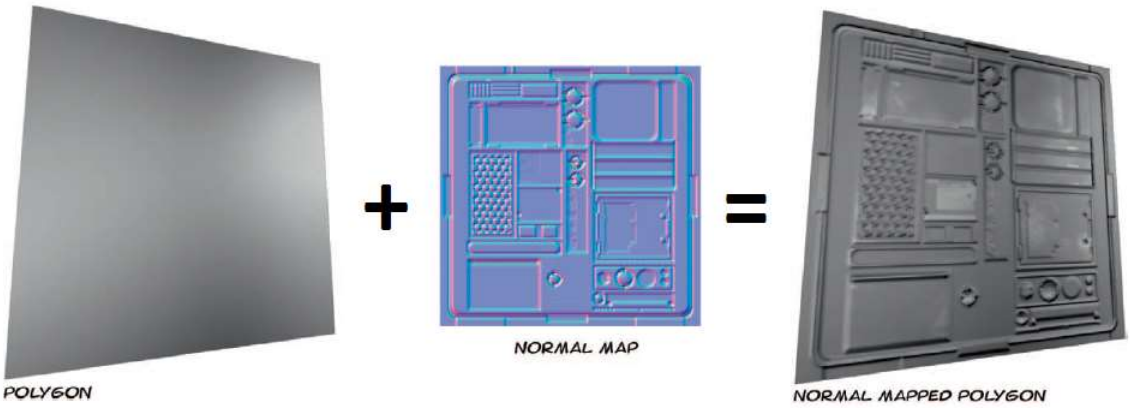
Barva obsažená v každém pixelu (resp. hodnoty barvy v jednotlivých barevných kanálech) této textury uchovává informaci o zdánlivé úpravě povrchu, který se pod ní nachází. Tato úprava spočívá v tom, že světlo, které na povrch modelu dopadá, se díky normal mapě může odrazit jiným směrem. Tím je možné i na rovném povrchu docílit výskytu odlesků a stínů dotvářejících přidané detaily, které by se na něm za normálních okolností neprojevily. Znamená to ale, že tyto detaily se na modelu zobrazí pouze v případě jeho nasvícení.

Tato textura bývá nejčastěji tvořena automaticky pomocí tzv. zapékání z modelu s vysokým počtem polygonů (vizte část 3.5 této práce), jelikož její ruční tvorba by byla velmi náročná. Toto zapékání umožňují 3D modelovací aplikace nebo aplikace pro digitální sochaření a jedná se tedy o typického zástupce textur tvořených pomocí techniky 3D kreslení. Během zapékání dojde k uložení detailů z modelu s vysokým počtem polygonů do normal mapy. Tato se poté aplikuje na model s nízkým počtem polygonů, přičemž detaily jsou s minimální ztrátou kvality zachovány.

V enginu Unity je nutné importovaný obrázek s touto texturou označit nejdříve jako normal mapu, jinak nebude správně plnit svou funkci.

Následující obrázek číslo dvanáct dokazuje, jak markantně dokáže normal mapa ovlivnit výsledný vzhled při její aplikaci na pouhý jeden polygon.

Zdroje [25], [27].



Obrázek 12: Ukázka použití normal mapy[7]

Displacement map

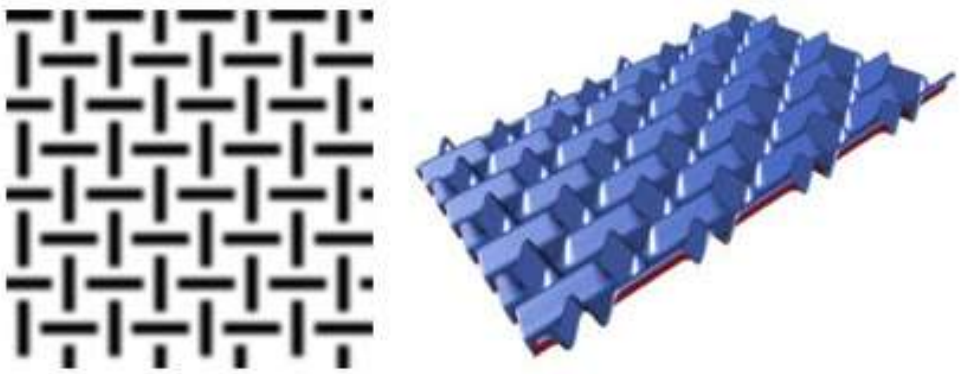
Displacement mapa stejně jako normal mapa dokáže na modelu dotvářet detaily. V případě této textury jsou ale za pomoci displacement mapingu tyto detaily fyzicky umístěny přímo do sítě mesh. Displacement mapping na objekt nejdříve aplikuje teselaci (vizte část 3.4 této práce) a až poté je zjemněná síť mesh na základě této textury upravena. V této textuře je tedy pomocí hodnot udávajících barvu jednotlivých pixelů uchována informace o výšce (kladná nebo záporná), do které bude vertex modelu posunut.

Tuto texturu lze nejlépe vytvořit automaticky podobně jako v případě normal mapy. V takovém případě je nutné vytvořit model s vysokým počtem polygonů obsahující všechny detaily, které budou následně převedeny do podoby displacement mapy. To umožňuje uchovávat v operační paměti pouze model s nízkým počtem polygonů a tuto texturu. O vytvoření všech detailů se poté již sama postará grafická karta.

Na obrázku číslo 13 je v levé části umístěn vzor, který obsahuje displacement mapa a vpravo je možné vidět výsledek její aplikace na povrch, který byl původně pouze rovný.

Zdroje [28], [29].

[7] Obrázek převzat z [25].



Obrázek 13: Ukázka použití displacement mapy[8]

5.4 Shader a materiál

Textury jako takové nelze jednu po druhé aplikovat na objekt. K tomu v herním enginu slouží materiál a s ním spojený shader.

Materiál je z hlediska herního enginu asset, který slouží ke specifikaci shaderu a nastavení jeho parametrů. Vytvořený materiál je poté možné aplikovat na objekty.

Shader je v počítačové grafice program, pro jehož vytvoření se používají jazyky jako Cg nebo HLSL. Takový program na základě různých algoritmů, výpočtů, vrženého světla a také svých parametrů určuje, jak bude vykreslen objekt, na který je aplikován materiál s daným shaderem. Parametry shaderu jsou jednotlivé textury, přičemž jednotlivé shadery se od sebe mohou lišit právě typy textur, se kterými pracují.

Zdroj [30].

5.5 Optimalizace výpočetního výkonu z hlediska textur a vykreslování

I v případě textur a vykreslování objektů existují techniky, které se starají o co nejnižší využití výpočetního výkonu.

Batching

Vykreslení každého objektu umístěného ve scéně probíhá v rámci tzv. draw call, přičemž jejich počet roste s počtem objektů ve scéně. Každý draw call je sám o sobě náročný především na výkon procesoru, a proto by měla být tendence jejich počet redukovat. Za tímto účelem používá herní engine Unity batching, který umožňuje vykreslit větší počet objektů v rámci jednoho draw call.

[8] Obrázek převzat z [17].

Batching exituje ve dvou variantách. První varianta tzv. dynamic batching umožňuje vykreslovat v rámci jednoho draw call pohyblivé objekty, které sdílí stejný materiál. Jedná se o automatickou záležitost, se kterou vývojář nemá žádnou práci. Druhá varianta s názvem static batching umožňuje během jednoho draw call vykreslit ty objekty, které se nehýbají a sdílí stejný materiál. V tomto případě je již nutný zásah vývojáře, který musí příslušné objekty přímo v enginu označit jako statické.

Pro zvýšení počtu objektů, které spolu sdílí stejný materiál je možné použít tzv. texture atlas probraný dále.

Zdroj [31].

Texture atlas

Pouze objekty, které sdílí stejný materiál a tím pádem také textury, mohou být vykresleny v rámci jednoho draw call. Pokud tedy bude ve scéně umístěn automobil a dům, k žádnému ušetření výkonu nedojde. Tento problém řeší použití texture atlasu, což je textura, která by v tomto případě obsahovala celou texturu automobilu i texturu domu (modely automobilu a domu samozřejmě musí být namapovány tak, aby se části UV mapy objektů nepřekrývaly). Taková textura je sice větší a zabírá více místa v paměti, ale je ji možné použít pro vytvoření jen jednoho materiálu a tím pádem je docíleno vykreslení automobilu i domu v rámci jednoho draw call.

Zdroj [25].

MIP mapping

S rostoucí vzdáleností modelu od hráče může být upravována nejen kvalita modelu (vizte část 3.4 této práce věnované technice Level of Detail), ale také kvalita textur. Technika MIP mapping spočívá ve vytvoření několika verzí stejné textury, z nichž každá má menší rozlišení než předchozí a tím pádem zabírá méně místa v paměti a je možné ji rychleji načíst.

Po importu textury do enginu Unity je možné vybrat automatické vytvoření MIP map.

Zdroj [25].

6 Tvorba assetů pro herní engine Unity

Následující část této práce je zaměřena na tvorbu převážně grafických assetů připravených pro nasazení v herním enginu Unity.

6.1 Stylizace assetů

Assety vytvořené v rámci této práce jsou určené pro tvorbu hry pro více hráčů (tzv. multiplayerové) na motivy známého herního titulu Bomberman. Ačkoli původní verze této hry je pouze dvourozměrná a herní prostředí se podobá šachovnici o čtvercových polích (vizte obrázek číslo 14), v případě této práce jsou vytvářeny assety pro 3D počítačovou hru zasazenou do sci-fi prostředí vesmírné lodi. Hráč dění ve hře vnímá z pohledu očí postavy a princip hry spočívá v eliminaci protivníka pomocí bomb s různým dosahem výbuchu, přičemž herní prostředí je plné překážek, které lze výbuchem zničit a uvolnit tak případně cestu do nových části herní mapy. Zničitelné překážky pak mohou ukrývat předměty, které hráči přinášejí různé výhody.

Obrázek 14: Obrázek ze hry Bomberman ve 2D[9]

[9] Obrázek převzat z [32].

6.2 Seznam vytvořených assetů

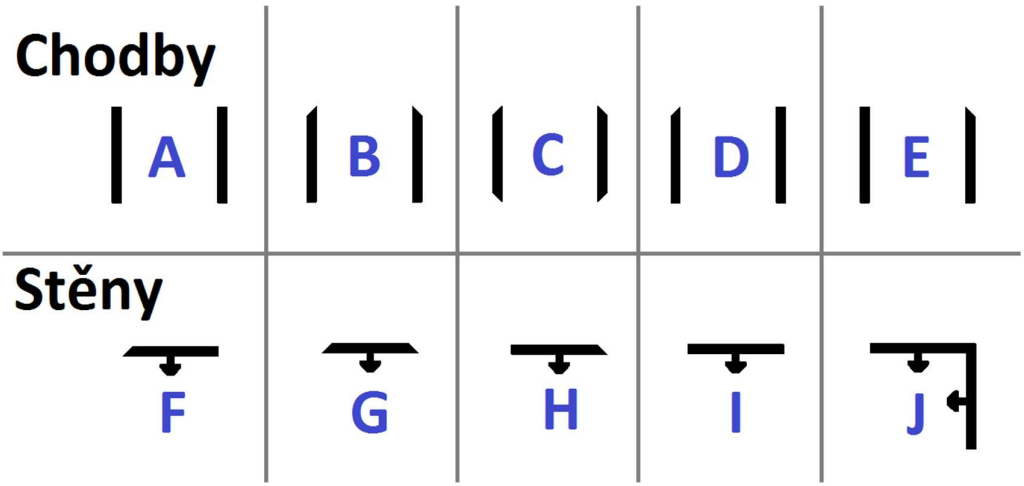
Následující seznam obsahuje všechny assety vytvořené pro tuto práci.

- Deset modelů stěn a chodeb (pro každý model dva další s různou úrovní detailu), tři textury v podobě texture atlasu a společný materiál pro tyto modely,
- model stropu s podlahou (dva další modely s různou úrovní detailu),
- samostatný model podlahy, dvě textury a materiál,
- jedenáct modelů značek pro systém umisťování herního prostředí, scipt zajišťující funkci tohoto systému a společný materiál pro tyto značky,
- model žárovky, materiály pro rozsvícenou a zhasnutou žárovku,
- model překážky (dva další modely s různou úrovní detailu), dvě textury a materiál,
- model bomby (dva další modely s různou úrovní detailu), tři textury, materiál a script pro explozi bomby spouštějící zvuk odpočtu před explozí,
- model hráče, tři textury, materiál, čtyři animace, Animation Controller a script pro ovládání animací,
- šest modelů předmětů zlepšujících schopnosti hráče, animace pro každý model, materiál pro každý model a Animation Controller pro každý model,
- částicový efekt jisker, textura a materiál,
- částicový efekt energetické vlny, textura a materiál,
- částicový efekt výbuchu, script pro spuštění zvuků výbuchu a odstranění překážek v jeho blízkosti,
- částicový efekt kouře, textura a materiál,
- plátno pro uživatelské rozhraní,
- tlačítko pro uživatelské rozhraní, textura jeho pozadí, animace a Animation Controller,
- přechod uživatelského rozhraní mezi scénami, Animation Controller a script pro zobrazení nové scény,
- kamera se scriptem pro ovládání pohybu hráče ve scéně,
- čtyři scény poskládané z vytvořených assetů.

6.3 Tvorba herního prostředí

Herní prostředí, ve kterém se hráči budou pohybovat je ztvárněno ve stylu chodeb a místností uvnitř vesmírné lodi. Vytvořeno je proto několik typů chodeb a stěn, ze kterých je možné výslednou herní mapu poskládat jakýmkoli způsobem. Jednotlivé typy stěn a chodeb se od se liší především tím, jaký další typ na ně může navázat. Např. chodba, za kterou se očekává

odbočka vpravo (chodba typu E na obrázku číslo15), má pravý roh na svém konci zkosený pod úhlem 45 stupňů.



Obrázek 15: Půdorys jednotlivých typů chodeb a stěn

Modelování chodeb a stěn

Označení typů stěn a chodeb převzato z obrázku číslo 15.

Základním předpokladem je fakt, že všechny stěny a chodby budou pravoúhlé, přičemž délka hrany atomického elementu tohoto pravoúhelníku bude:

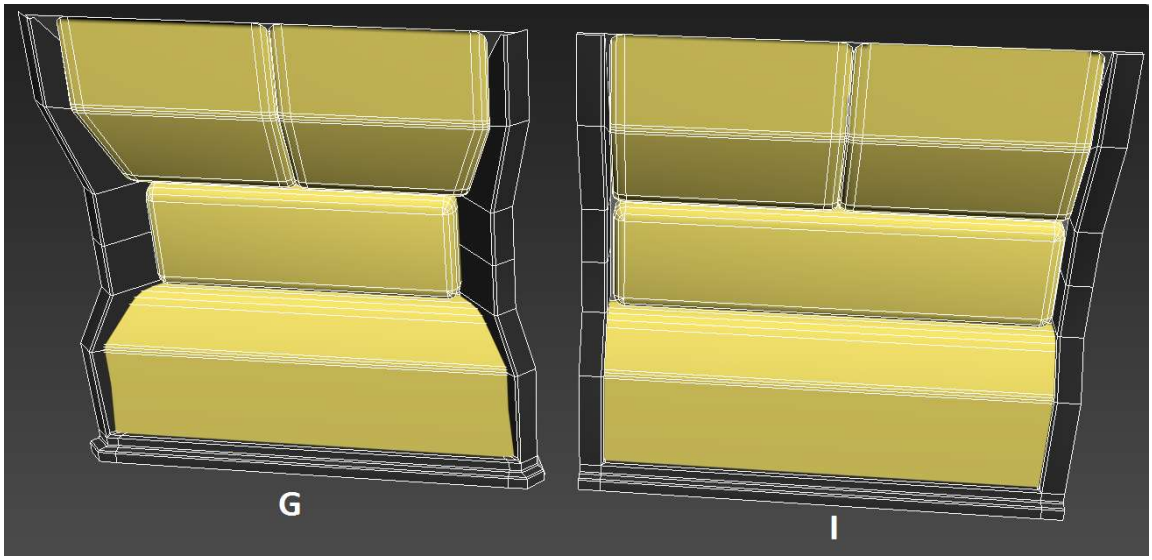
- 100 centimetrů v softwaru 3ds Max,
- jedna jednotka v enginu Unity.

Prvním krokem při modelování je vytvoření kovového rámu stěny, který kopíruje její obvod vlevo, vpravo a dole. Nejjednodušší je začít se stěnou typu I, která nemá zkosené rohy tohoto rámu. Jeho levá i pravá část je stejná a je tedy možné vymodelovat pouze jednu a druhou vytvořit zrcadlovým převrácením. Následně stačí spojením příslušných hran vytvořit spodní část rámu.

V dalším kroku je vymodelována zeď. Ta je tvořena pouze třemi polygony vytvarovanými tak, aby mezi ní a rámem nevznikly žádné mezery.

Dále jsou vymodelovány objekty, které jsou na zdi umístěny. Všechny jsou jednoduše vytvořeny z primitivního objektu chamfer box, což je pouhý kvádr nebo krychle se zaoblenými hranami. Polygony, které tvoří zadní část těchto objektů hráč nikdy neuvidí, a proto jsou z důvodu šetření s polygony odstraněny.

Takto vytvořenou stěnu je poté již velmi jednoduché na okrajích zkosit, upravit tvar objektů na zdi a vytvořit tak stěnu typu G (oba typy znázorněny na obrázku číslo 16, který je pořízen ze softwaru 3ds Max). Jejich kombinací je posléze možné již bez modelování nových objektů další typy stěn pouze složit.



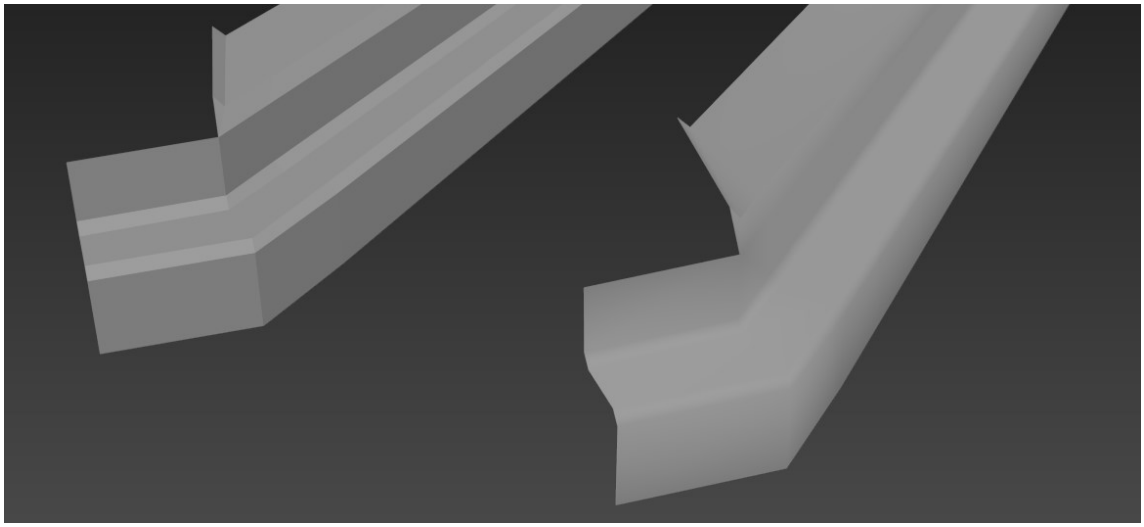
Obrázek 16: Ukázka modelů stěn herního prostředí

Podobně jednoduché je také vytvoření chodby, kdy pouze stačí umístit dvě stěny požadovaného typu naproti sobě v určité vzdálenosti (v tomto případě se jedná o vzdálenost 100 centimetrů v softwaru 3ds Max). Velmi důležité je tuto vzdálenost zachovávat při vytváření všech typů chodeb z důvodu jejich vzájemné návaznosti.

Další části herního prostředí, které je nutné vymodelovat jsou podlaha a strop. Jako podlaha postačí pouhá deska, tedy jeden čtvercový polygon. Tento lze také použít pro tvorbu stropu. Stačí ho uprostřed rozdělit na několik menších polygonů a ty následně vytvarovat do podoby polokoule, která nemusí být v horní části uzavřená, protože se v ní ve hře bude nacházet model žárovky osvětlení. Každý model stěny a chodby je o část stropu doplněn již v modelovací aplikaci a podlaha je posléze přidána pomocí scriptu, který je popsán dále, až v enginu při spuštění hry.

Po vytvoření všech základních modelů jsou jejich hrany v některých místech moc ostré. Počítačová grafika ale nabízí způsob, kterým lze tento nedostatek odstranit jednoduše a rychle. Polygonům, které spolu sdílí ostrou hranu, postačí přímo v softwaru 3ds Max nastavit hodnoty nazývané smoothing group na stejnou číselnou hodnotu a problém je vyřešen. Pro lepší

představu vizte obrázek číslo 17. Vlevo se na něm nachází část rámu stěny bez nastavení smoothing group a vpravo s ním, přičemž rozdíl je patrný na první pohled.



Obrázek 17: Nastavení hladkých hran

Mapování a texturování

Ve chvíli, kdy jsou modely vytvořeny, následuje proces mapování, který je připraví na následnou aplikaci textur.

V tomto případě se jednotlivé části v některých modelech často opakují. Např. celý model chodby typu A, B nebo C je tvořen stejnými modely stěn, které jsou pouze zrcadlově převrácené. V takové chvíli je jednoduší namapovat pouze jednu část (v rámci předchozího příkladu jednu stěnu) a až poté ji zkopírovat, neboť duplikát si již nastavené mapování dále uchovává a grafik tak nemusí dělat stejnou věc na více místech (modelech). Všechny modely jsou rovněž namapovány do jedné společné UV mapy, ze které lze vytvořit textury v podobě texture atlasu.

Samotné mapování v případě modelů herního prostředí je možné provést s použitím nejjednodušší planární projekce. Pro ještě větší usnadnění práce lze použít např. nástroj Relax, který navíc ve velkém množství případů dokáže příslušnou část mapy po aplikaci projekce upravit na správný tvar nebo nástroj Rescale, který namapované části ve správném poměru zvětší tak, aby na mapě (a dále i na textuře) zabírali co nejvíce místa.

Po dokončení mapování je již možné model exportovat do herního enginu. Vytvořená UV mapa dále poslouží pro tvorbu textur.

V rámci texturování bývá jako první je vytvořena diffuse mapa. Modely jsou exportovány do aplikace Mudbox, kde je na ně pomocí štětců nanesena barva, která se ihned uloží do textury. V případě objektů na stěně, které vypadají jako vyrobené z plastu, jsou to barvy oranžová a bílá. Pro podlahu, strop a rám stěny je použit odstín šedé barvy získaný z tabulek v dokumentaci enginu Unity, který bude ve výsledku dávat těmto objektům kovový nádech. Na tyto kovové části jsou posléze přes šablonu naneseny drobné detaily zašpinění s barvou ještě tmavšího odstínu šedé barvy.

Další vytvořenou texturou je speclar mapa. Tuto již není nutné na model kreslit, ale částečně odvodit z diffuse mapy a upravit ručně v editoru Photoshop. Pro kovové části je tato textura vytvořena použitím inverze barev a následnou úpravou kontrastu. V případě částí plastových byla jejich spekulární barva nalezena v tabulkách v dokumentaci herního enginu.

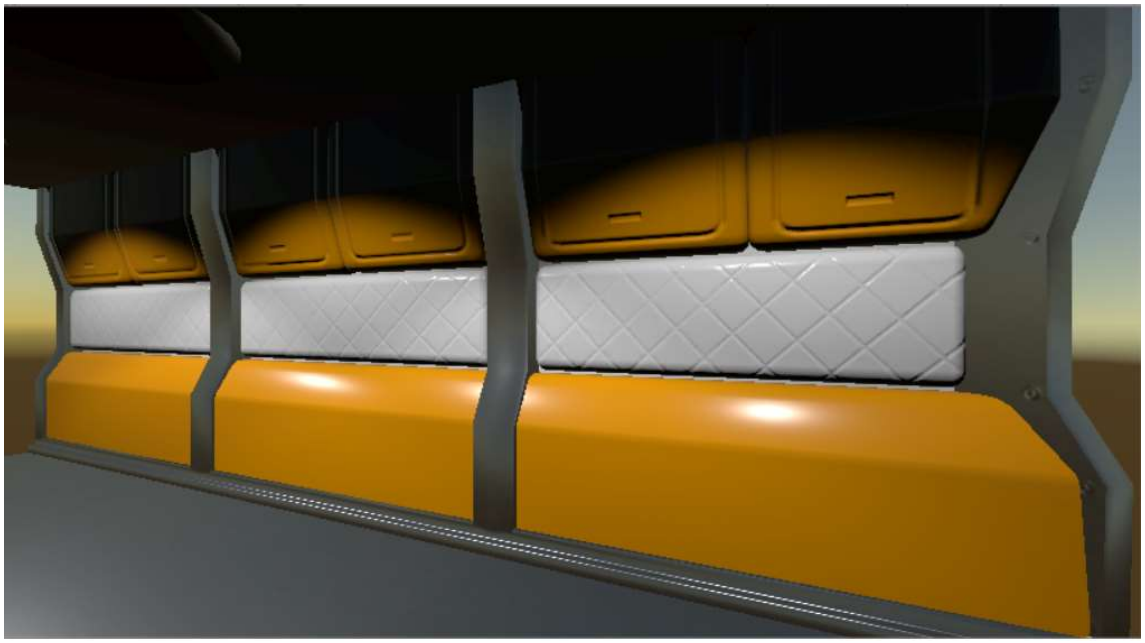
Poslední použitou texturou je normal map. Ta je použita pro vytvoření nýtů na kovových částech a dalších detailů na plastových objektech, které se na nich ale fyzicky nenacházejí. Každý z detailů je nejprve vytvořen v softwaru Mudbox jako skutečný model s vysokým počtem polygonů a posléze jsou detaily zapečeny do této textury.

Všechny vytvořené textury jsou posléze uloženy s rozlišením 2048px × 2048px do projektu v enginu. Normal mapu je navíc nutné přímo v enginu označit jako typ Normal map, aby se její vliv na model projevil správně.

Tvorba materiálu

Materiál je asset, který uživatel může vytvořit přímo v herním enginu. Stejně tak engine disponuje několika již připravenými shadery. V nastavení materiálu stačí pouze vybrat daný typ shaderu, který nabízí sloty pro potřebné textury. V případě materiálu pro objekty herního prostředí, byl vybrán shader s názvem Standard (specular setup). Do příslušných slotů pak stačí umístit připravené textury.

Výsledný vzhled osvětlených objektů herního prostředí s aplikovaným materiálem je znázorněn na obrázku číslo 18. Tento záběr zachycující tři stěny (zleva stěny typu I, I a H) je pořízen ze scény v enginu Unity.

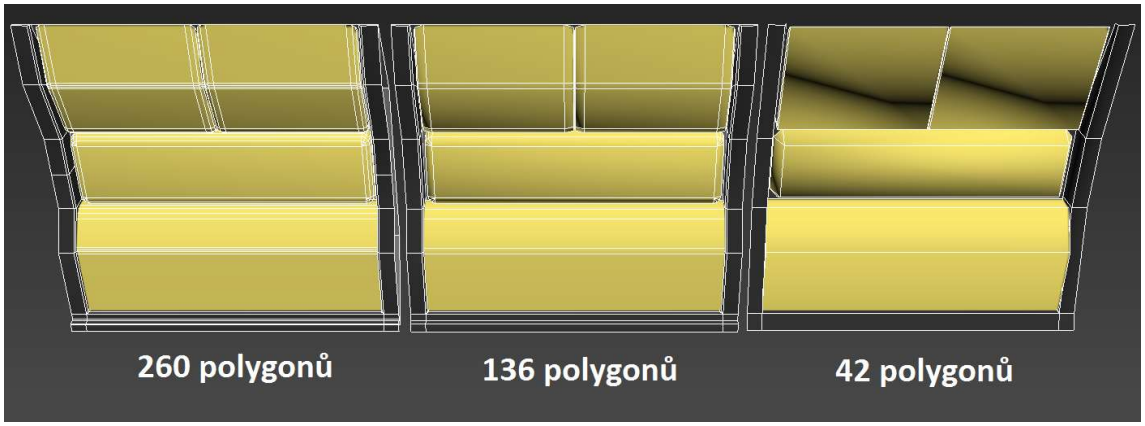


Obrázek 18: Modely herního prostředí s aplikovaným materiálem

Level of Detail

Takto vytvořené modely s aplikovaným materiálem je již možné ve hře použít. S přihlédnutím na výpočetní výkon lze však pro každý model vytvořit navíc varianty s nižší úrovní detailu. V takovém případě je nutný návrat zpět do modelovací aplikace, vytvořit kopii objektu a odstranit hrany, jejichž přítomnost na modelu vzhledem k větší vzdálenosti není potřeba.

Následující obrázek číslo 19 je pořízený ze softwaru 3ds Max a zachycuje model zdi typu I ve třech úrovních detailu a počet polygonů pro každou úroveň.



Obrázek 19: Ukázka modelů s různou úrovní detailu

Posléze je nutné i tyto další modely exportovat do herního enginu a říct mu, aby je mezi sebou na základě vzdálenosti přepínal. Toho lze docílit pomocí vytvoření tzv. prázdného objektu.

Tomuto objektu je nutné nastavit tři modely s různou úrovní detailu jako potomky a také mu nastavit komponentu zvanou LOD Group. Nakonec stačí na tuto komponentu všechny tři modely umístit a o zmíněné přepínání se již sama postará.

Nastavení kolizí a tvorba prefabu

Pro objekty skutečného světa je naprosto přirozené, že jimi nelze procházet. Pro objekty v herním enginu to již tak přirozené není. Aby skrz vytvořené objekty procházet nešlo, je nutné pro ně nastavit tzv. collidery neboli kolize. Tyto se v herní scéně nacházejí, ale hráč je nevidí.

K nastavení kolizí dochází až v herním enginu. V tomto případě stačí kolize nastavit pouze modelu s nejvyšší úrovní detailu, protože jedině s ním se hráč ve hře dostane do kontaktu. V případě modelů herního prostředí byl zvolen tzv. box collider, což znamená, že místo, kterým hráč nesmí projít udává tvar kvádru nebo krychle. Pro stěnu byla použita kolize pouze jedna, pro chodbu pak pochopitelně dvě. Pro kolizi podlahy pak stačí vytvořit jeden polygon, který je roztažen tak, aby se nacházel pod všemi objekty herního prostředí a má na sobě aplikovaný mesh collider, který automaticky přesně kopíruje jeho tvar. Tím je jednoduše zajištěno, že hráč pod podlahu nespadne.

Posledním krokem je vytvoření prefabu, který obsahuje assety herního prostředí s aplikovaným materiálem, nastaveným LOD Group a kolizemi. Rovněž ten lze připravit přímo v enginu a assety, které má obsahovat se do něj umístí pouhým přetažením. Navíc je vytvořen prefab složený pouze z modelu podlahy a modelu střechy umístěného 100 centimetrů nad ním. Ten lze umístit např. v místech, kde jsou stěny od sebe vzdáleny více než dva metry a vznikl by tak mezi nimi prostor, ve kterém by strop a podlaha chyběly.

Systém umístění prefabů herního prostředí ve scéně

Vytvoření mapy z připravených prefabů je velice závislé na jejich správném vzájemném umístění. Je totiž potřeba, aby např. dlouhá chodba složená z několika za sebou umístěných čtvercových částí vypadala celistvě a mezi jednotlivými modely nebyly žádné mezery. Přímé ruční umístění prefabů by mohlo být z tohoto hlediska nevhodné a především nepřehledné. Z toho důvodu byl vytvořen systém značek a scriptu, který skládání herní mapy zjednodušuje.

Každá značka je znázorněna 3D modelem, který svým tvarem odpovídá jednomu z typů chodeb nebo stěn znázorněných na obrázku číslo 15. Místo skutečných prefabů herního prostředí tedy vývojář do scény ručně vkládá pouze tyto značky, které se lépe umisťují a orientace ve scéně je velmi jednoduchá. Na každou z těchto značek je poté v enginu navázán script napsaný v jazyce C#.

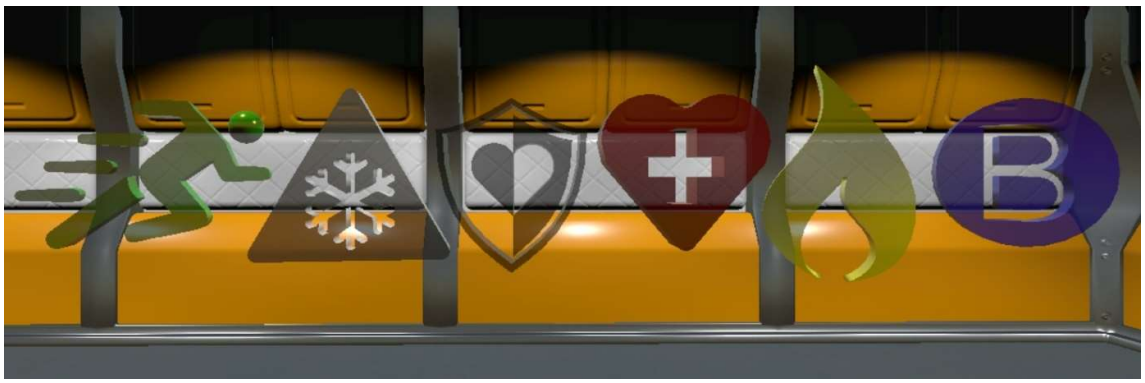
Tento script nabízí vývojáři přímo v enginu pro každou značku nastavit několik parametrů, kterými jsou prefab chodby nebo stěny, model podlahy, prefab zhasnuté žárovky, prefab rozsvícené žárovky a prefab efektu jisker. Dále může vývojář zvolit, jestli má žárovka umístěná ve stropě svítit nebo ne a také možnost zapnout efekt jisker. Po spuštění se tento script automaticky postará o umístění prefabu stěny nebo chodby na původním místě značky a správně ho otočí. Stejně tak správně umístí podlahu. Na základě volby také umístí do stropu žárovku se světlem nebo bez něj a případně také přidá efekt jisker, které vycházejí od žárovky. Žárovku a efekt jisker nastaví jako potomka prefabu herního prostředí, a nakonec odstraní značku, na které je navázán a tím pádem i sám sebe. Značka pro každý typ prefabu herního prostředí je poté i s tímto scriptem uložena také jako prefab a připravena pro znovupoužití.

6.4 Tvorba předmětů zlepšujících schopnosti hráče

Předmětů, které hráči po jejich získání zlepšují schopnosti a zvýhodňují ho ve hře je celkem šest. Model každého z nich je jednoduše vytvořen vytažením předem připraveného tvaru definovaným křivkami do 3D prostoru.

Následující obrázek číslo 20 zachycuje všechny tyto předměty. Zleva jsou to předměty pro:

- Zvýšení rychlosti hráče,
- zastavení pohybu nepřítele na určitý čas,
- nesmrtelnost hráče na určitý čas,
- doplnění zdraví hráče,
- zvýšení rozsahu výbuchu bomby,
- zvýšení počtu bomb, které může hráč nezávisle na sobě umístit.



Obrázek 20: Předměty zlepšující schopnosti hráče

Vytvoření animace

Pro předměty zlepšujících schopnosti hráče je v softwaru 3ds Max vytvořena jednoduchá animace, která pouze model otočí o 360 stupňů kolem osy Z za 60 snímků. V tomto případě je nutné při exportu modelu vybrat možnost exportovat rovněž tuto vytvořenou animaci.

Dále je nutné tuto animaci v enginu nastavit. Po výběru předmětu je v záložce enginu Animations změněn její název a možnost Loop Time označena jako aktivní, což zajistí, že se konkrétní animace bude přehrávat dokola a předmět se tedy bude stále otáčet.

Takto nastavená animace ale při spuštění hry nebude fungovat. Tento problém řeší tzv. Animator Controller, který modelu se kterým je spojen říká, kdy a jak se má animovat. Tento asset lze opět vytvořit až v enginu. Následně je do něj umístěna nastavená animace a v záložce enginu s názvem Animator lze Animator Controller nastavit za pomoci grafického rozhraní. V tomto případě jen stačí vytvořit spojnici (spojnice udává, které animace na sebe můžou navazovat a také může obsahovat omezení jako např. spuštění animace pouze ve chvíli stisknutí klávesy) mezi objektem Entry a objektem s názvem animace, čímž je docíleno, že se daná animace spustí při startu hry (obrázek 21 tento Animation Controller znázorňuje). Takto vytvořené Animator Controllery stačí pouze aplikovat na příslušný model.

Obrázek 21: Ukázka Animation Controlleru

Vytvoření materiálu

Materiál je v tomto případě je velice jednoduchý, neboť nepotřebuje žádné textury a v přímo v enginu tak stačí pouze nastavit několik parametrů. Prvním je volba shaderu, který zůstává stejně jako v případě objektů herního prostředí nastaven na Standard (specular setup). Dále je místo diffuse mapy nastavena čistá barva povrchu pomocí parametru Albedo a hodnota jeho alfa kanálu je nastavena na hodnotu 200. Místo specular mapy je na určitou barvu také nastaven parametr Specular a jeho volba s názvem Source je změněna na Albedo Alpha, čímž je zajištěno, že průhlednost materiálu bude určena hodnotou alfa kanálu nastavenou pro parametr Albedo. Zapnutí průhlednosti je nakonec docíleno pomocí nastavení parametru Rendering Mode na hodnotu Transparent. Výsledný vzhled modelů v enginu Unity po aplikování materiálu je znázorněn na obrázku číslo 20.

Nastavení kolizí a tvorba prefabu

Jako poslední jsou nastaveny kolize pomocí box collideru (pro vysvětlení vizte nastavení kolizí v modelů herního prostředí). Navíc je v tomto případě vybrána možnost Is Trigger, která umožňuje, aby hráč mohl projít skrz model, ale k zaznamenání kolize dochází stále, aby mohl programátor pomocí dalších scriptů upravit danou schopnost hráče.

Pro každý z takto nastavených předmětů je rovněž vytvořen prefab, který obsahuje model s aplikovaným materiálem, Animation Controllerem a nastavenými kolizemi.

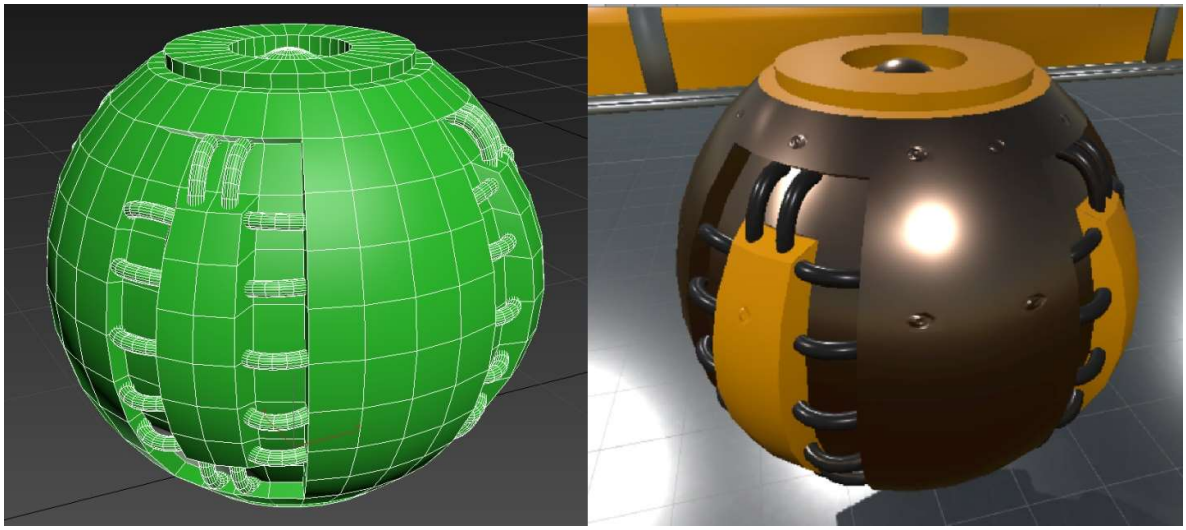
6.5 Tvorba bomby a částicových efektů

Model bomby pro tento projekt je navržen tak, aby ho hráč mohl umístit uprostřed čtvercové podlahy. Po jeho umístění je spuštěn zvukový odpočet, po jehož ukončení dojde k odstranění bomby ze scény a následné explozi, která je doprovozena dalším zvukem a efektem kouře. Poté se na místě bomby a v jejím okolí objeví efekt výbuchu, který zraňuje hráče a ničí překážky.

Modelování a texturování bomby

Model bomby je připraven z primitivního objektu koule o poloměru 22 centimetrů, ze které je ale zachována pouze jedna její čtvrtina. Ta je nejprve na horní i spodní části zploštěna a polygony umístěné na dně jsou z důvodu šetření polygony smazány. Poté je její povrch upraven za pomoci nástroje Extrude, který umožňuje polygony zatlačit nebo naopak vytlačit ven z modelu. Jednotlivé kabely, které jsou v těchto vtlačených částech umístěny jsou vytvořeny pomocí techniky tažení 2D profilu (v tomto případě kruhového) po křivce. Takto vytvořená čtvrtina je následně namapována a poté jsou vytvořeny její tři duplikáty, z nichž každý je oproti předchozímu otočen v ose Z o 90 stupňů. Výsledný model bomby je nakonec z těchto čtvrtin jednoduše složen. Po vytvoření základního modelu je tento upraven na další dvě verze s nižším počtem polygonů pro LOD.

Textury i materiál jsou vytvořeny stejným způsobem jako v případě textur herního prostředí. Pro výsledný vzhled tohoto assetu vize následující obrázek s číslem 22, který ve své levé části zobrazuje model v softwaru 3ds Max a vpravo asset umístěný ve scéně v enginu a aplikovaným materiálem.



Obrázek 22: Ukázka assetu bomby

Tvorba částicových efektů

Částicové efekty lze vytvořit přímo v enginu a do prostoru posílají velké množství textur nebo modelů (dále jen částic) a vývojáři nabízí široké spektrum nastavení. Pro bombu jsou vytvořeny celkem tři částicové efekty. Prvním je kouř, který se zobrazuje na místě bomby při explozi. Druhým je efekt výbuchu spojený s třetím efektem energetické vlny.

Pro tyto efekty jsou vytvořeny dvě textury. První textura zobrazující kouř je zhotovena v editoru Photoshop pomocí štětců a druhá je vytvořena ve stejném softwaru jako kruhový přechod mezi bílou a černou barvou. Pro použití v částicovém efektu jsou tyto textury vloženy do materiálu, jehož shader je nastaven na hodnotu Particles/Additive. Barva nebo výsledný tvar textur jsou závislé na nastavení konkrétního efektu.

Nastavením dalších parametrů efektů jako je například jejich časové trvání (nastaven v obou případech na 5 sekund), maximální množství částic, které se ve scéně může v jednu chvíli nacházet, připravený prefab bodového světla, které každá částice může nést, stopa, kterou za sebou částice zanechává a dalších hodnot, jejichž celkový výčet by byl velmi dlouhý, jsou efekty vytvořeny do podoby, která je znázorněna na obrázku číslo 23. Efekty jsou posléze uloženy jako prefab.

Obrázek 23: Ukázka vytvořených částicových efektů

Zvuky a script pro bombu a výbuch

Aby v mohl být v enginu Unity přehrán jakýkoliv zvuk, musí mít na sobě příslušný objekt aplikovanou komponentu Audio Source, tedy zdroj zvuku, kterou lze jednoduše ovládat pomocí scriptu jako je tomu v případě bomby. Vstupními parametry tohoto scriptu je zdroj zvuku, zvuk odpočtu do exploze a částicové efekty již zmíněné v předchozí části práce. Script pak zajišťuje spuštění zvuku odpočtu ihned po položení bomby na podlahu, který hráč může zaslechnout maximálně ze vzdálenosti tří metrů a se zvyšující se vzdáleností se jeho hlasitost snižuje. Ihned po odpočtu script zajistí umístění částicových efektů, znázorněných na obrázku 23, do scény.

Každý samostatný efekt výbuchu obsahuje další script a dva zdroje zvuku. Jeden pro zvuk s výbuchem, který lze slyšet maximálně do vzdálenosti deseti metrů a také zdroj zvuku elektrického výboje s maximálním dosahem tři metry. Oba zvuky jsou spuštěny současně. V případě, že se výbuch dotkne modelu překážky se script postará o to, aby byla tato překážka odstraněna ze scény a na jejím místě byl umístěn jeden náhodně vybraný prefab předmětu pro zlepšení schopnosti hráče. Rovněž se script postará o odstranění samotného efektu výbuchu ze scény po čase, který je dán parametrem tohoto scriptu.

Všechny zvuky jsou pořízeny z webu www.freesound.org, kde jsou volně k dispozici.

6.6 Tvorba postavy hráče

Model postavy hráče z hlediska počítačové hry představuje významnou část a žádná hra se bez ní neobjede. V závislosti na její důležitosti je ale stejně vysoká náročnost na její vytvoření.

Modelování a texturování hráče

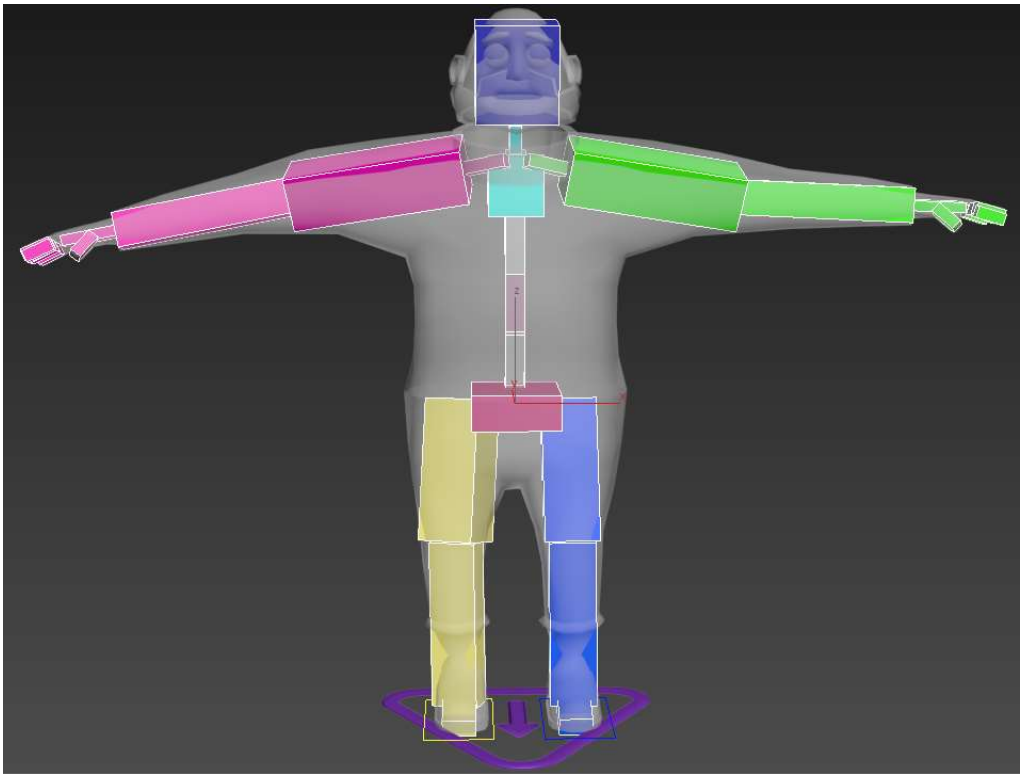
Model hráče je příkladem, že i technikou modelování z primitivního objektu lze vytvořit jakkoliv složitý objekt, neboť i postava hráče je vytvořena z objektu jednoduchého kvádru, který je postupným dělením na další polygony vytvarován do podoby, která je zachycena na obrázku číslo 24 nahoře. Postavy jsou pro počítačové hry obecně modelovány v tzv. poloze T, která se vyznačuje rozpažením rukou postavy. Důvodem je připravenost modelu pro umístění kostí a animování, jež je popsáno v následující části této práce.

Způsoby mapování, tvorby textur a materiálu se nijak neliší od těch, které již byly probrány v rámci tvorby assetů bomby nebo herního prostředí (vytvořena byla diffuse mapa, specular mapa a normal mapa). Model postavy hráče s aplikovaným materiálem přímo v enginu zachycuje obrázek číslo 24 dole a tvoří ho celkem 2304 polygonů.

Obrázek 24: Model postavy hráče

Příprava modelu hráče pro animování

Tvorbě samotných animací postav předchází nutnost připravit kosti, s jejichž pomocí lze s jednotlivými částmi těla hýbat. K tomu lze v softwaru 3ds Max použit plug-in s názvem CAT (zkratka anglického Character Animation Toolkit). Tento umožňuje do scény umístit již hotovou kostru, jejíž kosti jsou nastaveny tak, aby se při svém pohybu chovaly, pohybovaly a otáčely stejně jako lidské. Jednotlivé kosti je také nutné umístit na pozice uvnitř modelu postavy. Umístěnou kostru znázorňuje následující obrázek s číslem 25.



Obrázek 25: Umístění kostí modelu postavy hráče

Následně je na model postavy aplikován modifikátor Skin. Tento umožňuje ručně nastavit jednotlivým kostem, které vertexy modelu hráče budou ovlivněny pohybem těchto kostí (neboli která část lidského těla patři k dané kosti). Jedná se tedy o zdlouhavou a pro začátečníka náročnou část tvorby herní postavy.

Tvorba animace postavy hráče

Po správném nastavení kostí lze přistoupit k animování postavy. Animace jsou tvořeny za pomoci časové osy a klíčů, které se na ni umisťují. Každý klíč nese informace o pozici, rotaci a měřítku kosti v čase, který je určen aktuální polohou klíče na časové ose. Každá kost může, a pro vytvoření složitějších animaci nutně musí, mít tedy v rámci celé časové osy více klíčů.

Např. pro animaci upažení ruky o délce deseti snímků je tedy nejprve nutné přesunout se na konkrétní místo časové osy (snímek s číslem deset), posléze ruku nastavit do příslušné polohy a klíč se automaticky vytvoří.

Tímto způsobem jsou vytvořeny následující animace postavy hráče, které jsou umístěny za sebou v rámci jedné časové osy:

- Dvě animace postoje postavy (tyto se přehrávají ve chvíli, kdy hráč pouze stojí na místě),
- animace chůze postavy dopředu (animace chůze dozadu dosažena jejím přehráním pozpátku),
- animace smrti postavy.

Každá z těchto animací je doplněna o mrkání očí. Tato animace je vytvořena pomocí tzv. blend shapes (v softwaru 3ds Max za tímto účelem slouží modifikátor Morhp), kdy je vytvořen navíc model postavy, která má zavřené oči. Výsledné animace je docíleno pozvolným přechodem mezi původním a nově vytvořeným modelem.

Export postavy hráče a nastavení animací v enginu Unity

Po vytvoření animací je možné model exportovat do herního enginu. Stejně jako v případě assetu bomby je nutné během exportu nastavit, aby se s modelem exportovaly i jeho animace a navíc také, aby jednotlivé kosti v enginu nebyly vidět.

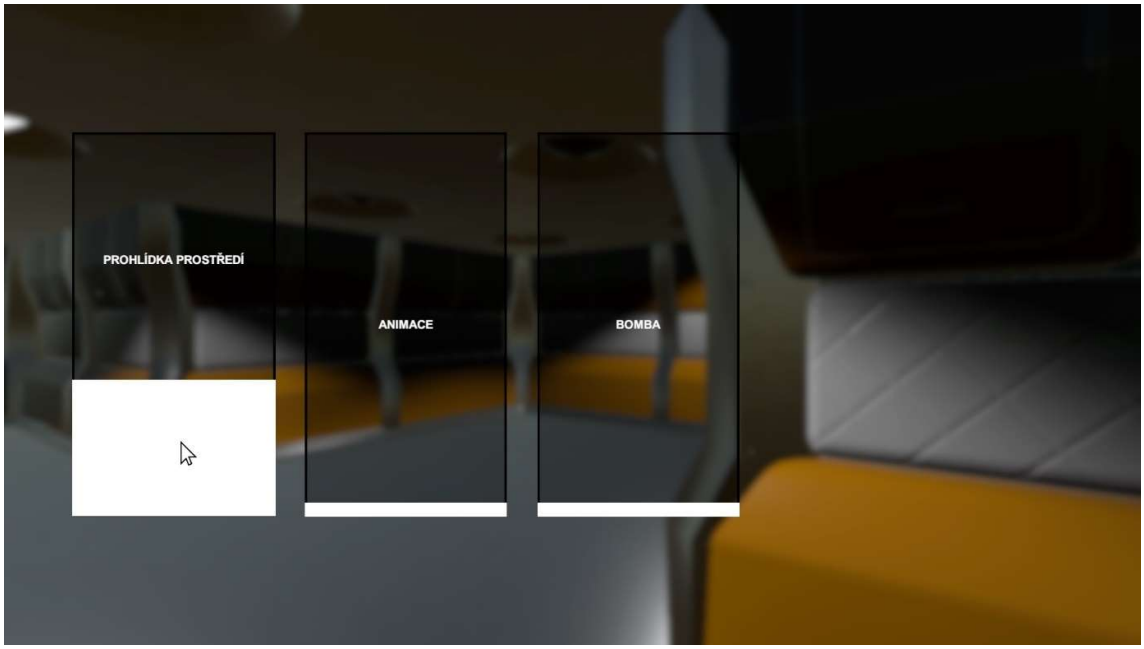
Jelikož jsou všechny animace umístěny v rámci časové osy za sebou, je nutné je od sebe oddělit. Po výběru exportovaného modelu je v záložce Animations nastaven pro každou animaci její první a poslední snímek na časové ose, čímž je problém vyřešen.

Takto nastavené animace již lze použít pro vytvoření Animation Controlleru (jeho význam je popsán v části animace předmět zlepšujících vlastnosti hráče), který pro účely ukázky umí spouštět jednotlivé animace na základě scriptu, který reaguje na stisknutí kláves.

Model postavy hráče s aplikovaným materiálem, Animtor Controllerem a připojeným scriptem pro ovládání animací je uložen jako prefab.

6.7 Tvorba uživatelského rozhraní

Uživatelské rozhraní zahrnuje hlavní nabídku, ze které lze přecházet do jiných scén (herních levelů) a nabídku pozastavení, která zahrnuje možnost návratu do hry, návratu do hlavní nabídky a ukončení celé hry. Ukázku hlavní nabídky složené z assetů vytvořených pro tuto práci zachycuje obrázek číslo 26. Tvorba těchto assetů je více probrána dále.



Obrázek 26: Ukázka hlavní nabídky

Plátno uživatelského rozhraní

Aby bylo možné vytvořit uživatelské rozhraní, je nutné použít tzv. Canvas neboli plátno, do kterého lze vkládat jednotlivé prvky jako například tlačítka. Společně s ním je automaticky vytvořen také tzv. Event System, který je zodpovědný za zpracování událostí od vstupních zařízení jako je např. myš nebo klávesnice.

V tomto plátně jsou navíc vytvořeny dva panely s názvem MenuPanel určený pro umístění tlačítek hlavní nabídky a KonecPanel pro umístění tlačítek nabídky zobrazené při pozastavení hry. Dále je na plátno připojen script, který se stará o to, aby se panel s nabídkou pozastavení zobrazil nebo skryl při stisku klávesy Escape. Script také zajišťuje skrytí této nabídky při kliknutí na tlačítko, které lze nastavit pomocí parametru tohoto scriptu.

Pozadí plátna je realizováno po vzoru některých dnešních počítačových her jako přímý náhled kamery do připravené scény. Náhled je navíc rozmazaný, čehož lze docílit pomocí aplikace příslušného scriptu s daným shaderem (tyto jsou v rámci enginu Unity volně dostupné v rámci balíčku standartních assetů) na kameru.

Takto připravené plátno je uloženo jako prefab a připraveno pro znovupoužití v jiných scénách.

Přechod mezi scénami

Aby přepínání mezi scénami vypadalo lépe, byl vytvořen přechod aktuální scény do bílé barvy, který je zobrazen před tím, než dojde k samotnému přepnutí.

Přechod je realizován jako samostatné plátno bílé barvy (tato barva se dá kdykoliv v nastavení plátna změnit) s nastavenou průhledností na hodnotu nula a je tedy naprosto průhledné. Pro toto plátno je vytvořena animace, která umí za dobu dvou vteřin postupně zvyšovat průhlednost na hodnotu jedna a tím pádem celou scénu překrýt bílou barvou. Takto vytvořený přechod je uložen jako prefab.

Dále je vytvořen Animation Controller s názvem Prechod_zakladni, který obsahuje vytvořenou animaci přechodu a je nastaven tak, aby se spustila pouze ve chvíli, kdy je událostí vyvolanou například kliknutím na tlačítko aktivována spoušť s názvem Prechod. Controller rovněž po přehrání animace přechodu zajišťuje spuštění scriptu s názvem JinaScena, který se postará o přepnutí do jiné scény.

Takto připravený přechod lze spolu s jeho Animation Controllerem a scriptem zkopírovat a jednoduše pomocí úpravy scriptu (přepnutí do jiné scény) vytvořit nový přechod pro další tlačítko uživatelského rozhraní

Tlačítko s animací

Pro uživatelské rozhraní je rovněž upraveno klastické tlačítko, které herní engine Unity nabízí. Vytvořena je pro něj nová textura pozadí, jejíž barvu lze přímo v enginu kdykoliv změnit. Je také doplněno o bílý spodní okraj, který díky vytvořenému Animation Controlleru s názvem MenuTlacitko_Controller po označení tlačítka myší mění svou velikost a mění polohu textu. Stejně jako v předchozích případech, i v tomto je tlačítko uloženo jako prefab.

6.8 Kamera hráče

Pro účely volného pohybu hráče po scéně složené z assetů vytvořených v rámci této práce je jako prefab uložena také kamera, která je vybavena scriptem pro ovládání pohybu pomocí kláves šipek nebo také kláves W, S, A a D. Rovněž je pro ni také nastavena kolize, aby nebylo možné procházet objekty umístěnými ve scéně.

6.9 Balíček vytvořených assetů

Herní engine unity umožňuje assety, prefaby a všechny další položky kteréhokoliv projektu exportovat ve formě balíčku. Jeho obsah je následně možné importovat do jiných projektů a jakýkoliv z assetů znovu použít pro vývoj jiné počítačové hry.

Takový balíček je také výstupem této práce a obsahuje všechny assety pro ni vytvořené spolu s několika ukázkovými scénami.

Závěr

Herní engine Unity je kvalitním nástrojem určeným k vývoji počítačových her pro velké množství platforem a díky jeho volné licenci dnes může začít vytvářet hry prakticky každý. Základním předpokladem a také problémem je však schopnost vývojářů vytvořit obsah, který s pomocí enginu ve hru promění, tedy tvorba assetů.

V rámci této práce se mi nejdříve podařilo čtenáři pojem asset (a související pojem prefab) vysvětlit a také představit širokou paletu herního obsahu, který lze tímto pojmem označit. Z toho ale vyplívá základní problém, kterým je neschopnost jediného vývojáře zcela ovládnout tvorbu všech druhů assetů, které se dnes pro vývoj počítačových her používají. V herních studiích bez ohledu na jejich velikost je proto možné nalézt zaměstnance, z nichž každý se specializuje pouze na tvorbu určitého druhu assetů. Stejně tak i tuto práci jsem se rozhodl zaměřit na vývoj audiovizuálního obsahu, tedy grafických objektů statických i pohyblivých, které hráč po spuštění hry vidí na obrazovce a s nimi spojené doprovodné ozvučení.

V teoreticky zaměřené části této bakalářské práce byl splněn cíl předat čtenáři znalosti z oboru počítačové grafiky, z nichž některé jsem načerpal již dříve díky samostudiu, avšak tato práce mi je umožnila ve velké míře značně rozšířit. Podařilo se mi objasnit modelování objektů pro počítačové hry společně s představením jeho technik a související optimalizací výpočetního výkonu. Věnoval jsem se rovněž následným nastavením modelů a jejich exportem do herního enginu Unity. Dále jsem v práci zahrnul mapování modelu, které ho připraví na aplikování textur. V neposlední řadě jsem se zabýval tvorbou těchto textur a představením nejpoužívanějších typů. Rovněž jsem v práci popsal jejich nasazením přímo v herním enginu Unity a způsoby, kterými se sám engine snaží hru optimalizovat. S těmito získanými znalostmi si čtenář může lépe udělat představu o náplní práce herního grafika, který je nepochybně důležitou součástí vývojářského týmu.

V části praktické jsem se zabýval konkrétním popisem tvorby audiovizuálních assetů pro počítačovou hru pro více hráčů inspirovanou známým herním titulem Bomberman. Tyto assety jsou stylizovány do prostředí vesmírné lodi, vytvořeny s využitím poznatků obsažených v teoretické části a exportovány do herního enginu Unity. Pro tuto práci jsem celkem vytvořil 120 assetů a 44 prefabů, na základě kterých jsem připravil čtyři scény z nich složené. Vše je výstupem této práce ve formě balíčku připraveném pro znovupoužití v jakémkoliv projektu herního enginu Unity.

Použitá literatura

[1] UNITY TECHNOLOGIES. Unity – Manual: Asset Workflow. *unity3d* [online]. 2017 [cit. 2017-02-28]. Dostupné z: https://docs.unity3d.com/Manual/AssetWorkflow.html

[2] UNITY TECHNOLOGIES. Unity – Manual: Cubemap. *unity3d* [online]. 2017 [cit. 2017-02-28]. Dostupné z: https://docs.unity3d.com/Manual/class-Cubemap.html

[3] UNITY TECHNOLOGIES. Unity – Manual: Prefab. *unity3d* [online]. 2017 [cit. 2017-02-28]. Dostupné z: https://docs.unity3d.com/Manual/Prefabs.html

[4] GALUZIN, Alex. 23 Recommended and Available 3D Game Engines (Updated). *World of Level Design* [online]. 2012, Updated: July 28, 2016 [cit. 2017-03-02]. Dostupné z: http://www.worldofleveldesign.com/categories/level_design_tutorials/recommended-game-engines.php

[5] KŘÍŽ, Jan. *Mistrovství v 3ds Max: [kompletní průvodce profesionálního grafika]*. Brno: Computer Press, 2010. ISBN 978-80-251-2464-2

[6] CAD STUDIO A.S. Autodesk Mudbox. *Cadstudio* [online]. 2015? [cit. 2017-03-02]. Dostupné z: http://www.cadstudio.cz/mudbox

[7] ADOBE SYSTEMS. Adobe Photoshop CC. *adobe* [online]. © 2017 [cit. 2017-03-03]. Dostupné z: https://www.adobe.com/cz/products/photoshop.html

[8] MICROSOFT. Game Development | Visual Studio. *visualstudio* [online]. 2016 [cit. 2017-03-03]. Dostupné z: https://www.visualstudio.com/cs/vs/game-development

[9] AUTODESK. About Polygon Meshes. *The Softimage Wiki* [online]. 2011 [cit. 2017-03-04]. Dostupné z: http://softimage.wiki.softimage.com/xsidocs/poly_basic_PolygonMeshes.htm

[10] UNITY TECHNOLOGIES. Unity – Manual: Meshes. *unity3d* [online]. 2017 [cit. 2017-03-04]. Dostupné z: https://docs.unity3d.com/Manual/class-Mesh.html

[11] EPIC GAMES. UDK | UT3CustomCharacter*s*. *unrealengine* [online]. © 2001–2012 [cit. 2017-03-04]. Dostupné z: https://docs.unrealengine.com/udk/Three/UT3CustomCharacters.html

[12] ZACH, Correa. 3D printing: convert a Picture into a 3D Model. *sculpteo* [online]. 2016 [cit. 2017-03-12]. Dostupné z: https://www.sculpteo.com/blog/2016/01/20/turning-a-picture-into-a-3d-model

[13] CAD STUDIO A.S. Autodesk 123D. *Cadstudio* [online]. © 2017 [cit. 2017-03-15]. Dostupné z: http://www.cadstudio.cz/123d

[14] UNITY TECHNOLOGIES. Unity – Manual: Art Asset best practice guide. *unity3d* [online]. 2017 [cit. 2017-03-15]. Dostupné z: https://docs.unity3d.com/Manual/HOWTO-ArtAssetBestPracticeGuide.html

[15] VALVE CORPORATION. Level of Detail. *Valve Developer Comunity* [online]. 2012 [cit. 2017-03-18]. Dostupné z: https://developer.valvesoftware.com/wiki/Level_of_detail

[16] UNITY TECHNOLOGIES. Unity – Manual: LOD Group. *unity3d* [online]. 2017 [cit. 2017-03-18]. Dostupné z: https://docs.unity3d.com/Manual/class-LODGroup.html

[17] NVIDIA. DirectX 11 Tessellation. *nvidia* [online]. © 2017 [cit. 2017-03-18]. Dostupné z: http://www.nvidia.com/object/tessellation.html

[18] NVIDIA. Chapter 7. Adaptive Tessellation of Subdivision Surfaces with Displacement Mapping. *nvidia* [online]. 2005 [cit. 2017-03-18]. Dostupné z: https://web.archive.org/web/20170102230742/http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter07.html

[19] UNITY TECHNOLOGIES. Unity – Manual: 3D Formats. *unity3d* [online]. 2017 [cit. 2017-03-20]. Dostupné z: https://docs.unity3d.com/Manual/3D-formats.html

[20] GAHAN, Andrew. *Game art complete: all-in-one : learn Maya, 3ds Max, zBrush, and Photoshop winning techniques*. Boston: Elsevier/Focal Press, 2009. ISBN 02-408-1147-X

[21] KLEKNER, Martin. Jak začít s vfx #14: Jak na UV mapping? *vizualniefekty* [online]. 2015 [cit. 2017-03-20]. Dostupné z: http://vizualniefekty.cz/jak-zacit-s-vfx-14-jak-na-uv-mapping

[22] AUTODESK. UVW Map Modifier. *Autodesk Knowledge Network* [online]. 2016 [cit. 2017-03-20]. Dostupné z: https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/3DSMax/files/GUID-78327298-4741-470C-848D-4C3618B18FCA-htm.html

[23] UNITY TECHNOLOGIES. Unity – Manual: Textures. *unity3d* [online]. 2017 [cit. 2017-03-25]. Dostupné z: https://docs.unity3d.com/Manual/class-TextureImporter.html

[24] KLEKNER, Martin. Jak začít s vfx #13: Jak na CG texturing? *vizualniefekty* [online]. 2014 [cit. 2017-03-25]. Dostupné z: http://vizualniefekty.cz/jak-zacit-s-vfx-13-jak-na-cg-texturing

[25] AHEARN, Luke. *3D game textures: create professional game art using Photoshop*. 2nd ed. Boston: Focal Press/Elsevier, 2009. ISBN 978-0-240-81148-2

[26] UNITY TECHNOLOGIES. Unity – Manual: Specular Mode: Specular parameter. *unity3d* [online]. 2017 [cit. 2017-03-26]. Dostupné z: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterSpecular.html

[27] UNITY TECHNOLOGIES. Unity – Manual: Normal Map (Bump Mapping). *unity3d* [online]. 2017 [cit. 2017-03-26]. Dostupné z: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterNormalMap.html

[28] PLURALSIGHT. Elliminate Texture Confusion: Bump, Normal and Displacement Maps. *pluralsight* [online]. 2014 [cit. 2017-03-30]. Dostupné z: https://www.pluralsight.com/blog/film-games/bump-normal-and-displacement-maps

[29] FIALA, Lukáš. Hardwarová teselace je cestou k lepší 3D grafice. *cnews.cz* [online]. 2010 [cit. 2017-03-30]. Dostupné z: https://www.cnews.cz/hardwarova-teselace-je-cestou-k-lepsi-3d-grafice

[30] UNITY TECHNOLOGIES. Unity – Manual: Creating and Using Materials. *unity3d* [online]. 2017 [cit. 2017-04-02]. Dostupné z: https://docs.unity3d.com/Manual/Materials.html

[31] UNITY TECHNOLOGIES. Unity – Manual: Draw call Batching. *unity3d* [online]. 2017 [cit. 2017-04-02]. Dostupné z: https://docs.unity3d.com/Manual/DrawCallBatching.html

[32] Maria. 13 Games Like Bomberman. *Similar Games To* [online]. 2017 [cit. 2017-04-20]. Dostupné z: https://www.similargamesto.com/bomberman

Přílohy

Příloha A – *Obsah přiloženého CD*

Na přiloženém CD se nachází následující obsah:

- Elektronická verze této práce v souboru HouzvickaJan_TvorbaAssetuProHruProHerniEngineUnity3D_2017.pdf,
- balíček HouzvickaJan_Assety.unitypackage, který je umístěn ve složce Assety. Tento je exportovaný z enginu Unity a obsahuje všechny assety vytvořené pro tuto práci (pro podrobný seznam assetů vizte část 6.2 této práce).