Univerzita Pardubice
Fakulta elektroniky a informatiky

Tvorba assetů počítačových her pro herní engine Unity 3D

Bc. Milan Horák

Bakalářská práce
2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2018/2019

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

Jméno a příjmení:	**Milan Horák**
Osobní číslo:	**I16088**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Tvorba assetů počítačových her pro herní engine Unity 3D**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

V teoretické části bakalářské práce bude přehledné seznámení s tvorbou assetů počítačových her (3D modely, textury, animace, zvuky, skripty, atd.) pro herní engine Unity 3D. Úvodní část práce nás seznámí s celkový postupem tvorby assetů, od počátečního návrhu vzhledu modelů, až po finální podobu použitelnou v počítačové hře. Práce se bude věnovat také softwarovým aplikacím umožňujícím tvorbu assetů. V implementační části bude popsána tvorba assetů pomocí softwarových aplikací a bude vytvořeno a implementováno několik assetů pro vytvoření prostředí a postav v herním enginu Unity 3D pro počítačovou hru Bulánci.

Rozsah pracovní zprávy: **(min. 30 stran)**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

DILLE, Flint. The ultimate guide to video game writing and design. New York: Watson-Guptill Publications, 2007. ISBN 158065066
GOLDSTONE, Will. Unity game development essentials. Birmingham, U.K.: Packt Pub., 2009. From technologies to solutions. ISBN 184719818
WAGNER, Jaroslav, KOPECKÝ, Lubor. Bulánci [online]. 2017 [cit. 2018-10-25]. Dostupné z: http://bulanci.cz/

Vedoucí bakalářské práce: **Ing. Zbyněk Kopecký**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2018**
Termín odevzdání bakalářské práce: **12. května 2019**

Univerzita Pardubice Česká republika

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 14. prosince 2018

Prohlašuji:

Tuto práci jsem 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.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 3. 9. 2019

Milan Horák

PODĚKOVÁNÍ

Tímto bych rád poděkoval celé své rodině za podporu během studia, dále své přítelkyni za trpělivost a psychickou podporu při psaní této práce. V neposlední řadě také děkuji za udělení mého tématu a možnost konzultovat odbornou problematiku mému vedoucímu bakalářské práce, kterým je pan Ing. Zbyněk Kopecký.

ANOTACE

Bakalářská práce se zabývá tvorbou vizuálních assetů pro herní engine Unity 3D. Cílem teoretické části je seznámit čtenáře s vývojem a problematikou tvorby 3D modelů. Práce nejdříve představí použitý software a následně vysvětlí principy a techniky, které se využívají při tvorbě 3D modelů pro počítačové hry.

Nakonec se práce věnuje praktické tvorbě grafických assetů pro 3D verzi hry známé jako Bulánci a to od počátečního návrhu vzhledu modelů, až po finální podobu použitelnou v počítačové hře.

KLÍČOVÁ SLOVA

assety, počítačová grafika, herní engine, textury, animace, optimalizace, skripty

TITLE

Creating computer game assets for game engine Unity 3D

ANNOTATION

Bachelor thesis deals by creating visual assets for game engine Unity 3D. Theoretical part deals with the development and problems of creating 3D models. The work first includes used software and then explains principles and techniques, which are used for creating 3D models for computer games.

Finally, the thesis dedicates to practical making of graphic assets from basic model designs to its final appearance usable in computer game. These models are based on game named Bulánci.

KEYWORDS

assets, computer graphics, game engine, textures, animations, optimization, scripts

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

C#	C Sharp
JS	JavaScript
GUI	Graphical User Interface
LOD	Level of Detail
NURBS	Non-uniform rational basis spline
CV	Control Vertex
PDF	Portable Document Format

ÚVOD

V dnešní době zná počítačové hry již asi každý, jejich tvorba však může být pro neznalou osobu záhadou. Cílem bakalářské práce je seznámit čtenáře s tvorbou assetů. Nejdříve je vysvětlen pojem asset v kontextu s vývojem počítačových her. Dále jeho postupné zpracování a nasazení do herního enginu Unity 3D. Práce je zaměřena především na assety vizuálního charakteru.

Dále práce představí profesionální software, který byl využit pro tvorbu assetů v této práci a také jeho současně možné alternativy, které jsou využívány ve velkých herních studiích. Tato část bude také obsahovat výhody a nevýhody jejich použití.

Další důležitou částí je teoretické vysvětlení nejrůznějších využívaných technik a principů z oblasti počítačové grafiky, které se při tvorbě herních assetů využívají. Dále bude popsáno několik důležitých pojmů, které se zabývají touto problematikou.

V poslední praktické části je cílem stylizace assetů do hry pro více hráčů známé jako Bulánci, které se v současné době nikdo nepokusil přetvořit do 3D verze. Následuje jejich výčet a zároveň ukázka jejich tvorby s využitím předchozích znalostí získané během této bakalářské práce.

1 ASSET

1.1 Vysvětlení pojmu

Asset obecně představuje nějaký předpřipravený objekt, který lze využít ve vaší počítačové hře nebo projektu. Může se jednat například o 3D model, zvukový záznam, obrázek, animaci, skript nebo jinou položku využívající více z těchto prvků. Bakalářská práce bude zaměřena výhradně na assety vizuálního charakteru.

Zdroj [2]

1.2 Grafické assety

Mezi grafické assety patří takové assety, které mají diváci možnost ve hře nebo projektu zahlédnout a patří mezi ně následující zástupci.

Typickým příkladem mohou být 3D modely představující vizuální objekty simulovaného světa. Jako například lidské postavy, budovy, vegetace a další. Tyto modely jsou často doprovázeny i sadou animací.

Dalším neméně důležitým příkladem mohou být textury. Textury mohou být použity jako materiály a následně aplikovány na modely, čímž poslouží k určení toho, jak bude model barevně zpracován.

Dalšími představiteli grafických assetů mohou být různé ikonky použité pro uživatelské rozhraní. Dále částicové efekty, pomocí kterých lze dosáhnout požadované atmosféry jako například efektu mlhy, ohně, exploze, kouře a dalších.

1.3 Zvukové Assety

Zvukové assety jsou důležitou součástí pro zlepšení herního zážitku a celkovou uvěřitelnost světa. Mezi zvukové assety patří hudba, dabing postav nebo různé prostorové audio efekty.

1.4 Skripty

Bez skriptů by se náš výsledný svět choval pouze staticky a nemohl nijak reagovat na interakci hráče. Pomocí skriptů lze popsat pravidla světa jako je například gravitace, umělá inteligence některých tvorů, obchodní podmínky, reakci na uživatelský vstup, zaznamenávat statistické údaje, pravidla pro výhru a další různé herní mechanismy.

2 POUŽITÝ SOFTWARE

Zde bude představen veškerý software použitý v bakalářské práci. Dále bude rozebrán důvod, proč byl tento software vybrán a také uvedeny jeho současně možné alternativy.

2.1 Blender

Blender je volně dostupný Multimediální software zaměřující se na 3D modelovaní, který je pod právy Open Source licence. V bakalářské práci je tento software použit pro tvorbu postav, zbraní a objektů prostředí.

Blender obsahuje širokou škálu nástrojů pro tvorbu 3D obsahu. Mezi tuto škálu patří modelování, renderování, animace, úpravy videa, VFX, kompozice, texturování, výstroje, simulace a tvorba her.

Jako další možné známé alternativy lze použít software Autodesk Maya, Autodesk 3ds Max a Maxon Cinema 4D. Jmenované programy od společnosti Autodesk lze získat na dočasnou dobu zdarma pod studentskou licencí.

Zdroj [1]

2.2 Adobe Photoshop CC

Adobe Photoshop je v současné době nejznámější profesionální software, který je specializovaný pro práci s rastrovou grafikou. V bakalářské práci je tento software použit pro tvorbu textur a ikonek pro uživatelské rozhraní.

Tento software byl vybrán, protože je standardem pro všechny, kdo chtějí zpracovávat digitální obrázky na vysoké úrovni, a vyznačuje se vysokým výkonem, řadou užitečných funkcí pro úpravu obrazu, intuitivním rozhraním a rozsáhlou škálou manuálů. Dalším důvodem je výuka s tímto editorem v rámci předmětu vyučovaného na této fakultě.

Jedinou nevýhodou je snad jen vysoká cena licence. Jako vhodný alternativní software lze použít například volně dostupný editor GIMP, který je volně distribuován pod právy Open source licence.

Zdroj [4]

2.3 Unity 3D

Unity 3D je nástroj používaný pro tvorbu jak 2D tak i 3D her od společnosti Unity Technologies. Dnes již podporuje spoustu platforem jako jsou například osobní počítače, mobilní telefony, weby, herní konzole a další.

Hlavním důvodem vybrání tohoto herního enginu je obrovská komunita a množství návodů, které pomůže začátečníkům při tvorbě nových her. Dále má rozsáhlou dokumentaci, a i jeho bezplatná verze obsahuje veškeré potřebné funkce, které budou potřeba pro vývoj vlastní hry. Krom toho je jeho grafické rozhraní uživatelsky přívětivé a práce s ním je skutečně pohodlná.

Jako alternativní herní engine lze použít například známý Unreal Engine nebo například GameMaker.

Zdroj [9]

2.4 Microsoft Visual Studio

Microsoft Visual Studio je vývojové prostředí od společnosti Microsoft určené pro vývoj aplikací v programovacích jazycích. V bakalářské práci je tento software použit pro psaní skriptů v jazyce C#.

Hlavním důvodem vybrání tohoto vývojového prostředí jsou integrované nástroje pro herní engine Unity 3D. Další výhodou je bezplatná studentská licence a velmi přehledné rozhraní.

Jako vhodné alternativní vývojové prostředí, které obsahuje nástroje pro herní engine Unity 3D lze použít vývojové prostředí MonoDevelop, které je dodáváno v základu spolu s enginem Unity 3D.

Zdroj [10]

3 MODELOVÁNÍ

V rámci počítačové grafiky se pojmem modelování rozumí vytváření 3D modelů pomocí vhodných počítačových nástrojů. Modelování tedy umožňuje přenést jakýkoliv objekt z reálného světa nebo představ grafiků do počítačové podoby.

3.1 Polygony

Polygony v 3D grafice značí jednoduché prostorové těleso, které se nachází ve 3D zobrazení. Polygon je složen z bodů (vertex), hran (edge) a plošek (face). Hrany představují přímky mezi jednotlivými body a ploška je vyplněná rovina. Nejjednodušší polygon musí mít alespoň 3 body a 3 hrany, který by v tom případě znázorňoval trojúhelník. Z topologického hlediska mohou existovat následující druhy polygonů.

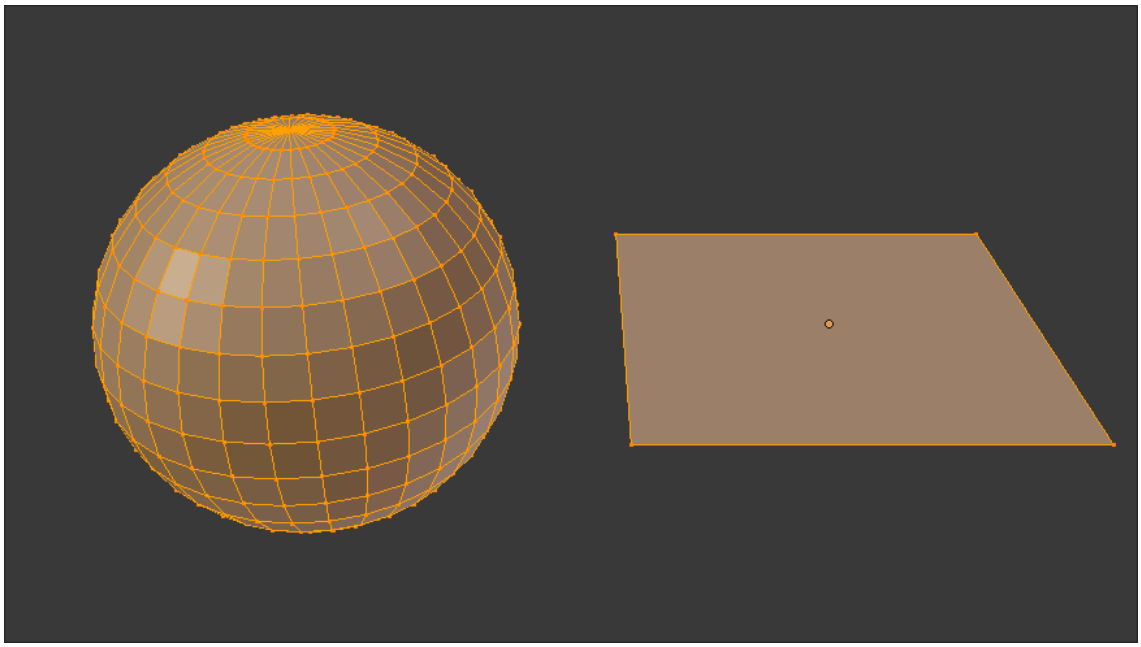
- Triangles (3 body)
- Quadrilateral (4 body)
- Pentagon (5 bodů)
- Hexagon (6 bodů)
- n-gon (n bodů)

Teoreticky může existovat polygon s libovolným vyšším počtem bodů než s dvěma body. V praxi se ale nejčastěji využívají Triangles nebo Quadrilateral polygony. Nejvhodnější druh polygonu pro animace je Quadrilateral, který je nazýván quad. Důležitým pravidlem je dodržení stejného druhu polygonu v rámci celého modelu. Grafik si tak usnadňuje práci a využití nástrojů.

Hry obvykle vyžadují trojúhelníkové polygony. Trojúhelníkové polygony se označují pozitivní vlastností, kde každý jednotlivý polygon je planární. Jakkoliv umístěné 3 body určují v prostoru vždy rovinu. Polygon určující rovinu je grafickými kartami mnohem snadněji a rychleji vykreslován. Praxe však dovoluje modelovat bez obav pomocí 4 bodů, kde lze snadno převést každý čtverec na 2 trojúhelníky.

Následující obrázek představuje objekt sestavený z jednotlivých polygonů a vpravo od něj samostatný polygon se čtyřmi body a vyplněnou ploškou.

Zdroj [11]



Obrázek 1: Ukázka polygonového objektu a samostatného polygonu.

3.2 NURBS

Geometrie NURBS je typ matematického modelu popisující tvar modelovaného objektu pomocí křivek. Objekt popsaný pomocí NURBS křivek má mnohem vyšší náročnost na výpočet. Pro počítačové hry je tento model kvůli náročnosti na výpočet nevhodný. Můžeme však objekt takto namodelovat a ve výsledné fázi převést na polygony.

Objekt NURBS má od polygonů rozdílné komponenty používané k manipulaci s objektem. Kontrolní vrcholy (control verticles), obalové křivky (hulls) a izoparmy (isoparms). Kontrolní vrcholy dále označovány jako CV jsou vrcholy, které neleží přímo na křivce, ale jejich změnou je ovlivněn tvar křivky. Obvykle se grafik snaží využít co nejmenší stupeň křivky, který ještě dostatečně popisuje daný tvar. Šetří tak výpočetní čas. Pro stupně se také používají následující slovní vyjádření.

- Lineární (1. stupeň)
- Kvadratické (2. stupeň)
- Kubické (3. stupeň)

Teoreticky může mít stupeň jakékoliv kladné číslo. Počet CV odpovídá stupně křivky + 1. Například přímka může být vyjádřena pomocí 2 CV.

Zdroj [6]

3.3 Modelovací techniky a nástroje

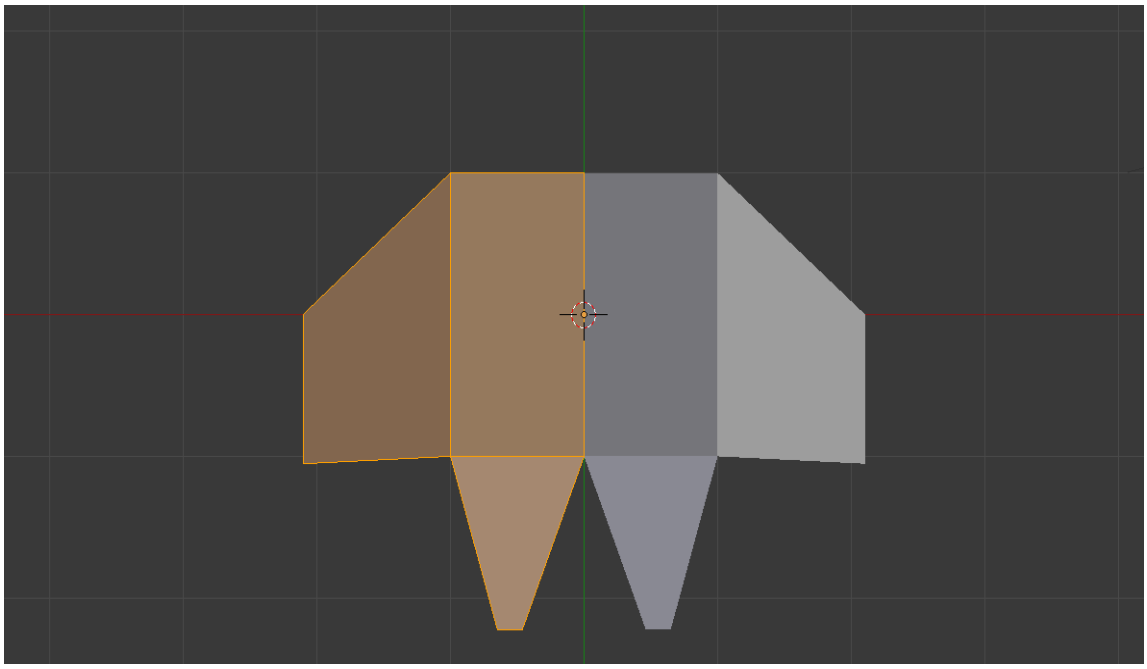
Zde budou popsány některé z populárních technik a nástrojů, které jsou k dispozici ve většině modelovacích softwarů.

3.3.1 Skládání primitivních objektů

Tato technika využívá předpřipravených primitivních objektů, které obsahuje námi používaný modelovací nástroj. Obvykle mezi tyto primitivní objekty patří alespoň koule, krychle a válec. Pomocí skládání těchto primitivních objektů lze snadno vytvořit některé jednodušší modely během velmi krátkého modelovacího času.

3.3.2 Zrcadlení

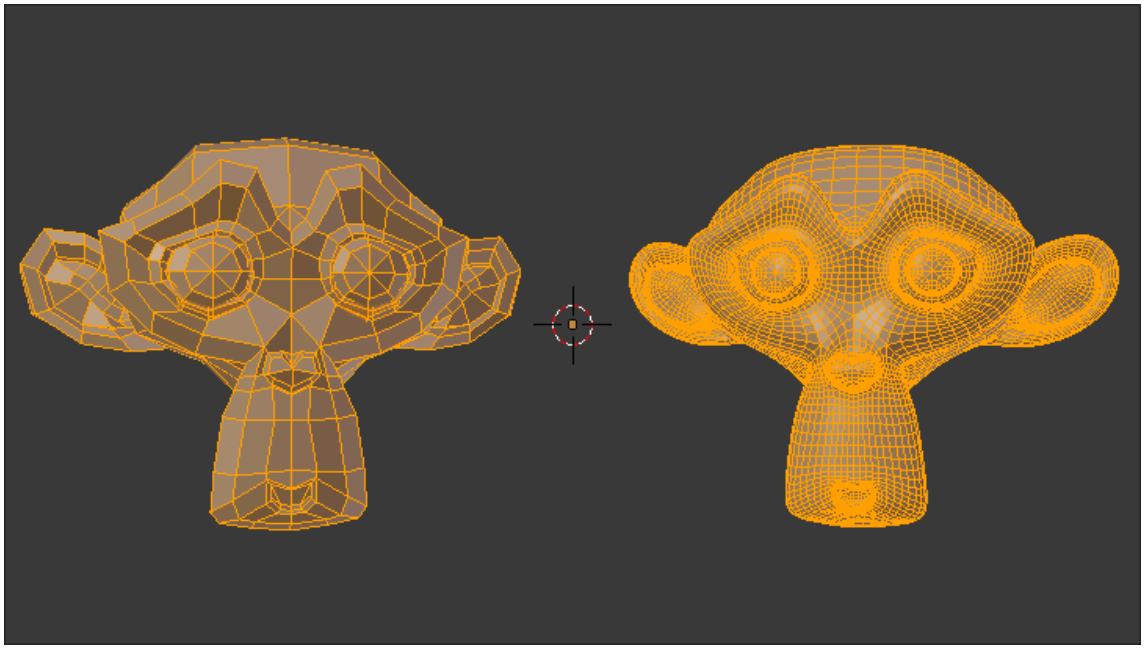
Zrcadlení patří mezi techniky, které zjednodušují práci při modelování symetrických objektů. Technika je velmi prostá, modelář potřebuje pouze správně nastavit takzvaný pivot point (vysvětlen v kapitole 3.4), který bude umístěn ve středu symetrie. Dále je nutno namodelovat polovinu objektu a následně využít nástroj zrcadlení, který vytvoří zrcadlově druhou stranu objektu.



Obrázek 2: Ukázka použití nástroje zrcadlení.

3.3.3 Dělení polygonů

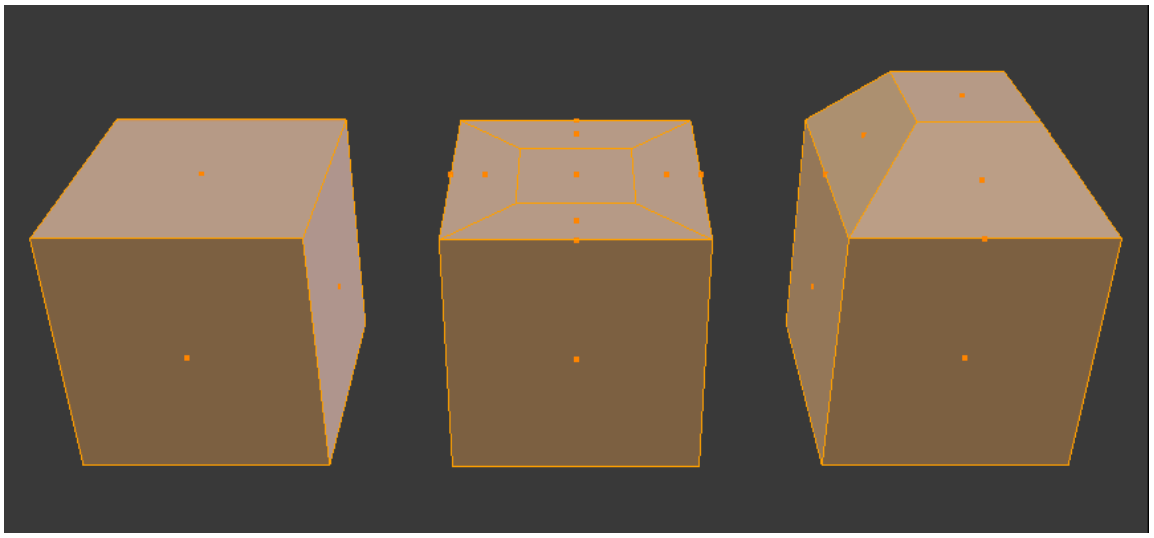
Dělením polygonů je označována operace, při které je vytvořena nová hrana přes plošku. Dělení může probíhat přes jeden polygon, nebo přes celý polygonální objekt. Použití této operace na celý objekt lze využít pro celkové navýšení detailů modelu.



Obrázek 3: Ukázka modelu s nízkým a vysokým počtem polygonů.

3.3.4 Vytažení (Extrusion)

Vytažení přidá geometrii na povrch mnohoúhelníku vytvořením nového povrchu, jeho následným vytažením a vytvořením nových ploch mezi vysunutým povrchem a původním povrchem. Pomocí tohoto nástroje lze vytahovat jak body, hrany tak i plošky. V následujícím obrázku je ukázka postupného vytažení horní plošky krychle.



Obrázek 4: Ukázka postupné vytažení krychle.

3.3.5 Modelování z primitivních objektů

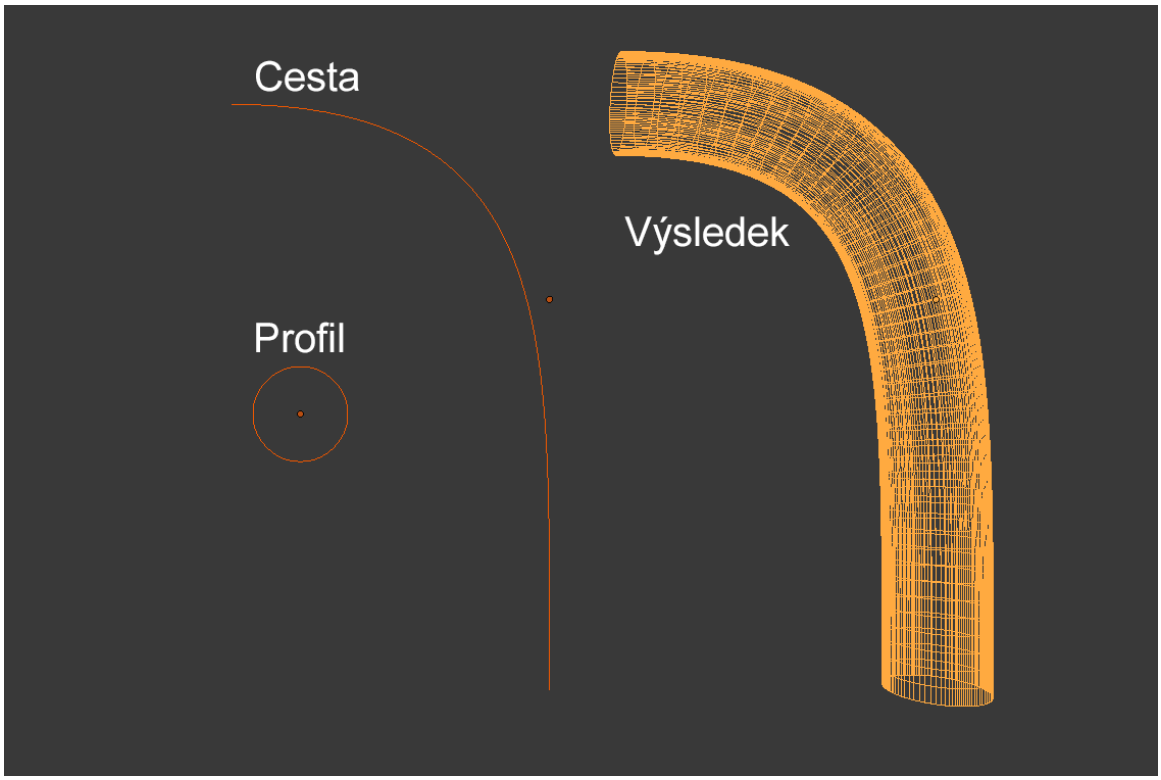
Tato metoda se podobá technice skládání primitivních objektů, kde namísto využití více primitivních objektů je rozšiřována geometrie pouze jednoho primitivního objektu. Často je tímto základním primitivním objektem krychle. Pomocí již předchozích zmíněných operací jako je vytažení a dělení polygonů je objekt rozšiřován, což dává grafikovy více geometrie pro tvarování modelu do požadované podoby.

3.3.6 Modelování pomocí křivek

Existuje mnoho způsobů, jak modelovat pomocí křivek. Zde bude vysvětleno jen několik typických technik, které mohou při správném použití velice usnadnit práci modelování některých specifických objektů.

Velmi užitečnou technikou je použít modifikátor křivky. Pro použití je potřeba jedna křivka jako cesta, která určí tvar výsledného modelu. Druhá křivka je použita jako takzvaný profil, který je pomocí modifikátor křivky nanesen podél celé námi vytvořené cesty. Tento nástroj velmi usnadní práci při modelování například vodovodního potrubí.

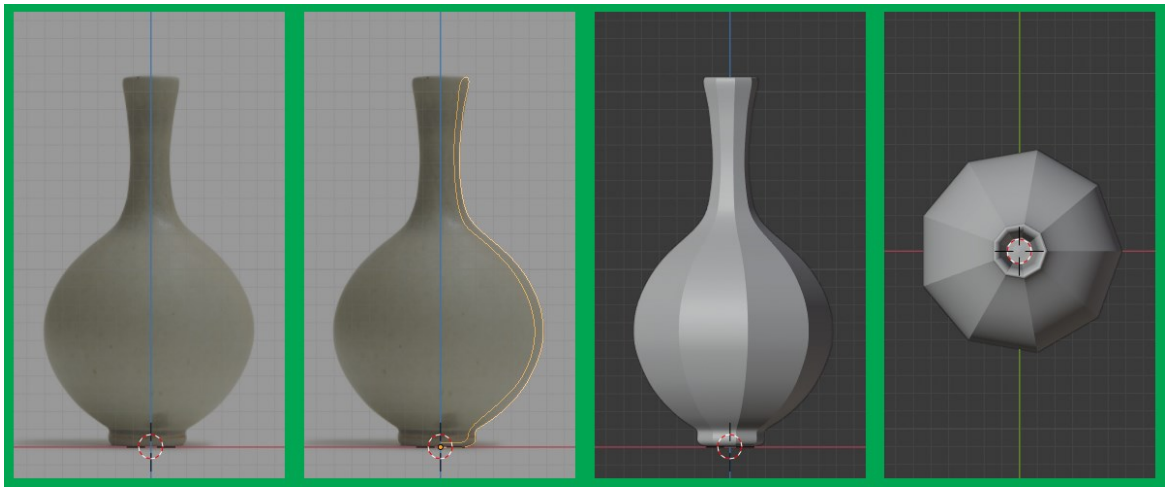
Zdroj [23]



Obrázek 5: Ukázka využití modifikátoru křivky.

Další velmi oblíbenou technikou je použití takzvaného nástroje Spin, který extruduje vybrané prvky a rotuje s nimi kolem konkrétní osy souřadnicového systému. Pomocí rotace křivky kolem určené osy lze snadno vymodelovat některá válcovitá tělesa jako je například váza nebo šachová figurka. Profilová křivka tvoří daný profil. Následně je potřeba zvolit osu rotace podle které bude křivka rotována. Oblíbeným postupem je nejprve vložit rastrový nebo vektorový obrázek předlohy a následně začít kreslit křivku podle obrysu obrázku. Po dokončení nákresu křivky mohou některé editory (například Blender) pro použití nástroje Spin nejprve požadovat převedení křivky na polygonální tvar. Následně lze aplikovat nástroj Spin, který vytvoří 3D model z profilové křivky. Následující obrázek znázorňuje postupný fáze tvorby modelu vázy pomocí této techniky.

Zdroj [19]



Obrázek 6: Postupné fáze použití nástroje Spin.

3.4 Pivot point

Každý vytvořený model obsahuje také bod, který určuje místo, podle kterého určité nástroje vykonávají své operace. Mezi tyto operace patří rotace, posun, změna měřítka modelu, zrcadlení a další. Lze měnit jak pozici pivot pointu, tak i jeho natočení.

Primitivní objekty mají automaticky nastavený pivot point ve svém středu. Změna tohoto bodu nám může být užitečná například pro animaci otevírání dveří, kde budeme chtít využít prostou rotaci kolem místa, kde by dveře měli umístěný závěs. Jednoduše přesuneme pivot point na bod určující místo závěsu a následně animujeme s využitím rotace.

4 LIMITY A OPTIMALIZACE

Pro jakoukoliv hru je potřeba dodržet určitou maximální výpočetní složitost. Hra totiž musí být vykreslovaná v reálném čase. Při tvorbě modelu musí grafik předem myslet na to, kde bude model umístěn, a z jaké vzdálenosti ho bude moci uživatel spatřit. Jestliže se například hráč k modelu skály nebude moci přiblížit na krátkou vzdálenost, je nežádoucí vytvářet model detailně propracovaný. Hráč by tyto detaily nijako nepoznal a hra by se stala náročnější na výpočet.

Zdroj [24]

4.1 Omezení polygonů

Vhodný počet polygonů, který by měl být použit pro postavu závisí na kvalitě kterou požadujeme, a hlavně na platformě, pro kterou daný objekt modelujeme. Jestliže hra bude obsahovat více stejných modelů postav v jedné scéně, bude kvůli výpočetnímu času potřeba snížit počet polygonů. Grafik musí dopředu počítat s tím, jestli jeho model bude vykreslován v reálném čase, nebo bude určen pouze pro film, kde se scéna vypočítá předem.

4.2 Optimalizace polygonů

Pro šetření výkonu je důležité, aby byl model vyvíjen už od počátku s co nejmenším počtem polygonů (tzv. low poly). Detaily lze přidávat později na místa, kde si jich hráč všimne. Jednoduchý model lze snadněji rozšířit na model s více detaily, než aby byl složitý model zjednodušen na jednoduchý model. Dodržování této zásady minimalizujeme množství polygonů, které by bylo na modelu zbytečné.

4.3 Využití textur pro optimalizaci

Textury lze využít jako nástroj pro zvláštní detaily, které nechceme na objektu modelovat. Mohou skrýt nedokonalé části polygonových objektů.

Jako příklad si představme poškrábanou dřevěnou stěnu. Místo aby grafik musel předělávat model stěny do požadované podoby, nanese na něj odlišnou texturu, která by zobrazovala škrábance. Takto vytvořený efekt na daném modelu není dokonalý, pro hráče je však dostačující. Stejnou technikou lze například vytvořit praskliny v zemi nebo jednoduché vrásky na obličeji postavy.

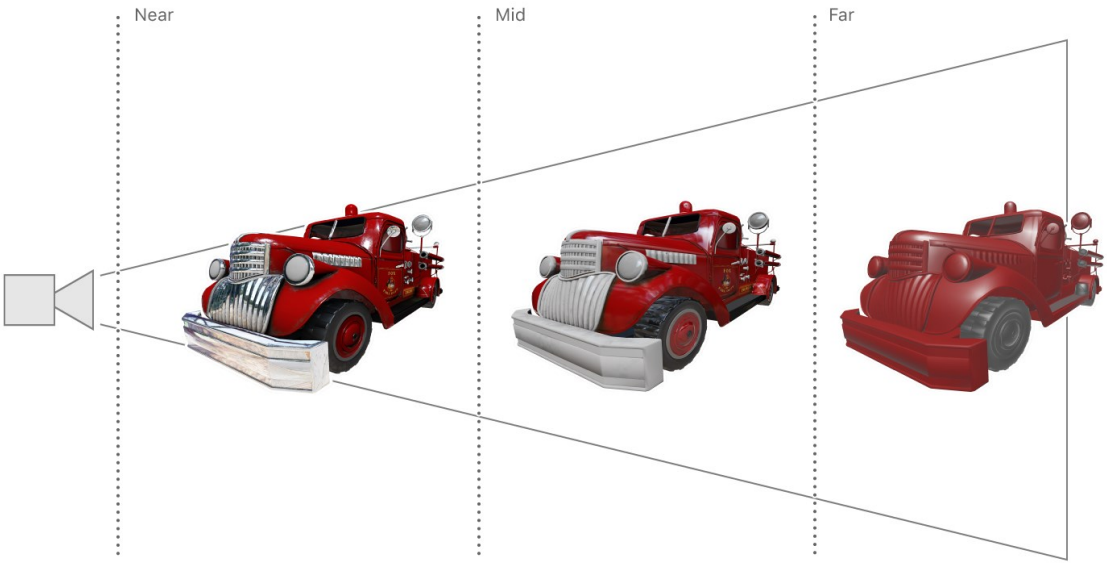
4.4 Level of detail

Tato technika výrazně snižuje potřebný výkon pro výpočet současného snímku v závislosti na pozici kamery. Obecně není důležité, aby byl model z velké vzdálenosti příliš detailní. Problém nastává v situaci, kdy dopředu nevíme, jak daleko se hráč bude nacházet od objektu. Může se na stejný objekt dívat z dlouhé ale i z krátké vzdálenosti.

Pro takové situace využijeme tuto techniku, která je nejčastěji známá pod zkratkou LOD. Český překlad znamená úroveň detailu. Princip této techniky je velmi prostý. Grafik vytvoří nejdříve jeden plnohodnotný model daného objektu. Následně vytvoří duplikát, který zjednoduší o detaily, které vyhodnotí za nepotřebné pro větší vzdálenost od kamery. Často se grafik snaží zjednodušit model přibližně o 50 %. Následně je tento postup několikrát opakován. Tímto je vytvořeno několik variant stejného modelu, kde každý model má jiný počet polygonů, tím pádem i jiné požadavky na potřebný výkon. Herní engine následně přepíná tyto modely podle vzdálenosti kamery od daného modelu.

V praxi je potřeba, aby si hráč co nejméně všiml okamžiku změny modelu. Je proto potřeba vhodně zvolit části modelu, které se při přechodu změní. Například při změně modelu tanku si hráč jistě všimne chybějícího kanónu nebo věže, ale naopak úbytek polygonů na pásech nemusí být vůbec patrný.

Zdroj [13]



Obrázek 7: Ukázka vykreslování modelu s využití LOD technikou.

[7] Obrázek převzat ze zdroje [13]

5 TEXTUROVÁNÍ

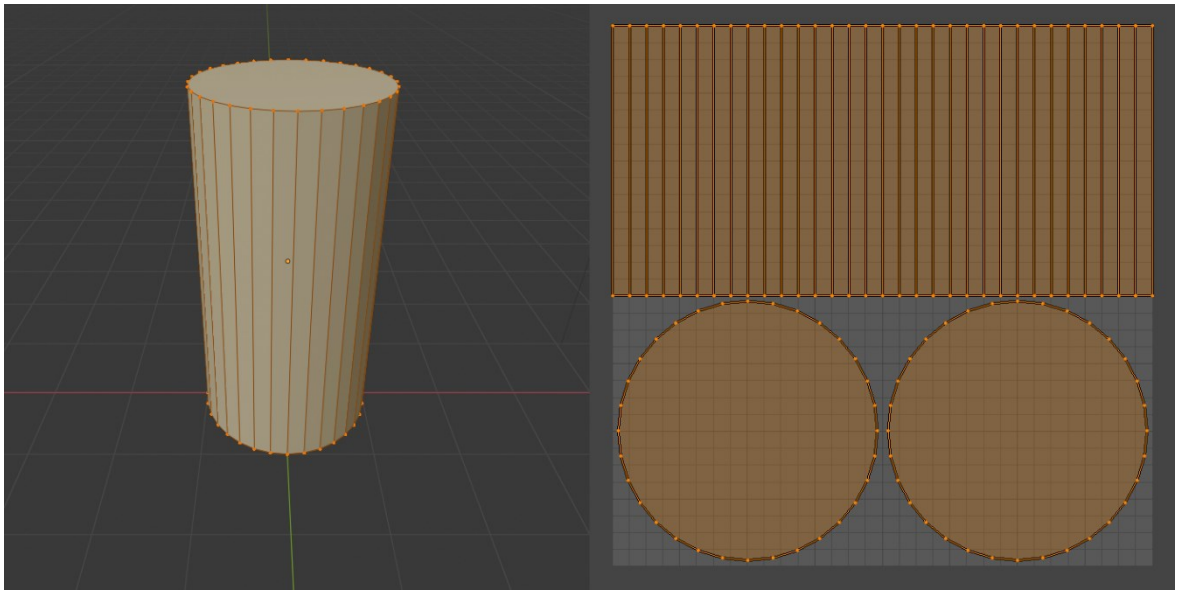
Texturování si lze představit jako oblečení, které se aplikuje na daný model. Jedná se o 2D obrázek, který je nanesen na model a určuje tak jeho barvu nebo vlastnosti. Všechny tyto parametry nelze vyjádřit jen jednou texturou, proto se jich na model aplikuje hned několik.

5.1 UV mapování

UV mapování je fáze, ve které je model připravován na texturování. Jedná se o převedení trojrozměrné sítě (mesh) do dvojrozměrné podoby takovým způsobem, aby se žádné části sítě nepřekrývali. Písmena U a V značí dvojrozměrný prostor stejně tak jako jsme zvyklí písmeny X a Y. Jelikož X a Y jsou již obsažena v 3D prostoru jako X, Y a Z, je tento 2D model nahrazen písmeny U a V. Model si uchovává informace o svém mapování, kde plocha uvnitř takto převedené sítě odpovídá místům, která budou na modelu pokryta texturou.

Editor dokáže sám velice slušně rozložit své primitivní objekty do UV sítě. Složitější tvary musejí být dále děleny grafikem na menší části. Na tyto části pak grafik může zvlášť nanést textury. Následující obrázek představu rozložení sítě válce do UV mapy.

Zdroj [14]



Obrázek 8: Ukázka rozložení sítě válce do UV mapy.

5.2 Co dělat s UV mapou modelu?

Jestliže je UV mapa našeho 3D objektu připravená, můžeme na ni nanést vlastní texturu. Velké množství modelovacích editorů umožňuje texturování přímo na námi vygenerovanou UV mapu. Tento způsob je však doporučen pouze pro jednodušší textury. Kvalitnějších výsledků lze dosáhnout exportováním UV mapy jako rastrový obrázek, který můžeme následně otevřít v libovolném grafickém editoru jako je například Adobe Photoshop a namalovat texturu v něm. Po namalování textury je nutné obrázek importovat zpět do našeho modelovacího editoru.

Zdroj [14]

5.3 Stínovač (Shader)

Stínovač obsahuje instrukce, které software používá k výpočtu efektů vykreslování a umožňuje definovat vzhled objektu a způsob, jakým se jeho povrch chová v konečném vykreslení. Vzhled může zahrnovat atributy, jako je barva objektu, okolní barva, průhlednost, odrazivost, lom, průsvitnost, vyzařování a mnoho dalších.

Zdroj [15]

5.3.1 Barva (Color)

Obsahuje informaci o barvách na objektu. Informace může obsahovat plnou barvu, plynulé přechody barvy nebo může být načítána z textury.

Zdroj [15]

5.3.2 Okolní barva (Ambience)

Okolní barva znázorňuje množství barvy, které ovlivní povrch objektu.

Zdroj [15]

5.3.3 Průhlednost (Transparency)

Určuje informaci o průhlednosti objektu, což znamená, jak moc bude objekt průhledný. Tímto efektem lze například vytvořit sklo.

Zdroj [15]

5.3.4 Odrazivost (Reflectivity)

Určuje, jak moc bude objekt odrážet světlo.

Zdroj [15]

5.3.5 Lámání světla (Refraction)

Lámání světla je termín používaný k popisu změny směru světla v důsledku změny jeho přenosového média. Jinými slovy, když světlo přichází do kontaktu s určitými povrchy, jako je voda nebo sklo, světlo je ohýbáno, protože tyto plochy ovlivňují rychlost, kterou světlo prochází.

To způsobuje zkreslení všeho, co je vidět za průhledným objektem. Například při pohledu skrz sklenici vody je vidět, že objekt za sklenicí je zkreslený.

Zdroj [15]

5.3.6 Vyzařování (Incandescence)

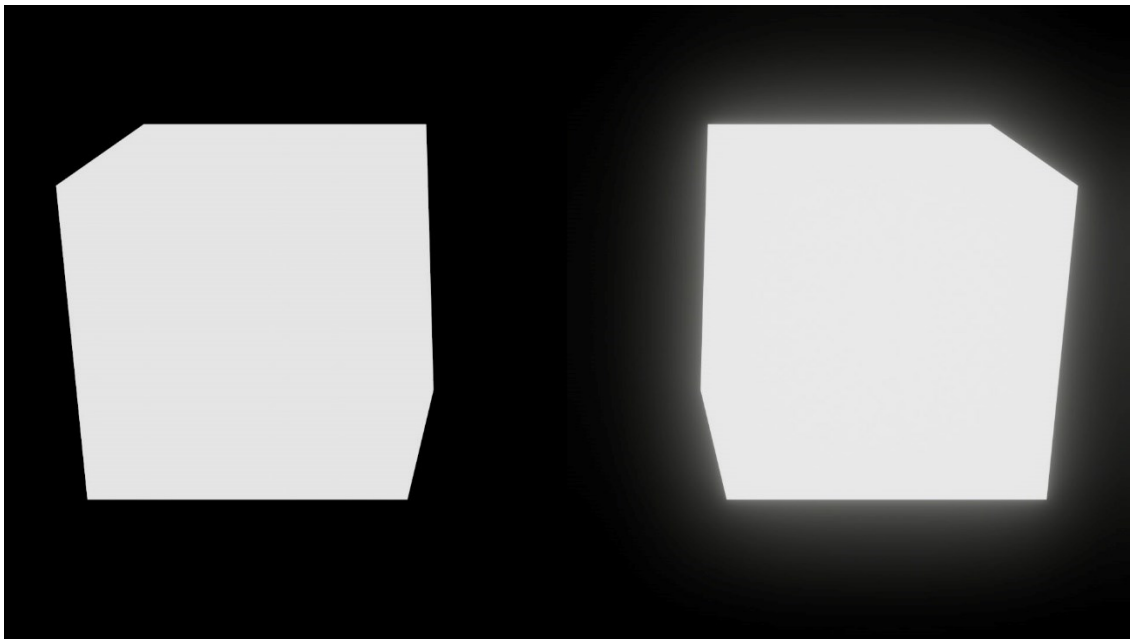
Vlastnost, která u objektu vyvolává dojem o vydávání vlastního světla. Toto světlo však neosvětluje okolí objektu. Ve skutečnosti pouze sjednocuje barvu objektu. Lze použít například pro vyvolání efektu zářivky.

Zdroj [15]

5.3.7 Záře (Glow)

Záře je vlastnost, pomocí které lze simulovat vlastní osvětlení. V následujícím obrázku je umístěna levá krychle bez využití efektu záře a pravá krychle s využitím efektu záře.

Zdroj [15]



Obrázek 9: Ukázka krychle s využitím efektu záře.

6 TVORBA 3D ANIMACÍ

6.1 Klíčování (Key frame)

Metoda klíčování využívá přechodu mezi klíčovými snímky. Klíčový snímek popisuje rozdíl vůči jinému snímku a uchovává hodnoty vlastností objektu jako je pozice, rotace a měřítko. Mezilehlé snímky jsou vypočítány v čase mezi těmito klíčovými snímky, aby vytvořily simulaci pohybu.

Zdroj [3]

6.2 Zakostění (Bones)

Kostra má za úkol manipulovat s částí modelu různou silou. Skládá se z řady virtuálních kloubů a kostí, které se grafik na modelu snaží co nejvíce napodobit objektu skutečného světa. Tato hierarchie propojených kostí je přiřazena k síti modelu a deformuje ji na základě polohy, měřítka a orientace každé kosti.

Zdroj [7]

6.3 Motion Capture

Motion Capture je procesem označující techniku pro tvorbu 3D animací, která využívá snímání pohybu skutečné živé entity a jeho následnou transformaci na digitální model. Tato technika je často využívána ve filmech, kde lze tímto způsobem kvalitněji zpracovat pohyb člověka včetně jeho mimiky obličeje. Animace postavy vytvořená touto technikou vypadá více realisticky a ušetřuje práci animátorů.

Herci jsou snímáni v optických systémech pomocí speciálně nalepených kuliček (markers), které jsou zaznamenávány speciálními kamerami. Tyto kamery snímají pouze kuličky. Každá kulička představuje na počítačovém modelu určitý bod. Další možností snímání pohybu je pomocí gyroskopů umístěných na individuálních částech těla herce.

Zdroj [25]

7 SKRIPTOVÁNÍ V UNITY 3D

V herním enginu Unity 3D jsou dostupné celkem 2 skriptovací jazyky. C Sharp dále označován jako C# a JavaScript dále označován jako JS. Bakalářská práce bude zaměřená na jazyk C#, kde znak „**//“** označuje řádek pro komentář. Dále bude vysvětlena základní myšlenka způsobu fungování herního enginu Unity 3D a ukázka důležitých funkcí, bez kterých by se programátor neobešel.

7.1 Základní myšlenka

Hra je spíše jako animace, kde jsou animační snímky generovány za běhu. Klíčovým konceptem v programování her je provádění změn polohy, stavu a chování objektů ve hře těsně před vykreslením každého snímku.

Zdroj [5]

7.2 Popis nejdůležitějších událostí (Events)

Pomocí událostí lze komunikovat mezi objekty. Dokáží zavolat námi zvolenou funkci při změně stavu tohoto objektu. Umožňují reakci hry na základě určitého vstupu jako je například stisknutí klávesnice, nebo detekce kolize dvou objektů.

Zdroj [5]

7.2.1 Start

Do funkce start zapisujeme takový kód, který má být vykonaný vždy před příchodem dané entity na scénu. Odborně řečeno při inicializaci objektu. Pro představu si lze například představit, že vyvíjíme strategickou hru a chceme, aby po výcviku nějaké postavy, například vojáka vydala tato postava zvukový efekt. Jednoduše do funkce Start vepíšeme kód, který spustí námi vybraný zvuk z pozice vojáka. Alternativou funkce start je funkce Awake, která může být spuštěna pouze jednou za celou dobu běhu scény. S implementováním zvukového efektu v události Awake by tedy v tomto příkladu pouze první vycvičený voják vydal zvukový efekt. Obě funkce jsou volány vždy před prvním zavoláním funkce Update, o které se píše v další kapitole.

- Start – Je volána vždy při inicializaci objektu.
- Awake – Je volána pouze jednou za hru při inicializaci objektu.

Zdroj [8]

7.2.2 Update

Funkce je prováděna každý snímek před veškerým vykreslením objektu. Programátor však musí počítat s tím, že tato funkce není volána vždy po uplynutí stejného časového intervalu. Hrají zde roli další pro nás náhodné jevy jako je náročnost výpočtu, které mohou tento čas ovlivnit. Pomocí vlastnosti Application.targetFrameRate lze nastavit takovou frekvenci volání funkce Update, o kterou se herní engine bude snažit.

Jestliže chceme provádět nějakou operaci při uplynutí vždy stejného časového intervalu, je potřeba do výpočtu zahrnout rozdíl časové jednotky od poslední vykonané funkce Update pomocí vlastnosti Time.deltaTime. Tato vlastnost poskytuje čas mezi aktuálním a předchozím snímkem.

Zdroj [8]

```
void Update() {
// vlastní skript
}
```

7.2.3 FixedUpdate

Funkce je volána nezávisle na snímkové frekvenci pro výpočet fyziky a to vždy po uplynutí stejného časového intervalu. Ve výchozím nastavení je funkce FixeUpdate volána jednou za 0,02 sekund tedy 50krát za sekundu. Změnu počtu volání za sekundu lze provést ve skriptu změnou hodnoty vlastnosti Time.fixedDeltaTime, nebo přímo v možnostech editoru nastavení scény.

Zdroj [8]

```
void FixedUpdate() {
// vlastní skript
}
```

7.2.4 LateUpdate

Funkce LateUpdate funguje stejným způsobem jako funkce Update, a to s jednou důležitou změnou. Skript v události je vykonávaný až po dokončení animace. Tato událost může být použita například pro změnu pozice kamery, kterou chceme aplikovat až po dokončení animace objektu, na který je kamera umístěna.

Zdroj [8]

```
void LateUpdate() {
// vlastní skript
}
```

7.2.5 Reakce na uživatelský vstup

Je možné také detekovat události myši, ke kterým dochází přes herní objekt umístěný ve scéně. To lze použít například k zobrazení informací o postavě, která je aktuálně pod ukazatelem myši. Stejně tak lze reagovat na grafické prvky uživatelského rozhraní. Následující funkce reagují na kliknutí myší za odlišných podmínek.

Zdroj [5]

- OnMouseDown – Je volána, když uživatel stiskne tlačítko myši nad objektem.
- OnMouseUp – Je volána, když uživatel uvolní tlačítko myši nad objektem.
- OnMouseEnter – Je volána, když myš začne být v kolizi s objektem.
- OnMouseExit – Je volána, když myš přestane být v kolizi s objektem.
- OnMouseOver – Je volána každý snímek, když je myš nad objektem.
- OnMouseDrag – Je volána každý snímek, když bylo tlačítko myši stisknuté nad objektem a stále nebylo uvolněno.

7.2.6 Fyzikální události

Fyzikální události nám velice usnadňují práci s herní mechanikou. Lze například detekovat kolizi dvou objektů. S událostmi lze různě pracovat. Do funkce je předán parametr poskytující podrobnosti o kolizi (poloha, identita přicházejícího objektu atd.). Následující události jsou vyvolány, jakmile dojde ke kolizi dvou objektů za odlišných podmínek.

- OnCollisionEnter – Je volána jednou při detekci kolize.
- OnCollisionStay – Je volána jednou za každý snímek během kolize.
- OnCollisionExit – Je volána jednou při ukončení detekce kolize.

Následující události budou vyvolány tehdy, když dojde ke kolizi collideru nakonfigurovaného jako spínač (trigger). Collider je zvláštní druh objektu, který definuje tvar pro kolize. Nemusí mít stejný tvar jako je tvar samotného objektu. Spínač je možné nastavit jako vlastnost collideru. Následující funkce jsou volány za stejných podmínek jako předchozí zmíněné, ovšem s podmínkou nastaveného spínače collideru objektu.

Zdroj [5]

- OnTriggerEnter – Je volána jednou při detekci kolize.
- OnTriggerStay – Je volána jednou za každý snímek během kolize.
- OnTriggerExit – Je volána jednou při ukončení detekce kolize.

Existují také obdobné zjednodušené funkce pro detekci kolize, které fungují pouze v rovině. Výhoda těchto funkcí je zjednodušený algoritmus výpočtu. Funkce jsou tedy méně náročný na výkon a také rychlejší.

Zdroj [8]

- OnCollisionEnter2D
- OnCollisionStay2D
- OnCollisionExit2D
- OnTriggerEnter2D
- OnTriggerStay2D
- OnTriggerExit2D

7.3 Proměnné

Proměnné se dají představit jako stavební bloky kódu. Při vytváření hry je často potřeba ukládat informace jako je například jméno hráče, počet získaných bodů, aktuální stav životů postavy, zbývající čas a mnoho dalších. V Unity 3D lze navíc využít proměnnou jakou vlastnost, která je modifikovatelná přímo v herním inspektoru.

Zdroj [21]

7.3.1 Primitivní datové typy

Primitivní datové typy reprezentují elementární údaje, které udávají konkrétní hodnotu. Jejich výčet závisí na použitém programovacím jazyku. Následující seznam představuje často používané primitivní datové typy pro jazyk C#.

- bool – Logický datový typ (boolean v jazyku JS).
- string – Řetězec znaků (String v jazyku JS).
- char – Jeden znak.
- byte – Celé číslo (8 bitové).
- short – Celé číslo (16 bitové)
- int – Celé číslo (32 bitové).
- float – Desetinné číslo (32 bitové).
- double – Desetinné číslo (64 bitové).

V JS dále existuje speciální datový typ var, který se dokáže automaticky přizpůsobit až během přiřazování. Pro lepší pochopení je zde uveden následující příklad.

```
var name;

name = "Jan";
```

První řádek kódu deklaruje proměnou pojmenovanou name. Druhý řádek kódu přiřazuje řetězec Jan do proměnné name. S proměnou name by se tedy dále pracovalo jako by byla datový typ string.

Zdroj [20]

7.4 Vlastnosti

Vlastnost je speciální proměnná objektu, která může být za určitých podmínek upravovatelná za běhu hry v samotném inspektoru herního enginu. Unity vytvoří štítek v inspektoru zavedením mezer, kde se v názvu proměnné nachází velké písmeno a s velkým písmenem na začátku. Toto je však čistě pro účely zobrazení, ve zdrojovém kódu by měl být používán skutečný název proměnné.

Následující skript obsahuje vlastnost objektSize, která je veřejná. Neveřejná vlastnost není zobrazována v herním inspektoru. Tento kód tedy vytvoří v inspektoru upravitelné pole označené „Objekt Size“. Dále skript vypíše velikost objektu do konzole při inicializaci objektu.

Zdroj [8]

```
using UnityEngine;
using System.Collections;

public class MainPlayer : MonoBehaviour
{
   public int objectSize;

   // Use this for initialization
   void Start ()
   {
      Debug.Log("Size is " + objectSize + " meters");
   }
}
```

7.5 Atributy

Atributy jsou značky, které mohou být umístěny nad třídou, vlastností nebo funkcí ve skriptu. Tyto značky se vepisují do hranatých závorek a určují zvláštní chování. Následující atribut zamezuje zobrazení vlastnosti description v inspektorovi.

```
[HideInInspector]
public string description;
```

Zdroj [17]

7.6 Vlastní funkce

Jazyk C# podporuje psaní vlastních funkcí s následující syntaxí. Funkce začínají návratovým typem na začátku, následovaným názvem funkce a poté parametry v závorkách (pokud existují). Názvy funkcí začínají velkým písmenem a tělo funkce se nachází mezi složené závorky. Zde je příklad bezparametrické funkce, která nevrací žádný návratový typ. Klíčové slovo void značí nepoužití návratového typu.

```
void MyFunction () {
// tělo funkce
}
```

V dalším příkladu je uvedena funkce s návratovým typem pro sečtení dvou celých čísel. Parametry jsou dvě celá čísla, které má funkce sčítat. Návratový typ je int (celé číslo). Klíčové slovo return udává, co má funkce vrátit.

```
int Sum (int x, int y) {
return x + y;
}
```

Funkce, která by vypočítala vzdálenost dvou bodů v prostoru a tento výsledek navrátila by vypadala následujícím způsobem.

```
float Distance3D(int x1, int y1, int z1, int x2, int y2, int z2) {
return Mathf.Sqrt(Mathf.Pow(x1 - x2, 2) + Mathf.Pow(y1 - y2, 2) +
Mathf.Pow(z1 - z2, 2));  }
```

Definované funkce můžeme volat jejich názvem a doplněných parametrů které definuje. Předchozí funkci pro výpočet vzdálenosti dvou bodů v prostoru lze zavolat následujícím způsobem.

```
Distance3D (p1, p2);
```

Takto zavolaná funkce však nikam nezapíše svůj výsledek. Pro uložení stavu výpočtu lze použít proměnou. V následující ukázce je výpočet metody Distance3D uložen do proměnné distance, která je o řádek výše deklarována.

```
float distance;
distance = Distance3D (p1, p2);
```

7.7 Třídy

Zde budou představeny některé z nejdůležitějších tříd, které se běžně používají při skriptování v Unity 3D. Pokrývají některé z klíčových oblastí skriptovacích systémů a poskytují dobrý výchozí bod pro vyhledávání dostupných funkcí a událostí.

Zdroj [21]

7.7.1 MonoBehaviour

Výchozí třída pro všechny nové skripty, třída MonoBehaviour poskytuje seznam všech funkcí a událostí, které jsou k dispozici pro standardní skripty připojené k herním objektům. Obsahuje veškeré interakce a kontroly nad jednotlivými objekty ve scéně.

Zdroj [21]

7.7.2 Transform

Třída Transform obsahuje funkce pro práci s polohou, rotací a měřítkem herního objektu v prostoru nebo rovině.

Zdroj [21]

7.7.3 Rigidbody

Pro většinu herních prvků je dostupná základní sada nástrojů pro pohyb a rotaci objektů, detekci kolizí a spínačů a také použití sil. Třída Rigidbody poskytuje všechny vlastnosti a funkce pro tyto účely v prostoru.

Zdroj [21]

7.7.4 Rigidbody2D

Třída Rigibody2D je třída pro stejné účely jako třída Rigibody, ovšem pro práci v rovině.

Zdroj [21]

8 VLASTNÍ TVORBA ASSETŮ PRO HERNÍ ENGINE UNITY

Vhodným začátkem je nejprve prozkoumat samotnou originální hru a vybrat z ní takové prvky, ze kterých budeme chtít vytvořit assety do herního enginu Unity 3D.

V bakalářské práci budou vytvářeny assety pro nejznámější mapu ze hry Bulánci pojmenovanou „Na dobrou noc“. Pro tvorbu assetů byl vybrán skript pro střelbu a chůzi, dále velký domek ve tvaru muchomůrky, kmen poraženého stromu, kamení, velký a malý keř, zbraně, travnatý terén a v neposlední řade samotný Bulánek, který je ve hře označován jako extrémně agresivní polštář.

Obrázek 10: Ukázka originální mapy „Na dobrou noc“.

8.1 Nastavení kamery

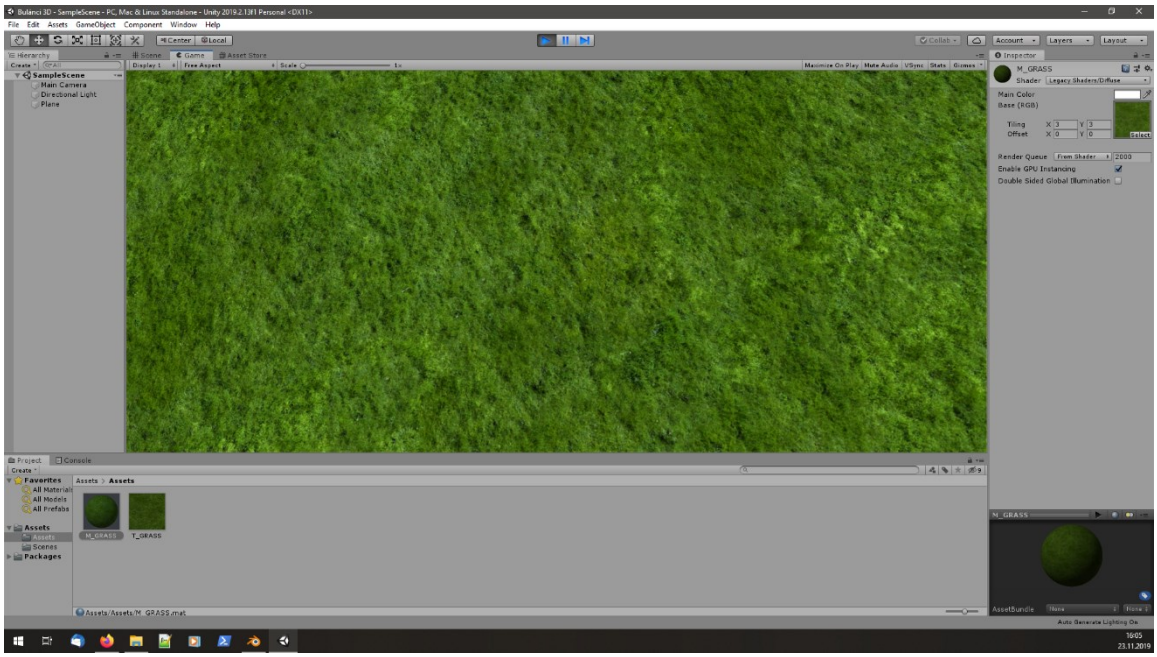
Kamera pro hru bude umístěna staticky nad scénou. V herním enginu tedy nastavíme pouze pozici a úhel, který bude kamera svírat. Není třeba žádný dodatečný skript, který by s kamerou manipuloval během hry.

8.2 Tvorba terénu

Terén ve hře bude uskutečněn pouze pomocí jedné polygonové plochy a vhodné textury. Jelikož se jedná o triviální model, není nutno jej tvořit žádným externím programem. Implementace terénu snadno provedeme přímo v herním enginu.

Do scény přidáme polygonovou plochu, kterou roztáhneme do potřebného rozměru. Dále je potřeba vytvořit materiál, na který naneseme námi zvolenou texturu. Aby naše textura vypadala důvěryhodně, je potřeba aby byla v takzvaném seamless tvaru. Seamless tvar označuje plynulou návaznost mezi koncem a začátkem textury. Můžeme tedy texturu po objektu nanášet s opakováním tak, aniž by bylo patrné místo spojení.

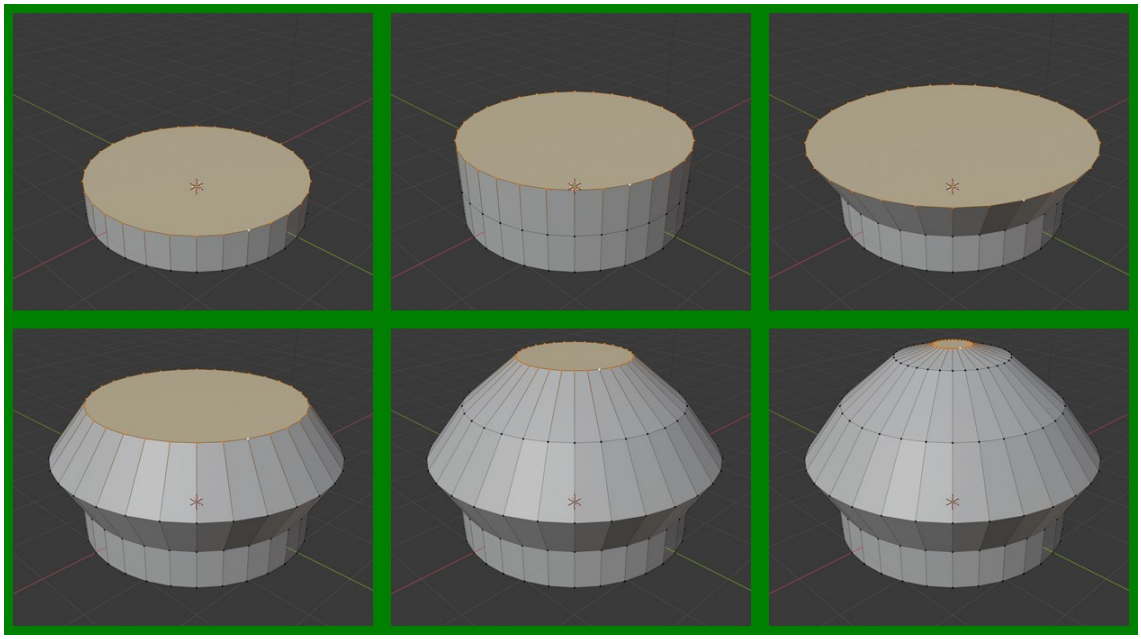
Po aplikování materiálu na plochu je potřeba texturu dodatečně nastavit. Stínovač materiálu byl nastaven jako „Diffuse“. Důležité parametry pro nás budou Tiling a Offset. Oba parametry se dají zvlášť nastavit pro souřadnici X a Y. Tiling určuje velikost dlaždice, pro kterou je textura opakována. Offset pouze posouvá počátek textury. V následujícím obrázku je ukázka výsledného terénu s nastavením v Unity 3D.



Obrázek 11: Výsledný terén s nastavením stínovače v Unity 3D.

8.3 Tvorba domku ve tvaru muchomůrky

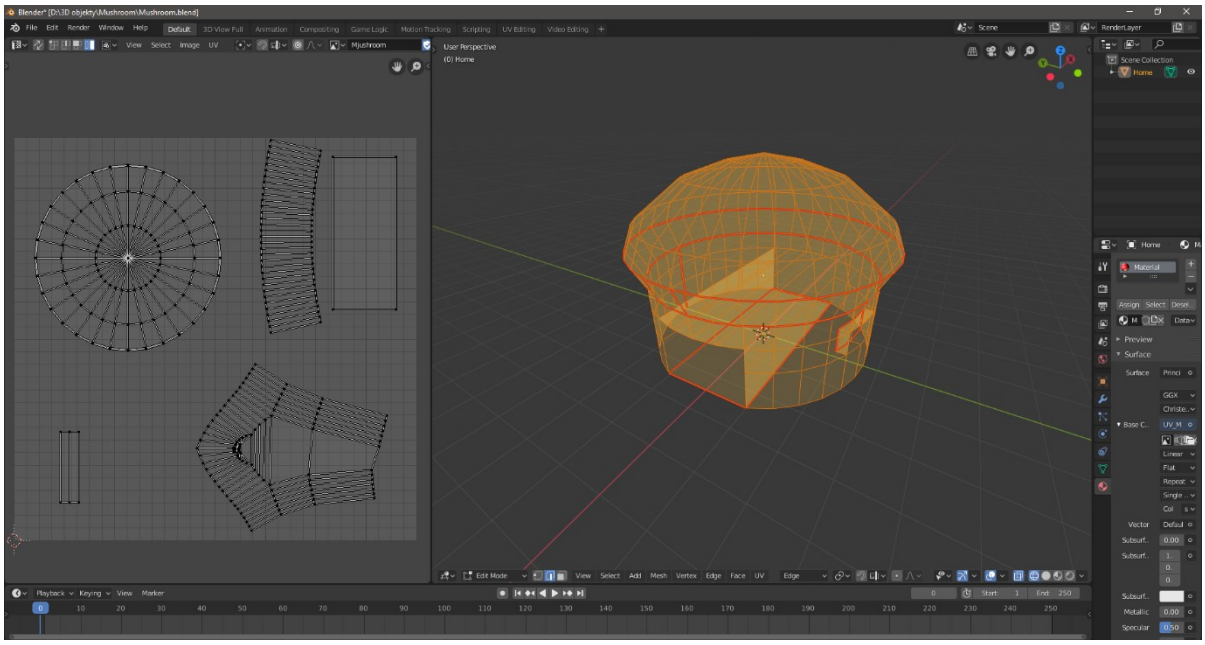
Na model muchomůrky byl použit modelovací software Blender. Jako výchozí primitivní tvar pro modelování byl použit válec. Válec byl nejprve zmenšen v ose Z a následně rozšiřován pomocí nástroje vytažení. Po každém vytažení byla změněna velikost obvodu kruhu. Tímto způsobem byl postupně modelován klobouk muchomůrky.



Obrázek 12: Postupné vytažení válce na model houby.

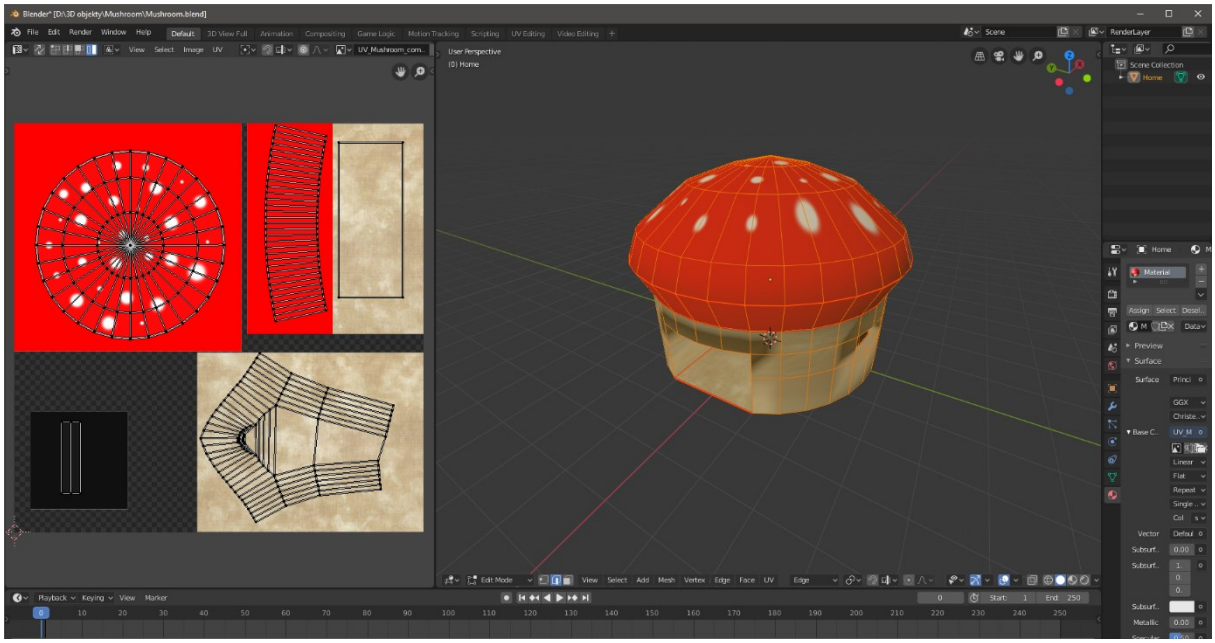
Originální tvar domku v muchomůrce ve hře Bulánci obsahuje ještě okno a přístupový vchod, ve kterém lze ve hře procházet s postavou. Vymodelování vchodu i okna docílíme snadno pomocí vymazání plošek v místech, kde se tyto prvky vyskytují. Dále pomocí vytažení rozšíříme okolní hrany dovnitř muchomůrky. Prázdný prostor v okně vyplníme ploškou, která bude později texturována tmavým odstínem šedi.

Po dokončení úprav na modelu je potřeba nanést texturu na model. Toto již není primitivní tvar, který modelovací software dokáže snadno převést na UV mapu. Jednotlivé části jako je okno, klobouk houby a podlaha je potřeba rozdělit a namapovat zvlášť. Následující obrázek představuje dokončený model s UV mapováním.



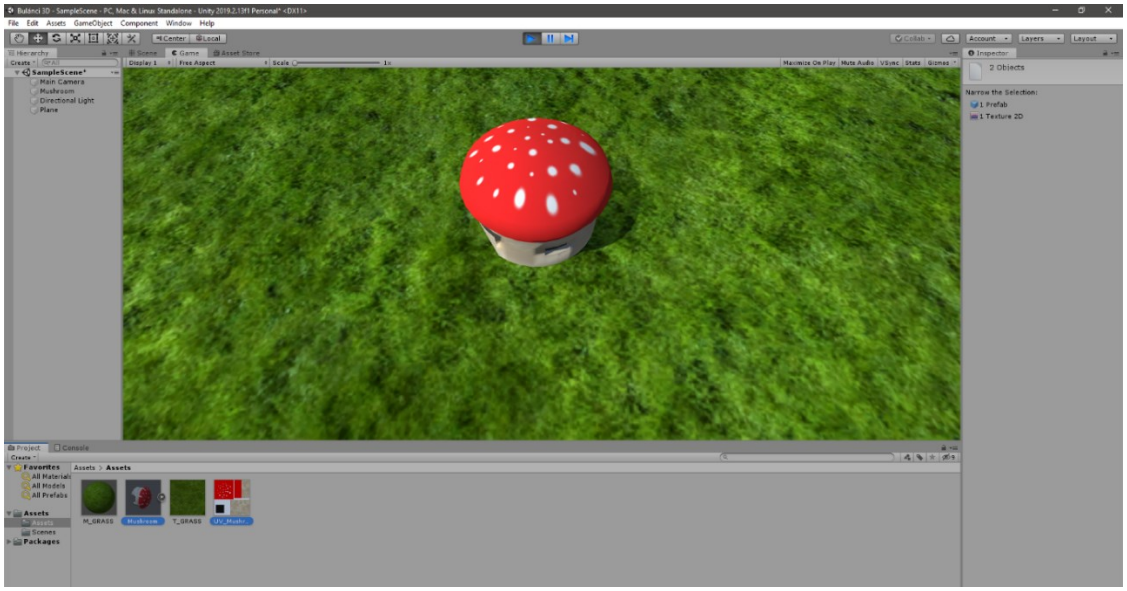
Obrázek 13: Model domku v Muchomůrce s UV mapou.

UV mapu našeho modelu je nyní potřeba exportovat jako rastrový obrázek, který je možné dále upravovat v libovolném grafickém editoru jako je Adobe Photoshop. Jednoduše na místa, která představují části modelu naneseme vhodnou barvu nebo texturu a obrázek následně importujeme zpět do modelovacího softwaru Blender.



Obrázek 14: Model domku v Muchomůrce s importovanou texturou.

Po úspěšné aplikaci textury zbývá exportování modelu do projektu v herním enginu Unity 3D. Model v Blenderu je nejprve potřeba exportovat do formátu .fbx, formát .fbx obsahuje pouze syrová data o modelu. Jeho výhodou je tedy nezávislost na modelovacím softwaru a také menší velikost souboru. Nevýhodou je však nutnost uchování původního souboru (Blender má formát .blend). Následně je potřeba soubor importovat do projektu v herním enginu Unity 3D spolu s texturou, která k modelu patří. Po těchto operací lze model libovolně používat v projektu jako asset. Herní engine automaticky rozdělí model na trojúhelníkové polygony.

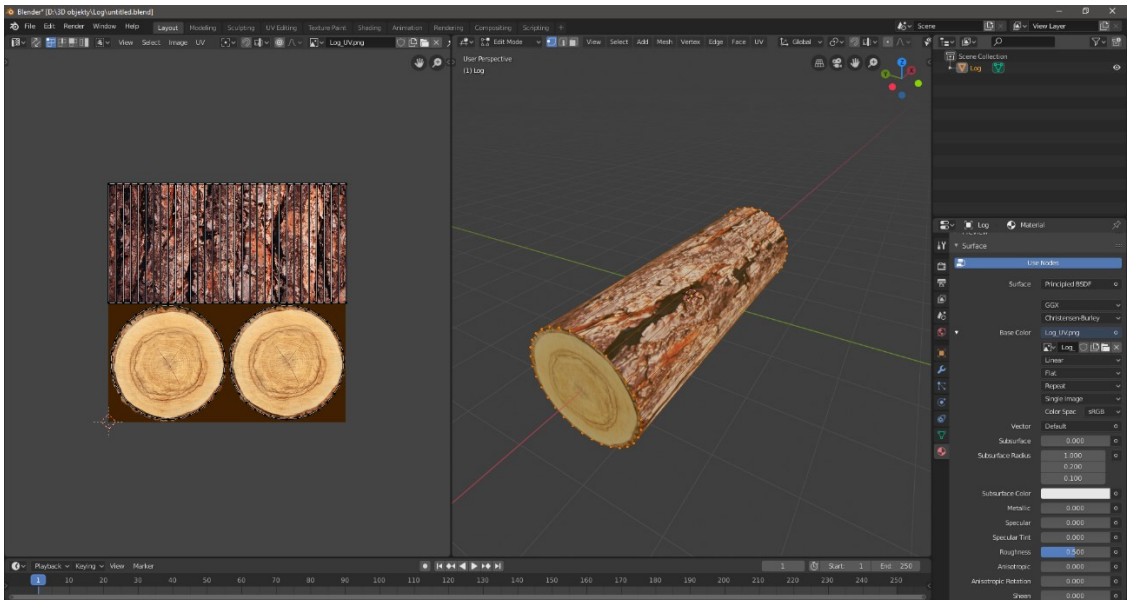


Obrázek 15: Použití importovaného assetu v Unity 3D.

8.4 Tvorba objektů prostředí

Na objekty prostředí bude opět použit modelovací software Blender. Jedná se pouze o statické modely, které ve scéně nepotřebují žádné animace ani skripty. V herním enginu bude potřeba nastavit vhodný collider, který zamezí hráči pohybu skrz model. Mezi modelované objekty prostředí patří kmen stromu, keř a kamení.

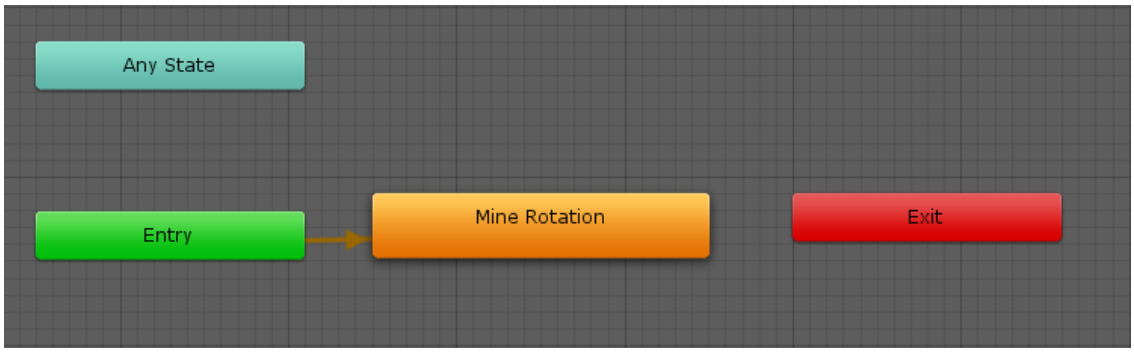
Na objekty prostředí byla použita technika modelování z primitivního objektu. Na model kamení je nejvhodnější primitivní objekt koule, u kterého stačí pouze měnit měřítko v různý ose a dále měnit pozice vrcholů a hran. Nejsnazší cesta k namodelování kmenu stromu je vycházet z primitivního objektu válce. Válec je potřeba pouze pootočit do požadovaného úhlu a rozšířit u kmenu. Jelikož se jedná o primitivní objekty, modelovací software dokáže sám velice dobře rozložit síť do UV mapy. Dále je potřeba nanést texturu stejným způsobem, jako při tvorbě domku v muchomůrce.



Obrázek 16: Kmen stromu.

8.5 Předměty

Předměty ve hře představují zbraně, které hráč může sebrat a zlepšit tak svoje šance na výhru. Pro všechny tyto modely byla použita jednoduchá animace pomocí metody klíčování, která rotuje s modelem o 360° kolem osy Z za 160 snímků. Pivot Point všech zbraní byl potřeba nastavit do středu modelu. Aby rotace byla animována stálou rychlostí, je potřeba v Blenderu nastavit tvar křivky animace v Grafickém editoru jako lineární. S výchozím nastavením by rotace opakovaně zrychlovala a zpomalovala. V herním enginu je dále potřeba spustit animaci. Animaci lze spustit přímo ve skriptu, nebo pomocí GUI komponenty zvané Animator Controller. Pomocí této komponenty lze napojit animace s událostí, při které má být animace spuštěna. Takto vytvořený Animator Controller stačí pouze aplikovat na daný model. Následující obrázek představuje napojení animace rotace při inicializaci objektu.



Obrázek 17: Ukázka napojení animace v Animator Controller.

8.6 Tvorba postavy

Ve hře Bulánci jsou hlavní postavy agresivní polštáře, za které má hráč možnost hrát. Originální hra omezuje maximální počet hráčů pouze na čtyři. V tvorbě assetu si tedy můžeme dovolit použít trochu více polygonů pro kvalitnější detaily modelu postavy.

Modelování postavy je značně složitější proces než předchozí již zmíněné modely. Je potřeba zkombinovat více technik. Začneme modelováním obyčejného polštáře, kterému následně doděláme ruce, obličej a oči. Na základní tvar polštáře byla použita opět technika modelování z primitivního objektu, v tomto případě z krychle. U krychle změníme měřítko v ose Z na 0.03. V dalším kroku je potřeba rozšířit geometrii modelu pro lepší možnost zpracování detailů.

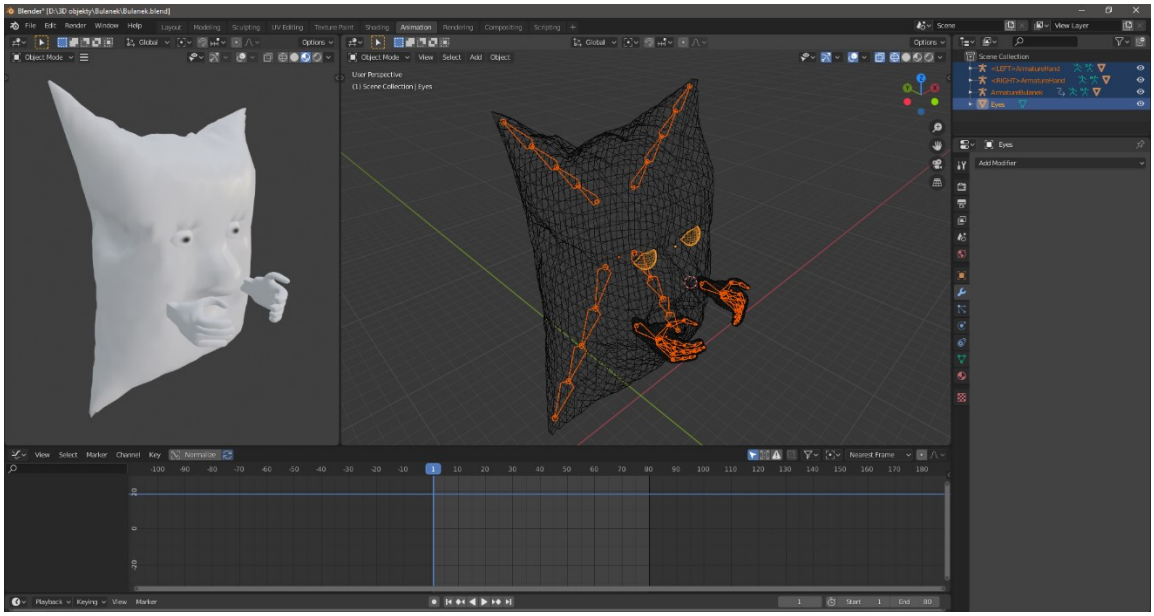
Blender obsahuje nástroje pro práci s fyzikální silou. Do středu modelu nyní vložíme silové pole (Blender označuje tuto komponentu jako Force), které necháme působit na model. Silové pole obsahuje velké množství vlastností k nastavení. Tato metoda je spíše experimentální. Dopředu člověk neodhadne přesný výsledek. Dále spustíme animaci, ve které se model začne roztahovat. V okamžiku, kdy je náš model v nejbližší podobě polštáře animaci zastavíme a necháme efekt aplikovat na model.

Základní model polštáře je tímto hotov. Následně je potřeba vytvarovat obličej. Jako oči použijeme samostatné primitivní tvary koule, které umístíme do vymodelovaných důlků a aplikujeme na ně nástroj obsažený v modelovacím softwaru Blender, booleovský modifikátor. Booleovský modifikátor ořízne zbytečné polygony koule uvnitř modelu polštáře, což je pro hru přínosem z hlediska optimalizace výkonu.

Pro dosažení kvalitní animace u tohoto modelu nebude stačit pouze pomocí samotného klíčování. Model je potřeba zakostit. Zakostíme postupně každý roh polštáře. Pohybem virtuálních kostí pak získáme mnohem důvěryhodnější pohyb, než kdybychom hýbali zvlášť s jednotlivými částmi polygonů. Následně použijeme techniku klíčování, kde však ukládáme stav jednotlivých virtuálních kostí, které manipulují s modelem.

Dále namodelujeme a zakostíme ruce. Kostra u rukou však nebude kvůli samotné animaci. Z pohledu kamery by pohyb rukou nebyl téměř vůbec patrný. Pomůže nám však dostat tvar ruky do požadovaného tvaru. Před importem modelu do herního enginu můžeme zachovat stav tvaru rukou a kostru následně odstranit.

Ve hře budou potřeba odlišné textury modelu. Každý hráč by měl mít svoji vlastní jedinečnou barvu postavy. Texturovat zde budeme pouze oči. Ostatní části modelu necháme s výchozím nastavením. Jejich výsledný materiál bude nanesen až v herním enginu, ve kterém bude vytvořeno několik barevných variant tohoto modelu.



Obrázek 18: Výsledný model postavy Bulánka.

Po namodelování postavy Bulánka je dále potřeba vytvořit skript, pomocí kterého bude mít hráč možnost ovládat postavu. Skript bude psán v jazyku C#. Ve funkci Update budeme odchytávat stisknutí určité klávesy. V originální hře má hráč možnost pohybovat se pouze do čtyř stran (nahoru, dolů, doleva, doprava). Pro natočení postavy na správnou stranu lze použít třídu Transform. Konkrétně vlastnost transform.eulerAngles, do které načteme požadovaný vektor s určitým směrem. Například pro natočení postavy do směru v ose X při stisknutí tlačítka D napíšeme do funkce Update následující kód.

```
Void Update()
{
if (Input.GetKey(KeyCode.D))
        {
        Transform.eulerAngles = new Vector3(0, 90, 0);
        }
}
```

Dále je potřeba během stisku některého z tlačítek určeného k pohybu spustit animaci chůze a transformovat pozici postavy dopředu v závislosti směru natočení postavy. K pohybu lze využít metodu Translate, která je také dostupná ve třídě Transform.

```
public void Transform.Translate(Vector3 translation);
```

Jelikož máme postavu již předem natočenou na správný směr, do parametru translation vložíme Vector3.forward, který představuje Vector3(0, 0, 1). Aby byl pohyb stálý a nezávislý na odchylkách volání funkce Update, vynásobíme tento vektor vlastností Time.deltaTime, která již byla zmíněna v předchozích kapitolách. Dále je potřeba vyladit vhodnou rychlost pohybu vynásobením další číselnou hodnotou představující rychlost.

Takovýmto způsobem lze snadno docílit k pohybu postavy pomocí klávesnice. Mnohem zajímavější je však pohyb postavy ovládaný počítačem. Základního pohybu by šlo docílit pomocí náhodně vygenerovaných čísel. Postava by po určitém časovém intervalu náhodně měnila směr pohybu. Takto však generovaný pohyb by vypadal na první pohled velice neprofesionálně. Postava by z velké části pouze narážela do zdi. Pro zlepšení algoritmu pohybu použijeme třídu Physics, konkrétně funkci Raycast. Tato funkce slouží k detekci collideru v určitém směru a vzdálenosti od konkrétního bodu. Následující ukázka znázorňuje předpis funkce Raycast.

```
public static bool Raycast(Vector3 origin, Vector3 direction, out
RaycastHit hitInfo, float maxDistance = Mathf.Infinity, int layerMask =
DefaultRaycastLayers, QueryTriggerInteraction queryTriggerInteraction =
QueryTriggerInteraction.UseGlobal);
```

Funkce Raycast vrací hodnotu true, jestliže došlo ke kolizi. Parametr origin určuje počáteční bod, direction určuje směr paprsku, výstupní parametr hitInfo vrací informaci o srážce v případě, když funkce vrátí hodnotu true. Mezi tyto informace patří například vzdálenost od počátku ke kolizi nebo kolizní bod v prostoru. Dále parametr maxDistance určuje maximální vzdálenost k hledanému collideru, layerMask určuje vrstvy, ve kterých má funkce detekovat kolize a queryTriggerInteraction určuje, jestli se má detekovat i collider, který je nastavený jako spínač.

Zdroj [26]

Pomocí této funkce můžeme detekovat blízkou překážku před postavou a následně reagovat změnou směru. Stejným způsobem lze detekovat nepřátelskou postavu na dostřel zbraně, nebo v jiném směru. Následující obrázek představuje princip fungování postav řízených pomocí počítače. Paprsky zde představují grafické znázornění funkce Raycast. Bílý paprsek zobrazuje dostřel zbraně, červený paprsek zobrazuje detekovanou nepřátelskou postavu v prostoru dostřelu zbraně. Zelený paprsek představuje strany, který si postava hlídá. Malá trojice purpurových paprsků určuje místa, která si postava hlídá, aby nenarazila do zdi.

Obrázek 19: Ukázka logiky postav řízených počítačem.

Tyto viditelné paprsky jsou určené pouze k vývoji a zkoumání správného chování postav. V hotové hře by se dále nezobrazovali. Postava na každý paprsek reaguje jiným způsobem. Když do bílého paprsku vejde nepřátelská postava, postava generující paprsek začne střílet. Jestliže purpurový paprsek detekuje jakoukoliv překážku, postava změní směr svého pohybu. Jestliže vstoupí postava do zeleného paprsku, postava generující paprsek se otočí ve směru narušení. Tyto reakce na detekci vstupů postav je potřeba delší dobu zkoumat a upravovat pro věrohodnější chování. Některé reakce jako například otočení postavy v případě že jiná postava naruší zelený paprsek bylo potřeba zpomalit v náhodném rozsahu desetin vteřin. V následujícím obrázku je navíc ukázáno rozdílné detekování postav v závislosti na používané zbrani postavy.

Obrázek 20: Ukázka rozdílné detekce nepřátelských postav.

ZÁVĚR

Herní engine Unity 3D je velmi rozsáhlým nástrojem pro tvorbu her na spoustu platformách. Díky volné licenci, rozsáhlé dokumentaci a velkému množství návodů může dnes začít vyvíjet hry prakticky každý.

Základním kamenem pro vytváření her jsou již zmíněné assety, které byli čtenáři vysvětleny včetně popisu vývoje od samotného počátku, až po nasazení do herního enginu. Spoustu assetů je volně distribuováno. Ovšem pro vývoj originální a kvalitní hry bude vždy potřeba vytvářet své vlastní.

Dále bylo ukázáno množství různorodosti druhů assetů, z čeho vyplývá základní problém nemožnosti vývojáře zcela ovládnout vývoj všech těchto druhů. Ve všech herních studií lze tedy nalézt zaměstnance, který se specializuje na konkrétní druh.

V teoretické části bakalářské práce byly vysvětleny základní principy tvorby assetů a to jak grafického charakteru, tak i skriptů, které určují chování objektů ve hře. Ve skutečnosti by si každý druh assetu zasloužil svoji vlastní práci, nicméně věřím, že si čtenář se získanými znalostmi po přečtení této bakalářské práce udělá lepší představu o náplni práce grafiků a programátorů, kteří jsou důležitou součástí vývojářského týmu.

V praktické části bylo ukázáno na konkrétních příkladech tvorba assetů pro počítačovou hru pojmenovanou Bulánci. Dále byl ukázán jejich export a nasazení do herního enginu včetně dalších úprav a dodatečných skriptů pro manipulaci. Celý výstup je ve formě balíčku, který je spustitelný v herním enginu Unity 3D.

POUŽITÁ LITERATURA

[1] Home of the Blender project: Free and Open 3D Creation Software [online]. Amsterdam: Blender Foundation, 2005 [cit. 2019-11-19]. Dostupné z: https://www.blender.org/

[2] *UNITY TECHNOLOGIES. Unity – Manual*: Asset Workflow. unity3d [online]. 2019 [cit. 2019-11-19]. Dostupné z: https://docs.unity3d.com/Manual/AssetWorkflow.html

[3] Keyframes: Blender Manual. *Blender Documentation* [online]. Amsterdam: Blender Foundation, 2005 [cit. 2019-11-26]. Dostupné z: https://docs.blender.org/manual/en/latest/animation/keyframes/index.html

[4] FAULKNER, Andrew a Conrad CHAVEZ. *Adobe Photoshop CC: oficiální výukový kurz*. Brno: Computer Press, 2016. ISBN 978-80-251-4741-2.

[5] Event Functions. *Unity User Manual* [online]. San Francisco: Unity Technologies, 2019 [cit. 2019-11-02]. Dostupné z: https://docs.unity3d.com/Manual/EventFunctions.html

[6] NURBS. Digital modeling. Berkeley, CA: New Riders, 2012, s. 109-111. ISBN 978-0-321-70089-6.

[7] Bones. Digital modeling. Berkeley, CA: New Riders, 2012, s. 135-136. ISBN 978-0-321-70089-6.

[8] UNITY TECHNOLOGIES. Unity – Manual: MonoBehaviour. unity3d [online]. c2019 [cit. 2019-11-19]. Dostupné z: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

[9] *Unity Real-Time Development Platform: 3D, 2D VR & AR Visualizations* [online]. Denmark: Copenhagen, c2019 [cit. 2019-11-19]. Dostupné z: https://unity.com/

[10] *Visual Studio: Nejlepší nástroje pro každého vývojáře* [online]. Redmond: Microsoft, c2019 [cit. 2019-11-19]. Dostupné z: https://visualstudio.microsoft.com/cs/

[11] WILEY, John. Polygons. *3D Animation Essentials*. Indianapolis: Inc, c2012, s. 136-150. ISBN 978-1-118-14748-1.

[12] Variables in C#: Unity Tutorial. *Mammoth Interactive* [online]. Vancouver: Bura, 2008, 2017 [cit. 2019-11-21]. Dostupné z: http://mammothinteractive.com

[13] LOD with Function Specialization. *Apple Developer Documentation* [online]. Cupertino: Apple, c2019 [cit. 2019-11-20]. Dostupné z: https://developer.apple.com/documentation/metal/lod_with_function_specialization

[14] WILEY, John. UVs. *3D Animation Essentials*. Indianapolis: Inc, c2012, s. 160-163. ISBN 978-1-118-14748-1.

[15] WILEY, John. Shaders. *3D Animation Essentials*. Indianapolis: Inc, c2012, s. 163-167. ISBN 978-1-118-14748-1.

[16] UNITY TECHNOLOGIES. Unity – Manual: Variables and the Inspector. unity3d [online]. c2019 [cit. 2019-11-19]. Dostupné z: https://docs.unity3d.com/Manual/VariablesAndTheInspector.html

[17] UNITY TECHNOLOGIES. Unity – Manual: Attributes. unity3d [online]. c2019 [cit. 2019-11-19]. Dostupné z: https://docs.unity3d.com/Manual/Attributes.html

[18] UNITY TECHNOLOGIES. Unity – Manual: Art Asset best practice guide. unity3d [online]. c2019 [cit. 2019-11-19]. Dostupné z: https://docs.unity3d.com/Manual/HOWTO-ArtAssetBestPracticeGuide.html

[19] Spin: Blender Manual. *Blender Documentation* [online]. Amsterdam: Blender Foundation [cit. 2019-11-20]. Dostupné z: https://docs.blender.org/manual/en/latest/modeling/meshes/editing/duplicating/spin.html

[20] Data types and variables in C#. *CodeMahal: Learn to code today* [online]. Sydney: Wood, 2014, 2015 [cit. 2019-11-20]. Dostupné z: https://www.codemahal.com/video/data-types-and-variables-in-c/

[21] UNITY TECHNOLOGIES. Unity – Manual: Important Classes. unity3d [online]. c2019 [cit. 2019-11-19]. Dostupné z: https://docs.unity3d.com/Manual/ScriptingImportantClasses.html

[22] WILEY, John. 3D Animation Overview. *3D Animation Essentials*. Indianapolis: Inc, c2012, s. 19-38. ISBN 978-1-118-14748-1.

[23] Geometry. *Blender Manual* [online]. Amsterdam: Blender Foundation, 2005 [cit. 2019-11-21]. Dostupné z: https://docs.blender.org/manual/en/latest/modeling/curves/properties/geometry.html

[24] UNITY TECHNOLOGIES. Unity – Manual: Modeling characters for optimal performance. unity3d [online]. c2019 [cit. 2019-11-19]. Dostupné z: https://docs.unity3d.com/Manual/ModelingOptimizedCharacters.html

[25] WILEY, John. Motion-Capture Animation. *3D Animation Essentials*. Indianapolis: Inc, c2012, s. 209. ISBN 978-1-118-14748-1.

[26] UNITY TECHNOLOGIES. Unity – Manual: Physics.Raycast. unity3d [online]. c2019 [cit. 2019-11-20]. Dostupné z: https://docs.unity3d.com/ScriptReference/Physics.Raycast.html

PŘÍLOHY

PŘÍLOHA A – OBSAH PŘILOŽENÉHO CD

Na přiloženém CD se nachází následující obsah:

- Elektronická verze této bakalářské práce ve formátu PDF v souboru HorakM_TvorbaAssetu_ZK_2019.pddf.
- Balíček HorakMilan_Assety.unitypackage, který je umístěn v adresáři assety. Tento balíček obsahuje všechny assety vytvořené v této bakalářské práci včetně ukázkové scény. Balíček byl vyexportován v herním enginu Unity 3D ve verzi 2019.2.13f1.