ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA PEDAGOGICKÁ
KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

TVORBA INTERAKTIVNÍ 3D ANIMACE

BAKALÁŘSKÁ PRÁCE

Bohuslav Bílý
Informatika se zaměřením na vzdělávání

Vedoucí práce: PhDr. Denis Mainz, Ph.D.

Plzeň 2018

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

V Plzni, 27. dubna 2018

.....

vlastnoruční podpis

Poděkování

Rád bych poděkoval svému vedoucímu práce PhDr. Denisovi Mainzovi, Ph.D. za nápady, které vnesl do mé práce, za všechnu pomoc, aby byla moje práce na kvalitnější úrovni.

ZDE SE NACHÁZÍ ORIGINÁL ZADÁNÍ KVALIFIKAČNÍ PRÁCE.

OBSAH

SEZNAM POJMŮ A ZKRATEK

2D	2 dimenze délka, šířka, x, y
3D	3 dimenze délka, šířka, objem, x, y, z
CAD	Computer aided design, programy pro technické kreslení na počítači
GPU	Graphic proccesing unity (grafický procesor)
CPU	Central processing unit (centrální procesorová jednotka)
RAM	Random Acces Memory (paměť počítače)
MEL	Maya Embedded language, skriptovací jazyk pro modelovací software Maya
SVG	Scalable Vector Graphics, typ souboru, který uchovává vektorovou grafiku
VR	Virtual Reality (virtuální realita)
AR	Augmented Reality (rozšířená realita)
IDE	Integrated Development Environment (integrované vývojové prostředí)

Vertex	bod v prostoru
Edge	hrana mezi dvěma body
Face	plocha vytvořená třemi body či dvěma hranama
Render	přetvoření modelu k reálnému zobrazení
Mesh	jednotlivé Vertex, Edge, Face, které tvoří model
Voxel	reprezentace objemu ve 3D prostředí
Sculpting	modelování ve virtuálním prostředí s virtuálním jílem
Addons	zásuvné moduly
Asset	materiály pro tvorbu v unity
Asset Store	obchod uvnitř unity kde se nacházejí Assety ke stažení

ÚVOD

V dnešní době už je těžké si představit, že by nebylo možné 3D modelovat. Výsledek práce grafiků, kteří se zabývají modelováním, vidíme všude kolem, filmy, videohry, figurky do deskových her a 3D tiskárny atd. 3D modelování je v podstatě základ pro každý výrobek, který je masově vyráběn. Nejdříve se vymodeluje jeho 3D reprezentace, poté se vyrobí testovací vzorek a po finální úpravě se data s 3D modelem pošlou výrobci a stroje za člověka vytisknou nebo vyfrézují požadovaný produkt, přesně jak ho modelář navrhl.

V průmyslové praxi se můžeme často setkat s CAD systémy, spadající do kategorie 3D modelování založené na konstruktivní geometrii těles vycházející z analytického popisu jejich objemu a matematicky vyjádřených křivkách. Tyto systémy využívají společnosti nebo jednotlivci zaměřující se na reálné výrobky, jako jsou např. strojní, stavební, elektrotechnické součástky, nebo rovnou celé automobily, stavby či procesory. Existuje ovšem i jiná kategorie modelovacích systémů používaná umělci a animátory, protože modelovaný výsledek nemusí mít nutně svou fyzickou podobu a nemusí fungovat podle fyzikálních zákonů reálného světa, ale mnohdy stačí, aby splňoval potřebná kritéria ve světe virtuálním, a může jít pouze o jeho vzhled a o takovou funkčnost, kterou modelu navrhne jeho modelář.

Předmětem a cílem této práce jsou oblasti vycházející především z druhé uvedené kategorie modelovacích systémů, konkrétně pak programový nástroj pro prostorové modelování Blender, vytvoření modelu, animování jeho pohybu v navržených situacích. Dále implementovat interaktivitu do vytvořených animací s využitím vývojového nástroje Unity, do kterého se importuje dříve vytvořený model se všemi jeho animacemi, materiály, texturami a pomocí C# skriptování zajistit výslednou funkčnost modelu, jeho interaktivitu se světem, a také zajistit ovládání klávesami spouštějící jednotlivé animace.

V první kapitole bude vysvětlena problematika 3D modelování, bude zdůvodněn výběr modelovacího nástroje na základě srovnání v praxi nejpoužívanějších nástrojů pro 3D modelování a představeno prostředí zvoleného nástroje prostorového modelování.

Druhá kapitola bude věnována podrobnému vysvětlení postupu tvorby vlastního modelu od tvorby polygonové sítě postupným přidáváním ploch na dvojrozměrný obraz

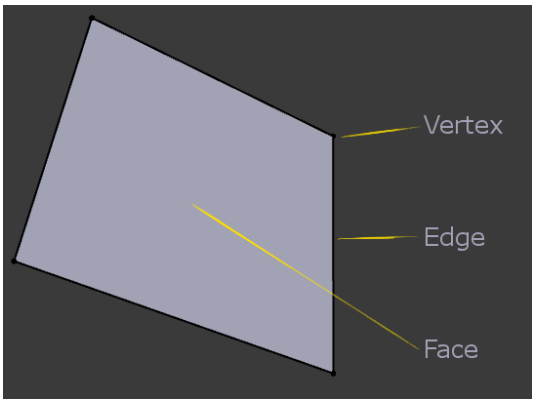
přes přidávání materiálů a textur až po návrh kostry modelu zajišťující jeho nezbytnou hybnost při animování.

Třetí kapitola bude vysvětlovat principy interaktivní animace v prostředí herního jádra společně s představením jejich hlavních zástupců umožňujících tvorbu interaktivit prostorových modelů, včetně silných a slabých stránek hlavních představitelů oblasti herních jader.

Poslední kapitola bude zaměřena na objasnění postupů tvorby vlastní interaktivní animace pomocí skriptů a ostatních náležitostí nutných k zajištění požadovaného chování modelu v simulovaných situacích. Výsledkem pak bude vytvoření jedné herní úrovně, kde se bude dříve vytvořený model pohybovat a interagovat s prostředím.

1 3D MODELOVÁNÍ

Je to proces, při kterém se vytváří objekt ve virtuálním prostředí, který je reprezentován ve třech dimenzích. Tím se získá objekt, na který se může nahlížet jako na reálný, z každé strany, z každého úhlu. Při vytváření 3D modelu se pracuje s body v prostoru, které se nazývají Vertex. Mezi dvěma Vertex vznikne hrana, neboli Edge. Pokud se spojí 3 a více Vertex nebo 2 Edge, vznikne plocha, dále Face, která reprezentuje polygon určitého stupně. (Slick, 2018) Tímto způsobem se vytváří topologie modelu, která udává, jak je model vytvořen, jeho detail. V běžné praxi je standardem vytvářet čistou topologii, uniformní a s co nejmenším počtem Face. (Martin, 2017)



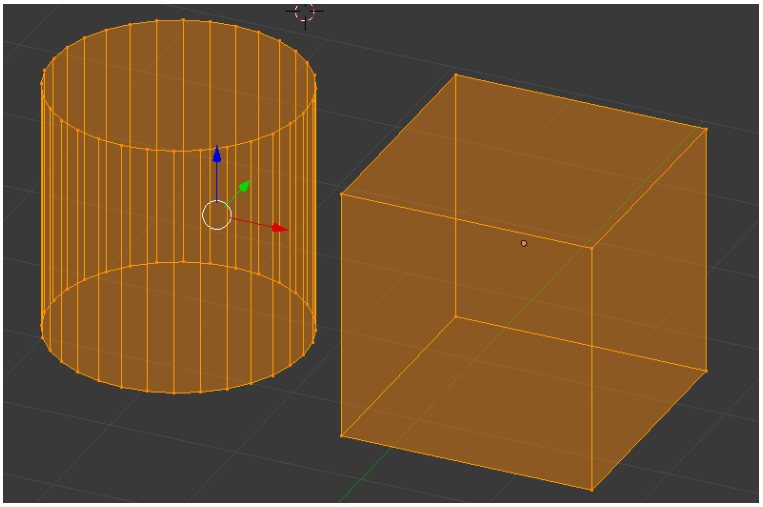
Obrázek 1: Vertex, Edge, Face. In: Blender Reference Manual [online]. [Cit. 2.3.2018]. Dostupné z WWW: https://docs.blender.org/manual/en/dev/_images/modeling_meshes_structure_example.png

V 3D modelovacím software jsou primitivní křivky či Meshe, které představují elementární matematické objekty, jako jsou krychle, plocha, kruh, koule, válec, jehlan a další, specifické pro jednotlivé 3D software. Každý z těchto objektů se skládá z Vertex, Edge a Face.

Poté se s modely pracuje nástroji, které má daný software k dispozici. Všechny Vertex, Edge či Face je možné posouvat, rotovat, zvětšovat, zmenšovat nebo vytvářet nové vytažením z již existujících, a tím se model formuje dle předlohy, kterou je možné vložit na jednu ze tří os (X, Y, Z) do pozadí. Modelovací softwary oplývají vysokým množstvím nástrojů pro úpravu modelu, které nejsou již na takové elementární úrovni.

Kvalitní topologie se skládá buď z trojúhelníků či čtverců. Nástroje pro úpravu modelu jsou přizpůsobené na práci se čtverci, proto je standardem, že se model vytváří právě s důrazem na to, aby bylo použito co nejvíce čtyřstranných polygonů neboli čtverců.

Provedení jednotlivého modelu závisí na jeho použití. Pro vytvoření interaktivní animace, by se měl modelář držet použití čtverců. (Martin, 2017)



Obrázek 2: Základní Mesh krychle a válce (Zdroj: vlastní)

Při práci s herním jádrem, se převádí celý model na trojúhelníky. Jádro pracuje automaticky a retopologii provádí nezávisle. Tento způsob ovšem nezaručuje nejlepší výsledek, jelikož je založen na kódu, který je matematicky propočítáván, a tím vznikají chyby. Pro kvalitní retopologii modelu se převádí model ručně ze čtverců na trojúhelníky a teprve až poté následuje import do herního jádra. (Slick, 2017)

Modelovací software pracuje s různými nástroji, jako jsou například smyčky, které ale nepracují správně, pokud nejsou aplikovány na čtverce, nebo také různé modifikátory, které rozdělí Face na dvě pro větší rozlišení a modifikovatelnost. Čtverce nám zajišťují, že se se model bude správně deformovat při animaci a jeho topologie bude mnohem čistější, lépe čitelná a modifikovatelná.

Existují různé techniky modelování, které využívají rozdílné nástroje k dosažení cíle. V této práci je zvoleno, polygonální modelování a jeho nejzákladnější forma *Box/Subdivision*. (Slick, 2017)

Tato technika pracuje s primitivními geometrickými útvary, které se následně upravují, přetvářejí do tvaru, který je požadován. Pracuje se po částech, nejdříve se model vytvoří v malém rozlišení, formuje se přidáním topologie, rozdělováním ploch pro vyhlazení hran a přidáním detailu. Proces se opakuje, do té doby, než Mesh obsahuje tolik polygonů, aby

odpovídal konceptu. (Slick, Seven common 3D modelling Techniques for Film and Games, 2017)

1.1 HARDWARE

Mimo software pro 3D modelování je také nezbytně nutné mít i dostatečně výkonný hardware. Nejdůležitější součástí jsou RAM paměti a GPU/CPU. Specifikace záleží na tom, jak složitý model bude vytvářen a jaké bude mít využití. Rozdíl je mezi tím, pokud je modelováno např. pro 3D tisk, kde není nutnost model renderovat nebo pokud je výsledkem 3D scéna a její finální render. (Daniel, Logical Increments, 2018)

RAM paměť je důležitá kvůli komplexnosti modelu. Když se v modelu přidává geometrie, zvyšuje se počet trojúhelníků v modelu. Modelovací software vše přepočítává na trojúhelníky kvůli renderování. Pokud bude systém mít nedostatek paměti a model bude příliš detailní, způsobí to zpomalení softwaru až do bodu, kdy nebude možné dále pracovat. (Daniel, Logical Increments, 2018)

GPU/CPU je hlavním činitelem při renderování. Každý modelovací software má vlastní interní renderovací jádro, vlastně jich může být i několik, s různými vlastnostmi. Některé z nich dokáží využít jak GPU, tak CPU pro renderování, avšak některé zvládnou práci pouze pomocí jedné komponenty. Složité a detailní scény renderují velice dlouho, v rámci hodin, měsíců či roků. V těchto případech se využívá více GPU/CPU, které si práci rozdělí a renderování se výrazně urychlí. (Daniel, Logical Increments, 2018)

Příkladem, proč je nutné mít tzv. výpočetní farmy, které se skládají z tisíce výpočetních jednotek, by mohl být filmový průmysl. Pro vytvoření plynulé 1 sekundy animace je nutné renderovat minimálně 23 snímků. Počet snímku se liší, ale obecně se udává 23–25 snímků za sekundu. Z toho se dá usoudit, že časy pro renderování mohou být astronomické, pokud je nedostatečně silný hardware. Zároveň je ale nutné dodržovat i chlazení jednotlivých komponent, jelikož při renderování pracují GPU či CPU na 100% své kapacity. (Daniel, Logical Increments, 2018)

1.2 3D MODELOVACÍ SOFTWARE

Pro vytváření modelu interaktivní animace existuje mnohem víc software s různými výhodami či nevýhodami. Mezi nejznámější patří 3ds Max, Maya, Blender či Zbrush. Dalo by se říci, že největší rozdíl mezi těmito softwary je ve finančních prostředcích. Jediný

Blender má platební model, který je zdarma, za používání jiných zmíněných softwarů je nutné platit poplatky, či si zakoupit licenci. (Daniel, Logical Increments, 2018)

1.2.1 3DS MAX

Je k dispozici pouze pro Windows a společnost Autodesk nemá v plánu rozšíření na ostatní platformy. Největší výhodou oproti ostatním softwarům jsou jeho modelovací schopnosti. (Autodesk, 2018)

3ds Max oplývá a zároveň exceluje s komplexními prvky pro modelování, obsahuje více nástrojů pro modifikaci modelu, velice kvalitní renderovací jádro Arnold, V-Ray či Iray. Prostředí je uživatelsky přívětivé, nováček by se měl snadno orientovat. V uživatelském prostředí se klade důraz na to, aby měl uživatel snadný přístup k nejdůležitějším modelovacím nástrojům. *UV mapování*, je velice sofistikované a nabízí například kombinování nekonečně mnoho textur, což jiné software nemusí podporovat. Specifický je přístup k váhám, které se používají při vytváření kostry. 3ds Max používá tzv. *Voxel Skinning*, tento přístup převádí Mesh na Voxely v daném rozlišení. Díky tomu se získá mnohem přesnější výpočet vzdálenosti Vertex k nejbližšímu kloubu, je to tedy mnohem přesnější a spolehlivější metoda než např. *Kreslení váh (Weight Painting)*. (Autodesk, 2018)

Největší nevýhodou tohoto softwaru je platební model. Je zde k dispozici i studentská verze na 3 roky a zkušební verze na 30 dní. Poté už ovšem je nutnost platit poplatky, které mohou být buď měsíční 190 $, roční 1,505 $ nebo 3 roční 4,515 $. (Autodesk, 2018)

1.2.2 MAYA

Patří do rodiny modelovacích software od společnosti Autodesk. Na rozdíl od 3ds Max je multi-platformní, jsou podporovány jak Windows, tak Macintosh i Linux. Maya se liší ve skriptovacím jazyku MEL, pokročilém vytvoření kostry pro model a také excelentnímu animačnímu prostředí. (Autodesk, 2018)

Maya má své vlastní funkce, které jiné modelovací software nemají, mezi ně mimo jiné patří After Effects live link. Tento plugin zprostředkovává spojení mezi Maya a Adobe Effects v reálném čase, tzn. že je možné vidět scénu v obou softwarech naráz a změny, které se provedou, jsou ihned promítnuty v obou prostředích. Pro animaci je zde určena celá škála pluginů a funkcí v balíčku Motion graphics. Obsahuje například možnost

importovat SVG soubory, vytvářet procedurální efekty, loga či texty. Mezi další zajímavé pluginy patří například Maya nCloth, který slouží k vytváření realistických deformací materiálu, Bifrost Ocean Simulation Systém, který ztvárňuje realistické oceány, Adaptive Aero Solver in Bifrost pro vytvoření atmosférických efektů jako je kouř nebo mlha. (Autodesk, 2018)

Jelikož tento software pochází od stejné společnosti jako 3ds Max, je zde stejný studentský model i trial verze na 30 dní. Platební model je nastavený totožně. Existuje i Maya LT, což je odlehčená verze, která neobsahuje všechny funkce a pluginy, ale pořád obsahuje důležité komponenty. Na druhou stranu její platební model už je přívětivější, 30 $ za měsíc, 245 $ za rok, 735 $ za 3 roky. (Autodesk, 2018)

1.2.3 ZBRUSH

Toto není klasický 3D modelovací software, ZBrush slouží pouze k jedné věci, a to je Sculpting. Patří mezi nejpropracovanější software ve svém oboru a je používán profesionálními vývojářskými studii ve filmu či video herním průmyslu. (ZBrush, 2018)

Dynamesh je v Zbrush digitální modelářská hlína. Přetváří topologii modelu v reálném čase. Skulptováním vznikají hladké povrchy nebo naopak přidáním objemu vzniká větší detail. (Zbrush, 2018)

Jelikož se ale model modifikuje pomocí objemu, kompletně se zruší daná topologie, která byla vytvořená, když vznikl mesh v 3D modelovacím software nebo není žádná, pokud se celý model vytváří v ZBrush. Topologii je proto nutné poté opět vytvořit. ZBrush nabízí základní prostředky pro vytvoření topologie a je nutné exportovat model do 3D modelovacího software, kde se provede retopologie tak, aby při následné animaci docházelo ke správnému deformování. (ZBrush, 2018)

Pro používání ZBrush je nutné si zakoupit licenci za 795 $. Zbrush má odlehčenou verzi ZBrush Core, která stojí 149.95 $. Je možné také zažádat o studentskou verzi přes podoporu ZBrush. Pixologic nabízí také Sculptris který je určen pro nováčky ve světě scultpingu a je kompletně zdarma. (ZBrush, 2018)

1.2.4 BLENDER

Blender je 3D modelovací software, který je kompletně zdarma, a Open source spadá pod General Public License. Tzn., že je možné Blender používat k jakýmkoliv účelům,

prohlížet zdrojový kód, provádět úpravy a následně tyto úpravy distribuovat. Pokud jsou ale takovéto úpravy provedeny, musíme je také zveřejnit a dovolit ostatním zájemcům si tyto změny prohlížet, případně je modifikovat. Tento přístup je obecně známý pod termínem Copyleft. Blender je zároveň multi-platformní, podporuje Linux, Windows či Macintosh. (Blender, 2018)

Blender obsahuje mnoho funkcí, které mohou být rozšířené ještě o tzv. Addons, které přidávají další možnosti a usnadňují práci. Mezi hlavní složky patří modelování, vytváření kostry, simulace, renderování, sledování pohybu, animace a dokonce vlastní herní jádro. Blender nabízí i možnost skriptování pomocí jazyka Python, což otevírá například možnosti modifikovat addony nebo si vytvořit vlastní, a tím si automatizovat vytvoření kostry pro model, upravit vypočítávání křivek či vytvářet náhodné modely procedurálním generováním. (Blender, 2018)

Blender má v základní vybavenosti i nástroje pro texturování přímo uvnitř softwaru. Avšak nejsou tak sofistikované a pro umělce na profesionální úrovni, je nutnost použití externích programů jako je Adobe Photoshop, Gimp či Krita. To ale neznamená, že přímo v Blenderu nelze vytvořit kvalitní textura. Jsou zde k dispozici palety barev, štětce, které si můžeme upravovat matematickými křivkami a také třeba vrstvy. Na stejném principu fungují i nástroje pro scuplting. Je zde i nabízena možnost propojení s grafickým tabletem.

Blender byl použit i pro výrobu krátkých filmů ve vysoké kvalitě. Elephants Dream, Big Buck Bunny a Sintel. Tyto filmy byly vytvořeny za účelem propagace Blender a ukázání jeho síly, že patří do profesionální sféry. (Flavell, 2010, str. 3) V současnosti se připravuje za pomoci získávání financí od podporovatelů Blenderu celovečerní film, který bude vznikat v Nizozemí v Blender institutu.

Obrázek 3: Agent 327. In: Agent 327: Operation Barbershop [online]. [Cit. 4.3.2018]. Dostupné z WWW: https://agent327.com/static/img/agent_barbershop_poster.jpg

1.3 VÝBĚR VHODNÉHO SOFTWARE

Při výběru vhodného nástroje pro prostorové modelování se projeví osobní preference, a finanční prostředky, které jsou k dispozici. Blender, který zdarma dokáže vytvořit profesionální modely stejně jako ostatní software.

Blender byl vybrán jako jediný zástupce software, které jsou zdarma. 3ds Max a Maya jsou pro modelování ve volném čase moc drahé a pro nezávislého grafika, který začíná, nemají žádnou větší výhodu. Takže jediný reálný výběr je tedy mezi Maya LT a Blender.

Pro nezávislou scénu či osobní použití se nabízí jako nejlepší řešení Blender, jehož softwarová výbava je na úrovni, která je schopná konkurovat profesionálním 3D modelovacím software. Profesionální studia ovšem mají jiné finanční prostředky a také nároky, pro ně je tedy výhodné si zaplatit za licenci či poplatky a využívat naplno možností těchto software.

Pokud jde o zkušenosti v daném software při zohlednění pracovního uplatnění, záleží na umění modelování dle svého portfolia a nezáleží, v jakém software bylo vytvořeno. Naučená schopnost přemýšlet a modelovat se kdykoliv dá přenést, a pokud je od budoucího zaměstnavatele zájem, v daném software zařídí důkladné školení. Neméně důležitým důvodem u výběru Blenderu je, že existuje nezměrné množství video návodů, knížek a tutoriálů oproti konkurenci, která díky svému platebnímu modelu není tak dostupná a rozšířená. Mít přístup k takovýmto informacím urychluje učící křivku a schopnost kvalitní produkce.

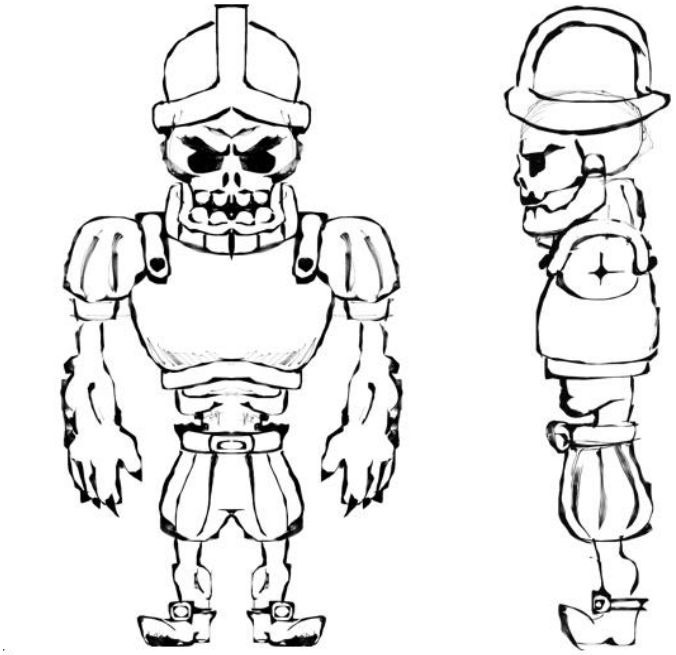
Z hlediska využitelnosti ve výuce, jsou všechny tyto software naprosto nevhodné pro výuku na základní škole. Softwary jsou příliš komplexní a pro uživatele, který není zběhlý v práci s počítačem, by byl ztracený a nedokázal by se orientovat. Pro střední a vysoké školy už bych doporučil jeden z těchto softwarů, žáci/studenti by už měli být schopní zvládnout komplikovanější softwary. Pro základní školy bych doporučil například SketchUp, který je mnohem méně komplikovaný.

2 POSTUP TVORBY MODELŮ A ANIMACE

Pro realizaci tvorby modelu bylo nejprve nutné si zvolit předlohu, podle které bylo modelováno. Výsledkem celé práce je interaktivní animace, proto bylo nutné se zamyslet, jaké všechny modely budou potřeba. Blender se nachází v původním nastavení, všechny použité klávesové zkratky jsou nezměněny.

2.1 TVORBA MODELŮ

Nejdůležitějším modelem, kterým je potřeba začít, je bezpochyby postava. Základ každé hry musí mít nějaký model, jež je kontrolován hráčem. Pro modelování byla vybrána předloha kostlivce, který má sobě kus brnění, oblečení a třímá v ruce meč. Koncept pro tento model byl převzat z online kurzu o 3D modelování. (Devslopes by Mark Price, 2017)



Obrázek 4: Předloha modelu In: 3D Game Modeling & Animation With Blender | Udemy [online]. [Cit. 28.3.2018]. Dostupné z WWW: https://www.udemy.com/blender3d/learn/v4/content

Ve vybrané předloze se nachází pohled zpředu a ze strany. Před samotným modelováním je nejprve nutné koncept importovat do prostředí Blender. Ten umožní, abychom v ortografickém zobrazení zepředu a ze strany měli pokaždé zobrazenou předlohu.

Po stisknutí klávesy ***N*** se otevře panel s *vlastnostmi (Properties)*, v záložce *obrázky na pozadí (Background Images)* je nutné zaškrtnout pole pro použití této komponenty a poté importovat obrázky.

Obrázek 5: Vlastnosti (Zdroj: vlastní)

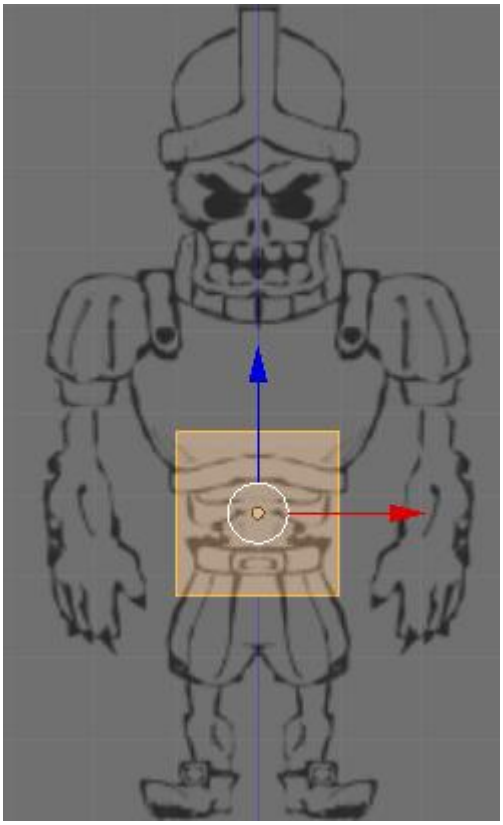
2.1.1 TĚLO

Jelikož se předloha skládá z více částí, je nutné modelovat postupně jednotlivé části. Největší částí je bezpochyby tělo, tak je vhodné začít zde. Pomocí klávesové zkratky ***Shift + A*** je vložena do scény krychle. Ta je následně zobrazena jak ve scéně, tak i v pravé části, kde se nachází *hiearchie (Outliner)*, který uchovává přehled o všech objektech ve scéně, zde je vhodné přejmenování na Body.

Obrázek 6: Hiearchie (Zdroj: vlastní)

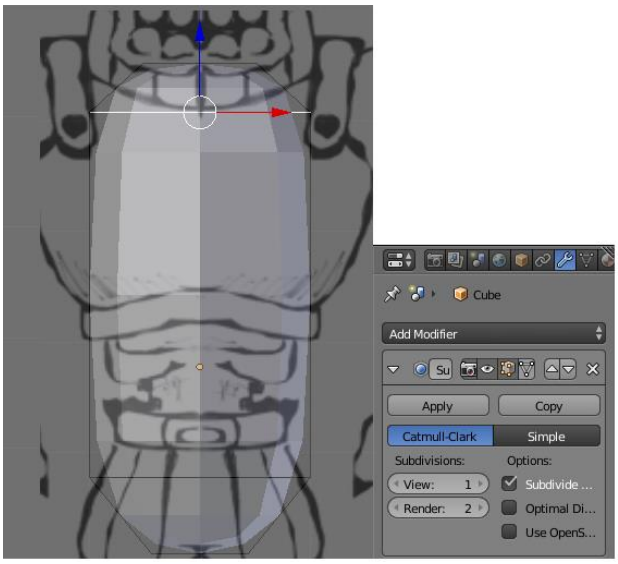
Krychle byla vložena na místo, kde se nachází 3D cursor, který byl umístěn na středu souřadného systému. Pro modifikaci krychle se stiskne klávesa ***Tab***, ta přepne model do Edit módu, teď je možné krychli modifikovat pomocí jednotlivých Vertex, Edge či Face.

Mesh je přesunut na místo, kde začne probíhat jeho úprava. Jelikož je modelováno tělo postavy, je vhodné začít u hrudního koše. Blender je přepnut pomocí klávesy ***5*** do ortografického zobrazení, dále pomocí klávesy ***1*** do pohledu zpředu. Následně je krychle přesunuta do požadované pozice táhnutím ***modré šipky***, reprezentující osu ***Z***.



Obrázek 7: Mesh pro tvorbu těla (Zdroj: vlastní)

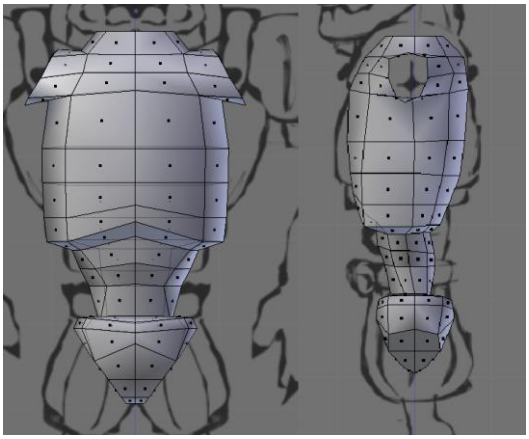
Pomocí nástroje *Extrude*, který dovoluje vytáhnout jednotlivé dílčí části. Z horní části se vytvoří hrudní koš, až po část, kde se bude nacházet krk, z dolní části vznikne spodní část těla. Pomocí *Circle Loop*, zmáčknutím zkratky ***Alt + pravé tlačítko myši*** dojde k označení skupiny Edge kolem modelu v horizontální poloze. V panelu vlastností se v pravé části okna nachází skupina modifikátorů. Je zvolen modifikátor *rozdělování ploch (Subdivison Surface)*, který rozdělí face na víc částí, a tím je model více uhlazený, ale dokud není potvrzen, původní geometrie je zachována a je stále možné s ní pracovat, modifikátor je aplikován v reálném čase.



Obrázek 8: Subdivison surface (Zdroj: vlastní)

Velikost modelu je upravena, aby přibližně odpovídala tvaru těla, a modifikátor se aplikuje. Poté, co je aplikován modifikátor, se reálná geometrie změní, a následují další úpravy jednotlivých Edge. Používají se základní úpravy změny pozice a velikosti. Pokud je žádoucí vytvoření další geometrie, použije se nástroj *smyčkový řez (Cut Loop)*, který udělá horizontální či vertikální řez, kterým vznikne další Edge, tím se zvýší geometrie a možnost modifikace modelu. Model byl upravován do doby, než připomínal referenční obrázek.

V této chvíli je model ještě velice hranatý, je tedy žádoucí abychom hranil zaoblili, aby tvar byl přiblížen více realistickému pojetí. Po označení Edge, které jsou vhodné k zaoblení, je zvolena možnost *uvolění (Relax)* ve *smyčkových nástrojích (Loop tools)* a následně se uvolní do křivky.

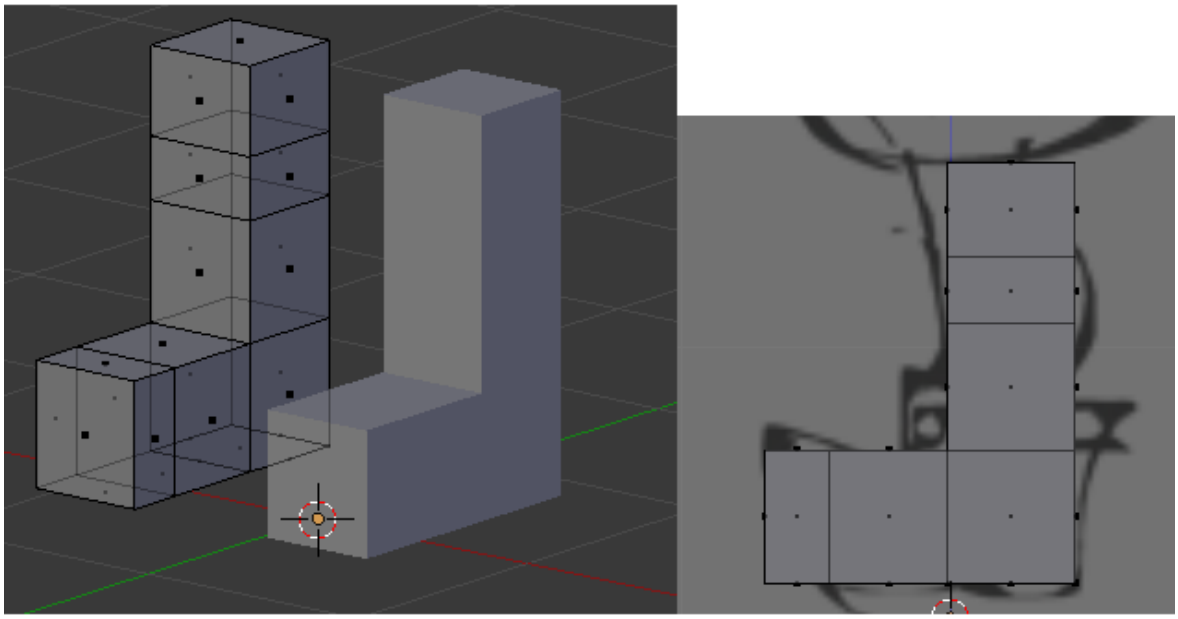


Obrázek 9: Tělo postavy (Zdroj: vlastní)

2.1.2 NOHY A RUCE

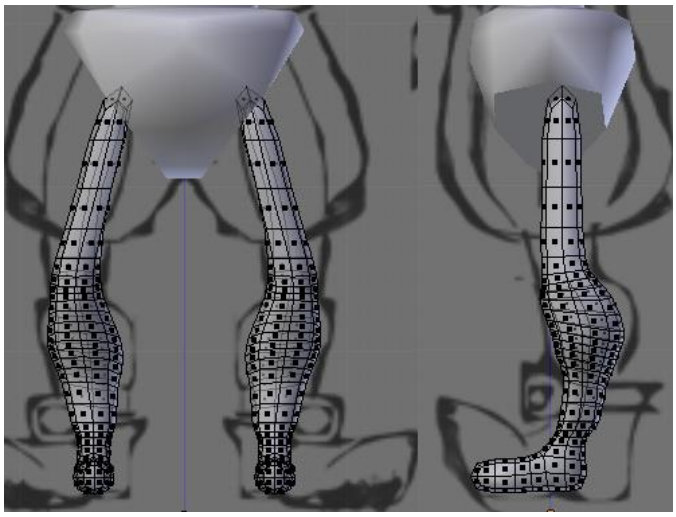
Poté co je vymodelováno tělo, začíná práce na nohou. Model se teď nachází v Edit módu, pokud by byl přidán další Mesh, z kterého budou modelovány nohy, byl by součástí těla, to je avšak nežádoucí. Z toho důvodu je nutné změnit na Objekt mód. Následným vložením krychle do scény se vytvoří v hiearchii nový objekt, kterému se změní název na Legs.

Každá humanoidní postava má dvě nohy, které jsou symetrické tak, aby nebylo nutné modelovat každou jednotlivě, je žádoucí použít modifikátor pro *zrcadlení (Mirror)*, ten zajistí přesnou kopii podle osy ***X***.



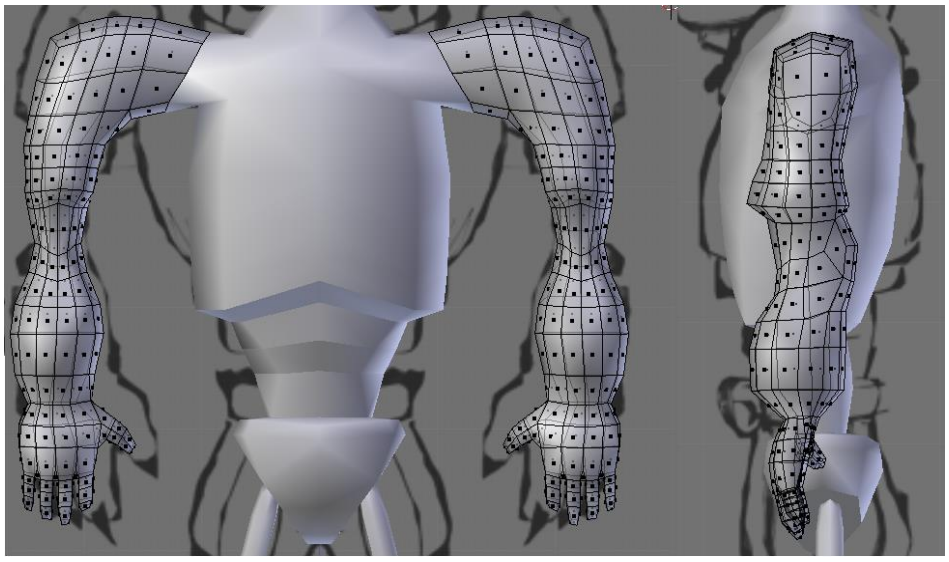
Obrázek 10: Zrcadlení (Zdroj: vlastní)

Následně je vytažen pomocí *vytažení* horní Face nohy, kde se nachází pánev, následují základní úpravy Edge a přesunutí do pozice, aby odpovídaly obrázku, poté se opět aplikuje modifikátor *rozdělení ploch*, tím vzniknou nohy, které už nejsou tak hranaté a mají reálnější tvar.



Obrázek 11: Nohy (Zdroj: vlastní)

Stejným postupem, kterým byly modelovány nohy, se vymodelují i ruce. Do scény se přidá Mesh krychle na úroveň ramene postavy. Postupnými úpravami této krychle, pomocí *vytažení, smyčkového řezu*, změnami pozice a velikosti Vertex, Edge nebo Face jsou vymodelovány paže, ramena a ruce. Následně byl použit nástroj *uvolnění* pro zaoblení hran.



Obrázek 12: Ruce (Zdroj: vlastní)

2.1.3 HLAVA

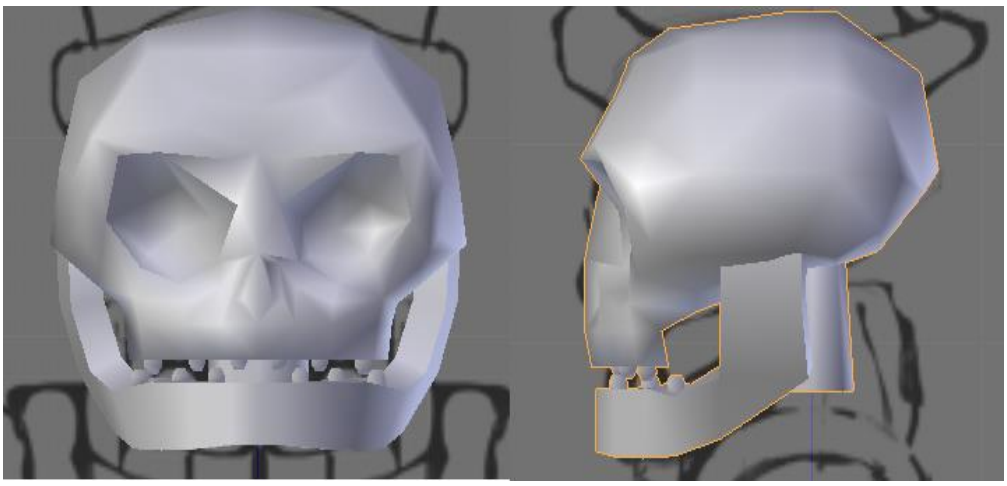
Poslední částí těla, která zbývá vymodelovat, je hlava. Do scény je vložena opět krychle a přesunuta na požadovanou pozici. Přidá se modifikátor pro *rozdělení ploch* a následně se označí polovina krychle užitím *boxového výběru (Box Selection)*, pomocí

klávesy ***B*** jsou označeny plochy a klávesou ***X*** je provedeno smazání. Přidáním modifikátoru *zrcadlení* je získána opět kompletní hlava, následné úpravy budou zrcadleny přes osu **X**.

Použitím smyčkového řezu se vytvoří horizontální řez uprostřed krychle a následně se na vzniklou Face aplikuje nástroj *protlačení(Insert)*, který vloží do Face novou geometrii, která je protlačena dovnitř, tím se vytvoří oční důlky. Aplikováním modifikátoru *rozdělení ploch* je získána geometrie, která se následně upravuje do podoby referenčního obrázku. Aby byla získána kulatá geometrie očí, je možné označit smyčku Edge a použít nástroj ve smyčkových nástrojích *Circle*, který nám geometrii těchto Edge převede do kulatého tvaru.

Pro vymodelování čelisti modelu je vložena do scény tentokrát Mesh plochy (Plane), které je přidán modifikátor *zrcadlení* a po přesunutí na pozici, kde se čelist nachází, je pomocí postupným vytahováním Edge vytvářen tvar. K tomu, aby byla získána geometrie z plochy, je nutné přidat ploše tloušťku. Aby ale bylo možné použít tento modifikátor, bylo nutné oddělit čelist od hlavy. Po označení čelisti se zmáčkne klávesa ***P*** a tím se otevře *menu pro oddělení (Separate menu)*, poté se zvolí možnost *výběrem (By Selection)*, tím vznikne v hiearchii nový model. Teď už je možné přidat modifikátor *Solidify*, který přidá objem podle požadované tloušťky a jeho aplikováním se vytvoří geometrie. Postupnými úpravami je čelist upravena do požadovaných rozměrů.

Následným přidáváním krychle a úpravou velikosti vzniknou zuby, které jsou přesunuty na pozice, kde by se měly nacházet. Tímto je hotová celá hlava. Následně je změněn režim na objekt mód a po označení čelisti, hlavy a zubů zmáčknutím klávesy ***J*** jsou spojeny do jednoho meshe.

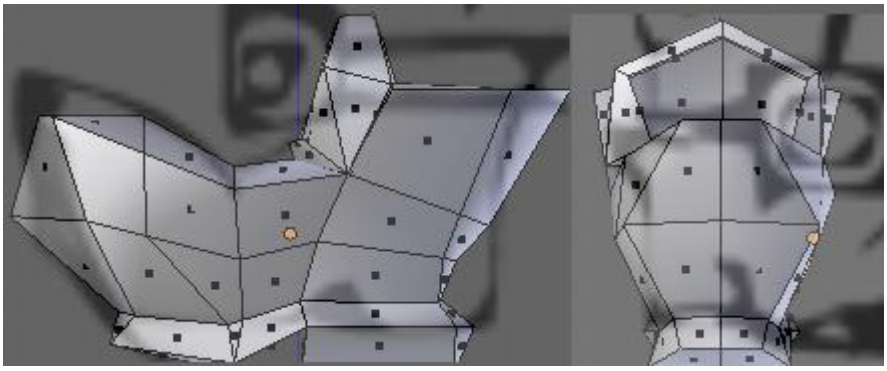


Obrázek 13: Hlava (Zdroj: vlastní)

2.1.4 BOTY

Bota byla vytvořena pomocí krychle a *smyčkovými řezy* byla přidána geometrie. Těchto řezů se přidalo několik a postupným upravováním jednotlivých Edge byl vymodelován tvar boty.

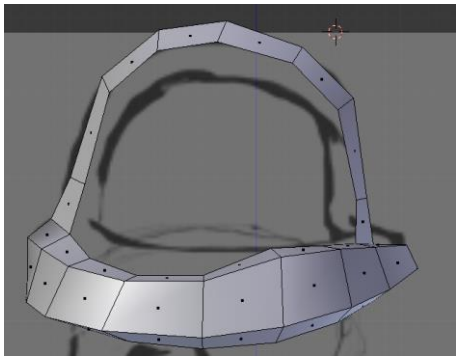
Pro vytvoření spony byl použit nástroj *vytažení*. V místě, kde má být spona, byl tedy vytlačen Face a nástrojem *protlačení* vytvořena geometrie uvnitř face, která byla protlačena na opačnou stranu. Následně byla spona formována dle konceptu.



Obrázek 14: Finální model boty (Zdroj: vlastní)

2.1.5 HELMA

Helma má kruhový tvar, proto je dobré začít její modelování pomocí kruhu. Do scény je tedy vložen kruh, který je umístěn nad hlavu. Polovina kruhu je smazána a přidáním modifikátoru *zrcadlení* je opět dokončena. Nástrojem *vytažení* Edge po ose ***Z*** vznikne tvar helmy, přidáním modifikátoru *Solidify* vznikne objem a po jeho aplikaci je připravena geometrie k další modifikaci. Následným postupným vytahováním Edge obvodu po vertikální ose vznikne tvar helmy který je nutno následně spojit.

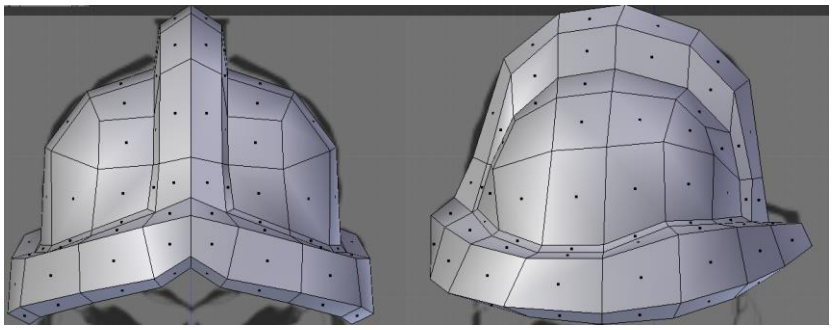


Obrázek 15: Tvar helmy (Zdroj: vlastní)

Žádoucí je, abychom vytvářeli čtvercové face, postupným označováním 4 Vertex jsou následně spojeny pomocí klávesy ***F***, která vytvoří face. Takto se vyplní zbylé části helmy a postupně je helma dotvořena s úpravami dle reference.

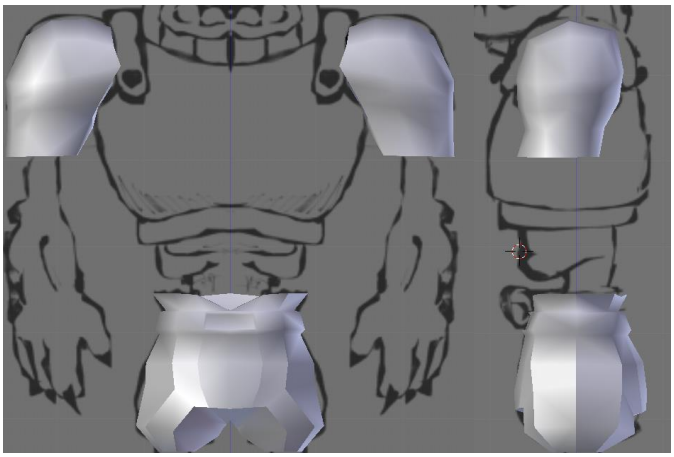
2.1.6 KALHOTY, BRNĚNÍ, POKRÝVKY RAMEN

Modelování kalhot probíhá stejným způsobem jako u již vytvořených modelů, začne se s krychlí a postupným přidáváním geometrie a modifikátorů vznikne model, který reprezentuje kalhoty. Nástroje pro vytvoření modelu jsou *rozdělení ploch*, *smyčkové řezy*, *vytažení* a *protlačení*. U kalhot se zvýraznily jednotlivé Edge a pomocí ***Ctrl + e*** v nabídce je zvoleno *ostrost stopy (Mark sharp)*, tím vznikla ostrost hran.



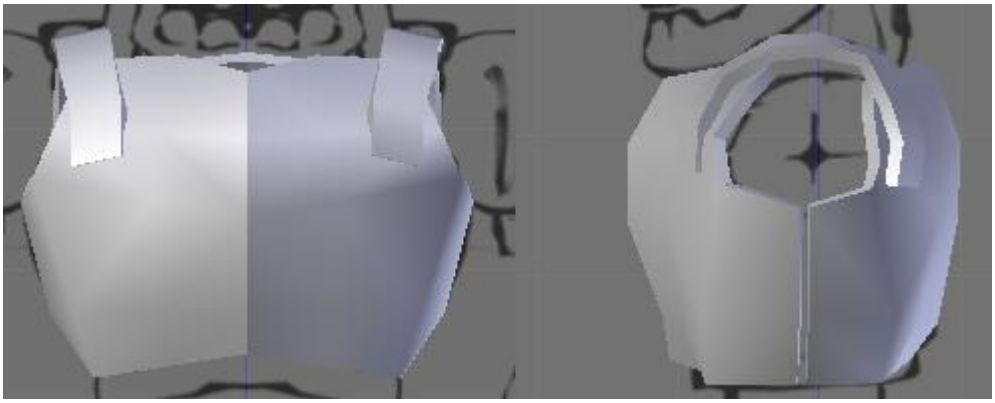
Obrázek 16: Helma (Zdroj: vlastní)

Aby pokrývky ramen přesně kopírovaly jejich tvar, je možné zvolit požadované face na rameni a poté je duplikovat pomocí kláves ***Shift + D*** a potvrdit klávesou ***Enter*** a poté se oddělí pomocí klávesy ***P*** pro separaci do samostatného modelu. Následně postačí pouze upravit velikost, vzhled a uzavřít případné mezery pomocí nových face.



Obrázek 17: Kalhoty a krytí ramen (Zdroj: vlastní)

Stejným způsobem jako pokrývky ramen bylo vytvořeno brnění. Část těla, kterou má kopírovat, byla duplikována, oddělena a následně upravena. Jelikož ale u brnění je požadována tloušťka, protože je rozdělené na přední a zadní část, spojené koženými popruhy, bylo nutné použít modifikátor *Solidify*.



Obrázek 18: Brnění (Zdroj: vlastní)

2.1.7 VÁZA A OSTATNÍ MODELY

Zbytek modelů byl vytvořen pomocí stejných technik, kterými se modelovala postava. Byly vytvořeny modely pro meč, vázu, minci, podlahu, zeď, dveře, truhlu a past.

Váza, jako jediný model ve hře, se stane předmětem destrukce. Aby bylo možné animovat rozpad vázy, je nutné ji rozbít na více částí. K tomu v Blender slouží addon *buňkových zlomů (Cell Fracture)*, který rozdělí vázu na několik částí pomocí již předem naskriptované funkce.

2.1.8 DOKONČENÍ MODELŮ

Poté co jsou vytvořeny všechny části postavy, je nutné si všechny modely projít, podívat se, jestli nevznikly nějaké problémy nebo je potřeba modely ještě upravit, aby odpovídaly představám autora. Mohou vzniknout problémy při úpravách modelu, např. že jsou omylem vytvořeny identické Vertex na stejném místě, to způsobuje v modelu zbytečnou geometrii navíc, a proto je dobré v každém modelu, použít nástroj *odstranění dvojníků (Remove Doubles)*, který odstraní Vertex dle nastavené vzdálenosti od sebe.

2.2 UV MAPOVÁNÍ

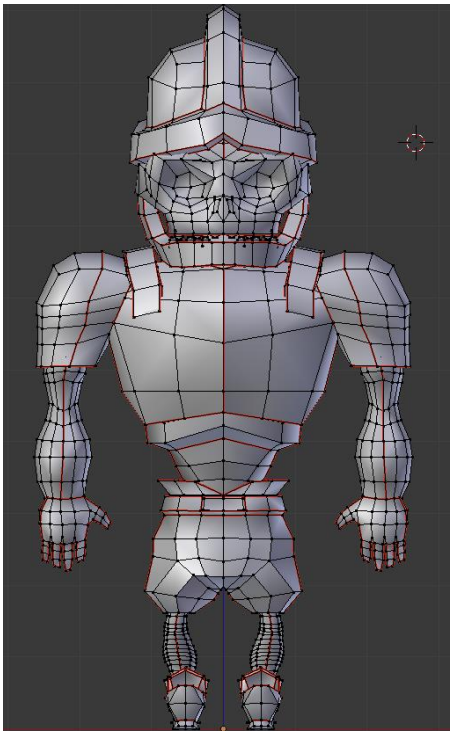
UV značí osy, kde má Blender vnitřně uložené pozice všech Vertex v osách ***X***, ***Y***, ***Z*** které se používají právě pro projekci na 2D plochu. Je to proces, který převádí model na 2D plochu za účelem texturování. Bez této úpravy by software nebyl schopen poznat, jak

má texturu správně vykreslit. UV jsou tedy 2D pozice Vertex 3D modelu. Tento proces se nazývá UV unwrapping. (Villar, 2017)

Blender obsahuje nástroje určené pro UV unwrapping, který je velice populární a dokonce sbírá chválu i z profesionálních sfér, např. z filmových projektů v Hollywoodu, kde animátoři používájí Blender pro vytváření UV. Proces unwrapping funguje na principu vkládání řezů, podle kterých se následně model rozloží na 2D plochu. (Villar, 2017, stránky 159-161)

2.2.1 UV ROZBALENÍ (UNWRAPPING) POSTAVY

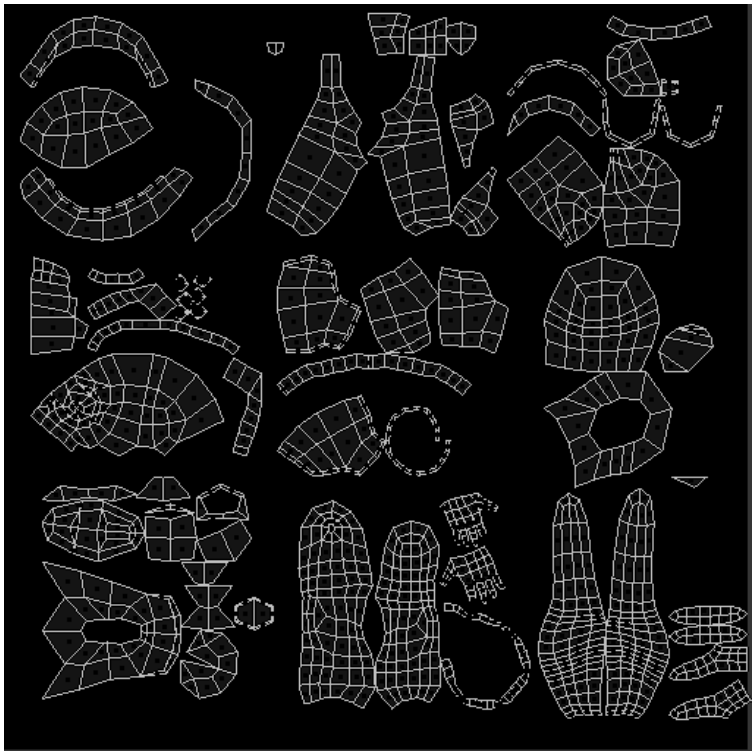
Aby bylo tedy možné texturovat model, je nutné provést UV rozbalení. Pro každou jednotlivou část je proveden samostatně. Nejprve se označí jednotlivé Edge podle kterých bude model rozložen, pomocí kláves ***Ctrl + E*** je otevřena nabídka pro úpravu Edge, je zvolena možnost *zvýraznění stopy (Mark seam)* a všechny označené Edge se červeně zvýrazní.



Obrázek 19: UV mapování (Zdroj: vlastní)

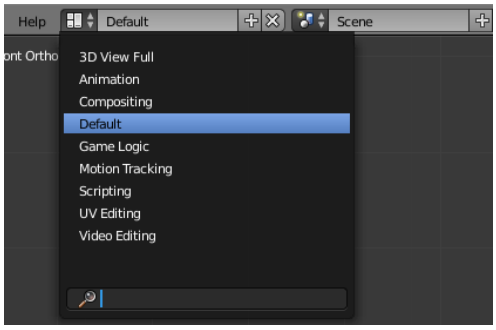
Na Obrázek 19: UV mapování je vidět celý model, který byl označen řezy. Jakmile je hotový tento proces, v horní části Blenderu je nutné přepnout do řežimu UV editing.

Klávesou **U** je otevřena nabídka pro rozbalení, po jednotlivých částech je model rozbalován.



Obrázek 20: UV rozbalení (Zdroj: vlastní)

Poté, co je model *rozbalen*, je možné začít texturovat. Je zde více možností, buď je vytvořena textura mimo Blender, kdy se UV mapa exportuje do programů, jako jsou Adobe Phostohop, Gimp či Krita a tam se následně vytvoří textura nebo je možné použít nástroje Blenderu pro kreslení.



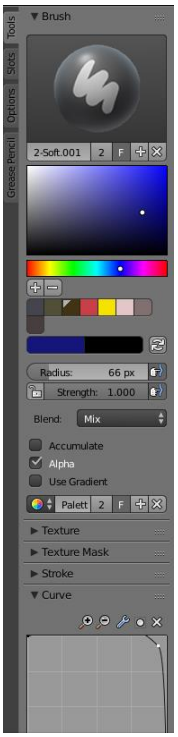
Obrázek 21: UV Editing (Zdroj: vlastní)

V této práci bude využit Blender. Je možné kreslit přímo na UV mapu a tím se obarví model v reálném čase, ale to není velice přesné, lepším řešením je přepnout Blender do režimu *kresby textury (Texture paint)* a zvolit možnost *výběr plochy pro kreslení (Face*

selection masking for painting). Následně jsou zvoleny Face, na které je možné kreslit na okolních, které nejsou vybrány, nebudou změny provedeny.

Obrázek 22: Kreslení textury (Zdroj: vlastní)

Blender obsahuje *nástroje pro kresbu (Paint tools)*, kde si uživatel zvolí štětec, a jeho vlastnosti jako např. tloušťka, barva, velikost. Je možné definovat křivku, podle které se štětec bude chovat, pokud budeme chtít tvrdý, po celé velikosti se bude aplikovat barva stejně, naproti tomu měkký štětec bude mít barvu sytější uprostřed, která bude méně sytá u krajů.



Obrázek 23: Blender paint tools (Zdroj: vlastní)

Všechny barevné změny takto provedené se zaznamenávají na UV mapu v reálném čase. Jakmile je model otexturovaný, je spojen do jednoho Mesh a připravený pro tvorbu kostry.

2.2.2 UV ROZBALENÍ ZBYLÝCH MODELŮ

UV mapy byly vytvořeny ještě pro modely meče, truhly, dveří, zdi a truhlu. Ostatní modely získají texturu až ve vývojovém prostředí Unity, proto není nutné pro ně tvořit UV mapování. Meč byl texturován pomocí Paint tool nástroje v Blenderu, pro zbytek modelů byla stažena textura z internetového zdroje. (Paulo, 2018)

2.3 KOSTRA MODELU

Aby bylo možné model animovat, je nejdříve nutné vytvořit tzv. kostru. Kostra se skládá z kostí, které mají konec *Tip*, tělo *Body* a začátek *Root.* Aby bylo možné celou kostrou hýbat, následující *Root* kosti vždy vychází z jejího *Tip*.

2.3.1 VYTVOŘENÍ KOSTRY

Klávesovou zkratkou ***Shift + A*** vložíme do scény *kostru (Armature)*. Je vytvořena první kost, která je umístěna do spodní části těla, kde se nachází pánev, následně pomocí nástroje *vytažení* jsou vytaženy další kosti. Tyto kosti se budou starat o pohyb těla. Stejným postupem jsou vytvořeny kosti pro ruku a nohu.

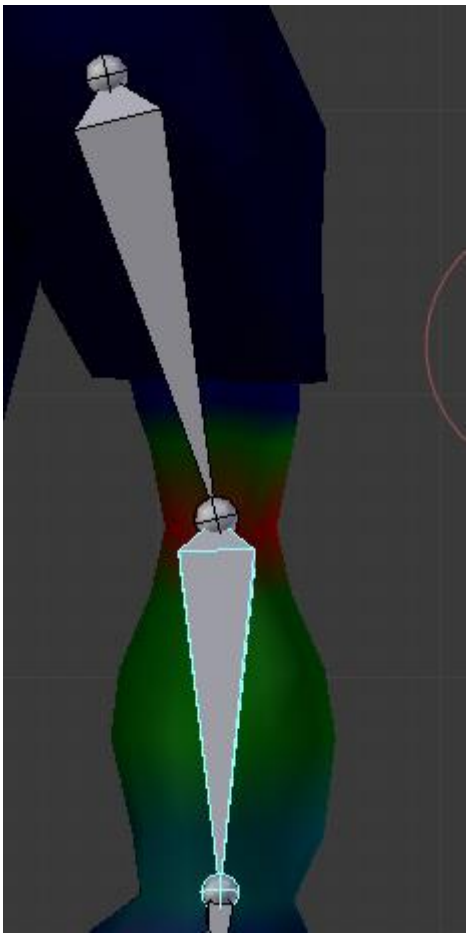
Důležité je pojmenování kostí, jelikož nohy a ruce jsou v páru, bude zapotřebí mít kosti pro obě strany. Je možné duplikovat jednu stranu, a pokud je kost pojmenována správně, Blender automaticky pozná, že jde o pár a přehodí název. Správný název by tedy měl mít vždy příponu .L nebo .R podle toho, jaká strana je právě tvořena. Po duplikaci Blender přehodí .L na .R a opačně.

Obrázek 24: Kostra postavy (Zdroj: vlastní)

2.3.2 KRESLENÍ VÁHY (WEIGHT PAINTING)

Jelikož jsou vytvořené všechny kosti, které se budou starat o deformaci modelu, musí být spárovány s modelem. V objektu módu se nejprve označí model a následně kostra, klávesou C***trl + P*** se otevře *menu pro párování (Parent menu)* a je zvolena možnost *deformace kostry s prázdnými skupinami (Armature deform with empty groups)*, ta nám vytvoří skupiny Vertex s názvem dané kosti.

Když jsou vytvořeny skupiny Vertex, je teď možné každé kosti určovat váhu, tím daná kost deformuje část modelu. Označením modelu a přepnutím do režimu *kreslení váhy* je možné zvolit kost a následně si vybrat skupinu Face a pomocí štětce přidávat váhu. Štětec funguje na stejném přístupu jako při texturování, model obarví podle intenzity váhy, která se pohybuje od 0 do 1. Takto je zvolena každá kost a přiřadí se jednotlivé váhy, tak jak je potřeba deformovat model.



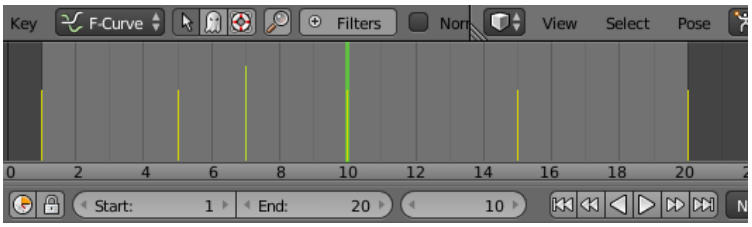
Obrázek 25: Váha pro ohyb ruky (Zdroj: vlastní)

Na Obrázek 25: Váha pro ohyb ruky, jsou vidět jednotlivé váhy pro ohyb ruky. Kde červená odpovídá hodnotě 1, na této části se bude deformace provádět na 100%, zelená odpovídá hodnotě 0.5, světle modrá 0.25 a tmavě modrá 0.

Jakmile jsou vytvořeny váhy pro každou kost, je možné modelem pohybovat v *pózovacím módu (Pose mode)*, každou jednotlivou kostí. V této chvíli ale není možné, aby se model ohnul v kolenou, je proto nutné vytvořit tzv. ovládací kosti. K tomu je zapotřebí vytvořit tzv. *IK řetězec (Inverse Kinematics)*, to dovolí, aby se z jedné kosti dala ovládat celá končetina. Ovládací kosti pro nohu byly vytvořeny pomocí video návodu. (Lile, 2015)

2.4 ANIMACE

Poté, co je vytvořena kostra a model se nyní deformuje podle představ, je možné začít animovat. Animace probíhá úpravami kostí do pozic, které se následně uloží do tzv. *klíčových snímků (Keyframe)*, který se nachází na časové ose. Je tedy nutné si zvolit, kolik *snímků (Frame)* bude animace obsahovat.

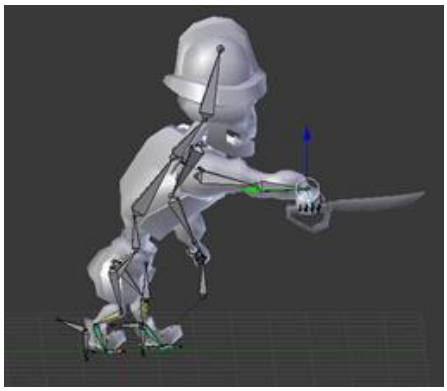


Obrázek 26: Časová osa (Zdroj: vlastní)

Zapnutím automatického ukládání *klíčových snímků* se po jakékoliv změně kostry automaticky uloží. Nejprve se tedy zvolí daný *snímek* animace, poté se upraví model do požadované polohy a proběhne uložení.

Animace byly tvořeny postupně, nejdříve se vytvořila póza pro začátek a konec. Následně byl upravován model v různých *snímcích* mezi koncem a začátkem, aby vznikla plynulá animace daného pohybu.

Tímto způsobem byly vytvořeny animace pro útok, smrt, zásah, stání na místě postavy a nepřítele, pohyb, běh, zmáčknutí tlačítka a otevření truhly.



Obrázek 27: Póza při útoku (Zdroj: vlastní)

2.5 Export

Vše je připraveno pro export do vývojového prostředí, kde se následně vytvoří interaktivita a vznikne počítačová videohra. Jelikož je zvoleným softwarem Unity, není nutné modely exportovat. Unity se samo postará o import modelů se všemi nezbytnými věcmi, jako jsou animace a UV mapování.

3 VÝVOJ MULTIPLATFORMNÍCH 3D INTERAKTIVNÍCH ANIMACÍ

K vytvoření 3D interaktivní animace je zapotřebí využít herní jádro nebo si vytvořit vlastní. V praxi ale herní jádro vytváří velká studia, která k tomu mají dostatečně prostředků, jak pracovní sílu, tak i finance. Vývoj kvalitního herního jádra se pohybuje v rámci miliónů $.

Herní jádro umožňuje převedení vytvořených modelů a jejich komponentů na interaktivní úroveň pomocí skriptování. V dnešní praxi se představuje speciální varianta skriptování, u které nejsou zapotřebí znalosti z programování, tzv. *vizuální skriptování (Visual scripting)*. Jedná se o sestavování a propojení bloků, které mají uvnitř naprogramované chování, uživatel se musí rozhodnout, jaké budou vstupy a výstupy.

3.1 HERNÍ JÁDRO

„Herní jádro je software, v kterém je možné vytvořit videohru. Tyto software se skládají z renderovacího engine pro zobrazování herních assetů, fyzikální engine pro kontrolu interakce mezi objekty, práci se zvuky, skriptování, animace, a mnoho dalších funkcí které jsou spojovány s moderními videohrami.“ (Totten, 2012, str. 20)

Mezi nejznámější a nejpoužívanější 3D multiplatformní herní jádra patří Unreal Engine, Unity, Cry Engine a Godot jako open source zástupce. Každý z těchto uvedených nástrojů pro vytvoření interaktivní animace je v základu totožný, avšak oplývají unikátními vlastnostmi, které ostatní nemají.

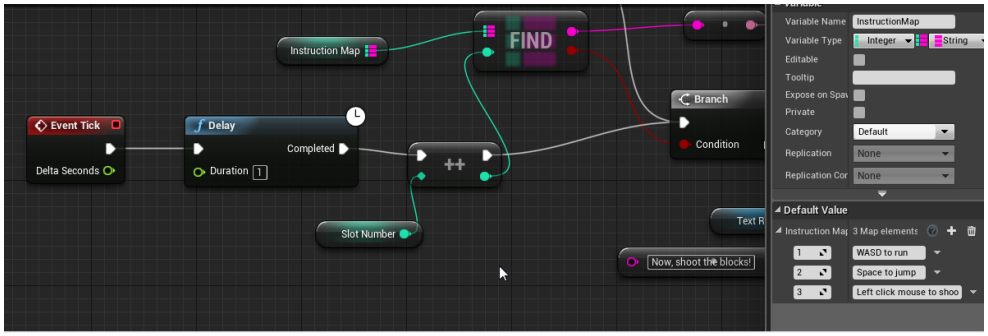
3.1.1 UNREAL ENGINE

Unreal Engine vytvořila společnost Epic Games, jedná se o nejnovější verzi z dlouhodobě vydávaných a aktualizovaných nástrojů pro vývoj videoher. Unreal Engine 4 je profesionální prostředí, které kombinuje špičkové fyzikální renderování materiálů v reálném čase, odrazy a osvětlení s robustní sadou nástrojů k vytvoření atraktivního interaktivního zážitku. Tyto nástroje pracují se simulací fyziky, osvětlením, stínováním, uživatelským prostředím, propagací zalesnění, renderováním, masivními terény, komplexními materiály, vizuálním skriptováním, animacemi, simulací částic, prací s filmem, multiplayer a mnoho dalšího. (Shannon, 2018, stránky 1-3)

Celý zdrojový kód Unreal Engine 4 je otevřený a psaný v C++, tím dává možnost uživateli zasahovat do kódu, studovat a případně ho i modifikovat. Unreal editor je možné

pouštět na platformách Windows, OS X a Linux. Herní engine je multi-platformní a je možné vyvíjet pro platformy Windows PC, Playstation 4, Xbox One, Mac OS X, iOS, Android, VR, Linux, SteamOS a HTML5. (Epic Games, 2018)

Platební model je zdarma, je zde ale povinnost platit poplatky, pokud oficiálně vydáváme dílo za účelem zisku. Pokud čtvrtletní příjem přesáhne 3 000 $ za videohru, je nutnost zaplatit 5% poplatek. (Epic Games, 2018)



Obrázek 28: Blueprint systém. In: Unreal Engine Manual [online]. [Cit. 5.4.2018].Dostupné z WWW: https://docs.unrealengine.com/portals/0/images/Engine/Blueprints/UserGuide/Maps/maps_topicBanner.png

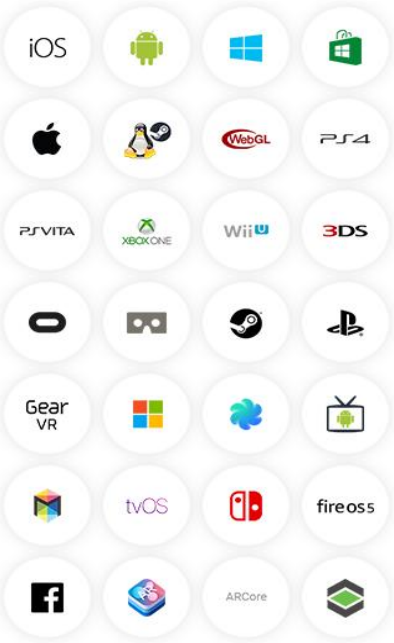
3.1.2 UNITY

Herní jádro Unity je vyvíjeno společností Unity Technologies, jako platforma která je schopná produkovat 2D, 3D, VR a AR videohry a aplikace. (Unity Technologies, 2018) Unity podporuje většinu známých platforem, jejich seznam je zobrazen na Obrázek 29: Unity platformy.

Nástrojová vybavenost závisí na tom, jakou verzi máme k dispozici. Unity má 3 rozdílné verze, Personal, Plus, Pro. Personal edice je zdarma, zbylé dvě vyžadují měsíční poplatky.

Základní výbava Unity je pro všechny verze stejná, obsahuje editor, vlastní fyzikální jádro, osvětlení, renderovací jádro, materiály, skriptování pomocí jazyka C#, animační editor, simulaci částic a mnoho dalšího. Hlavním rozdílem je, že Unity pracuje se systémem *drag and drop*. V herních Assetech jsou vytvořeny složky, které mají fyzické a identické umístění na disku. Pokud je nahrán obsah do těchto složek, je zobrazen v Unity a je připraven k použití, pokud je podporován formát souboru. Naopak pokud v Unity

je přetáhnut soubor do zobrazeného okna s Assety, automaticky se projeví změna ve složce na disku.

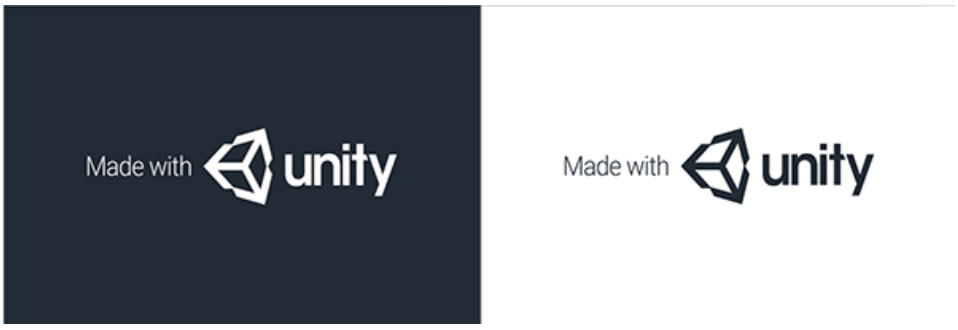


Obrázek 29: Unity platformy. In: Unity Products [online]. [Cit. 6.4.2018]. Dostupné z WWW: https://unity3d.com/unity

Personal

Tato licence je zdarma určená spíše pro začátečníky či studenty, avšak má i svá omezení podle ročního příjmu. Pokud je roční příjem z prodejů víc než 100 000 $ tak vzniká povinnost platit za licensi Plus. (Unity Technologies, 2018)

Každý vytvořený projekt obsahuje tzv. Splash screen, který se objeví pří zapnutí videohry či aplikace, a zobrazí se nám logo unity. Splash creen, unity logo ani opacity nemůže být vypnuto. (Unity technologies, 2018)



Obrázek 30: Unity logo. In: Unity User Manual [online]. [Cit. 9.4.2018]. Dostupné z WWW: https://docs.unity3d.com/uploads/Main/LogoStylesSideBySide.png

Plus

Licence plus je zpoplatněna 35 $ měsíčně a oproti personal verzi nabídne odstranění či customizaci *úvodní obrazovky (Splash screen)*. Zprávy o výkonu, které nám zajišťuje sbírat data o chybách v reálném čase ze všech zařízení a platforem. Dále jsou součástí vylepšené analytické nástroje, multiplayer pro 50 hráčů, Unity v tmavé barvě, reklamy v aplikacích, nákupy v aplikacích a přehled o tom kdo v týmu používá právě v danou chvíli Unity. Pokud příjmy přesáhnou 200 000 $, je povinnost zakoupit licensi pro. (Unity Technologies, 2018)

Pro

Za pro licenci je nutnost platit 125 $ za měsíc. Nabízí to samé co verze plus a k tomu ještě více analytických nástrojů, Multiplayer až pro 200 hráčů, možnost zakoupit zdrojový kód a prémiovou podporu od Unity teamu. (Unity Technologies, 2018)

3.1.3 CRY ENGINE

Cry Engine je vyvíjen společností Crytek a jeho současná verze už je pátou generací tohoto herního jádra. Je napsán v jazyce C++, a skriptování probíhá pomocí C#. Nabízí velice podobné nástroje jako jiné herní jádra, ale má i své vlastní nástroje, jako *Flowgraph*, což je vizuální skriptování podobné jako má Unreal Engine. Crytek se chlubí nástroji pro vizuální zpracování, které jsou nejlepší ve své třídě a realistickými animacemi postav ve video herním průmyslu. (Crytek, 2018)

Obrázek 31: Kingdomcome deliverence In: Cry Engine Showcase [online]. [Cit. 5.4.2018]. Dostupné z WWW: https://www.cryengine.com/files/showcase/images/_1280x/04.jpg

Všechny projekty mohou být produkovány, editovány, a testovány ihned díky sytému WYSIWYP (What you see is what you play) které má taktéž Unity. Jádro zvládne okamžitou konverzi a optimalizaci Assetů v reálném čase, umožňující multi-platformní změny všech elementů ve vývoji videohry. To vede ke zvýšení rychlosti a kvality vývoje, a razantně snižuje risk při vytváření multi-platformních her. (Crytek, 2018)

Vyvíjený produkt je možné publikovat na Windows, Linux, Playstation 4, Xbox One, Oculus Rift, OSVR, PSVR, HTC VIVE. V budoucnosti se plánuje i podpora mobilních zařízení. V Cry Engine byla dokonce vyvinuta videohra z českého herního studia War Horse Kingdomcome deliverence. (Crytek, 2018)

Největší nevýhodou a slabinou se stává dokumentace, která není příliš chválená, je špatně napsaná a nekompletní. Zároveň by mohl sužovat nového uživatele přístup k informacím. Knih, tutoriálů či video návodů, které by se zaobíraly tématikou tohoto jádra je pramálo.

Cry Engine je zdarma, ale podobně jako Unreal Engine, i zde je povinnost poplatků z prodeje. Prvních 5000 $ z projektu za rok je bez poplatku, poté se musí zaplatit 5%.

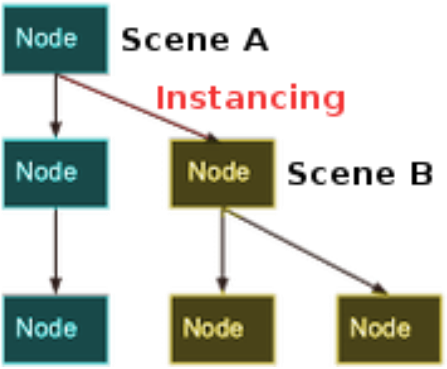
3.1.4 GODOT

Godot je open source jádro pro 2D či 3D tvorbu a je kompletně zdarma. Mezi podporované paltformy patří Windows, OS X, Linux, UWP, BSD, Haiku, iOS, Android a export na webové rozhraní (HTML a Web Assembly). Godot obsahuje všechny nutně potřebné nástroje pro výrobu 2D, 3D her, od práce s materiály až po animační složku. (Linietsky & Manzur, 2018)

Hlavním rozdílem, kterým se Godot liší, je, že konkurence má většinou omezené možnosti výběru programovacího jazyka. Godot má ovšem na výběr rovnou z několika, jako je GDscript, což je jazyk podobný python, vyvinutý přímo pro Godot, C#, C++ a vizuální skriptování pomocí bloků. Díky komunitě je možné používat jazyky jako Python, Nim, D a další. Godot disponuje vlastním IDE pro kódování, který zvýrazňuje syntaxi, a v reálném čase analyzuje a dokončuje kód. Zároveň má i integrovanou dokumentaci bez nutnosti jít na web. (Linietsky & Manzur, 2018)

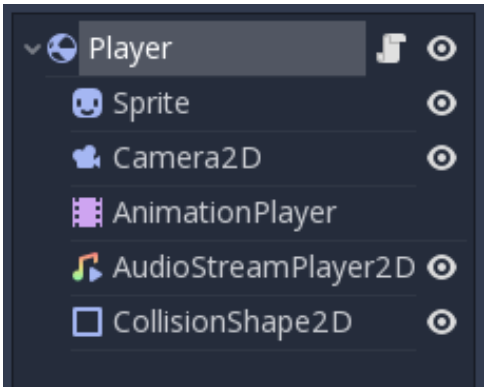
Godot pracuje s tzv. *Nodes* a *Scenes*. Node je schránka, která obsahuje elementy jako je 3D objekt, 2D obrázek, časovač, hudební přehrávač, věci, které vidíme, ale i které

jsou na pozadí. Může mít skript, který bude určovat chování. Zároveň node může mít dítě další node. (Linietsky & Manzur, 2018)



Obrázek 32: Instancování. In: Godot User Manual [online]. [Cit. 5.4.2018]. Dostupné z WWW: http://docs.godotengine.org/en/3.0/_images/instancing.png

Každý vytvořený Node je dítě scény a scéna může mít další scény jako děti, tomu se v Godot říká instancování. Scéna jménem Player, a jednotlivé node, sprite, Camera2D, AnimationPlayer, AudioStreamPlayer2D a CollisionShape2D. (Linietsky & Manzur, 2018)



Obrázek 33: Scéna In: Godot User Manual [online]. [Cit. 5.4.2018]. Dostupné z WWW: http://docs.godotengine.org/en/3.0/_images/scene_tree_example.png

3.2 VÝBĚR VHODNÉHO JÁDRA

Při výběru vhodného software je dobré sestavit seznam několika bodů a zjistit, zda je z nich alespoň většina splněna, a tím by mělo být jasné, které herní jádro by mělo být vhodné. (Florian, 2017)

- Zkušenosti
- Dostatečná dispozice informačních zdrojů
- Rozpočet
- Velikost týmu

- 2D, 3D
- Cíl, žánr. Např. pro RPG existuje speciálně vytvořený herní jádro RPG maker
- Cílová platforma
- Způsob skriptování, provedení

Každé jádro nabízí v základu ty samé vlastnosti, nejlepší volbou je vyzkoušet více herních jader a podle toho, jak se uživatel cítí komfortně a zorientovaně v prostředí, poté je vhodné si zvolit. Funkce v kterých se herní jádro zásadně liší a je v tom mnohem lepší než konkurence, se objeví až poté, co přijdou zkušenosti.

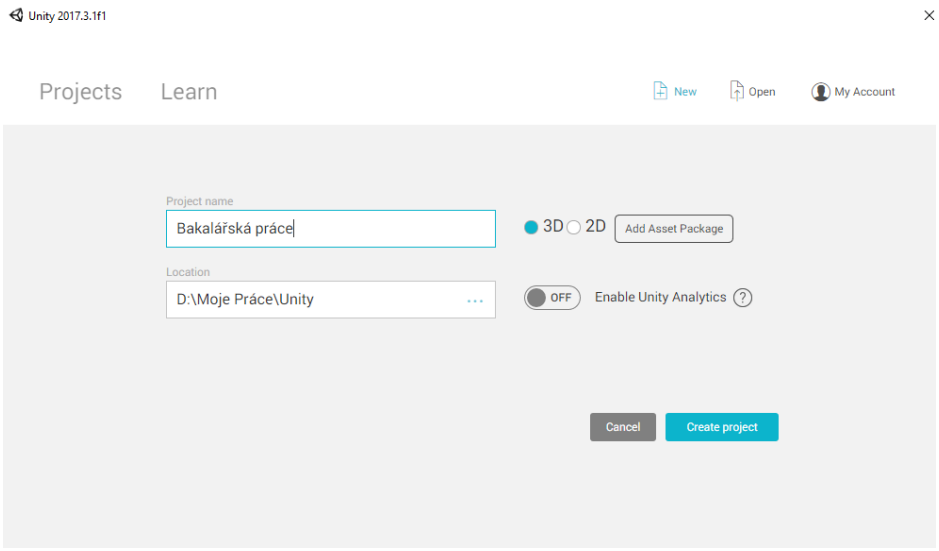
V základu jsou všechna herní jádra zdarma, pokud tedy uživatel, který chce provozovat vývoj videoher, aplikací pouze jako koníček, se o žádné poplatky zajímat nemusí, jelikož ty přichází většinou až od několika tisíc dolarů za prodej ročně. Pokud se ovšem bojí takovýchto poplatků a že bude mít omezené možnosti a funkce jádra, je tu vždy free scéna, kde je bezpochyby nejlepší Godot, u kterého se komunita rozrůstá exponenciálně.

Unity bylo zvoleno z důvodu existence mnoha videonávodů, knih a tutoriálů na webu, díky kterým je možné najít všechny informace potřebné k dokončení práce. Komunita je neskutečně rozšířená a nápomocná. Unity je dokonce i přívětivé pro začátečníky a je možné vytvářet videohry/applikace na všechny platformy. Dalším aspektem pro výběr byl programovací jazyk C#.

Z hlediska výuky, jsou tato jádra nevhodné pro základní školy. Práce v nich je natolik komplexní, žák již potřebuje mít potrénované logické uvažování a alespoň základní vlastnosti programování. Z toho důvodu by dle mého názoru bylo vhodné volit vývojové prostředí, které pracuje s bloky, které již obsahují dané funkce a metody, např. Scratch. Na střední a vysoké škole by měli být žáci či studenti schopni zvládnout profesionální herní jádra, která byai popsána v této kapitole.

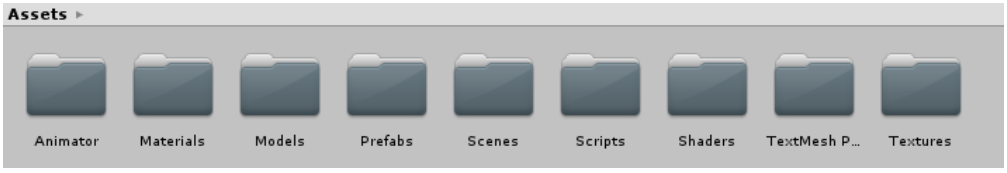
4 VYTVOŘENÍ INTERAKTIVNÍ ANIMACE

Nejprve je nutné vytvořit projekt v Unity. Pří otevření se zobrazí obrazovka, kde jsou všechny vytvořené projekty rychle k otevření tzv. rychlý přístup, možnost vytvořit nové či otevřít projekty, které již nejsou v rychlém přístupu. Při vytváření nového projektu je nutné zadat název, umístění, zda projekt bude 3D nebo 2D. Cílem této práce je vytvořit 3D interaktivní animaci, zvolena bude proto možnost 3D a vypnutí Unity Analytics, které jsou pro tento projekt nepodstatné a nevyužitelné.



Obrázek 34: Vytvoření projektu (Zdroj: vlastní)

Všechny modely, které byly vytvořeny v Blenderu, jsou následně importovány do Unity *drag and drop* systémem, který usnadňuje v Unity práci, jelikož odpadá nutnost importovat všechny assety. Nejdříve jsou v projektu vytvořeny patřičné složky, které nejsou nutné k dokončení interaktivní animace, ale předchází se tímto zbytečnému zmatku a nepořádku.

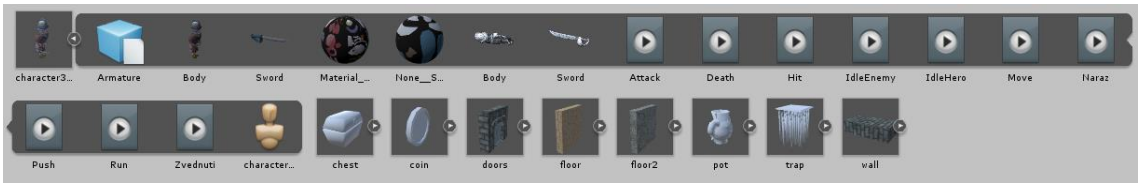


Obrázek 35: Herní assety (Zdroj: vlastní)

Z nabitých zkušeností z předešlé práce v Unity, byly vytvořeny tyto složky:

- Animator - animace vytvořené v unity a ovladácí prvky všech animací.

- Materials - materiály vytvořené v unity
- Models - 3D modely z Blenderu
- Prefabs – speciální objekty Unity
- Scenes – uložené scény, které fungují jako úrovně, menu, různé obrazovky
- Scripts – C# scripty pro kontrolu objektů
- Shaders – shadery pro úpravu vzhledu objektů
- TextMeshPro – úprava textů
- Textures – všechny obrázky a textury

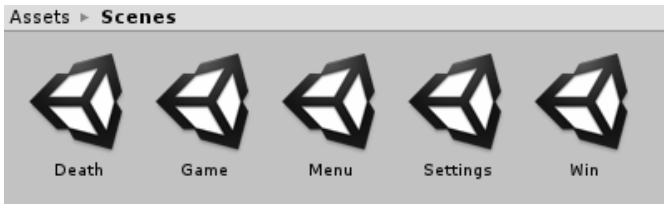


Obrázek 36: Importované modely (Zdroj: vlastní)

Pokud je rozkliknuta šipka vedle assetu, jsou vidět všechny dílčí komponenty modelu jako animace, uv mapování s texturou, kostra. V této chvíli je vše připraveno pro vytvoření 3D interaktivní animace.

4.1 VYTVOŘENÍ HERNÍ SCÉNY

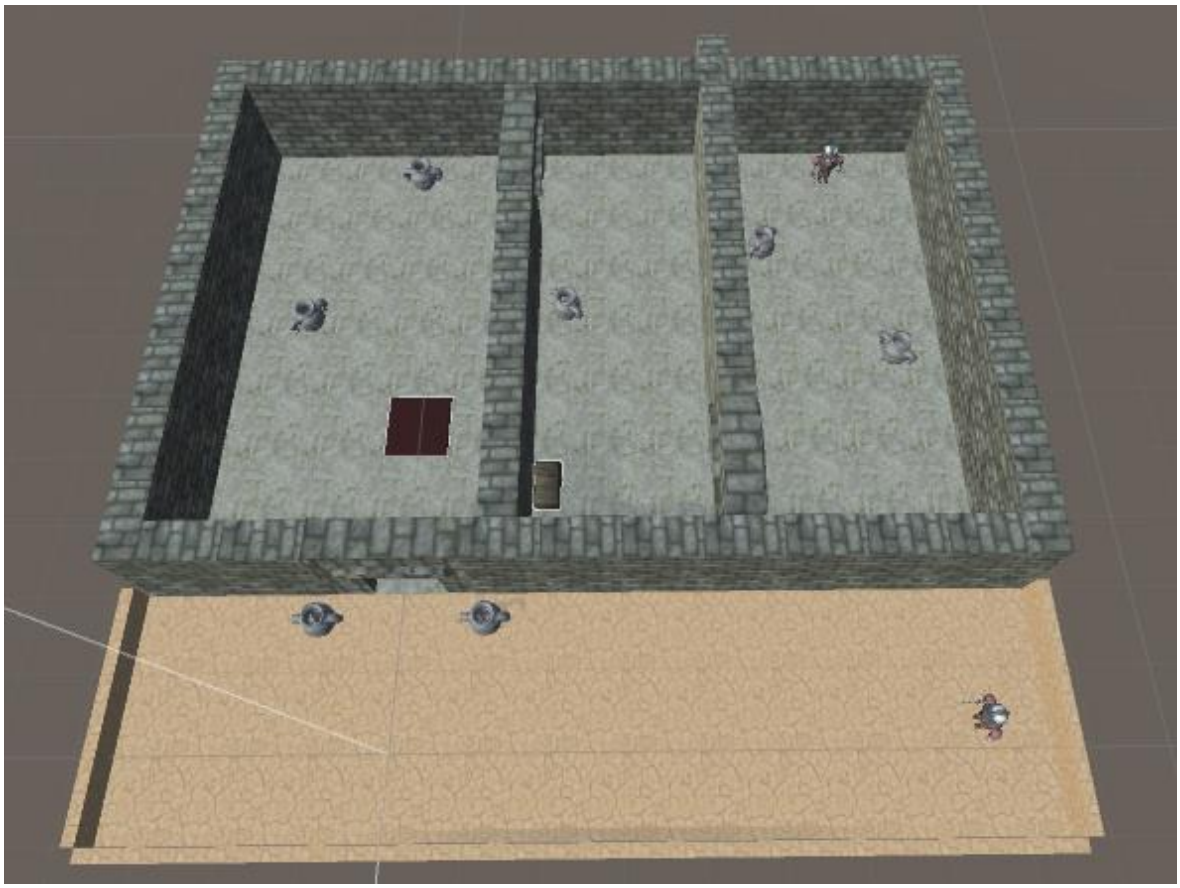
Nejprve je nutné vytvořit scénu, jinak by nevzniknul prostor, kam je možné vkládat modely. Zároveň jsou vytvořeny všechny scény pro rozdílné obrazovky, které budou potřeba. Cílem je vytvoření fungující videohry. Je tedy nutné mít scénu pro samotnou hru a menu, odkud se bude hra spouštět, vypínat a zároveň je možné prohlédnout nastavení ovládání. Pokud postava zemře, objeví se obrazovka určená pro tento případ a pokud uživatel splní podmínky, bude přesměrován na obrazovku určenou pro výhru.



Obrázek 37: Scény (Zdroj: vlastní)

Vytvořena je scéna videohry, ostatní se budou tvořit, až práce na videohře pokročí, v tuto chvíli jsou nepodstatné. Postava potřebuje nějaký povrch, po kterém se bude moci pohybovat.

Byly vytvořeny 2 typy podlahy, jedna venkovní, která má texturu jako z vyprahlé pouště a druhá je vnitřní, kamenná. Záměrem bylo vytvoření budovy, a k tomu poslouží modely zdi a dveří, které mají texturu, která připomíná staré kamenné budovy. V této úrovni byly rozmístěny náhodně vázy, které je možné rozbít, a vlastně i nutné ke splnění podmínek pro vítězství ve hře, past, tlačítko pro otevření dveří, truhlu skrývající minci a nepřítele.



Obrázek 38: Scéna – Game (Zdroj: vlastní)

4.1.1 TEXTUROVÁNÍ V UNITY

Ne všechny modely byly texturovány v Blenderu, některé jsou texturovány přímo v Unity. Na rozdíl od Blenderu nemá Unity žádnou funkci pro kreslení na model, tak byly nalezeny textury ze zdrojů na internetu, které jsou zdarma a k volnému použití. Takových

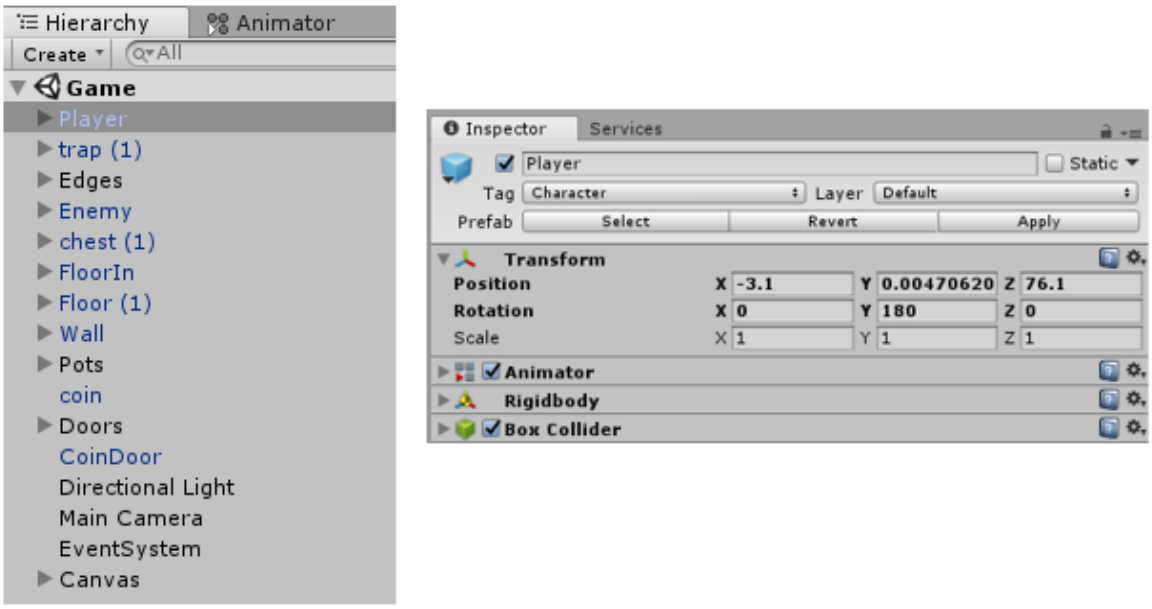
stránek existuje velké množství. (Hughes, 2017) Z tohoto webu byla vybrána webová stránka, kde byly nalezeny všechny textury, které jsou ve hře používány. (Paulo, 2018)

Jedná se o profesionálně vytvořené textury, které mají různé mapování pro vytvoření realistického podání. Pokud tedy není čas nebo zkušenost pro vytváření vlastních textur, je možnost stáhnout si takovéto textury.

4.2 OBJEKTY V UNITY

Všechny objekty, které jsou vloženy do scény, se zobrazí v hierarchii. Po vybrání objektu jsou zobrazeny všechny informace o jeho poloze, rotaci, rozměrech a komponentech zobrazeny v inspektoru.

Komponenty, které jsou na objektu *Player*, jsou *Transform*, který udává pozici, rotaci a velikost objektu v prostoru. *Animator* se stará o všechny animace které objekt bude vykonávat. *Rigidbody* je fyzikální jádro vytvořené pro Unity, které aplikuje fyzikální chování na objekt a umožní interakci ve scéně. *Box Collider* vykreslí neviditelnou mříž kolem modelu, která způsobí srážku s jiným *kolizním objektem (Collider)* a nedovolí modelu projít. *Collider* může fungovat i jako spouštěč, který pak nezpůsobí srážku, ale místo toho dá vědět jádru, že ke srážce došlo, a tato informace je použita ve skriptech pro definování chování.



Obrázek 39: Hiearchie, Inspektor (Zdroj: vlastní)

4.2.1 ŠABLONY (PREFABS)

Šablony (Prefabs) jsou vytvořeny z objektů ve scéně a fungují jako šablony. Při vložení objektu do scény vznikne jeho instance, při vložení dalšího stejného objektu vznikne jeho druhá instance. Případné změny jsou pouze viditelné pro danou specifickou instanci. Pokud je zamýšleno, aby se ve scéně nacházelo více identických instancí, bylo by nutné každou vložit a modifikovat, což by bylo neefektivní a matoucí.

Z tohoto důvodu existují tyto šablony, které nám dovolí uložit instanci jako *šablonu*, a jakékoliv změny, které byly provedeny. Následně při nutnosti vložit další instanci identického objektu postačí vložit do scény *šablona* a tato instance zdědí všechny komponenty, jediné co není zděděno, je pozice a rotace, která je individuální pro každou instanci.

4.3 SKRIPTOVÁNÍ HERNÍ SCÉNY

Skripty jsou v unity kódovány pomocí jazyka C# či Javascript a upravovány v IDE Microsoft visual studio 2017 či Monodevelop. V této práci je všechno skriptování prováděno pomocí C# ve Visual studiu. Skripty se přidávají na objekty jako komponenty, a tím umožní danému objektu interagovat s okolím.

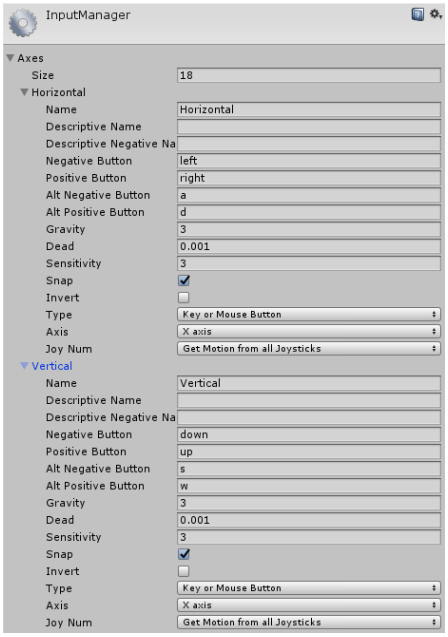
Unity má vlastní knihovny, které se využívají při skriptování. Každý nový skript je vytvořen se 2mi základními metodami *Start()*, která se volá pouze jednou, a to při spuštění aplikace či videohry. Metoda *Update()* se volá každý snímek. Aby bylo možné používat Unity knihovny, je nutné použít v hlavičce skriptu *using UnityEngine;*.

Všechny skripty použité při tvoření interaktivní animace se nachází v seznamu příloh na konci této práce.

4.3.1 POHYB HRÁČE

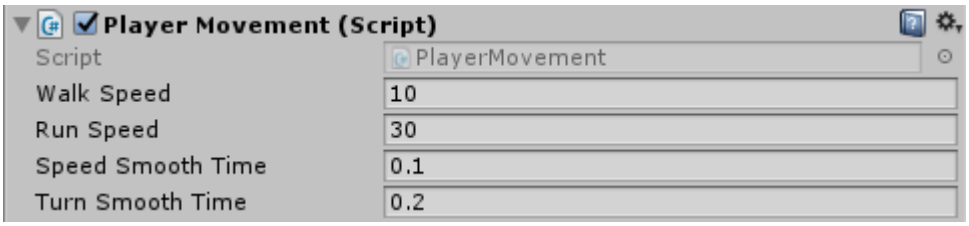
Všechny komponenty, modely jsou vloženy do scény, je tedy nutné vytvořit skript pro hráče, který se bude starat o ovládací funkce a interakce. Pro vytvoření pohybu pro postavu je nutné nejprve inicializovat proměnné, které budou potřeba pro vytvoření příslušných akcí.

Hráč je naprogramován tak, aby se pohyboval pomocí šipek či kláves **W**, **S**, **A**, **D**. Byla použita funkce Unity *Input.GetAxisRaw*, kde jsou předdefinované klávesy pro pohyb pomocí změny horizontální a vertikální osy.



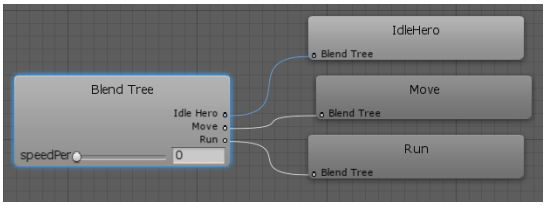
Obrázek 40: Input Manažer (Zdroj: vlastní)

Ve skriptu jsou použité public proměnné, které dovolují změny skriptu v Unity editoru, tyto hodnoty se zapíší do skriptu při volání metody *start()*, která se volá při spuštění hry.



Obrázek 41: Veřejné (Public) proměnné v editoru (Zdroj: vlastní)

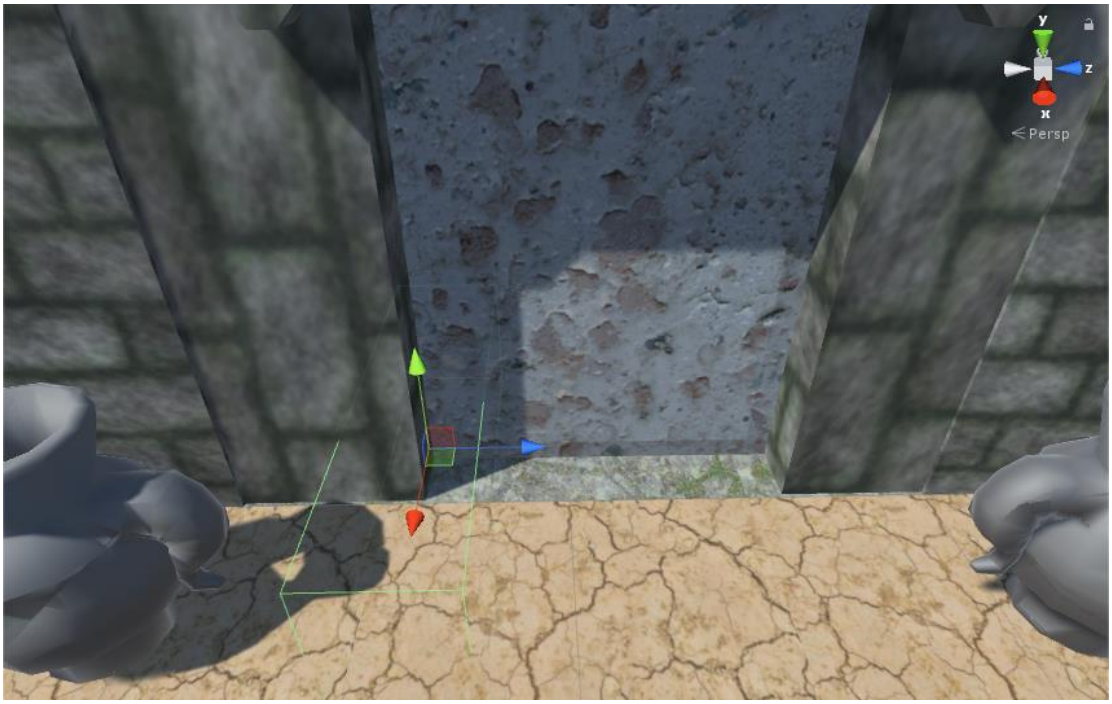
Funkce pro pohyb *PlayerMove()* využívá trigonometrii pro výpočet natočení hráče, následně spustí animaci pohybu a pohne postavou v daném úhlu o tolik jednotek, kolik je nastavena rychlost. Rozdílné animace se spouští pomocí proměnné *animationSpeedPercent*, která je předána animátoru, kde je použit *Blend Tree*, který na základě hodnot od 0 do 1 spouští animace pro *stání na místě (IdleHero)*, *chůzi (Move)* či *Běh (Run)*. (Lague, 2016)



Obrázek 42: Blend Tree (Zdroj: vlastní)

4.3.2 OVLÁDÁNÍ DVEŘÍ

Postava je nyní schopna se pohybovat ve scéně. Pro demonstraci byly zvoleny průchody trojími dveřmi, každý s jinou podmínkou pro její zdolání a následné otevření dveří. Byla vytvořena dvojice skriptů a přidány funkce do skriptu hráče.

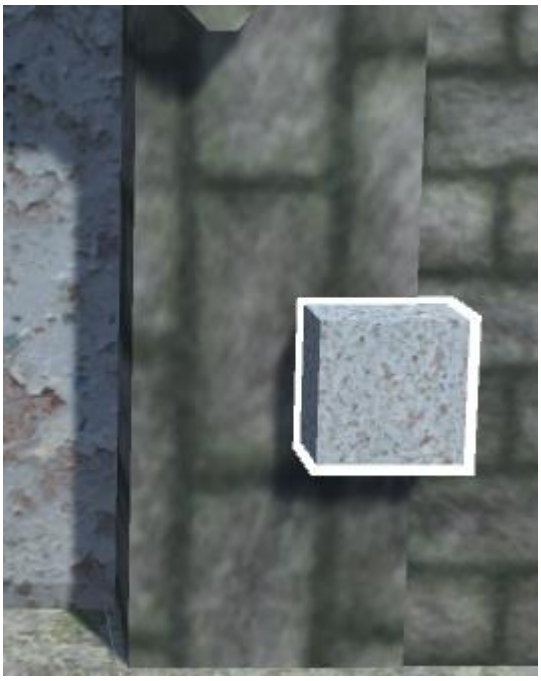


Obrázek 43: Collider pro vstup do dveří (Zdroj: vlastní)

Pro překonání první překážky musí hráč vstoupit do zelené oblasti, která zobrazuje hranice *Collideru*, po dobu, co se v ní nachází, zůstanou dveře otevřené a *Collider* dveří je vypnutý. Pokud se hráč nachází mimo oblast, dveře jsou zavřené a jejich *Collider* je aktivní, což znemožní průchod dveřmi.

V práci s kolizními objekty se využívá Unity metody *OnTriggetEnter()* a *OnTriggerExit()* které se volají, zda jiný *Collider* vstoupí do zóny *Collideru,* který obsahuje skript s těmito metodami.

V druhé místnosti má za úkol postava stisknout tlačítko, které otevře dveře a vypne *Collider*. Skript funguje na podobném principu jako v předchozím případě, pro přehlednost byl ale vytvořen samostatně. Kontrola zde probíhá stejnou metodou, ale po stisknutí tlačítka zůstávají dveře otevřené do té doby, než se tlačítko opět stiskne.



Obrázek 44: Tlačítko pro vstup (Zdroj: vlastní)

V další místnosti se nachází truhla, která obsahuje zlatou minci. Truhlu je nejdříve nutné otevřít, minci sebrat a při přiblížení k posledním dvěřím se automaticky mince vloží na panel vedle dveří, které se otevřou, pokud hráč opravdu minci sebral.

Obrázek 45: Truhla (Zdroj: vlastní)

4.3.3 UŽIVATELSKÉ ROZHRANÍ (USER INTERAFACE)

Aby bylo možné zobrazit, jaký počet váz musí hráč rozbít, je nutné vytvořit ve scéně plátno, které se nachází v Unity pod záložkou *uživatelské prostředí* (*UI)*. Pro zpřístupnění kvalitnějšího nastavení textu, pro úpravu stínování, jiných fontů byl stáhnut z *Asset Store*

balíček *TextMeshPro*, který byl vyvinut společností Unity Technologies, a je kompletně zdarma. Bohužel ale neobsahuje diakritiku, proto všechny texty ve hře jsou psané bez diakritiky. Po vložení plátna do scény, mu byl přidán text pro indikaci počtu rozbitých váz.

Obrázek 46: Plátno (Zdroj: vlastní)

V této chvíli má hráč pouze vyobrazen text na obrazovce, ale nedochází ke změně počtu váz, které musí být zničeny. K tomu je zapotřebí upravit skript hráče.

4.3.4 ROZBÍJENÍ VÁZ

Na špičce hráčova meče se nachází *Collider*, který slouží k desktrukci váz ve scéně. V ovládacím skriptu hráče byla vytvořena metoda *OnTriggerEnter()*, která kontroluje, zda *Collider* do kterého meč narazil, patří váze, pokud ano, spustí se animace rozbití vázy. Také musela být vytvořena funkce pro útok, která při stisknutí příslušného tlačítka zahájí animaci útoku.

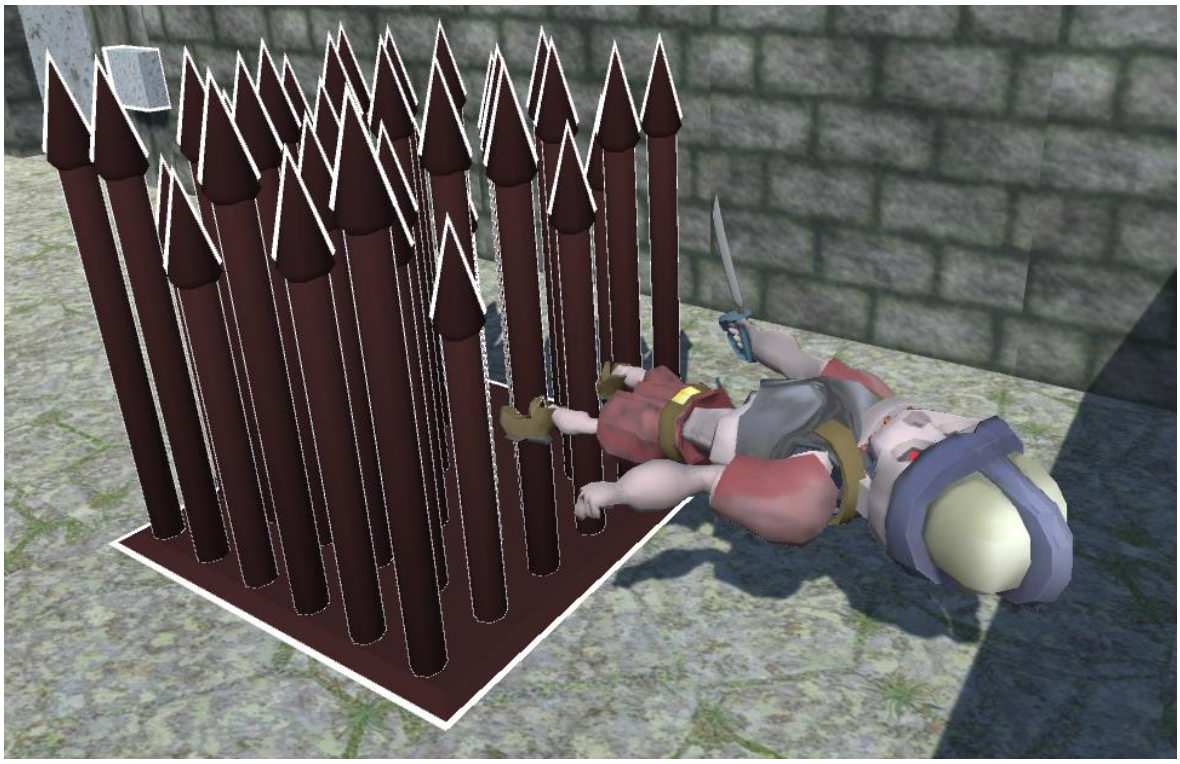
Zároveň byla vytvořena funkce *UpdatePotToDestroy()*, která mění text na plátně, podle počtu rozbitých váz. Hráč má za úkol zničit 5 kusů pokaždé, když je nějaká zničena, je volána tato metoda, která upraví text k zobrazení.

Pokud se stav počtu váz k rozbití nachází na 0, je splněna jedna ze dvou podmínek pro skončení hry hráčovým vítězstvím.

4.3.5 PAST

V každé videohře postava může zemřít, zde je tato interakce předvedena na pasti, na kterou může hráč vstoupit a poté se spustí animace vyjíždějících hrotů ze země, které hráče usmrtí.

Poté, co je spuštěna hráčova animace smrti, je skriptem vypnuto ovládání postavy. Po dobu 3 sekund je možné prohlížet okolí a hráče a poté se přepne hra na obrazovku smrti.



Obrázek 47: Smrt hráče (Zdroj: vlastní)

4.3.6 NEPŘÍTEL

Po překonání všech překážek čeká na hráče nepřítel, který neobsahuje ale žádnou umělou inteligenci. Poté, co ho hráč zasáhne, je spuštěna animace zásahu a obdrží zranění. Implicitně má nastavené 2 životy. Poté, co obdrží 2 zranění, je spuštěna animace smrti a hráč splní druhou podmínku pro vítězství. Pokud hráč v této chvíli splnil obě podmínky, hra se po 3 sekundách přepne na výherní obrazovku.

4.4 VYTVOŘENÍ ZBYLÝCH SCÉN

V této chvíli je hra skoro hotova, hráč je schopen se pohybovat, interagovat s prostředím a má určené podmínky pro výhru či prohru. Je ale nutné ještě vytvořit scény pro menu, ovládání, smrt a vítězství. O přepínání mezi jednotlivými scénami se stará skript, který je obsažen v každé scéně a kontroluje, zda byly splněny podmínky pro přepnutí na danou obrazovku. Tlačítka vytvořené ve scéně na plátně využívají funkci *onClick()*, která se volá při stisknutí tlačítka a spouští definovanou funkci ve skriptu.

4.4.1 MENU

V Unity si vytvoříme novou scénu menu, do které vložíme objekt plátna, na kterém jsou vytvořeny tlačítka a text. Pro text je opět použit *Asset* balíček *TextMeshPro*, aby bylo písmo ve hře unifikované.

V menu se nachází jméno práce a autora a tři tlačítka, spustit, ovládání a konec, které se po najetí myší zvýrazní. Po stisknutí daného tlačítka se hra spustí či se zobrazí ovládání nebo se hra ukončí.

Obrázek 48: Menu (Zdroj: vlastní)

4.4.2 OVLÁDÁNÍ, SMRT, VÍTĚZSTVÍ

Jelikož je zřejmé, že se na této obrazovce bude opět pracovat s textem, je možné pomocí tlačítek **Ctrl + D** duplikovat scénu. Poté je scéna duplikována, se všemi objekty a jejich komponenty. Stačí tedy poupravit text, odebrat tlačítka a nechat pouze tlačítko pro návrat do menu. Obdobně jsou vytvořeny i scény pro vítězství a smrt, s jejich originálními texty a návratovým tlačítkem do menu.

Obrázek 49: Ovládání, smrt, vítězství (Zdroj: vlastní)

4.5 TESTOVÁNÍ A BUILD

Poslední fází, která je nutná před publikováním hry udělat, je testování. Kromě testování vývojářem je vhodné hru nechat otestovat i jiné lidi, kteří mohou přijít na nějaké nedodělky či chyby ve hře. Testování by mělo probíhat za účelem rozbití hry, pokud se to nestane, hra je hotová a je možné jí publikovat. (Thorn, 2014, str. 20)

Zásady pro testování (Thorn, 2014, str. 20):

- Testovat průběžně
- Netestovat sám
- Používat různé verze hry
- Sledovat chyby

Když je hra otestována, je nutné, aby byla schopná fungovat jako samostatná aplikace na jiných zařízení, než byla vyvíjena. V unity je nutné tedy vytvořit build. Pod záložkou *File* je zvolena možnost *Build Settings*, zde se umístí všechny vytvořené scény, zvolí se cílová platforma. (Thorn, 2014, str. 21) V mém případě se jednalo o Windows a architektury x86 pro 32bitové procesory a x86-64 pro 64bitové.

ZÁVĚR

Hlavním cílem této práce bylo představit, jaké jsou možnosti při výběru software pro modelovaní a herního jádra. Následně prakticky vytvořit modely, které byly použity pro vývoj videohry a jak takový vývoj/modelování probíhá.

Pro zpracování této práce byly používány nejvíce zdroje na internetu a dokumentace jednotlivých software. Bohužel všechny dostupné zdroje jsou převážně v anglickém jazyce, český překlad či zdroj hledat je marné. Je proto nutná znalost anglického jazyka.

V teoretické části bylo vysvětleno v adekvátních kapitolách, co je 3D modelování a herní jádro. Poté byly představeny největší a nejpopulárnější software a jejich srovnání, aby bylo možné vybrat, ve kterém se následně bude modelovat a vyvíjet interaktivní animace.

Pokud se jedná o 3D modelovací software, tak z informací, které byly získány z dostupných zdrojů, byl vyvozen závěr, že jedinou reálnou možností pro konečný výběr se stane Blender. Ostatní software jsou výborně zpracované, ale jejich platební model je zaměřen spíše na modelářská studia, pro běžného uživatele jsou finančně nedostupné.

Naopak v oblasti herních jader je tato situace opačná. Všechna dostupná jádra jsou zdarma, je zde pouze nutnost platit poplatek z prodeje, ale pokud se nejedná o herní studio, je prakticky nemožné dosáhnout takové částky, aby bylo nutné platit poplatky. Jde tedy spíše o osobní preference. Po zkoumání a vyzkoušení daných software bylo zjištěno, že nejvíce vyhovujícím herním jádrem je Unity, kde probíhá skriptování v jazyce C#, v kterém jsem měl již předešlé zkušenosti.

V praktické části byla vytvořena videohra, na které je možné ukázat, jak funguje interakce mezi jednotlivými modely. Při práci na modelech bylo zjištěno, že je reálné strávit na jednom modelu desítky či stovky hodin práce, protože je žádoucí, aby model byl kvalitní, detailní a odpovídal představám autora. Zároveň bylo využito v Blenderu minimálně nástrojů pro práci s modely. Blender jich obsahuje nezměrné množství a aby bylo možné prozkoumat všechny možnosti, které nabízí, vyžadovalo by to stovky hodin další práce.

Co se týče Unity, to také nabízí mnoho rozdílných řešení jednotlivých úloh, které jsou více efektivní než ty, které byly použity při zpracování této práce. K tomu je zapotřebí ale získat zkušenosti. Jelikož se o funkci hry starají skripty a samotné kódování zabere mnoho času, pokaždé je možnost, jak napsat kód lépe a přehledněji. Toto zjištění pomáhá

k porozumění o optimalizaci a jak je velice důležitá pro běh hry. V mém malém projektu je optimalizace jen slovo a ve hrách, které vyvíjejí velké společnosti a jsou natolik komplexní, už opravdu záleží na tom, jak je strukturovaný kód a jestli jsou použity správné techniky pro zlepšení chodu hry.

RESUMÉ

The Main Goal of this Thesis was to produce Interactive 3D Animation. There is introduction to 3D modelling and different types of Software to make the actual Models, Blender was chosen to make the Models. Lastly there is an explanatory Guide how the Models and Animation were created. An explaination is given about what a Game Engine is what it does followed by an overview of different types. Amongst all of them, Unity was chosen for the development of the Game. A step-by-step Guide is shown on how to create the game with Blender Assets.

SEZNAM LITERATURY

Autodesk. (2018). *3ds max | 3D Modelling, Animation & Rendering Software | Autodesk*. Retrieved from www.autodesk.com: https://www.autodesk.com/products/3ds-max/overview

Autodesk. (2018). *Maya | Computer Animation & Modeling software | Autodesk*. Retrieved from www.autodesk.com: https://www.autodesk.com/products/maya/overview

Blender. (2018). *About - blender.org*. Retrieved from www.blender.org: https://www.blender.org/about/

Blender. (2018). *Features - blender.org*. Retrieved from www.blender.org: https://www.blender.org/features/

Crytek. (2018). *CRYENGINE Help & FAQs*. Retrieved from www.cryengine.com: https://www.cryengine.com/faq

Daniel, Logical Increments. (2018, Březen). *Building the Best for 3D Rendering and Animation.* Retrieved from www.logicalincrements.com: http://www.logicalincrements.com/articles/building-pc-3d-rendering-animation

Devslopes by Mark Price. (2017, Leden). *3D Game Modeling & Animation With Blender | Udemy.* Retrieved from www.udemy.com: https://www.udemy.com/blender3d/learn/v4/content

Epic Games. (2018). *Unreal Engine Frequently Asked Questions*. Retrieved from www.unrealengine.com: https://www.unrealengine.com/en-US/faq

Flavell, L. (2010). Beginning Blender: Open Source 3D Modelling, Animation, and Game Design. V L. Flavell. New York: Distributed to the book trade worldwide by Springer Science+Business Media, c2010. Expert's voice in open source.

Florian. (2017, Červen 2). *The Best Game Engines for Begginers*. Retrieved from www.websitetooltester.com: https://www.websitetooltester.com/en/blog/best-game-engine/

Hughes, K. (2017, Říjen). *Where to find free textures for 3D projects | Creative Bloq.* Retrieved from www.creativebloq.com: https://www.creativebloq.com/3d-tips/find-high-res-textures-1232646

Lague, S. (2016, Říjen 30). *Character Creation (E07: Unity character controller).* Retrieved from www.youtube.com: https://www.youtube.com/watch?v=ZwD1UHNCzOc

Lile, D. (2015). *Blender character rigging 7 of 10 - Youtube.* Retrieved from www.youtube.com: https://www.youtube.com/watch?v=VVHKBSknFhA&index=28&list=PLyelx0TsmSpf-8xZfjZveokSsIwj0F3lo

Linietsky, J., & Manzur, A. (2018). *Godot Engine - Features*. Retrieved from godotengine.org: https://godotengine.org/features

Linietsky, J., & Manzur, A. (2018). *Scenes and Nodes - Godot Engine latest documentation*. Retrieved from docs.godotengine.org: http://docs.godotengine.org/en/3.0/getting_started/step_by_step/scenes_and_nodes.html

Martin, J. (2017, Leden 15). *Topology Guides.* Retrieved from http://topologyguides.com/: http://topologyguides.com/

Paulo, J. (2018). *3D TEXTURES |Free CC0 textures with Diffuse, Normal, Displacement, Occlusion, Specularity and Roughness Maps.* Retrieved from 3dtextures.me: https://3dtextures.me/

Shannon, T. (2018). Unreal Engine 4 for design visualization: developing stunning interactive visualizations, animations, and renderings. Boston: Addistion-Wesley.

Slick, J. (4. Září 2017). *5 Common Pitfalls of Beginning Modelers.* Načteno z www.lifewire.com: https://www.lifewire.com/common-pitfalls-of-beginning-modelers-2052

Slick, J. (2017, Říjen 17). *Seven common 3D modelling Techniques for Film and Games*. Retrieved from www.lifewire.com: https://www.lifewire.com/common-modeling-techniques-for-film-1953

Slick, J. (2018, Březen 25). *The Definiton of 3D Modeling.* Retrieved from www.lifewire.com: https://www.lifewire.com/what-is-3d-modeling-2164

Thorn, A. (2014). Practical game development with unity and blender. Boston: Cenpage Learning.

Totten, C. (2012). Game character creation with Blender and Unity. Hoboken, N.J.: Wiley.

Unity Technologies. (2018). *Unity - Fast Facts*. Načteno z www.unity3d.com: https://unity3d.com/public-relations

Unity technologies. (2018). *Unity - Manual: Splash Screen*. Retrieved from docs.unity3d.com: https://docs.unity3d.com/Manual/class-PlayerSettingsSplashScreen.html

Unity Technologies. (2018). *Unity - Products*. Retrieved from www.unity3d.com: https://unity3d.com/unity

Villar, O. (2017). Learning Blender: A Hands-On Guide to Creating 3D Animated Characters. 2nd edition. Boston: Addison-Wesley.

ZBrush. (2018). *Pixologic > Zbrush 2018*. Retrieved from store.pixologic.com: https://store.pixologic.com/zbrush-2018/

ZBrush. (2018). *Polymesh | ZBrush Docs*. Retrieved from docs.pixologic.com: http://docs.pixologic.com/reference-guide/tool/polymesh/

Zbrush. (2018). *ZBrush 2018 Features*. Retrieved from www.pixologic.com: http://pixologic.com/features/index.php#getZBrush

SEZNAM OBRÁZKŮ, TABULEK, GRAFŮ A DIAGRAMŮ

SEZNAM PŘÍLOH

Příloha I – doplňující obrazový materiál

Příloha II – ukázky skriptů v jazyce C#

Příloha III – digitální datové optické médium s elektronickou verzí práce, vytvořené modely v Blenderu a Unity soubory včetně skriptů

Příloha I

- Elephants Dream
- Zdroj:https://orange.blender.org/wpcontent/themes/orange/images/media/gallery/s5_both_t.jpg?x53801

- Big Buck Bunny
- Zdroj: https://peach.blender.org/wp-content/uploads/its-a trap.thumbnail.png?x11217

- Sintel
- Zdroj: https://durian.blender.org/wpcontent/uploads/2010/06/03.1h_comp_000004-200x95.jpg

Příloha II

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enter : MonoBehaviour {
    public GameObject obj;
    public GameObject objCol;
    Animator ani;
    Collider col;

    private void Start()
    {
        ani = obj.GetComponent<Animator>();
        col = objCol.GetComponent<BoxCollider>();

    }

    private void OnTriggerEnter(Collider other)

    {

        if (other.gameObject.tag == "Character")
        {

            ani.SetBool("open", true);
            col.enabled = false;
        }

    }

    private void OnTriggerExit(Collider other)
    {
        if (other.gameObject.tag == "Character")
        {
            ani.SetBool("open", false);
            col.enabled = true;
        }


    }
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Button : MonoBehaviour {

    public GameObject obj;
    public GameObject objCol;
    Animator ani;
    Collider col;
    private bool open;


    private void Start()
    {
        ani = obj.GetComponent<Animator>();
        col = objCol.GetComponent<BoxCollider>();
        open = false;

    }

    private void OnTriggerEnter(Collider other)

    {

        if (other.gameObject.tag == "uk1" && open == false)
        {

            ani.SetBool("open", true);
            open = true;
            col.enabled = false;
        }

        else if (other.gameObject.tag == "uk1" && open == true)
        {

            ani.SetBool("open", false);
            col.enabled = true;
            open = false;

        }
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Health : MonoBehaviour {

    public int maxHealth;
    public int currentHealth;
    public Animator ani;
    public bool enemyDeath = false;


    // Use this for initialization
    void Start () {
        currentHealth = maxHealth;
    }

    // Update is called once per frame
    void Update () {
        if (currentHealth == 0)
        {
            ani.SetBool("death", true);
            enemyDeath = true;
        }
    }

    public void Hurt(int damage)
    {
        currentHealth -= damage;

    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HurtEnemy : MonoBehaviour {

    public int damage;
    public Animator ani;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }

    private void OnTriggerEnter(Collider other)
    {

        if (other.gameObject.tag == "Enemy")
        {
            other.gameObject.GetComponent<Health>().Hurt(damage);
            ani.SetBool("hit", true);

        }
    }

    private void OnTriggerExit(Collider other)
    {
        ani.SetBool("hit", false);
    }
}
```

```
private void PlayerMove()
{
    Vector2 input = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical")); // Unity ovládání wsad
    Vector2 inputDirection = input.normalized; // vrací 1tkový input, pro výpočet úhlu otočení pomocí trigonometrie

    if (inputDirection != Vector2.zero) // podmínka pro ověření pohybu hráče
    {
        float targetRotation = Mathf.Atan2(inputDirection.x, inputDirection.y) * Mathf.Rad2Deg + cameraT.eulerAngles.y; // výpočet úhlu otočení
        transform.eulerAngles = Vector3.up * Mathf.SmoothDampAngle(transform.eulerAngles.y, targetRotation, ref turnSmoothVelocity, turnSmoothTime); // otočení hráče
    }

    bool running = Input.GetKey(KeyCode.LeftShift); // běh
    float targetSpeed = ((running) ? runSpeed : walkSpeed) * inputDirection.magnitude; // přepíná rychlost mezi během a chůzí
    currentSpeed = Mathf.SmoothDamp(currentSpeed, targetSpeed, ref speedSmoothVelocity, speedSmoothTime); // vyhlazení rychlosti

    transform.Translate(transform.forward * currentSpeed * Time.deltaTime, Space.World); // pohyb hráče vypočtenou rychlostí

    float animationSpeedPercent = ((running) ? 1 : .5f) * inputDirection.magnitude; // rychlost animace pro běh a chůzi
    animator.SetFloat("speedPercent", animationSpeedPercent, speedSmoothTime, Time.deltaTime); // spuštění animace
}
```