Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Prezentace firmy pomocí počítačové 3D grafiky

Bc. Dalibor Pandula

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Dalibor Pandula

Systémové inženýrství a informatika
Informatika

Název práce

Prezentace firmy pomocí počítačové 3D grafiky

Název anglicky

Company presentation using 3D computer graphics

Cíle práce

Cílem diplomové práce je vytvoření 3D animace. Animace zobrazí podstavec na kterém bude umístěn škorpion. V první části videa se bude podstavec otáčet a modelové snímací zařízení provede snímaní (scan) zvířete. V druhé části laserový scaner přejede lineárně po objektu. V průběhu snímání bude v animaci zobrazen vznikající holografický obraz snímaného objektu. Video by mělo fungovat jako reklama pro firmu.

Metodika

Metodika DP bude založena na analýze odborných a vědeckých dokumentů (zejména monografií) a následně budou získané poznatky syntetický využity k návrhové části. Postupujte dle následujících bodů:

- Prostudujte literární zdroje se zaměřením na téma 3D grafika.
- Vytvořte 3D modely včetně textur a případných animací, vytvořte celkovou kompozici scény.
- Využijte moderních softwarových řešení pro vytvoření krátkého videa s prezentací počítačové 3D grafiky.
- Definujte závěry

Doporučený rozsah práce

60-80

Klíčová slova

Počítačová grafika, 3D, Modelování, Holografický obraz, Reklamní video, Textury a animace

---

Doporučené zdroje informací

Soft Computing: Integrating Evolutionary, Neural, and Fuzzy Systems,Andrea Tettamanzi, Marco Tomassini,Springer Science & Business Media, 7. 9. 2001

The Complete Guide to Blender Graphics: Computer Modeling & Animation

---

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

Ing. Josef Pavlíček, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 27. 11. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 27. 11. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 29. 11. 2020

Čestné prohlášení

Prohlašuji, že svou diplomovou práci Prezentace firmy pomocí počítačové 3D grafiky jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30.11.2020 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Josefu Pavlíčkovi, Ph.D. za jeho ochotné vedení a podporu, která mi pomohla se zpracováním této diplomové práce.

Prezentace firmy pomocí počítačové 3D grafiky

Abstrakt

V literární rešerši diplomové práce jsou představeny základní principy počítačové 3D grafiky, zejména pak modelování, animace a různé přístupy k vykreslení obrazu. Dále je představen program Blender a jeho technologický přístup k tvorbě digitálního 3D obsahu.

Praktická část diplomové obsahuje analýzu rychlosti a kvality vykreslení snímků u dostupných rendererů v prostředí Blenderu. V návaznosti na provedenou analýzu je představen pracovní postup tvorby počítačové animace. Jsou představeny základní modelovací postupy v podobě polygonálního modelování, modelování Booleovskými operacemi anebo Sculpting. Dále jsou představeny procedurální textury a materiály a jejich současný význam pro tvorbu snadno a rychle pozměnitelných materiálů. Následuje ukázka přímé animace i nepřímé kosterní animace objektů. Závěrem praktické části diplomové práce je vykreslení třiceti sekundového videa, které prezentuje technologické pokroky v oblasti počítačové 3D grafiky.

Klíčová slova: Počítačová grafika, 3D, Modelování, Holografický obraz, Reklamní video, Textury, Procedurální textury, Animace, Kosterní animace, Realtime renderování

Company presentation using 3D computer graphics

Abstract

In literary research of this thesis are introduced basic principles of computer 3D graphics, such as modeling, animation and different approaches to image rendering. Furthermore, program Blender is introduced with its technological approach to creating digital 3D content.

Practical part of the diploma thesis contains an analysis of speed and quality of image rendering using available renderers in the Blender software. Following the analysis, the workflow of creating computer animation is presented. Basic modeling methods are introduced in the form of polygonal modeling, modeling by Boolean operations or Sculpting. Furthermore, procedural textures and materials and their current significance for the creation of easily and quickly modifiable materials are presented. That is followed by an example of direct animation and indirect skeletal animation of objects. The conclusion of the practical part of the thesis is the rendering of a thirty second video, which presents technological advances in computer 3D graphics.

Klíčová slova: Computer Graphics, 3D Modeling, Holographic image, Advertisment video, Textures, Procedural textures, Animations, Skeletal Animation, Realtime rendering

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Počítačová 3D grafika je rozsáhlý obor informatiky věnující se tvorbě a využití digitálního trojrozměrného obsahu. V praxi se počítačová 3D grafika využívá ve tvorbě animací, vizuálních efektů, simulací anebo konstrukčních řešení. Počítačová 3D grafika vychází z principů vektorové 2D grafiky a využívá rastrové 2D grafiky pro vykreslení obrazu. Jedná se o dlouhodobě rozvíjející se obor, který se bude i nadále rozvíjet, zejména díky příchodu virtuální reality a umělé inteligence.

V této diplomové práci je věnována pozornost zejména počítačovým animacím a pracovnímu postupu tvorby počítačové animace sloužící jako krátká reklama. Zadání vzniklo z důvodu existence nových technologických postupů, které mění současnou situaci ve vývoji počítačových animací i jiného počítačového 3D obsahu.

2. Cíl práce a metodika

2.1 Cíl práce

Cílem práce je vytvoření reklamního videa pro nespecifikovanou firmu pomocí počítačové 3D grafiky. Animace by měla zachytit škorpióna stojícího na podstavci, který je snímán dvěma různými zařízeními. Ze zadání lze tedy vyčíst požadavek na tvorbu modelů včetně textur, tvorbu celkové scény včetně osvětlení a nastavení kamer a animaci modelů.

Dalším cílem práce je představení technologického pokroku v oboru počítačové 3D grafiky, přinášející rozsáhlé změny ve tvorbě počítačových animací.

2.2 Metodika

Zadání práce je velmi otevřené a umožňuje výběr vlastní metodiky i technologie pro tvorbu animace.

Součástí práce je představení základů počítačové 3D grafiky, základních modelovacích postupů, tvorby materiálů, přímé i nepřímé animace a vykreslení obrazu. Důraz je kladen především na moderní technologické postupy, například tvorbu procedurálních textur a výhod plynoucích z jejich využití a také vykreslování obsahu v reálném čase.

Použitá technologie v praktické části diplomové práce je zdarma dostupná pro vývoj animací i jiného 3D obsahu. Prezentované postupy lze aplikovat i v jiných oborech, například ve vývoji her nebo při tvorbě statických scén pro vizualizaci.

3. Teoretická východiska

3.1 Počítačová grafika

Počítačová grafika je vědní obor zabývající se tvorbou grafického obsahu za pomocí výpočetní techniky. Tok informací zpravidla pochází z počítače a směřuje k uživateli prostřednictvím výstupních zařízení jako jsou monitory. Je tak důležité, aby uživatel byl schopen sám zpracovat přijaté vizuální informace. V některých případech může být tok informací opačný a uživatel předává počítači grafický vstup. Práce s počítačovou grafikou může zasahovat do dalších vědních oborů, například fyzika, matematika nebo grafický design. Fyzika se používá pro modelování světelných podmínek nebo simulací například gravitace. Matematika se používá pro vyjádření tvarů. Grafický design zvyšuje efektivitu předávání informací uživateli.[1, s. 1]

Počítačová grafika již nemusí být výpočetně náročná jako to bylo v minulosti a je možné pracovat s počítačovou grafikou i na osobních počítačích. Dostupnost i kvalita programů pro práci s počítačovou grafikou se také zvýšila. Počítačovou grafiku lze využít v mnoha oborech a níže je k vidění jejich základní výčet:[7]

- Tiskoviny zahrnující časopisy i noviny běžně obsahují kromě textů také velké množství grafiky.
- Reklama se snaží předat co možná nejvíce informací za co možná nejkratší čas, proto obsahuje velké množství grafiky.
- Multimédia zahrnující například filmy a televizi běžně obsahují grafické prvky. Animované filmy se v současnosti vytváří výhradně pomocí počítačové grafiky.
- Internetové stránky běžně obsahují velké množství grafického obsahu. Tvorba grafiky pro internetové stránky se řídí trochu jinými pravidly než u jiných oborů. Důraz je zde kladen na jednoduchost a jasnost.
- 3D Modelování umožňuje vytvářet trojrozměrné scény, které lze uplatnit například při vizualizaci produktů před jejich výrobou. 3D Modelování je obecně výpočetně náročnější, než jiné obory využívající počítačovou grafiku.
- CAD a CAM projektování se soustřeďuje na tvorbu přesných technických výkresů produktů a jejich výrobu.

- Videohry se vytvářejí téměř vždy pomocí počítačové grafiky. Zábavní průmysl často tlačí na technologické pokroky počítačové grafiky, zejména pak na vykreslování obsahu v reálném čase.

Počítačovou grafiku lze rozdělit do dvou kategorií. Jedná se o počítačovou 2D grafiku a počítačovou 3D grafiku. Počítačová 2D grafika je omezena na dvojrozměrné objekty a dále se dělí na rastrovou a vektorovou grafiku. Počítačová 3D grafika pracuje s dvojrozměrnými i trojrozměrnými objekty. Počítačová 3D grafika je výpočetně náročnější a je schopna vytvářet složitější grafický obsah.[7]

3.2 Rastrová grafika

Rastrová grafika vychází z hardwarové implementace klasických počítačových monitorů. Počítačové monitory využívají luminiscenčních prvků pro zobrazení svého obsahu. Luminiscenční prvky jsou seskupeny do čtvercové mřížky pixelů, jejichž množství a velikost udává rozlišení a fyzickou velikost monitoru.[2, s. 51]

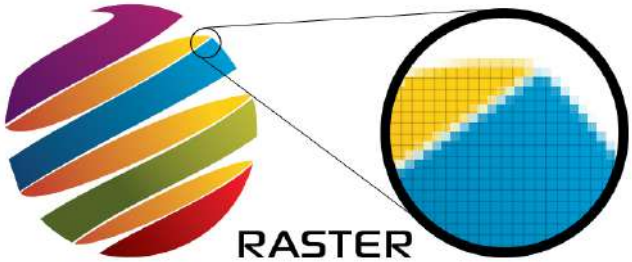
Rastrová grafika ukládá data jako jednotlivé pixely, kde každý pixel má svoji vlastní barvu. Sestavením všech pixelů vznikne požadovaný obrázek. Paměťová náročnost rastrového obrázku vychází zejména z jeho rozlišení. Úprava rastrových obrázků je možná prostřednictvím různých grafických programů, které v současnosti obsahují spoustu funkcí usnadňující práci s rastrovými obrázky.[7]

Jelikož má každý pixel svoji vlastní barvu, tak je možné vytvářet obrázky s vysokou mírou detailů. Rastrová grafika je ideální pro ukládání fotografií, naskenovaných dokumentů nebo digitálních obrazů.[7]

Největší nevýhodou rastrové grafiky je problematická změna rozlišení obrázku. Při zvětšení obrázku dochází k zvýraznění jednotlivých pixelů, tj. je viditelný takzvaný rastr. Naopak při zmenšení obrázku se ztrácí detaily.[8]

3.2.1 Formáty rastrové grafiky

Pro uložení obrázků počítačové 2D grafiky se používá mnoho různých formátů. V následující kapitole jsou popsány základní formáty pro rastrovou grafiku.



Obrázek 3.1: Ukázka vektorové grafiky[33]

Formát GIF (Graphics Interchange Format) je velmi jednoduchý datový formát využívající bezeztrátovou kompresní metodu LZW. Formát GIF podporuje animace, ačkoliv snímky jsou ukládány samostatně bez ohledu na jejich množství a jak moc se od sebe liší. Další nevýhodou je omezení na 256 barev z důvodu 8mi bitového formátu barevné hloubky. Jedná se tak o paměťově nenáročný datový formát, který se v současnosti hojně používá pro přenos webové grafiky jako jsou jednoduché obrázky nebo animace.[3, s. 71]

Formát PNG byl vyvinut hlavně pro přenos obrázků přes internet jako nástupce formátu GIF. Formát PNG podporuje dvourozměrné prokládání obrazu, což je metoda rozdělení obrázku na několik skupin čtverců a obdélníků. Pixely jsou při načítání obrázku rovnoměrně rozloženy mezi skupiny. Díky tomu je možné rozpoznat obsah obrázku dříve, než se obrázek celý načte. Dalšími vlastnostmi formátu PNG je podpora barevného modelu RGBA, velmi účinná bezeztrátová komprese anebo možnost navýšit počet bitů na kanál (až 16 bitů na kanál). Jedinou nevýhodou formátu PNG je absence podpory animací. Jedná se tak o velmi efektivní a moderní datový formát, který nahrazuje starší formát GIF.[3, s. 72]

Vedle formátu PNG se také často používá formát JPEG. Formát JPEG využívá ztrátovou kompresi, díky čemuž je výsledná velikost souboru podstatně nižší, než u formátu PNG. Formát JPEG také podporuje 24 bitovou barevnou hloubku. Tento formát byl vytvořen zejména pro účely digitálních fotografií, naskenovaných dokumentů a rentgenových snímků, kde není vidět efekt ztrátové komprese. Obrázky, které mají málo barev nebo mají ostré hrany nejsou vhodné pro formát JPEG. Díky své nízké velikosti je formát JPEG vhodný pro přenos obrázků přes internet.[9]

Pro tvorbu počítačových animací se kromě formátu GIF také používají formáty MPEG. Formátů MPEG je více a postupně se technologicky vyvíjely. Nejpoužívanější verzí je v současnosti MPEG-4. Pro MPEG-4 existuje mnoho kodeků, pomocí kterých je obraz

zakódován a je na něj aplikována komprese. Formát MPEG-4 definuje běžné snímky a rozdílové snímky. V případě že dekódovací zařízení (například televize) není schopné dekódovat včas všechny snímky, tak jsou rozdílové snímky přeskočeny a místo toho se načtou běžné snímky, které mají nižší míru komprese. Kromě obrazu je formát MPEG-4 také schopen zahrnout zvuk, statické i dynamické texty nebo popis 3D objektů ve scéně.[3, s. 76]

Obrázek 3.2: Ukázka vektorové grafiky[33]

3.3 Vektorová grafika

Vektorová grafika je ukládána v podobě matematického zápisu, který definuje tvary, barvy i polohu všech křivek a jiných objektů na obrázku. Obrázek se může skládat z velkého množství těchto objektů, které se mohou navzájem překrývat a doplňovat. Mezi výhody matematického zápisu patří nezávislost na rozlišení obrázku, snadná modifikovatelnost obrázku a nízká paměťová náročnost. Při změně rozlišení nedojde ke snížení kvality obrazu jako je tomu u rastrové grafiky a obraz zůstane vždy ostrý. Zároveň lze kdykoliv změnit parametry objektů, například jejich pozici nebo barvu. Další výhodou matematického zápisu je vysoká přesnost, proto lze vytvářet přesné technické výkresy.[7]

Nevýhodou vektorové grafiky je její jednoduchost. Objekty vektorové grafiky mají většinou ploché barvy a ostré hrany. Je tak těžké vyjádřit detaily na obrázcích. Vektorová grafika tak není vhodná pro uchování obrázků s vysokou mírou detailu, jako jsou například fotografie. Existuje také určitý horní limit pro množství a složitost objektů ve vektorovém obrázku, po jehož překročení vzrůstá výpočetní i paměťová náročnost velmi rychle a přesahuje náročnost rastrového obrázku.[10]

Vektorová grafika se používá zejména při tvorbě diagramů, reklamní grafiky, počítačových animací anebo technických výkresů.[10]

3.3.1 Formáty vektorové grafiky

Pro uložení vektorové grafiky existuje hodně formátů, ale nejčastěji se používají formáty AI nebo SVG. Formát AI je proprietární datový formát patřící společnosti Adobe. Pravděpodobně se jedná o nejkvalitnější formát pro vektorovou grafiku. Problémem tohoto formátu je jeho omezená podpora mimo programy od společnosti Adobe.[7]

Pro webovou grafiku se vektorová grafika často jeví jako lepší řešení, než je rastrová grafika. Důvody jsou nižší paměťová náročnost vektorové grafiky a zároveň možnost snadno měnit velikost vektorové grafiky beze ztráty kvality obrázků. Formát SVG byl navržen právě pro webovou grafiku, nicméně jeho užití se rozšířilo a formát se stal standardem pro uchování vektorové grafiky. Kromě základních funkcí jako je podpora animací nebo průhlednosti prvků, formát SVG také podporuje tvorbu interaktivních prvků. Lze tak vytvořit interaktivní webové aplikace běžící na technologii SVG. Formát SVG má strukturu XML dokumentu, tudíž je snadno čitelný a je možné jej snadno upravovat i pomocí textových editorů nebo pomocí JavaScriptu.[11]

3.4 Počítačová 3D grafika

Počítačová 3D grafika vychází z principů vektorové 2D grafiky. Jedná se v podstatě o nástavbu nad vektorovou grafikou, která je omezena na dvourozměrný prostor s osami x a y. Počítačová 3D grafika pracuje ve trojrozměrném prostoru a přidává tak třetí osu z.[12]

Prostředí, ve kterém uživatel pracuje se obvykle označuje jako scéna. Scéna se skládá z kolekce objektů $S = \{O_1, O_2, \ldots, O_i\}$. Každý objekt je definovaný jako kolekce polygonů (polygon) $P\{p_1, p_2, \ldots, p_j\}$, stran (face) $F\{f_1, f_2, \ldots, f_k\}$, hran (edge) $E = \{e_1, e_2, \ldots, e_l\}$ a vrcholů (vertex) $V = \{v_1, v_2, \ldots, v_m\}$. Každý vrchol v_n je definován pomocí tří souřadnic v prostoru $v_n = \{x_n, y_n, z_n\}$, každá hrana e_o je definována dvěma koncovými vrcholy $e_o = \{v_p, v_q\}$ a každá strana f_r je definována kolekcí svých hran $f_r = \{e\}$ nebo vrcholů $f_r = \{v\}$. Každý polygon p_s lze definovat jako kolekce jeho strany a všech jeho hran a vrcholů $p_s = \{f_s, e, v\}$.[4][13]

Další vlastností polygonů je jejich jednostrannost. Jedna strana je vždy určena jako přední strana a je viditelná zatímco druhá strana je zadní stranou. V závislosti na používané technologii může být zadní strana vykreslována nebo nemusí. To, zda-li se jedná o přední nebo zadní stranu určuje normála plochy daného polygonu. Je také vhodné zmínit, že každý vrchol má také svoji normálu, která určuje kam vrchol směřuje. Normály polygonů i vrcholů jsou běžně využívány při vývoji 3D obsahu.[12]

Polygony lze poskládat vedle sebe tak, aby spolu sdílely hrany. Takováto skupina polygonů se nazývá síť a jedná se o základ jakéhokoliv 3D objektu. Všechny 3D objekty jsou tak mnohostěny. Trojúhelníkové polygony, mající tři hrany a tři vrcholy, jsou nejjednodušší možné polygony, zejména proto, že všechny tři vrcholy takového polygonu budou vždy ležet na stejné ploše. Polygony s více stranami se zásadně nepoužívají pro vykreslování snímků, nicméně jsou běžně používány při modelování 3D objektů. Pokud se povrch objektu neskládá z trojúhelníků před jejich vykreslením tak je nutné povrch rozdělit na více polygonů. Například povrch krychle se obvykle skládá ze šesti čtverců. Každý z čtverců lze rozříznout dle jedné z diagonál (obvykle kratší diagonála), čímž vzniknou dva trojúhelníkové polygony pro každou stěnu krychle. Tento postup zjednodušení polygonů je zpravidla prováděn strojově na začátku procesu vykreslování, bez vědomí uživatele.[1, s. 187][14]

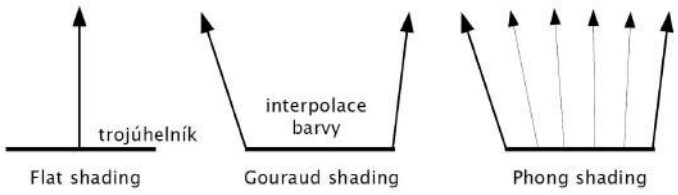
Z pohledu uživatele je práce s trojúhelníky poněkud složitá ale ne nemožná. Obvykle je lepší pracovat převážně se čtyřúhelníky, neboť tak lze vytvářet smyčky polygonů. Smyčky mají směr (dopředu a dozadu) a zároveň lze na smyčku jako na celek aplikovat různé funkce, například ji rozříznout napůl. Čtyřúhelníky se také lépe protahují nebo naopak stlačují, což je užitečná vlastnost pro animaci.[40][14]

Pokud se objekt skládá z trojúhelníků, tak lze běžně zpozorovat grafické chyby zvané artefakty na povrchu objektu. Artefakty se nezobrazí v případě, že je model vytvářen ze čtyřúhelníků a to i přesto, že jsou čtyřúhelníky zjednodušovány na trojúhelníky před jejich vykreslením. V praxi je ale skoro nemožné vyhnout se přítomnosti trojúhelníků. Proto jsou trojúhelníky cíleně vytvářeny na místech, kde jsou objekty plně ploché, nebo jsou špatně viditelné (například v dutinách). Polygony, které mají pět hran nebo více se obvykle nazývají jako n-gony. Tyto polygony vyvolávají stejné artefakty jako trojúhelníky, nicméně jejich artefakty jsou obvykle podstatně horší. V praxi je tak důležité se n-gonům plně vyhnout.[14]

Jelikož jsou trojúhelníky ploché, tak je vykreslení hladkých a oblých objektů problematické. Platí, že čím více polygonů utváří povrch objektu, tím přesnější je výsledný

povrch objektu. To se negativně odráží na množství výpočetního výkonu potřebného pro vykreslení scény.[12]

Pro vykreslení objektů se využívají různé metody stínování. Konstantní stínování zkoumá barvu každého polygonu zvlášť a aplikuje ji na celý polygon za využití normály plochy polygonu. Lze tak jasně vidět hrany rozdělující jednotlivé polygony. Existuje také dokonalejší Gouraudovo stínování a Phongovo stínování. Gouraudovo stínování vykresluje nejprve barvu vrcholů polygonu pomocí normál vrcholů a až poté dopočítá barvu samotného polygonu pomocí lineární interpolace mezi vrcholy. Phongovo stínování vypočítává povrchové normály polygonů pro každý pixel výsledného rastrového obrázku zvlášť. Gouraudovo i Phongovo stínování produkuje hladké hrany polygonů a lze tak docílit hladkého povrchu objektů i s nízkým množstvím polygonů. Nicméně rohy polygonů můžou být stále viditelné na siluetě objektu.[12]



Obrázek 3.3: Princip metod stínování[12]

3.4.1 Modelování

Modelování je zpravidla prvním krokem při vývoji 3D obsahu. V následující kapitole jsou popsány základní metody modelování.

Modelování často využívá objektová primitiva jako výchozí objekt pro modelování. Objektové primitivum je již hotový 3D objekt s optimalizovanou sítí polygonů, který je vytvarován do podoby nějakého velmi jednoduchého tvaru. Jedná se například o krychli, válec, jehlan, plochu nebo kouli. Objektová primitiva je možné snadno upravit, například změnit velikost krychle nebo určit množství vrcholů, které tvoří podstavu jehlanu.[41]

3.4.1.1 Krabicové a polygonální modelování

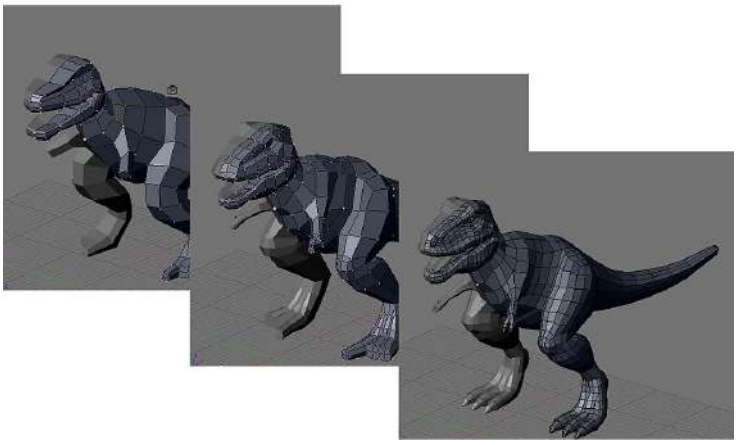
Krabicové modelování a polygonální modelování jsou dvě příbuzné metody modelování. Obojí vychází z myšlenky využití objektového primitiva a postupného přidávání polygonů pro vytvoření hrubého tvaru cílového předmětu. Detaily jsou objektu přidávány postupně až poté, co je hrubý tvar vytvořen. Krabicové a polygonální modelování přímo zasahují do sítě objektu. Ne každá modelovací metoda má tuto vlastnost.[15]

Krabicové modelování využívá několik objektových primitiv krychle, válce nebo jiné. Běžným postupem je tvorba nového objektového primitiva pro každou oblast objektu. Celý proces modelování je rozdělen do několika fází. Ve fázi hrubého náčrtu se uživatel snaží vytvořit obecný tvar výsledného objektu. Cílem je vytvarovat objekt tak, aby jej šlo identifikovat podle siluety objektu. Druhou a třetí fází je přidávání detailů. Ve druhé fázi se jedná o větší detaily, které ovlivňují siluetu (například prsty na tlapě dinosaura) a ve třetí fázi se jedná o nejmenší detaily, které již neovlivňují siluetu (například oko dinosaura nebo ostré hrany na kůži). Klíčové je dávat stejnou míru pozornosti všem oblastem objektu a vytvářet tak detaily rovnoměrně. Čtvrtou fází je očištění objektu. V této fázi jsou hledány chyby v síti objektu (například přítomnost polygonů s pěti a více stranami) nebo jsou vyrovnány velikosti polygonů. Poslední fází je fáze retopologie, která je popsána v pozdější kapitole.[15]

Polygonální modelování nevyužívá objektová primitiva tak často, jako krabicové modelování. Polygonální modelování místo toho pracuje se samostatnými polygony, nejčastěji čtyřúhelníky. Nejvhodnější objektové primitivum pro tuto metodu je objekt plocha. S objekty se pracuje jako s celky, které nejsou rozdělovány na samostatné oblasti jako je tomu u krabicového modelování. Na rozdíl od krabicového modelování je polygonální modelování podstatně pomalejší, ale běžně produkuje přesnější výsledky. Fáze očištění a retopologie běžně nejsou tak podstatné, neboť síť modelu je již optimalizovaná.[15][16]

Většina 3D modelovacích programů má podporu krabicového nebo polygonálního modelování. Podporou se myslí řada funkcí, které pomáhají uživateli tvarovat objekt. Například funkce extrusion vytváří novou síť tím, že kopíruje a vytahuje existující vrcholy, hrany nebo polygony. Další funkcí může být funkce bevel, která se aplikuje na hrany nebo vrcholy a vytváří efekt zkosení nebo zaoblení hrany.[16]

Jedná se o metody doporučené začátečníkům, ale zároveň se jedná o metody běžně používané i profesionály. Zkušenost uživatele a jeho znalost tvorby čtyřúhelníků často určuje kvalitu výsledného objektu.[16]



Obrázek 3.4: Průběh krabicového modelování[34]

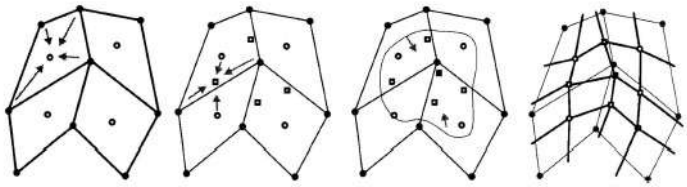
3.4.1.2 Subdivison modelování

Subdivison modelování se často využívá jako rozšíření krabicového nebo polygonálního modelování. Metoda subdivision modelování se aplikuje na celý objekt nebo na jeho část a lze ji aplikovat rekurzivně v několika úrovních. Existuje více algoritmů pro implementaci subdivision modelování, ale v praxi se nejvíce používá aproximační algoritmus Catmull-Clark, který rozděluje každý polygon na menší polygony. Například každý čtyřúhelník je rozdělen na čtyři menší čtyřúhelníky, proto každá úroveň zčtyřnásobí množství polygonů objektu za předpokladu, že se objekt skládá jenom ze čtyřúhelníků. Subdivison modelování tak může být výpočetně velmi náročné v případě, že má původní objekt již mnoho polygonů, nebo pokud je aplikováno více úrovní metody najednou.[14][3, s. 233]

Na druhou stranu tato metoda umožňuje ukládat objekty s nízkým množstvím polygonů, což podporuje přenositelnost objektů. Subdivision operaci je poté možno aplikovat až na straně klienta. Je také možné měnit úroveň subdivision v závislosti na požadované míře detailu. To je užitečné například u interaktivních aplikací. V praxi často není nutné aplikovat příliš vysokou úroveň. Obvykle stačí provést několik kroků a povrch výsledného objektu je dostatečně hladký. Vyšší úrovně subdivision se používají pouze pro tvorbu fotorealistického obsahu.[3, s. 228][14]

Subdivision modelování může využít buď aproximační nebo interpolační dělení polygonů. Interpolační dělení využívá původní sítě objektu a pouze ji dělí. Interpolační dělení tak nevyhlazuje ostré hrany, nicméně lze jej použít například pro procedurální modelování. Aproximační dělení nevyužívá původní sítě objektu a místo toho vytváří úplně novou síť.

Aproximační dělení se snaží přiblížit k limitě povrchu objektu, čímž zaobluje všechny hrany.[3, s. 230]



Obrázek 3.5: Průběh algoritmu Catmull-Clark[3, s. 234]

Metoda Subdivison modelování dokáže rozdělit polygon o jakékoliv velikosti, nicméně pouze čtyřúhelníky lze perfektně rozdělit. Rozdělené trojúhelníky nebo n-gony obvykle produkují nepříjemné výsledky. Rozdělené trojúhelníky zhoršují viditelné artefakty zatímco n-gony mohou velmi snadno deformovat celý povrch objektu. Na obrázku níže je možné vidět efekt aproximačního algoritmu. Na levé příšeře si lze všimnout zaoblené siluety levého ramene stvůry. Naopak na pravé příšeře je silueta levého ramene stvůry jasně hranatá. Podobný efekt je k vidění například na jazyce příšery.[14]

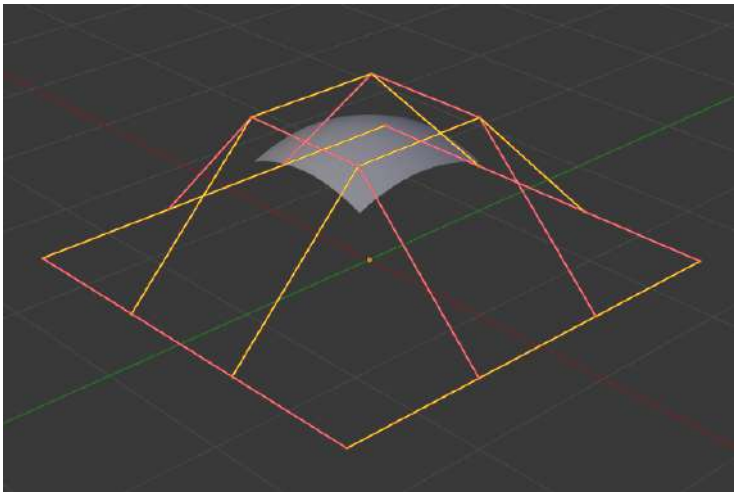
Obrázek 3.6: Subdivision modelování[14]

3.4.1.3 NURBS modelování

Metoda modelování pomocí NURBS křivek je jednou z nejstarších metod pro modelování 3D objektů. Jedná se o metodu používanou spíše u computer-aided design, kde je vyžadována vysoká přesnost při modelování, nicméně NURBS křivky lze použít i při modelování obecných nebo organických objektů.[17]

Neuniformní racionální B-spline křivky (NURBS) umožňují vytvářet dvojrozměrné i trojrozměrné povrchy. Generování NURBS křivky je velice triviální a intuitivní, což umožňuje rychlý vývoj 3D objektů. Osamocené NURBS křivky samy o sobě nejsou moc užitečné, ale propojením dvou a více NURBS křivek vzniká povrch, který lze snadno upravovat manipulací s kontrolními body křivek. Uživatel nemá přístup k samotné síti vygenerovaného povrchu, nicméně díky jednoduchosti této metody má uživatel jistotu, že povrch je vytvořen čistě z čtyřúhelníků. Výhody NURBS křivek jsou nízká paměťová náročnost a vysoká přenositelnost mezi programy. Nevýhodou je jejich procedurální přístup generování sítě, který zvyšuje časovou náročnost pro vykreslení snímků.[17][42]

Při modelování je možné využít objektových primitiv vytvořených z NURBS křivek. Může se jednat například o válec nebo kouli.[43]



Obrázek 3.7: NURBS modelování[35]

Zjednodušenou verzí NURBS křivek jsou Bézierovy křivky, které samy od sebe nevytváří povrchy. Bézierovy křivky mají podobné chování jako NURBS křivky a lze s nimi manipulovat podobným způsobem, tedy za pomocí kontrolních bodů. Bézierovy křivky lze

využít pro tvorbu loga, animace pohybu jiného objektu (například pohyb kamery) nebo pro vytažení profilu objektu.[42]

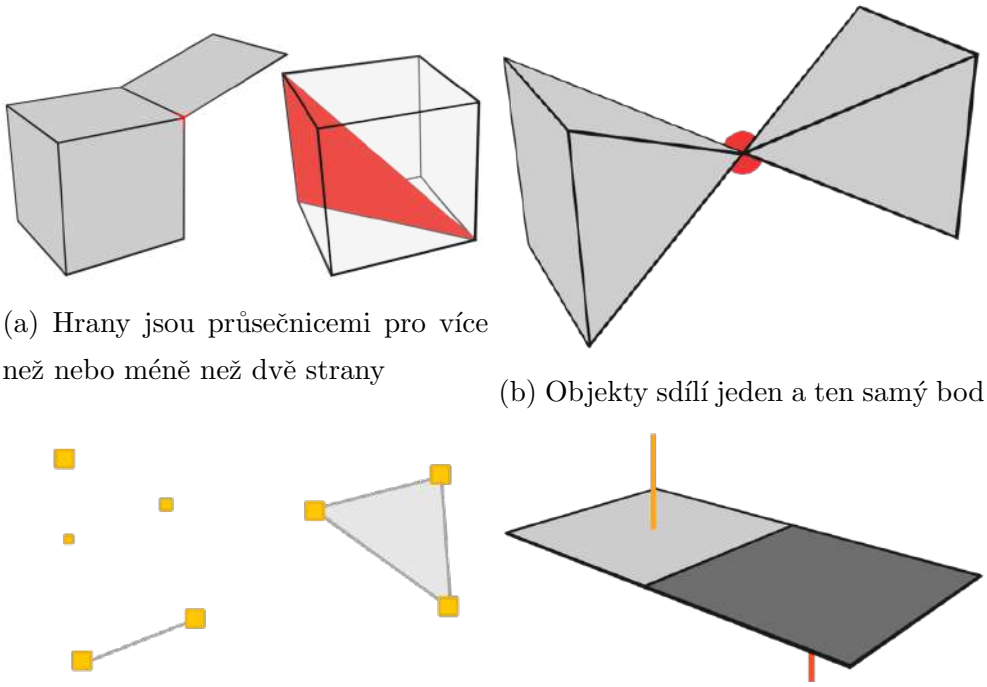
3.4.1.4 Modelování Booleovskými operacemi

Modelování Booleovskými operacemi využívá Booleovy operace k úpravě sítě jednoho objektu druhým objektem. Lze použít operace rozdílu (Difference), sjednocení (Union) a průniku (Intersect). Tato modelovací metoda se hodí pro úpravu objektů, které jsou příliš složité a ruční úprava prostřednictvím polygonálního modelování by trvala příliš dlouho.[44][18]

Podmínkou pro použití Booleových operací je vlastnost manifold pro oba objekty. Za manifold se považuje objekt, který může existovat ve skutečném světě. Jedná se tedy také o podmínku pro vývoj modelů pro 3D tisk, nebo pro provádění počítačových simulací jako jsou simulace tekutin. Manifoldy lze identifikovat tím, že objekt je nepřetržitý, tedy nemá začátek ani konec jako například krychle nebo koule. Opakem manifoldů jsou nonmanifoldy, tedy objekty které nemohou existovat ve skutečném světě. Například objektové primitivum plochy je nonmanifold, neboť lze jasně identifikovat kde objekt začíná a kde končí. Nonmanifoldy vycházejí z matematické a geometrické abstrakce, umožňující například dotyk dvou oddělených objektů v jednom a tom samém bodě. I přesto mají nonmanifoldy využití při vývoji objektů, kde je požadavek na nízké množství polygonů. Nonmanifoldy lze identifikovat různými způsoby:[19][3, s. 240]

- Hrana objektu je průsečnicí pouze pro jednu stranu. Objekt má jasně viditelný konec nebo může mít uvolněnou geometrii.
- Hrana objektu je průsečnicí více než dvou stran. Objekt má například vnitřní strany.
- Hrana objektu protíná jinou stranu.
- Normály dvou sousedících polygonů míří opačným směrem.
- Dva oddělené objekty se protínají v jednom a tom samém vrcholu nebo hraně nebo straně.

Výsledek modelování Booleovskými operacemi závisí na použité operaci. Operace rozdílu odečte jeden objekt od druhého. Operace sjednocení připojí jeden objekt k druhému. Operace průniku je inverzní k operaci rozdílu a zachová pouze tu síť, kterou objekty sdílí.[44]



(a) Hrany jsou průsečnicemi pro více než nebo méně než dvě strany

(b) Objekty sdílí jeden a ten samý bod

(c) Uvolněná geometrie jednoho objektu

(d) Otočené normály sousedících polygonů

Obrázek 3.8: Příklady nonmanifoldů[19]

Síť objektu je generována strojově a uživatel k ní nemá přístup během samotné operace. Síť tak nelze upravovat tak, jako je tomu u polygonálního modelování. Po skončení Booleovy operace je ovšem síť přístupná a lze ji upravovat polygonálním modelováním. Problémem modelování Booleovskými operacemi je generování n-gonů. N-gony se obecně špatně vykreslují pokud je použita některá metoda hladkého stínování a při aplikaci subdivision metody se navíc deformují. Proto se nedoporučuje použití Booleových operací u organických modelů, kde je běžné využití hladkého stínování. Booleovy operace se hodí pro modelování objektů s ostrými hranami jako například monitorů nebo klávesnic.[18]

3.4.1.5 Procedurální modelování

Procedurální modelování je poněkud obecný pojem pro řadu dalších modelovacích metod. Všechny metody mají podobnou vlastnost generování 3D obsahu za pomocí nějakého algoritmu. Spadá sem modelování pomocí jednoduchých procedur, částicových systémů,

fraktální geometrie nebo simulace (například tekutin). V této diplomové práci budou popsány pouze jednoduché procedury.[3, s. 265]

Procedurální modelování je založeno na takzvaně nedestruktivním přístupu. Ostatní metody popsané v této práci jsou destruktivní, to znamená, že jejich užitím lze udělat nenapravitelnou chybu. Taková chyba může mít různě závažné dopady na výsledný objekt a často vyžaduje nápravu. Nedestruktivní přístup stále produkuje chyby, ale je možné kdykoliv se vrátit o několik kroků zpět a chybu napravit bez výrazné ztráty obsahu. Je také možné kdykoliv změnit nastavení každé procedury. Další výhodou procedurálního modelování je automatizace některých činností při modelování, které by jinak zabraly příliš mnoho času.[45]

Jednotlivé procedury jsou spouštěny jedna po druhé a jsou ukládány do zásobníku. Procedury, které se nachází na vrchu zásobníku jsou spouštěny před procedurami na konci zásobníku. Pořadí, ve kterém jsou procedury spouštěny je tak důležité a změna pořadí procedur může vyvolat velkou změnu na objektu. Procedury mohou vytvářet novou síť objektu, upravovat existující síť objektu nebo upravovat další data objektu jako jsou normály ploch.[45]

Obrázek 3.9: Objekt vytvořený pomocí řady procedur[45]

Popsané metody subdivision modelování a modelování Booleovskými operacemi lze považovat za procedurální modelování, ale záleží na konkrétní implementaci těchto metod v daném programu.[45]

3.4.1.6 Sculpting

Modelování pomocí Sculpting nástrojů generuje a upravuje síť objektu pomocí štětců. Štětce mají vždy svoji velikost, která určuje míru vlivu štětce na síť objektu. Štětce zároveň

mají sílu, která určuje intenzitu efektu štětce na objekt. Poslední vlastností štětce je jeho typ, určující, jak se štětec chová. Uživatel nemá přímý přístup k síti objektu tak, jako je tomu u polygonálního modelování. Jelikož se pracuje s trojrozměrnými objekty, tak štětce běžně následují zakřivení povrchu objektů.[46]

Modelování Sculpting nástroji svým průběhem připomíná skutečné modelování s hlínou. Běžně se začíná s obecnými tvary a až poté se uživatel soustřeďuje na přidávání detailů modelu. Zde si lze vzít příklad z polygonálního modelování, kde jsou detaily rovnoměrně přidávány na celý objekt. Sculpting se používá pro tvorbu organických modelů (například postavy) nebo jiné přírodou vytvořené tvary (například kmeny stromů).[20]

Dostupné štětce mohou síť objektu přesunout z místa na místo, otočit nebo přidat novou vrstvu polygonů. Je také možné vytvořit nové štětce za pomocí rastrových obrázků použitých jako textury. Takové štětce mohou aplikovat vzorek uschlých listů nebo tkaniny na povrch objektu.[20]

Nevýhodou modelování Sculpting nástroji je vysoká výpočetní i paměťová náročnost. Ta se zvyšuje s přibývajícím množstvím polygonů potřebných pro vyjádření detailů na objektu. Nejmenší možné detaily mohou být samostatné póry na lidské kůži. V takovém případě může množství polygonů vzrůst na miliony.[20]

Obrázek 3.10: Objekt modelovaný Sculpting nástroji[46]

3.4.1.7 Retopology

Poslední modelovací metodou, která bude popsána je metoda retopology. Jedná se o metodu příbuznou polygonálnímu modelování, která nejčastěji funguje jako rozšíření metody

Sculpting. Retopology lze ale použít i pro jiné účely, než jako rozšíření Sculpting metody. Metoda se používá pro vytvoření zjednodušené kopie objektu, pokud má původní objekt příliš velké množství polygonů. Cílem je vytvoření optimalizované sítě, která je vhodná pro vykreslení anebo animace.[21]

Toho lze dosáhnout vytvořením nového objektu a jeho úpravou pomocí klasického polygonálního modelování. Správným nastavením vývojového prostředí lze docílit připínání nově vytvářených polygonů k původnímu objektu s vysokým množstvím polygonů. Tak dojde k překrytí původního objektu novým objektem, který má optimalizovanou síť tvořenou z čtyřúhelníků. Polygony lze šikovně rozložit po povrchu původního objektu tak, aby byl nový objekt vhodný pro animaci. Je například možné zvýšit hustotu polygonů v místech, kde se bude pokožka postavy protahovat, například v oblasti kloubů.[21]

Nový objekt je obvykle vytvářen manuálně uživatelem. Výhodou tohoto postupu je plná kontrola nad rozložením polygonů. Existují ale také nástroje, které zrychlují práci s polygony například tím, že vyplní uživatelem označený prostor stejně velikými polygony. Jiné nástroje jsou schopny kompletně automatizovat celou metodu retopology, ovšem tyto metody často produkují nekvalitní výsledky.[21]

3.4.2 Textury

Textura je obecné označení pro obrázek, který přenáší nějakou informaci a je aplikovatelný na jakýkoliv 3D objekt. Přenášené informace mohou mít vysokou vypovídající hodnotu o objektu za relativně nízkou cenu výpočetního výkonu. Je tak běžnou praxí použití objektů s nízkým množstvím polygonů spolu s vysoce kvalitními texturami. Textury lze rozdělit na základě jejich rozměru.[3, s. 379]

- Jednorozměrné textury obsahují jednoduché vzorky nebo křivky.
- Dvojrozměrné textury se mapují na povrch objektu a jedná se o nejčastěji používané textury. Dvojrozměrné textury dovolují přenášet podstatně větší množství informací, než jednorozměrné textury.
- Trojrozměrné textury se také nazývají objemové textury, a jedná se o textury definující trojrozměrný prostor.
- Čtyřrozměrné textury fungují jako rozšíření trojrozměrných textur o hodnotu času. Používají se tak v animacích.

Textury lze dále dělit dle způsobu jejich uložení.[3, s. 380]

- Textury uložené v jednorozměrné, dvourozměrné nebo trojrozměrné tabulce jsou ukládány jako klasické rastrové obrázky nebo jako voxely.
- Textury je také možné definovat pomocí procedury, tedy pomocí nějakého algoritmu.

Poslední způsob dělení textur je dle informace, kterou předávají. Textury nemusí obsahovat pouze barevnou informaci, ale můžou přenášet i další informace potřebné pro vykreslení povrchu. Níže je výčet základních typů textur:[22][5, s. 167]

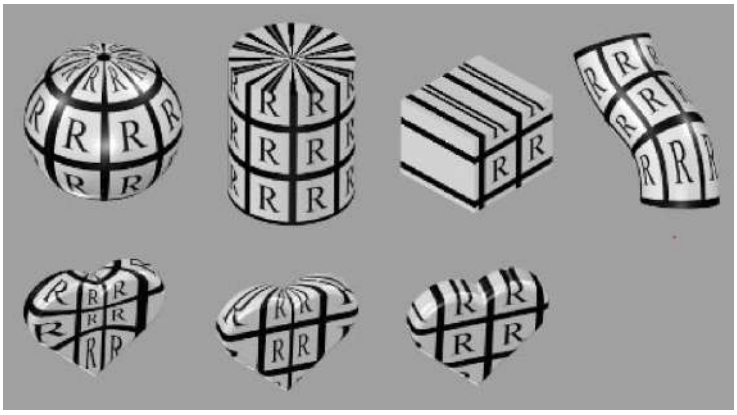
- Albedo textura obsahuje barevnou informaci povrchu. Jedná se o nejzákladnější texturu, která neobsahuje žádné další informace.
- Diffuse textura obsahuje barevnou informaci povrchu stejně jako Albedo textura, ale zároveň obsahuje informaci o rozptylu světla po povrchu. Podstatné je rovnoměrné rozložení světla na textuře. Diffuse má tak větší barevnou hloubku v porovnání s Albedo texturou, která se při jejich srovnání může zdát bledá.
- Luminosity textura způsobuje rozsvícení povrchu. V závislosti na použité technologii může být povrch pouze rozsvícen ale nebude ozařovat ostatní objekty a nebo může povrch fungovat jako samostatný zdroj světla ve scéně.
- Specular textura zobrazuje jak je povrch objektu lesklý, pokud se na něj namíří světlem. Textura zároveň zobrazuje vady na povrchu, které snižují lesklost.
- Glossiness textura upravuje velikost Specular odrazu.
- Reflection textura určuje kde povrch odráží své okolí a zároveň ostrost odrazu. S ostrostí odrazu se to často přehání a odrazivé plochy jsou tak nerealistické.
- Transparency textury určují průhledné povrchy. Zároveň s tím vyžadují index lomu světla.
- Bump textury vytvářejí nerovnost povrchu. Taková nerovnost může být pouze optická iluze, neboť textura neovlivní pozici vrcholů tvořící síť objektu. Jiné textury jsou ale schopny přesouvat vrcholy objektu a vyvolávat tak skutečnou nerovnost povrchu.

3.4.2.1 Mapování textur

Pro aplikaci textury na povrch objektu je nejdříve nutné provést projekci povrchu do souřadného systému textury, což je zpravidla dvojrozměrný prostor s osami u, v. V tomto prostoru je možné pracovat s jednotlivými vrcholy, hranami i stranami promítnutého objektu stejně, jako u polygonálního modelování. Projekcí povrchu objektu do UV prostoru lze mapovat konkrétní bod textury přímo na povrch objektu. V případě užití obrázkové

textury se body textury označují jako texely a jedná se o klasickou zobrazovací informaci (pixel). Pokud je použita procedurální textura tak je její hodnota vypočtena pro každý bod. Projekci objektu do UV prostoru lze provést pomocí nějaké projekční funkce nebo pomocí procesu unwrap.[5, s. 170]

Projekční funkce promítá objekt do UV prostoru pomocí různých algoritmů. Jedná se například o projekci koule, válce, krychle nebo projekce z pohledu kamery. Tyto funkce jsou rychlé, ale většinou nevhodné pro složitější objekty.[5, s. 171]



Obrázek 3.11: Projekční funkce koule, válce a krychle[5, s. 171]

Projekční funkce často vyvolávají zkreslení textur v místech, kde jsou polygony objektu natočené do podobného úhlu, jako je projekční směr. To lze vidět na obrázku nahoře, kde projekční funkce koule, válce i krychle je aplikována na objekt srdíčka. V určitých místech je textura na srdíčku silně zkreslená. To řeší metoda unwrapp.[5, s. 172]

UV unwrapping je proces podobný projekčním funkcím. Opět zde dochází k projekci trojrozměrného objektu do dvojrozměrného prostoru u, v. Rozdíl oproti projekčním funkcím je v tom, že UV unwrapping zároveň rozřezává objekty na menší části. Jedná se tak o vhodnější metodu pro složitější objekty. Opět existuje více algoritmů pro rozřezání objektu. Například je možné zkoumat úhel, který mezi sebou svírají přilehlé polygony a pokud je úhel vyšší než zadaná vstupní hodnota, tak dojde k řezu v místě společné hrany. Je také možné aby uživatel sám od sebe určil, které hrany budou rozříznuty.[22]

3.4.2.2 Procedurální textury

Procedurální textury jsou rychle se rozvíjející odnož počítačové 3D grafiky. Procedurální textury jsou generované prostřednictvím algoritmu a ačkoliv nejsou nijak zvlášť náročné na paměť zařízení, mohou se stát velice snadno náročné na výpočet. Proto se spíše používají u vývoje počítačových animací nebo statických snímků, než u vývoje interaktivních aplikací. Nejčastěji se využívají pro generování trojrozměrných (objemových) textur, protože ukládání trojrozměrných textur do voxelů je velmi náročné na kapacitu úložného prostoru. Obrovskou výhodou procedurálních textur je jejich nezávislost na rozlišení. Procedurální textury lze běžně zvětšovat i zmenšovat bez snížení kvality obrazu. Další výhodou je jejich modifikovatelnost. Hotové procedurální textury lze často rychle měnit a vytvářet tak alternativní textury.[5, s. 198][3, s. 390]

Procedurální textury často závisí na funkci náhodného šumu, která přidává chaotickou (a tím přírodní) složku do textury. Taková funkce je předem vypočítaná a ostatní části procedury se na ni běžně odkazují. Existuje více algoritmů pro generování náhodného šumu, ale nejvíce používaným je Perlin.[5, s. 199]

Zásadní nevýhodou procedurálních textur je jejich až příliš umělý vzhled a to i přesto, že jsou běžně aplikovány různé funkce náhodného šumu. Procedurálním texturám běžně chybí drobné chyby na jejich povrchu. Může se jednat například o škrábance, otisky prstů nebo poškrábané hrany. Takové drobné detaily často nelze vyjádřit prostřednictvím algoritmu.[22]

Nicméně vývoj procedurálních textur značně pokročil v posledních několika letech. V současnosti jsou dostupné programy pro míchání fotorealistických textur mezi sebou, čímž vznikají nové textury. Použité textury jsou naskenované povrchy kamení, betonu, dřevěné kůry a další. Textury jsou míchány různými způsoby, například je možné vytvořit jednoduchou černobílou masku nebo zkoumat výšku jedné textury a aplikovat do jejích dutin druhou texturu. Výsledná textura je tak fotorealistické kvality a i přesto je procedurální.[23]

3.4.3 Shadery

Pro vykreslení povrchu nějakého objektu je nutné využít počítačový kód nazvaný shader. Shader využívá výpočetní výkon grafické karty pro výpočet dopadu a chování světla na povrchu objektu. Shadery lze rozdělit do různých kategorií, podle jejich chování.[24]

Obrázek 3.12: Fotorealistický procedurální materiál[23]

Shadery definující povrch objektu jsou v současnosti nejčastěji založené na Dvousměrové odrazové distribuční funkci (BRDF). Takové shadery zkoumají odrazové vlastnosti povrchu materiálu v určitém bodě. K tomu je potřeba znát úhel dopadu a úhel odrazu světla a také výchozí zář povrchu. BRDF shadery mohou být rozšířeny o další parametry, jako je odraz světla nebo průchod a lom světla. BRDF shadery jsou tak založeny na skutečnostech známých z fyziky a umožňují vykreslovat realistické povrchy.[3, s. 325][47]

Kromě povrchových shaderů také existují například emission shadery, které mění povrch objektu na zdroj světla, volume shadery (objemové shadery), které popisují jak se světlo rozptyluje uvnitř povrchu a shadery pozadí, které definují vzhled a chování prostředí scény.[47]

3.4.4 Počítačová Animace

Počítačová animace je proces, ve kterém je statický 3D objekt donucen se pohybovat nebo měnit tvar v průběhu času. Objekt lze ovlivňovat přímo nebo nepřímo. Přímá animace zasahuje do vlastností samotného objektu. Například je možné animovat pozici nebo rotaci objektu. Nepřímou animací se myslí ovlivnění objektu prostřednictvím nějakého

jiného objektu. Nehledě na zvolenou metodu, animace se zaznamenává do časové přímky prostřednictvím takzvaných klíčových snímků.[49]

3.4.4.1 Kosterní animace

Kosterní animace je nepřímý typ animace, kdy je objekt ovlivňován svojí abstraktní reprezentací. Prvním krokem tohoto typu animace je rozdělení sítě objektu do skupin vrcholů, které reprezentují pohyblivé části objektu. Celá síť nemusí být využita pro animace, například vnější stěny chránící vnitřní díly stroje mohou být statické. Celý model se také může skládat z více objektů. V tomto případě je možné brát každý objekt jako samostatnou skupinu vrcholů, nebo každý objekt dále dělit na několik menších skupin vrcholů. Množství skupin je závislé na složitosti požadované animace. Například animace chůze lidského těla si může vystačit se skupinami pro končetiny. Jakékoliv složitější animace mohou vyžadovat více skupin, například páteř, krk, hlavu nebo jednotlivé prsty na ruce postavy. Každá skupina vrcholů má svoji vlastní abstraktní reprezentaci v podobě kosti. Soubor všech kostí se nazývá kostra a kosterní animace je tedy proces transformace jednotlivých kostí pomocí transformačních operací posunu, rotace a škálování a mapování transformace zpět na síť objektu.[6, s. 64, 68]

Kosti jsou propojené prostřednictvím kloubů a mají přesnou hierarchickou strukturu, která by měla být podobná struktuře objektu. Každá kost má svého vlastního rodiče v podobě jiné kosti, kromě kořenové kosti. Kořenová kost je v hierarchii kostí na nejvyšší úrovni a jakákoliv transformace této kosti vyvolá transformaci všech ostatních kostí. Kořenová kost by měla být umístěna ve středu celého modelu a ukládá svoji globální pozici a orientaci. Všechny ostatní kosti také ukládají informace o své pozici a orientaci, ale ty jsou vždy relativní k jejímu rodiči. Transformace jakékoliv kosti tudíž vyvolá transformaci i všech jejích potomků.[6, s. 69, 71]

Mapování transformace kosti zpět na síť objektu je proces, který následuje po transformaci samotné kosti a vyvolává transformaci připojené skupiny vrcholů v síti objektu. V potaz je brána vždy jen kost, která reprezentuje danou skupinu vrcholů, ovšem z důvodu hierarchického uskupení celé kostry, transformace jedné kosti vyvolá transformaci i všech skupin vrcholů připojených ke kostem potomků. Transformace skupin vrcholů lze předem spočítat pro celý průběh animace a uložit.[6, s. 72]

Každému vrcholu ve skupině vrcholů je přidělena číselná váha z intervalu $< 0, 1 >$. Tato váha určuje sílu působení připojené kosti na daný vrchol. Pokud je váha velmi nízká, tak

transformace kosti vyvolá jenom malou změnu pozice vrcholu. Pokud je váha blízká hodnotě 1, tak vrchol téměř kopíruje transformace kosti. Vrchol může mít více číselných vah pocházejících od více kostí, zejména pokud se nachází v blízkosti kloubu spojujícího dvě a více kostí. Váhu každého vrcholu je možné automaticky vypočíst z relativní vzdálenosti vrcholu od kosti ve výchozí poloze kosti, nebo je možné ji určit ručně.[6, s. 75]

3.4.4.2 Klíčové snímky

Pro vytvoření animace se obvykle používá soubor klíčových snímků pro zaznamenání transformací na časové přímce. Klíčové snímky mohou zaznamenávat transformace objektu v přímé animaci, nebo transformace jednotlivých kostí a kloubů v nepřímé animaci. Klíčový snímek se tak stává časovým razítkem, které obsahuje transformační údaje či jiné údaje (například barvu) a které je zachyceno v určitém snímku animace. Celkové množství klíčových snímků určuje složitost i délku animace. Například animace lidské chůze může být animována jen za pomocí pouhých čtyř klíčových snímků zaznamenávající pohyby osmi kostí v končetinách modelu. Lze ovšem vytvořit i složitější animaci lidské chůze.[6, s. 77][49]

Obsah snímků, které nejsou klíčové, ale nacházejí se na časové přímce mezi dvěma klíčovými snímky, lze vygenerovat pomocí interpolace sousedních klíčových snímků. Lze k tomu využít lineární funkci, funkci křivky, funkci kroku nebo jiné. Funkce kroku kopíruje předchozí klíčový snímek do všech snímků následujících, dokud animace nenarazí na další klíčový snímek. Lineární interpolace generuje přímku spojující sousedící klíčové snímky a na této přímce generuje nové snímky animace. Funkce křivky funguje na stejném principu jako lineární interpolace, akorát generuje křivku. Tvar této křivky může být různý a vyvolává dojem plynulejšího pohybu objektu.[6, s. 79]

3.4.5 Fotorealistické zobrazování

Fotorealistické zobrazování je způsob vykreslování takových snímků, které nelze rozeznat od fotografie. Fotorealistické zobrazování se v současnosti běžně používá pro tvorbu vizualizací (například interiérů, exteriérů nebo samostatných produktů) a pro vizuální efekty. Pro tvorbu fotorealistického obsahu je využito řady postupů, které se společně podílejí na výsledném snímku.[3, s. 413]

Samotné objekty musí mít realistickou formu. To zahrnuje jejich proporce i celkovou velikost. Zejména se to týká organických tvarů (například lidé), u kterých lze snadno odhalit nepřesnosti. Je také vhodné často využívat fotografie jako referenční snímky, které zachycují objekt z více úhlů.[25]

Pro vykreslení fotorealistických materiálů je vhodné využít fotorealistické textury a zároveň fyzicky založené shadery. Fotorealistické textury zachycují povrchy objektů a zejména jejich nedostatky, které způsobují variaci chování povrchu v různých místech. Může se jednat například o šmouhy nebo smetí na povrchu objektu. Společně balíček textur vytváří materiál. Fyzicky založené shadery jsou schopny takové balíčky textur využívat a vykreslit jejich informaci.[25]

Jedním z nejdůležitějších shaderů současnosti je Disney Principled BRDF Shader (zkráceně Disney BRDF). Disney BRDF je označován za fyzicky založený shader, ačkoliv to není úplně správně. Disney BRDF byl vytvořen s důrazem na jednoduché ovládání a ztrácí trochu fyzikální přesnosti. Proto je často označovaný také jako zásadový. Níže je výčet zásad, podle kterých byl shader vytvořen:[26]

- Parametry shaderu mají být spíše intuitivní spíše než fyzikálně přesné.
- Parametrů shaderu by mělo být co nejméně.
- Parametry shaderu by měli podporovat hodnoty v rozsahu $< 0, 1 >$, nicméně parametry by měli přijímat i jiné hodnoty, pokud to bude potřeba.
- Všechny parametry by měly být kompatibilní mezi sebou.

Podle definovaných zásad společnost Disney vytvořila shader, který obsahuje jednu barevnou informaci určující barvu povrchu a deset číselných parametrů definujících různé fyzikální vlastnosti povrchu. Jedná se například o míru hrubosti povrchu, intenzitu odrazu světla nebo zda-li je povrch kovový nebo ne. Disney BRDF se tak stal standardem ve vykreslování fyzicky založených materiálů pro účely fotorealistického zobrazování. Disney BRDF byl převzat mnoha vývojáři a je tak dostupný v řadě programů.[26][48]

Pro osvětlení fotorealistické scény se používá pouze globální osvětlení. Jedná se o takové osvětlení, které zahrnuje přímé i nepřímé osvětlení. Přímé osvětlení způsobuje osvětlení objektů z nějakého zdroje světla, zatímco v rámci nepřímého osvětlení se objekty osvětlují navzájem. Velmi lesklé a jemné povrchy mohou také vyvolat ostrý odraz jiných objektů. Patří sem také stíny a polostíny, které vznikají když je mezi jedním objektem a zdrojem světla jiný objekt. Kromě toho je vhodné využívat takových světelných zdrojů, které mají stejnou barvu, intenzitu i směr, jako skutečné zdroje světla.[3, s. 413][25]

Posledním krokem při tvorbě fotorealismu je přidání defektů, které jsou běžné u fotoaparátem pořízených snímků. Jedná se zejména o rozmazání pohybu, hloubku ostrosti, odlesky a chromatická aberace. Rozmazání pohybu vzniká ve chvíli, kdy se nějaký objekt zachycený kamerou pohybuje vysokou rychlostí. Hloubka ostrosti zajišťuje ostrost některých prvků na snímku na úkor jiných, které se zdají rozmazané. Odlesky vznikají ve chvíli, kdy se na snímku nachází jasný zdroj světla, který je namířený na snímek. Chromatická aberace vzniká v čočkách objektivů, které lámou jednotlivé barvy paprsků světla různým způsobem.[25]

Fotorealistické zobrazování obvykle vyžaduje použití nějaké ray tracing technologie. Hlavním důvodem pro využití ray tracing technologie je právě globální osvětlení, které je zpravidla nižší kvality pokud je využita technologie vykreslování v reálném čase.[25]

Obrázek 3.13: Příklad fotorealistického snímku[36]

3.4.6 Ne-fotorealistické zobrazování

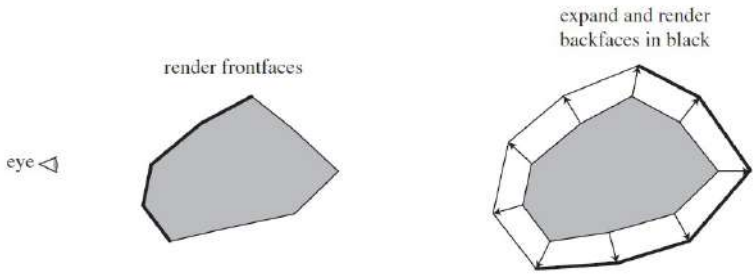
Zatímco fotorealistické zobrazování se snaží vytvořit obraz nerozeznatelný od skutečného obrazu, ne-fotorealistické zobrazování cílí na stylizaci objektů počítačové 3D grafiky. Praktické využití ne-fotorealistického zobrazování může být tvorba informační grafiky například v manuálech nebo jiných technických dokumentech, kde je požadavek na jed-

noduchost a jasnost. Pomocí ne-fotorealistického zobrazování je také možné simulovat malířské styly, například kresbu perem nebo malbu vodními barvami.[5, s. 651]

Jednou z metod používaných pro ne-fotorealistické zobrazování je cel shading nebo také toon shading. Cel stínování zjednodušuje barvy scény a zároveň pomáhá rozeznávat tvary, neboť objekty jsou obtažené zvýrazňující čárou. Metoda má velké využití u komiksů i animovaných filmů a lze ji pěkně kombinovat spolu s klasickými 2D kresbami. Cel stínování kombinuje více metod pro dosažení výsledku.[5, s. 652]

Ve své nejjednodušší podobě má povrch objektu vykreslovaného cel stínováním dvě barvy. Jedna reprezentuje osvětlenou část objektu a druhá reprezentuje tu část objektu, která je ve stínu. Přechod mezi světlou a tmavou částí může být lineární nebo ostrý a může obsahovat více tónů. Zda-li je část objektu osvětlená nebo ne lze získat ze skalárního součinu normály stínovaného povrchu a směru přicházejícího světla.[5, s. 652]

Obtažením objektu čarou lze docílit pomocí různých metod. Nejjednodušší metodou se jeví procedurální geometrická silueta. Jedná se o vytvoření kopie celého objektu, jak je vyobrazeno na obrázku níže. Kopie bude rozšířena rovnoměrně do všech stran podle směru normál všech vrcholů původního objektu. Míra rozšíření reprezentuje tloušťku hraniční čáry. Normály všech polygonů kopie musí být otočené, musí tedy směřovat směrem dovnitř předmětu. Jelikož nejsou zadní strany polygonů vykreslovány, tak kopie objektu nebude vidět. Po obvodu původního objektu bude nicméně viditelná přední strana kopie objektu, byť to je jenom malá část kopie předmětu. Přiřadíme-li tmavý materiál této kopii tak vyvoláme efekt tmavé čáry okolo původního objektu.[5, s. 657]

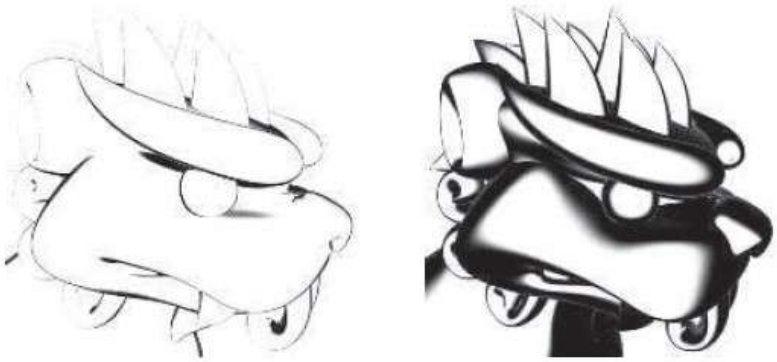


Obrázek 3.14: Metoda procedurální geometrická silueta[5, s. 659]

Nevýhodou této metody je zdvojnásobení množství polygonů pro každý objekt, který má mít ohraničení. Zároveň jsou zvýrazněny pouze takzvané hrany siluety objektu, které

oddělují objekt od jeho pozadí. Obrysové hrany, které vyjadřují obrysy všech ostrých hran objektu, nejsou viditelné.[5, s. 654, 659]

Metoda stínování normál obrysových hran je výpočetně nenáročná a je schopna vykreslit tmavé čáry na siluetě objektu a zároveň na jeho obrysových hranách. Metoda funguje na podobném principu jako již popsané vykreslení barvy na objektu. Dochází ke skalárnímu součinu normály stínovaného povrchu a směru kamery scény. Násobením skalárního součinu lze dosáhnout kontroly nad tloušťkou hraniční čáry. Nevýhodou metody je špatné vykreslení hraniční čáry u plochých nebo mírně zakřivených povrchů.[5, s. 656]

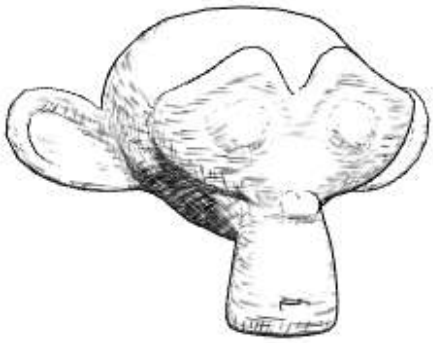


Obrázek 3.15: Metoda stínování normál obrysových hran[5, s. 656]

Hraniční čáry lze také vykreslovat jako post processing efekt. Takovéto metodě se říká detekce hran pomocí zpracování obrazu. Metoda nepracuje s geometrií objektů a místo toho pracuje s daty uloženými do paměti při vykreslování snímku. Mnoho obrysových hran je možné identifikovat hledáním nesouvislostí v z-bufferu, který se používá pro určení vzdálenosti polygonů od kamery. Po vyhledání všech hran siluety a obrysových hran jsou vykresleny hraniční čáry již na hotový rastrový obrázek. Výhodou této metody je vyšší kontrola nad tvarem hraničních čar. Je například možné aplikovat funkci náhodného šumu a vytvořit tak klikatou hraniční čáru.[5, s. 660]

Cel shading také často využívá šrafování povrchu. Šrafování je možné aplikovat na osvětlené i tmavé části objektu a je možné dynamicky měnit hustotu šrafování. Je také možné aplikovat šrafování na odraz světla. Vzorek šrafování se vyjadřuje prostřednictvím klasické textury, nebo pomocí procedurální textury. Důležité je, že textura šrafování se

neaplikuje na UV mapu jako to běžně bývá, nýbrž je aplikována na objekt z aktuálního pohledu kamery.[5, s. 669][37]



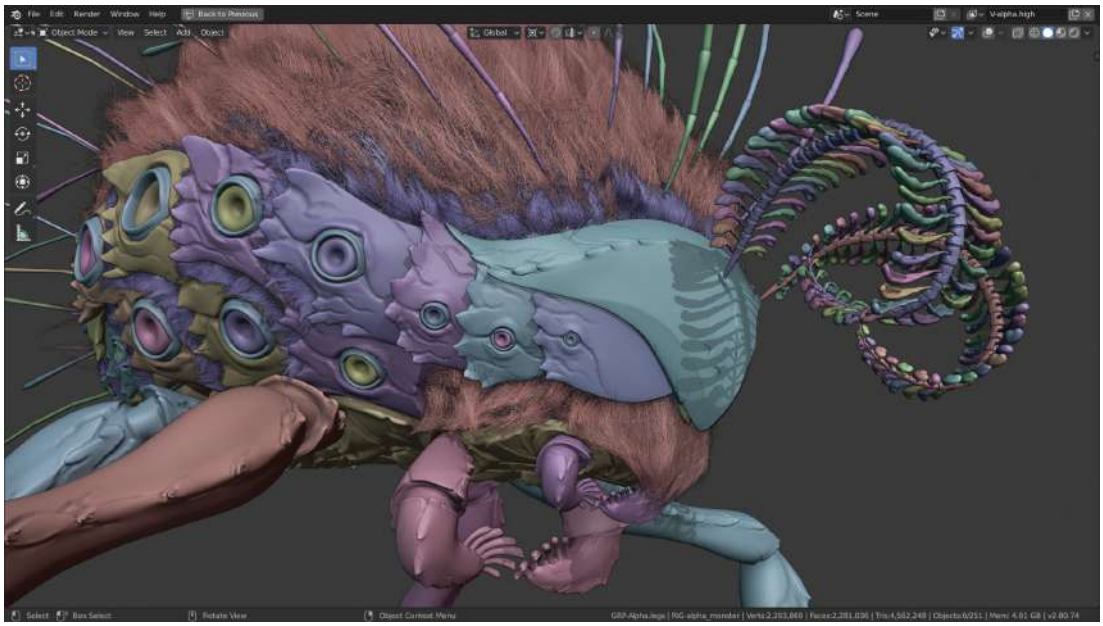
Obrázek 3.16: Šrafování stínů[37]

3.4.7 Zobrazování v reálném čase

Zobrazování v reálném čase je způsob rychlého vykreslení dvojrozměrného rastrového obrazu z trojrozměrných dat. Používá se zejména u interaktivních aplikací, kdy uživatel může reagovat na výstup aplikace a jeho reakce již ovlivňuje následující chování aplikace. Zobrazovací cyklus by měl být co možná nejrychlejší a obvykle se měří ve snímcích za vteřinu. Pokud aplikace vykresluje alespoň 6 snímků za vteřinu, tak uživatel začíná cítit interaktivitu aplikace. Interaktivní aplikace ale často míří na vyšší množství vykreslených snímků. Toto se týká zejména videoher, které se snaží dosáhnout vykreslování alespoň třiceti snímků za vteřinu.[5, s. 1]

Kromě videoher se zobrazování v reálném čase používá i při vývoji 3D obsahu. Vývojové renderery jsou zpravidla jednodušší a nejsou určené pro vykreslování konečného obrazu. Místo toho mají funkce pro podporu vývoje, jako je například X-ray režim, nebo zvýraznění světlých hran a tmavých dutin.[50]

Zobrazování v reálném čase je založeno na grafickém řetězci, který je rozdělen do několika fází. Jedná se o aplikační fázi, geometrickou fázi, rasterizační fázi a fázi zpracování pixelů. Fáze jsou spouštěny paralelně v systému, nicméně pozdější fáze jsou závislé



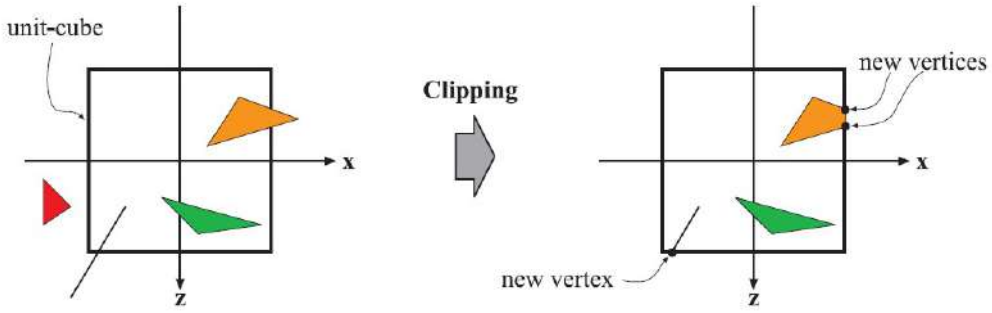
Obrázek 3.17: Workbench renderer pro vývoj 3D obsahu[50]

na výsledku předchozích fází. Pokud nějaká fáze trvá déle než by měla, tak to zpomaluje celý řetězec.[5, s. 12]

Aplikační fáze většinou pracuje s procesorem a je spouštěna samotnou aplikací. Aplikační fáze se používá pro výpočet velkého množství výpočetně náročných kalkulací, jako je detekce kolizí, deformace objektů v animaci, simulace fyziky, nebo zpracování informací ze vstupních zařízení. Výsledkem aplikační fáze je soubor primitiv pro vykreslení, tedy trojúhelníky.[5, s. 13]

Geometrická fáze pracuje zejména s vrcholy a polygony, které získá z aplikační fáze. V geometrické fázi dochází k výpočtu barvy vrcholů v závislosti na použité metodě stínování. To zahrnuje jak barvu samotného polygonu tak přicházející světlo. Mezi další úkoly geometrické fáze patří úprava existujících objektů, například přidání polygonů nebo posun některých vrcholů. Součástí geometrické fáze je také oříznutí celé scény. Objekty, které nejsou k vidění na kameře, jsou kompletně odstraněny a objekty u kterých je viditelná pouze jejich část jsou ořízнuty. Oříznuté a odstraněné polygony jsou uvolněny z paměti.[5, s. 14]

Fáze rasterizace přijímá transformační data všech vrcholů, které mají být vykresleny a zároveň data ze stínování trojúhelníků. Cílem fáze rasterizace je promítnout všechny zbývající trojúhelníky na rastrový obrázek. Toho lze docílit projekcí každého trojúhelníku

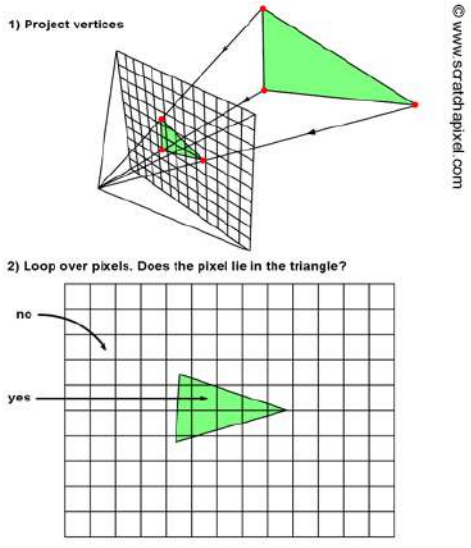


Obrázek 3.18: Oříznutí objektů scény[5, s. 20]

na kameru scény, kde každý pixel samostatně zaznamená barvu promítnutého trojúhelníku nebo jeho hrany nebo jeho vrcholu.[5, s. 21]

Pokud dojde k projekci dvou trojúhelníků na jeden a ten samý pixel, je potřeba určit, který z trojúhelníků je blíže kameře a má se tedy vykreslit. To probíhá za pomoci z-bufferu, což je dvojrozměrné pole o stejné velikosti, jako je vykreslovaný rastrový obrázek. Z-buffer ukládá vzdálenost nejbližších trojúhelníků pro každý pixel rastrového obrázku a přepisuje ji v případě, že algoritmus nalezne bližší trojúhelník.[27]

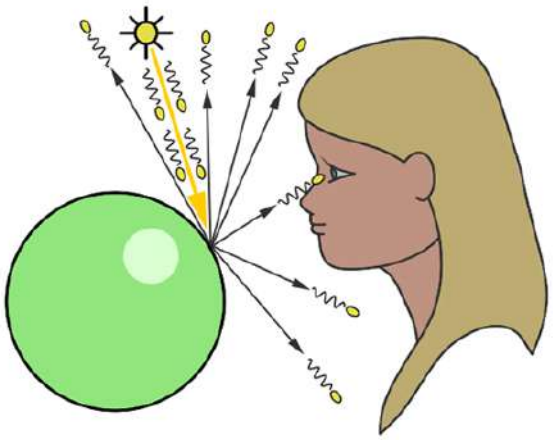
Výsledkem fáze rasterizace je rastrový obrázek zobrazující objekty v jejich stínované podobě. Poslední fází je fáze zpracování polygonů, ve které jsou načteny další barevné informace, zejména textury. Textury jsou poté aplikovány na stínované modely.[5, s. 22]



Obrázek 3.19: Fáze rasterizace[27]

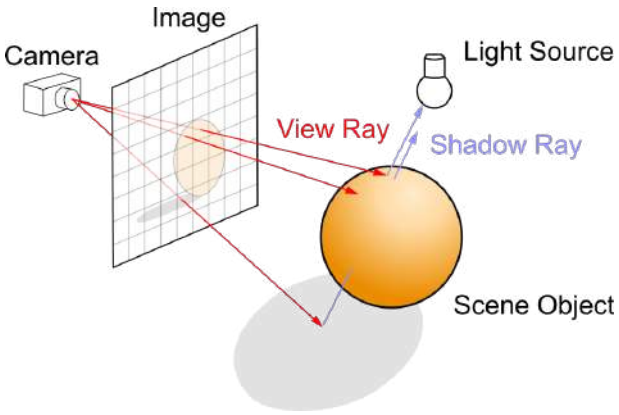
3.4.8 Zobrazování sledováním paprsků

Zobrazování sledováním paprsků (ray tracing) je druhou běžně používanou metodou pro vykreslení 3D obsahu. Ray tracing vychází ze skutečností známých o chování světelných paprsků vycházejících například ze slunce. Světelné paprsky se po dopadu na předmět odráží do různých směrů. Pokud jeden z odražených paprsků zasáhne lidské oko, tak jsme schopni daný předmět spatřit. Využití tohoto postupu při vykreslení obrazu je téměř nemožné. Problémem je množství generovaných paprsků zatěžující výpočetní systémy, které vycházejí ze zdroje světla a náhodně se odrážejí od povrchů objektů. Dalším problémem je generování řady paprsků, které mají minimální nebo nulový význam pro vykreslení obrazu.[28]



Obrázek 3.20: Chování světla[28]

Na rozdíl od skutečných poznatků o chování světla, ray tracing používá postup opačný, při kterém paprsky vycházejí z pomyslného oka uživatele a směřují směrem do 3D scény. Díky tomu lze minimalizovat množství paprsků, které je potřeba sledovat pro vykreslení celého rastrového obrázku. Bod ze kterého paprsky vychází se obvykle ilustruje jako lidské oko nebo kamera. Před tímto bodem se nachází plocha reprezentující finální 2D rastrový obrázek, tudíž lze o této ploše také uvažovat jako o mřížce, jejíž velikost odpovídá rozlišení rastrového obrázku. Zpravidla je kamera vykreslována spolu s rastrovým obrázek jako jeden celek připomínající položenou pyramidu, kde rastrový obrázek je základna pyramidy směřující do zbytku 3D scény a kamera je vrcholem pyramidy.[1, s. 387]



Obrázek 3.21: Princip ray tracing[38]

Paprsky vycházející z pomyslného oka procházejí skrze jednotlivé pixely generovaného rastrového obrázku. Jakmile je trasa jednotlivých paprsků spočtena, tak je zkoumáno, zda-li daný paprsek prochází skrze nějaký trojúhelník ve 3D scéně. Pokud paprsek neprochází žádným trojúhelníkem, tak je pixel vyplněn dle výchozího nastavení 3D scény (například nějakou základní barvou) a algoritmus začne zkoumat další paprsek. Pro každý trojúhelník, kterým paprsek prochází je nutné spočítat jeho vzdálenost od zdroje paprsku, díky čemuž je možné najít trojúhelník nejblíže ke kameře, tedy trojúhelník, jehož materiál bude vykreslován na zpracovávaný pixel výsledného rastrového obrázku.[1, s. 391]

Paprsek směřující do vybraného trojúhelníku se na místě dopadu odráží směrem do zdroje světla ve 3D scéně. Pokud scéna obsahuje více zdrojů světla, tak se paprsek rozdělí a postupně se odrazí do všech zdrojů světla. Tyto paprsky zkoumají intenzitu a barvu přicházejícího světla a také velikost zdroje světla a zda-li je mezi zdrojem světla a zkoumaným trojúhelníkem nějaký další trojúhelník, jehož působením vzniká stín. Pokud scéna neobsahuje žádné zdroje světla, a ani pozadí scény nevytváří žádné světlo, tak je trojúhelník vykreslen plně černý, bez ohledu na materiál, stejně jako zbytek celé scény. Ray tracing je výpočetně náročná metoda vykreslování, která ovšem dokáže prezentovat 3D obsah ve vysoké kvalitě, někdy i ve fotorealistické kvalitě.[1, s. 414]

Speciální pozornost je možné věnovat trojúhelníkům, které jsou průhledné a světlo jimi prochází anebo jsou schopny světlo odrážet. Odraz světla lze spočítat z normály plochy trojúhelníku a úhlu dopadu paprsku. Lom světla potřebuje ty samé údaje, ale zároveň vyžaduje index lomu, numerickou hodnotu vyjadřující jak moc se paprsek světla lomí při průchodu trojúhelníkem. Například index lomu u skla je obvykle 1,5 zatímco diamant má index lomu 2,4.[29][30]

Zatímco zobrazování v reálném čase pracuje na úrovni samostatných primitiv (trojúhelníků), zobrazování sledováním paprsků pracuje na úrovni jednotlivých pixelů. To značně zpomaluje celý proces vykreslení snímků a jedná se o metodu nevhodnou pro většinu interaktivních aplikací. Výhodou ray tracing je celkově jednodušší implementace, vyšší kvalita obrazu a přesnější efekty jako jsou stíny, odrazy nebo lom světla.[2, s. 201]

3.5 Blender

Program Blender je zdarma dostupný open source projekt, běžící na platformách Windows, Linux a MacOS, určený pro samostatné umělce a malé týmy. Blender má podporu skriptů napsaných v jazyce Python, které upravují chování programu nebo přidávají nové funkce. Blender lze využít k tvorbě krátkých i dlouhých počítačových animací, vizuálních efektů pro hrané filmy, objektů pro video hry a jiné interaktivní aplikace, pro vývoj modelů pro 3D tisk i pro kreativní uměleckou činnost.[39]

Blender lze označit za program všeuměl, neboť jeho funkční vybavení je velmi rozsáhlé v porovnání s jinými programy. Blender podporuje všechny metody popsané v kapitole Modelování a další, jako například metaballs, částicové systémy a procedurální modelování pomocí skriptů. Kromě toho Blender podporuje grease pencil pro tvorbu 2D animací. Blender dále umožňuje tvorbu UV map pro objekty, mapování textur, tvorbu materiálů a tvorbu procedurálních textur. Přímá i nepřímá animace je také podporována, včetně kosterní animace. Existuje také podpora základních simulací, jako je simulace ohně, vody nebo látky. Blender využívá tři renderery. Renderer Workbench se používá pro vývoj. Ray tracing renderer Cycles se běžně používá pro vykreslení snímků pro produkci. Real-time renderer Eevee je novinkou na trhu a nabízí unikátní funkcionalitu. Blender dále podporuje kompozici vykreslených snímků a také funguje jako video editor.[39]

Blender zaznamenal v posledních letech velký nárůst uživatelů. Důvodů je hned několik. Blender měl dlouhá léta špatně přístupné uživatelské prostředí, které bylo plně předěláno ve verzi 2.80, která byla vydána 30. července 2019. Spolu s tím dostal Blender renderer Eevee, podporu 2D animací a optimalizace rendereru Cycles. Od té doby dostal Blender řadu aktualizací. V současnosti je nejnovější verze Blenderu verze 2.90, která přináší další optimalizace různých systémů, nové funkce a lepší podporu grafických karet Nvidia.[39]

Přestože je Blender prezentován jako program pro samostatné umělce a malé týmy, tak je Blender používán i u větších společností. Blender je k nalezení například u společnosti Mojang, která vyvíjí hru Minecraft.[31]

3.5.1 Eevee

Real-time renderery jsou používány zejména u interaktivních aplikací a často nejsou schopny produkovat fotorealistické snímky. Programy pro vývoj počítačové 3D grafiky využívají velmi triviální renderery pro vykreslení svého obsahu. Například renderer Workbench, který běží v Blenderu, je schopen vykreslovat miliony polygonů ve scéně, ale nemá podporu PBR materiálů. Takové renderery lze označit jako vývojové. Pro renderování počítačových animací se tak běžně používá pomalá technologie sledování paprsků, která produkuje podstatně lepší výsledky. Takové renderery lze označit jako produkční.[50]

Eevee vykresluje obsah v reálném čase. Jedná se o unikátní renderer mezi ostatními renderery svého druhu, protože se jedná o kombinaci vývojové a produkční technologie. Eevee je schopen vykreslovat snímky rozsáhlých scén ve vysoké, někdy i fotorealistické kvalitě a podporuje řadu dalších efektů jako jsou volumetrics nebo ambient occlusion. Eevee schválně vykresluje snímky v nižší kvalitě, pokud je použit při vývoji aby byl schopen produkovat snímky co nejrychleji. Naopak během vykreslení výsledného snímku nebo animace se Eevee vzdává svojí vlastnosti vykreslení v reálném čase a vykreslení každého snímku může trvat i několik vteřin.[39]

3.6 Barevné modely

Pro vyjádření barev se v počítačové grafice využívá barevných modelů. Barevné modely definují barvy, které lze zobrazit například na monitoru. Celková množina všech barev v barevném modelu se nazývá barevný prostor.[32]

3.6.1 Model RGB

Jedním z nejčastějších barevných modelů je model RGB. Barvy jsou zde vyjádřeny pomocí číselných nebo procentuálních hodnot v jednotlivých kanálech modelu RGB, nebo pomocí

šestimístného hexadecimálního zápisu zahrnující všechny tři kanály. Kanály modelu RGB jsou barvy červená (R), zelená (G) a modrá (B), kde každý kanál má stejnou velikost. Nejčastěji se používá 24 bitový RGB formát, tj. každý barevný bod je zaznamenán pomocí 24 bitů. Každý kanál má vyhrazených 8 bitů, tj. existuje $256^3 = 16777216$ různých barev.[3, s. 24]

Modelu RGB se často říká aditivní model, neboť složením všech tří barev vznikne barva bílá. Tato vlastnost je převzata z technických vlastností monitorů, kde luminiscenční prvky reprezentují červenou, zelenou a modrou barvu a společně rozsvěcí daný bod. Jedná se tedy o model, který využívají všechny počítačové monitory, displeje mobilních telefonů nebo televize. Existuje také rozšíření barevného modelu RGB, zvané RGBA. Barevný model RGBA obsahuje navíc kanál α, který určuje průhlednost obrazového bodu. Lze jej tak využít pro skládání více obrazů na sebe. Model RGBA se nejčastěji používá ve 32 bitovém formátu.[3, s. 25]

3.6.2 Model CMY

Barevný model CMY funguje na opačném principu než model RGB. Model CMY také obsahuje tři kanály. Jedná se o kanály tyrkysová (C), fialová (M) a žlutá (Y). Jejich skládáním vznikají tmavé barvy až po černou barvu, proto se model označuje jako subtraktivní. Model CMY se používá převážně pro tisk, kde jsou výchozí barvy aplikovány na papír. Častým rozšířením modelu CMY je model CMYK, který obsahuje navíc kanál pro černou barvu (K). Toto rozšíření je vytvořeno z důvodu častého užití černé barvy při tisku, kde klasický model CMY je ekonomicky nevýhodný. Model CMY založen na míchání skutečných barev tak, jak to dělají malíři.[3, s. 26]

3.6.3 Modely HSV a HSL

Modely RGB a CMY fungují na principu míchání barev pro získání požadované barvy. Takovéto míchání ovšem nemusí být intuitivní pro uživatele. Například změna odstínu barvy beze změny sytosti barvy je v modelu RGB velmi náročná. Tyto problémy řeší barevné modely HSV a HSL. Oba modely se opět skládají ze tří složek, ovšem na rozdíl od předchozích modelů, tyto složky nereprezentují samostatné barvy. Model HSV obsahuje složky barevného tónu (H), sytosti (S) a jasu (V). Model HSL obsahuje složky barevného

tónu (H), sytosti (S) a světlosti (L). Model HSV obsahuje určité technické nedostatky, které řeší model HSL, nicméně v praxi jsou modely zaměnitelné a záleží spíše na preferenci uživatele. Modely HSV i HSL jsou přívětivé vůči uživateli a umožňují mu rychle měnit odstín barvy beze změny ostatních parametrů.[32]

4. Vlastní práce

4.1 Analýza zadání

Zadání diplomové práce neurčuje umělecký styl a nevyžaduje fotorealistické zpracování. Lze tak experimentovat s různými nápady a odchýlit se od klasického realistického zpracování proporcí objektů a jejich vykreslení. Jelikož má výsledná animace fungovat jako reklama pro firmu, je vhodné určit několik omezení. Předně se jedná o časové omezení celé reklamy. Ta by neměla přesáhnout délky třiceti vteřin. Reklama by také měla využívat pestré barvy pro zachycení pozornosti. Dále by zde měl být prostor pro reklamní sdělení.

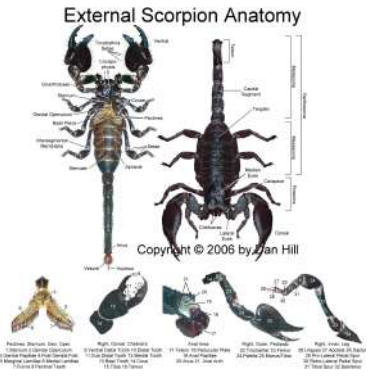
Firma a její oborové zaměření není specifikována, ale lze odhadnout, že se bude jednat o firmu soustřeďující se na skenování skutečných objektů. Posledních pět vteřin reklamy je obvykle vyhrazeno reklamnímu sdělení.

Zadání diplomové práce identifikuje jako základní objekty v animaci škorpióna, podstavec, snímací zařízení, laserový paprsek a hologram.

Pro tvorbu škorpióna je vhodné využít grafické podklady reprezentující anatomii, vzhled a chování škorpiónů. Zdrojem hlavních grafických podkladů je stránka The Scorpion Files soustřeďující se na tvorbu a udržování katalogu všech škorpiónů světa. Zdrojem grafických podkladů pro animaci bylo použito video How to animate a scorpion in motion. Jako vzorový model byl zvolen veleštír císařský z důvodu své popularity i velikosti.[54]

(a) Veleštír císařský[51]

(b) Veleštír císařský[52]



(c) Anatomie škorpióna[53]

Obrázek 4.1: Grafické podklady pro tvorbu škorpióna

Ostatní objekty scény nemusí být věrnou kopií existujících věcí. Snímací zařízení, které bude škorpióna skenovat, může tvarem připomínat diagnostický nástroj pro automobily, který je rozšířen o objektiv. Snímací zařízení bude vysílat proud světla, který bude přejíždět přes škorpióna.

Laserový paprsek potřebuje nějaký zdroj. K tomu lze vytvořit objekt podobný 3D skeneru. Objekt se tak bude skládat z objektivu, který bude snímat škorpióna a zároveň z plochy, na které bude škorpión posazen. Tato plocha bude zároveň fungovat jako podstavec definovaným zadáním, tudíž se musí otáčet.

Hologram škorpióna potřebuje pro existenci nějaký zdroj. Lze vytvořit statický projektor, ze kterého budou vycházet paprsky tvořící hologram.

Kromě zmíněných objektů je nutné vytvořit další objekty, které budou fungovat jako výplň pozadí. Všechny zmíněné objekty by se měly nacházet na nějaké rovné ploše. Lze tak vytvořit jednoduchý stůl. Snímací zařízení, laserový skener i hologram mohou být napojeny na nějaký počítač, který je bude ovládat. Dalším objektem bude tudíž malý notebook, ke kterému budou ostatní zařízení napojena. Posledním objektem bude podlaha a stěna místnosti, ve které se bude stůl nacházet. Není nutné vytvářet celou místnost, pokud bude animace zachycovat místnost pouze z jednoho úhlu.

Škorpión by měl vykazovat nějaký pohyb pro prezentaci kosterní animace. Lze tak například vytvořit animaci škorpióna, jak vylézá na podstavec.

4.2 Použitá technologie

Důležitým cílem diplomové práce je prezentace moderních postupů při tvorbě počítačové animace. V nadcházejících kapitolách budou představeny v současnosti standardní postupy tvorby 3D obsahu a budou identifikovány jejich silné a slabé stránky.

Jako zvolená technologie pro tvorbu animace byl zvolen program Blender. Důvodem této volby je zejména osobní zkušenost s programem a znalost jeho uživatelského prostředí zvyšující celkovou produktivitu. Dalšími důvody pro využití Blenderu je jeho licence, umožňující stažení programu zdarma i pro komerční účely a množství nástrojů, které lze použít pro tvorbu téměř jakéhokoliv 3D obsahu. V prostředí Blenderu budou vytvářeny všechny modely, animace, osvětlení scény a bude použit pro vykreslení animace i následné úpravy.

Posledním důvodem pro využití Blenderu je jeho nízká náročnost na hardware. Počítač na kterém bude diplomová práce zpracovávána má již zastaralý hardware a Blender v porovnání s jinými programy svého druhu vyžaduje mnohem méně operační paměti na fungování i výpočetního výkonu na vykreslení obrazu.

V praktické části diplomové práce se bude nacházet velké množství obrázků v nízkém rozlišení. Nízké rozlišení bylo zvoleno proto, aby diplomová práce významně nepřesáhla doporučený rozsah stran. Ovšem v některých případech může být rozlišení obrázků nedostatečné pro předání informace. Proto jsou všechny přiložené obrázky dostupné ve vysokém rozlišení v souborové příloze diplomové práce, která je k nalezení na přiloženém CD nebo nahraná na informačním systému České zemědělské univerzity.

4.3 Analýza rendererů Cycles a Eevee

Blender disponuje dvěma produkčními renderery. Jedná se o renderery Cycles a Eevee. V následující kapitole bude provedena analýza zobrazovacích schopností těchto dvou rendererů. Bude zkoumána rychlost vykreslení a kvalita výsledného obrázku na řadě několika scén. Scény jsou jednoduché a jsou vytvořené pro prezentaci specifického efektu. Cílem analýzy je zhodnocení rychlosti a kvality nového rendereru Eevee, zejména pak zda-li je Eevee schopen vykreslovat fotorealistický obsah.

Každý snímek je vykreslený v rozlišení 1000x1000. Cycles vykresluje každou cestu pro každý pixel 128 krát, zatímco Eevee pouze 64 krát. Cycles dále využívá OpenImageDenoise funkci pro odstranění šumu. Pokud není popsáno jinak u každého testu, renderery využívají své výchozí nastavení.

4.3.1 Ambient Occlusion

Ambient Occlusion je efekt zkoumající intenzitu přicházejícího světla, které je tlumeno jinými povrchy. Ambient Occlusion vykresluje jemné stíny na místech, kde je přicházející světlo tlumeno. Jedná se o nepřesnou ale rychlou alternativu ke globálnímu osvětlení.

Eevee ovšem aplikuje Ambient Occlusion jako post-processing efekt, tedy až poté, co je snímek vykreslen. Eevee tak dokáže pracovat jenom s výsledným dvojrozměrným obrázkem. Na snímku (b) lze zpozorovat, že mnohé strany jsou podstatně světlejší, než by měly být. V některých případech se zdá, že jsou osvětleny přímým zdrojem světla.

(a) Cycles (b) Eevee (c) Eevee celá scéna

Obrázek 4.2: Ambient Occlusion

Důvodem je oříznutí pravé stěny modré krychle, ve které se celá scéna nachází. Ta byla ve grafickém řetězci oříznuta a není zahrnuta ve výpočtu Ambient Occlusion efektu. Pouze v případě, že je vykreslována celá scéna (c), tak jsou zmíněné stěny správně stínované. V porovnání s Cycles jsou přechody světla do stínu podstatně kratší a i při vykreslení celé scény nejsou vykresleny všude, kde by měly být. Například vnitřní hrany modré krychle mají ve svém okolí příliš málo stínu.

Přestože efekt Ambient Occlusion prezentuje několik chyb, tak je vykreslený velice pěkně a neobsahuje žádné grafické artefakty.

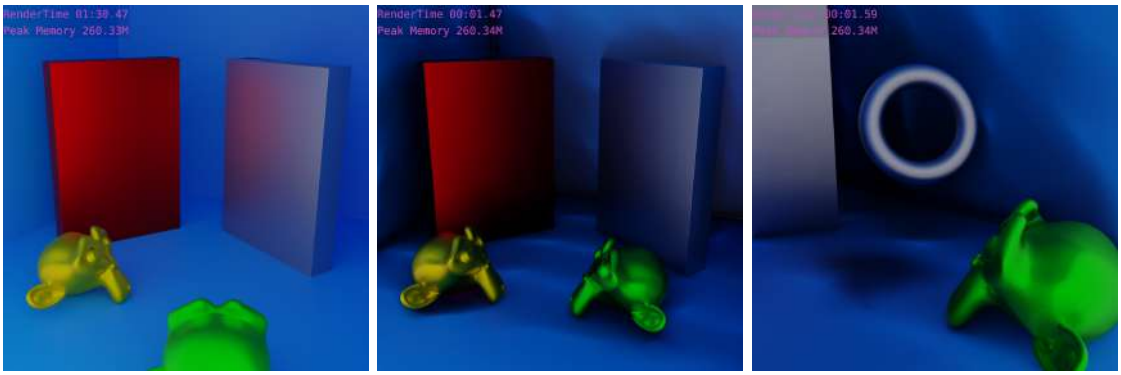
4.3.2 Globální osvětlení

Globální osvětlení se skládá z přímého a nepřímého osvětlení. Přímé osvětlení je vytvořeno zdroji světla a oba renderery jej plně zvládají. Problémy ale mohou vzniknout v případě užití nepřímého osvětlení.

Eevee nedokáže vykreslit nepřímé osvětlení v reálném čase, kvůli své technologické limitaci. Místo toho se používají světelné sondy, které zkoumají pozici blízkých objektů. Ty jsou užity pro pečení nepřímého světla a uložení světelných informací na pevný disk ve formě trojrozměrné textury.

Na snímku (a) vykresleném Cycles rendererem je k jak bílý kvádr obsahuje značné množství modrého i červeného světla pocházejícího od sousedních objektů.

Snímek (b) zachycuje kvalitu upečeného nepřímého osvětlení v Eevee. To má řadu nedostatků. Zejména se jedná o nutnost upečení nové trojrozměrné textury nepřímého osvětlení



(a) Cycles (b) Eevee (c) Eevee přiblížení chyb

Obrázek 4.3: Nepřímé osvětlení

pokaždé, když dojde ke změně ve scéně. Na snímku (c) je k vidění černá šmouha na modré podlaze. Ta ukazuje místo, kde se původně nacházela zelená hlava, která byla pro snímek (c) přesunuta. Jedná se tak o metodu nevhodnou pro animace, kdy dochází ke změnám v každém snímku animace.

Upečené nepřímé osvětlení bylo vytvořeno v nízkém rozlišení (lze si všimnout, že přechody na bílém kvádru (b) jsou kostičkované) a i přesto trvalo pečení přibližně 15 sekund. Pečení nepřímého osvětlení s vyšší kvalitou pro každý snímek může značně navýšit čas potřebný pro vykreslení, ale nelze s přesností odhadnout jak moc. To záleží na velikosti scény, množství světel i kvalitě pečené trojrozměrné textury.

Některé světelné sondy byly schválně umístěny dovnitř stěn modré krychle, aby vyvolaly další chybu. Takové sondy nepřijímají žádné světlo a tudíž generují plně černé oblasti, které vykreslují vnitřní hrany modré krychle plně černé. To je k vidění zejména na snímku (c).

Pokud je do scény vložen nový objekt po upečení nepřímého osvětlení, tak tento objekt přijímá informace z uložené trojrozměrné textury. Na snímek (c) byl přidán bílý anuloid, který je jasně osvětlen modrým světlem.

4.3.3 Odrazy

Odrazy na objektu lze vyvolat využitím jednoduchého Glossy shaderu, jehož parametr hrubosti je nastaven na 0. Cycles nevyžaduje žádné další úpravy nastavení a vykresluje velice přesné odrazy i na objektech, které mají zakřivené plochy.

Eevee je značně limitován. V teoretické části diplomové práce byl popsán grafický řetězec, v němž je trojrozměrný obsah oříznut a jsou vykreslovány pouze viditelné polygony.

(a) Cycles (b) Eevee (c) Eevee světelná sonda

Obrázek 4.4: Odrazy

Důsledek oříznutí je viditelný na snímku (b). Při porovnání snímku (b) se snímkem (a) si lze všimnout chybějících odrazů v zrcadle i na obličeji hlavy. Týká se to také polygonů, které jsou v záběru kamery, ale jsou ukryty za jinými polygony, proto na snímku (b) není vidět v zrcadle zadní strana hlavy a zadní strana koule.

Odrazy je možné do určité míry vyvolat pomocí světelné sondy Reflection Plane. Taková světelná sonda se chová jako nová kamera ve scéně a vykresluje svůj vlastní snímek, který je poté umístěn na uživatelem vykreslovaný snímek. Na snímku (c) je možné vidět efekt světelné sondy na zrcadle. Ale i zde je stejná chyba, která byla popsána v předchozím odstavci. Obličej zadní hlavy, který je viditelný v zrcadle, neodráží žádné zelené světlo, protože z pohledu světelné sondy byly potřebné polygony ořízнuty.

Světelnou sondu není možné ohnout, tudíž pokud je umístěna na zakřivený povrch, tak je odraz vykreslován pouze na části zakřiveného povrchu. Na obličej hlavy před kamerou byla také umístěna světelná sonda, ale s větší šířkou tak, aby sonda obalila celou hlavu. I přesto se ovšem odraz vykresluje značně deformovaný.

Dalším problémem je výpočetní náročnost. S každou další světelnou sondou použitou pro tvorbu odrazu je nutné celou scénu znovu vykreslovat. Teoretické použití pěti světelných sond pro pět odrazových ploch znamená, že jeden snímek případné animace by se vykresloval šestkrát, pokaždé z jiného úhlu.

4.3.4 Lom světla

Pro vytvoření průhledného materiálu byl použit Principled BSDF shader, který je určený pro fyzicky založené zobrazování. V tom byla nastavena hodnota Transmission na 1. Skleničky mají hodnotu Index of Refraction 1,5, která je běžná u skla, zatímco tekutina uvnitř skleniček má hodnotu 1,333, která je běžná u vody.

(a) Cycles (b) Eevee (c) Eevee

Obrázek 4.5: Lom světla

Na snímku (a) je k vidění výsledek rendereru Cycles. Snímek obsahuje drobnou chybu na hladině tekutiny, kde je lom světla vykreslen na podivném místě a velmi náhle oříznut. Nejedná se o chybu rendereru Cycles, nýbrž o chybu při modelování. Hladina tekutiny je totiž vytvořena jako n-gon.

Eevee není schopen správně vykreslit několik vrstev materiálů, které společně vytváří lom světla. Na snímku (b) je vidět, že hlava nalevo je protáhlá do stran. To je správné chování, které je způsobeno modrou tekutinou uvnitř skleničky. Samotná tekutina ovšem není vykreslena. Při odstranění jedné skleničky je možné tekutinu spatřit tak, jako je tomu u snímku (c).

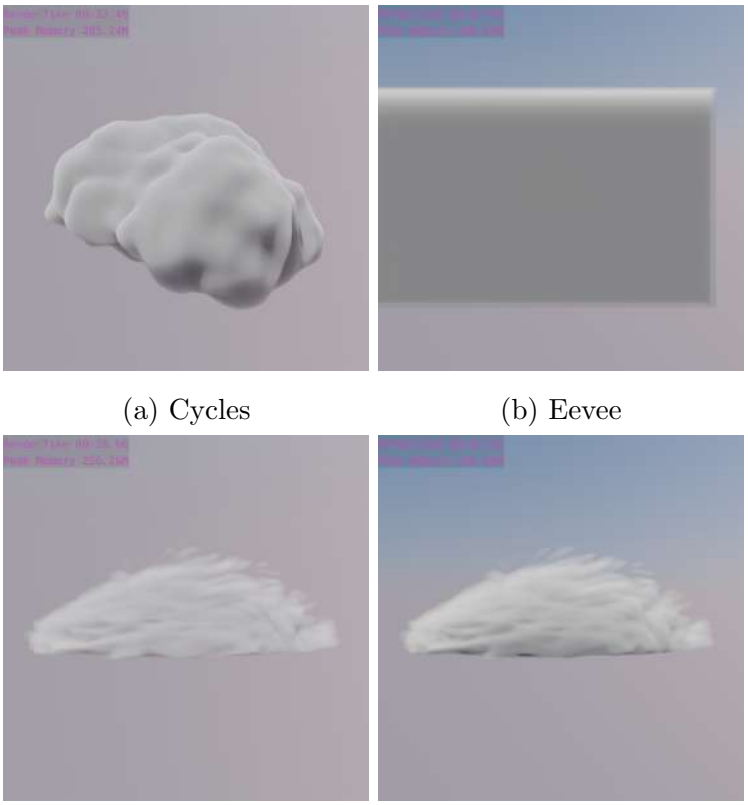
Na samotných skleničkách chybí vykreslení podstavy skleničky i dna skleničky. Horní hrana skleniček taktéž není vykreslena.

Pro tuto scénu byl také použit efekt Ambient Occlusion pro tvorbu jemných stínů na hlavách. Stíny jsou ale značně deformovány kvůli lomu světla a na povrchu hlavy tak vzniká spousta artefaktů.

Další limitací Eevee je neschopnost kombinovat lom světla spolu s odrazy. Na snímku (a) je na levé skleničce vidět mírný odraz pravé skleničky. Na snímcích (b) a (c) ovšem nejsou vidět žádné odrazy, pouze lomy světla.

4.3.5 Volumetrics

Volumetrics jsou takové objekty, které využívají objemové shadery k vykreslení svého obsahu, namísto obvyklých povrchových. Pomocí volumetrics shaderů je možné vytvořit prach, plyny nebo oblaka.



(a) Cycles (b) Eevee

(c) Cycles (d) Eevee

Obrázek 4.6: Volumetrics

Na snímku (a) je k vidění jak Principled Volume shader vyplňuje prostor deformované koule. Snímek byl vykreslen rendererem Cycles. Renderer Eevee ovšem nedokáže umístit volumetric shader dovnitř vytvořeného objektu. Eevee místo toho vyplňuje takzvanou

bounding box objektu, což je pomyslný kvádr, který ohraničuje celý objekt. To je k vidění na snímku (b).

Jedinou možností jak lze dát tvar volumetrics shaderu v rendereru Eevee je prostřednictvím samotného shaderu. Do Principled Volume shaderu je možné napojit různé trojrozměrné procedurální textury, které ovlivňují vzhled vytvořeného plynu. Výhodou tohoto postupu je velká kontrola nad tvarem shaderu. Další výhodou je relativně nízká náročnost na uložiště, neboť shader lze aplikovat i na obyčejný kvádr vytvořený z osmi vrcholů a šesti stran.

Takový postup je k vidění na snímcích (c) a (d), u kterých nelze najít žádný významný rozdíl. V tomto ohledu se Eevee zdá jako vynikající volba pro tvorbu volumetrics.

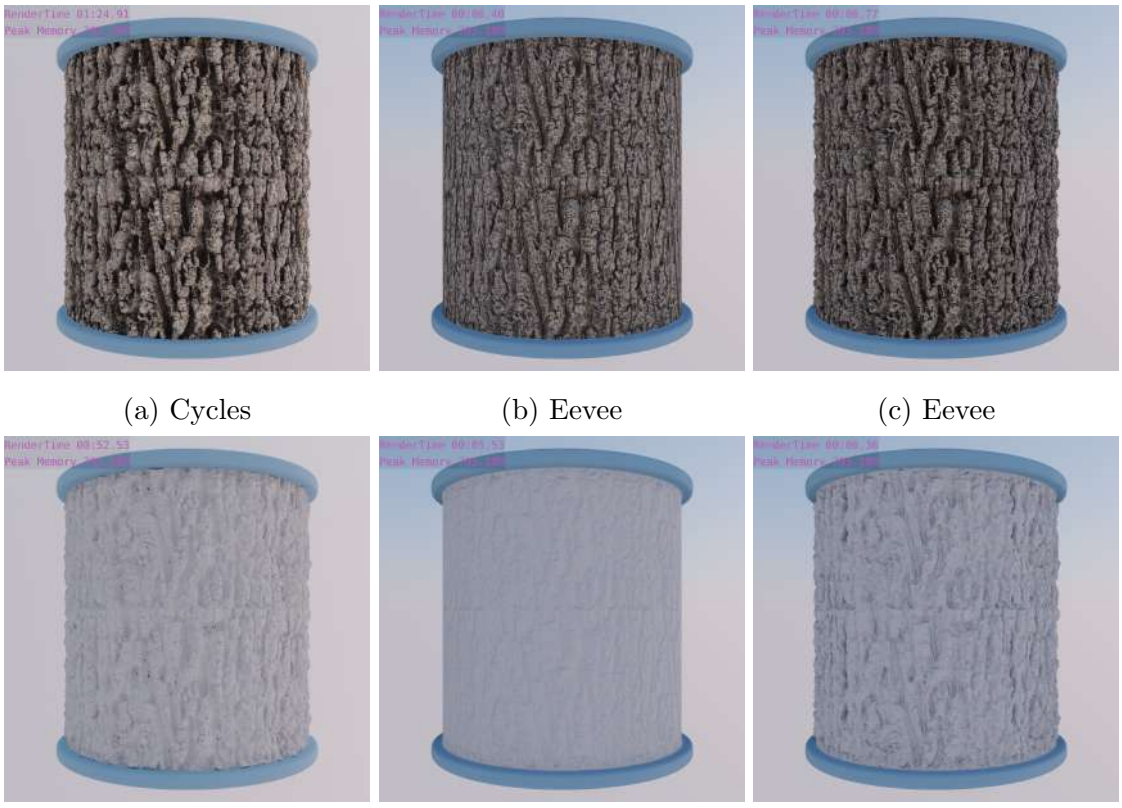
4.3.6 Kvalita materiálů

V rámci tohoto testu byl využit veřejně dostupný materiál Bark na platformě BlenderKit od uživatele Julio Sillet. Materiál obsahuje pět textur, které společně vytvářejí fotorealistický povrch kůry stromu.[55]

Eevee podporuje většinu typů textur, takže je možné na povrch objektu aplikovat diffuse texturu, texturu hrubosti, normální mapu a další textury. Textury jsou vykreslované ve vysoké kvalitě, srovnatelné s rendererem Cycles. Jediným problémem jsou displacement textury. Displacement textury spadají do kategorie bump textur, které vytvářejí nerovnost povrchu. Na rozdíl od normální mapy, která ovlivňuje pouze normály vrcholů a nevytváří skutečnou nerovnost povrchu, displacement textura ovlivňuje pozici vrcholů a tím vytváří skutečnou nerovnost povrchů.

Cycles renderer podporuje takzvaný shader displacement, tedy metodu, která dokáže využít displacement texturu a ovlivnit síť objektu skrze materiál. Na snímku (a) je k vidění vykreslený materiál v Cycles a na snímku (d) efekt shader displacement bez přidání dalších informací, jako je například barva povrchu.

Eevee nepodporuje shader displacement a místo toho se displacement informace vykresluje jako normální mapa. To je k vidění na snímcích (b) a (e). Falešnou nerovnost normální mapy lze nejlépe odhalit soustředěním se na siluetu objektu. Levá či pravá strana válce na snímku (d) jasně obsahuje nerovnost povrchu, zatímco na snímku (e) jsou obě strany plně ploché.



(a) Cycles (b) Eevee (c) Eevee

(d) Cycles shader displacement (e) Eevee shader displacement (f) Eevee displacement modifikátor

Obrázek 4.7: Kvalita materiálů

Součástí Blenderu je také modifikátor Displacement, který funguje na stejném principu jako Shader Displacement a je možné do něj nahrát displacement texturu materiálu. Eevee plně podporuje Displacement modifikátor a výsledek tohoto postupu je k vidění na snímcích (c) a (f).

4.3.7 Subsurface scattering

Subsurface scattering umožňuje přicházejícímu světlu proniknout do povrchu objektu a rozptýlit se uvnitř a zároveň vykreslit vnitřní barvu objektu. Jedná se o vlastnost organických povrchů, jako je lidská kůže nebo ovoce.

Eevee dokáže změnit barvu povrchu objektu v závislosti na vnitřní barvě objektu, ale nedokáže vykreslit rozptyl světla uvnitř povrchu objektu. Na snímku (a) je možné si všimnout rozptylu světla v uších opičí hlavy, který chybí na snímku (b).

(a) Cycles (b) Eevee

Obrázek 4.8: Subsurface Scattering

4.3.8 Výpočetně náročné scény

Scény obsahující milióny polygonů jsou výpočetně náročné na vykreslení. Pro tento test byla vytvořena scéna procedurálním modelováním, která obsahu skoro pět miliónů polygonů. Scéna byla také renderována ve 4K rozlišení.

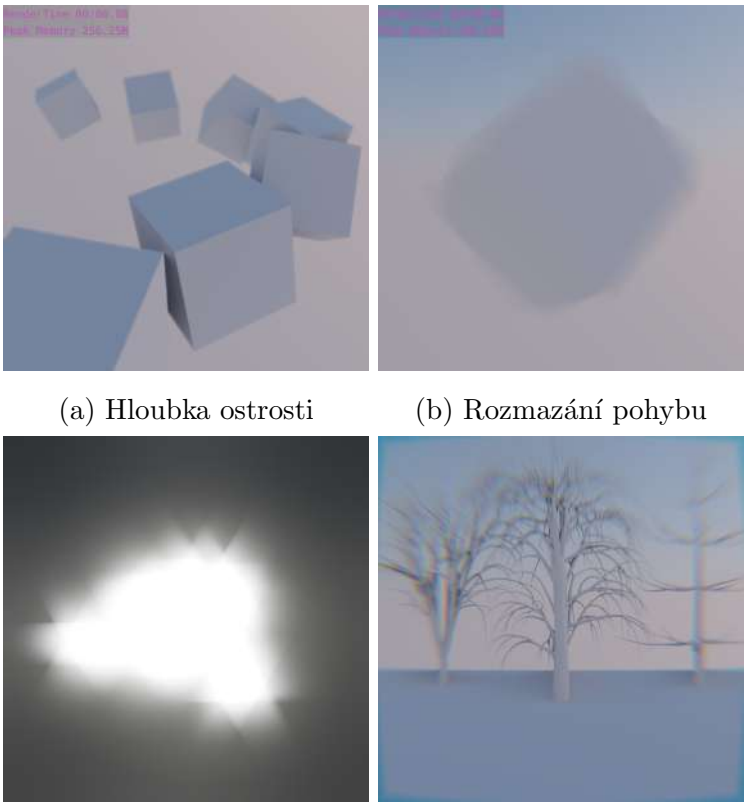
Renderer Cycles si s scénou bez problému poradil a vykreslil scénu za 37 minut. Renderer Eevee nedokázal scénu vykreslit a program spadl přibližně po patnácti minutách. V dokumentaci rendereru Eevee lze najít zmínku o správě paměti. Podle dostupné dokumentace má Eevee problémy s paměťově náročnými scénami, které mohou vyvolat právě pády programu nebo ovladače grafické karty. Jediné řešení je tedy optimalizace celé scény, nebo užití jiné technologie.

Jeden vykreslený snímek je dostupný v souborové příloze.

4.3.9 Defekty skutečných objektivů

Fotografové se snaží odstranit chyby na svých fotografiích, které snižují jejich kvalitu. V rámci fotorealistického zobrazování často takové chyby neexistují a proto jsou chyby přidávány ručně, aby se vykreslený snímek zdál více skutečný.

Defekty rozmazání pohybu a hloubka ostrosti jsou vykresleny použitým rendererem. Eevee podporuje oba efekty a ty jsou k vidění na snímcích (a) a (b). Odlesky na objektivu (c) a chromatická aberace (d) se ovšem přidávají dodatečně jako post-processing efekt a použitý renderer nehraje roli.



(a) Hloubka ostrosti (b) Rozmazání pohybu

(c) Odlesky na objektivu (d) Chromatická aberace

Obrázek 4.9: Defekty skutečných objektivů

4.3.10 Zhodnocení provedených testů

Tato kapitola se věnuje shrnutí provedených testů a prezentace jejich výsledků. Renderer Eevee využívající grafického řetězce pro vykreslení snímků v reálném čase se projevil jako působivý konkurent pro stávající ray tracing renderer Cycles.

Rychlost vykreslení snímků v Eevee je velmi působivá. Možnost vývoje 3D obsahu v reálném čase značně urychluje vývoj ve všech fázích. Naopak při užití ray tracing rendererů je nutné čekat několik vteřin až minut po každé provedené změně na vykreslení snímku. Tabulka níže zachycuje časy vykreslení pro všechny testovací snímky a zároveň shrnutí prezentovaných chyb.

Pokud je vykreslován výsledný (produkční) snímek, tak Eevee svoji rychlost poněkud ztrácí. Snímky se mohou vykreslovat i několik sekund. I přesto má Eevee veliký náskok nad ray tracing renderery.

	Cycles		Eevee	
Provedený test	**Čas vykreslení**	**Hodnocení snímku**	**Čas vykreslení**	**Hodnocení snímku**
Ambient Occlusion	01:07.73	Realistické jemné stíny.	00:01.61	Stíny mohou záviset na ořízntutých polygonech.
Globální osvětlení	01:30.47	Automaticky generované nepřímé osvětlení.	00:01.47	Ruční pečení nepřímého osvětlení.
Odrazy	01:24.71	Vynikající podpora odrazů světla a nastavení množství odrazů.	00:01.87	Odrazy mohou záviset na ořízntutých polygonech.
Lom světla	01:18.06	Vynikající podpora lomů světla.	00:01.26	Grafické artefakty, špatné vykreslení více objektů a odstranění jakýchkoliv odlesků.
Volumetrics	00:26.66	Plná podpora volumetrics.	00:07.03	Podpora shader volumetrics v reálném čase.
Kvalita materiálů	01:24.91	Plná podpora všech textur.	00:06.77	Chybí podpora shader displacement.
Subsurface scattering	00:44.08	Realistický rozptyl paprsků světla pod povrchem.	00:00.66	Rozptyl světla pod povrchem není správně vykreslen.
Zátěžový test	37:41.24		x	Pád programu po patnácti minutách.

Tabulka 4.1: Hodnocení provedených testů

Síla Eevee spočívá kromě v rychlosti také v kvalitě vykreslení povrchových i objemových materiálů a v možnosti pečení nepřímého osvětlení. Ovšem Eevee má značné nedostatky ve vykreslování efektů, jejichž vzhled je závislý na přicházejícím světle. Eevee se snaží různě napodobit ray tracing renderery, ale výsledky jsou chabé. To bylo prezentováno například na průhledných objektech.

Zda-li lze dosáhnout fotorealismu při užití Eevee je sporné a závisí to na vyvíjené scéně. Pokud se jedná o scénu, která je silně závislá na kvalitě povrchových materiálů, tak lze Eevee doporučit. Může se například jednat o snímky exteriérů, nebo některých interiérů. Jakmile ale scéna závisí na odrazech nebo je scéna výpočetně náročná, tak se stává Eevee nevhodným. U vizualizace produktů záleží na vlastnostech vykreslovaného produktu.

Také je možné uvažovat nad kombinací technologie Eevee s jinou technologií. Například je možné využít Eevee k tvorbě pozadí, tedy prostoru, kterému divák nebude věnovat pozornost. Popředí v tu chvíli může být vykresleno pomocí jiné a přesnější technologie.

I přesto je u výsledného snímku možné rozpoznat, že se jedná o počítačem generovaný snímek, nejčastěji kvůli osvětlení a stínování objektů, které bývá nerealistické. Pokud je cílem vytvoření skutečného fotorealismu, tedy takového snímku, který ani po jeho analýze není možné rozeznat od fotografie, tak Eevee nelze doporučit.

Renderer Eevee lze stále považovat za novou technologii, která je ještě ve vývoji. Konkurenčních rendererů pro vykreslování v reálném čase je velmi málo a drtivá většina z nich se soustřeďuje pouze na vývojovou fázi a pro produkční fázi se používají ray tracing renderery.

Díky své rychlosti lze Eevee doporučit pro rychlé a zároveň levné vykreslování snímků. To je důležité zejména u počítačových animací, které mohou být tvořeny deseti tisíci snímky. Týká se to také uživatelů, kteří mají omezené finanční prostředky (například menší studia).

Z výše uvedených důvodů bude animace vytvořena právě pomocí rendereru Eevee. Vzhledem k problémům s dosažitelnosti fotorealismu bude animace provedena za užití ne-fotorealistického zobrazování.

4.4 Modelování

Program Blender podporuje řadu modelovacích metod, které jsou přístupné v různých vývojových režimech, které Blender obsahuje. V praktické části je užito mnoha modelovacích metod pro dosažení výsledku. V některých případech jsou objekty vytvořeny pomocí jedné modelovací metody, jindy se jedná o kombinaci více metod. Užití té či oné metody závisí na použitelnosti metody pro tvorbu daného objektu.

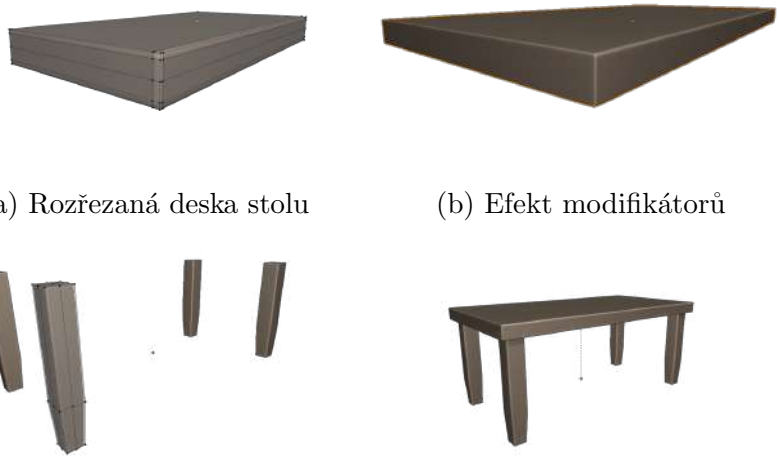
Jakýkoliv renderer je vždy omezován výpočetním výkonem počítače i svojí vlastní architekturou. Vždy existuje nějaký horní limit pro množství polygonů, objektů i materiálů ve 3D scény. Kromě horního limitu, kdy renderery začínají selhávat, je také nutné vzít v potaz čas vykreslování snímku. Z těchto důvodů je veškerý 3D obsah v praktické části optimalizován tak, aby jej šlo rychle vykreslit a zároveň aby měl obsah vypovídající hodnotu.

4.4.1 Stůl

Blender disponuje několika režimy. Objektový režim (Object mode) umožňuje práci s jednotlivými objekty 3D scény na jejich globální úrovni. Objektový režim umožňuje pouze základní transformační operace nad objekty. Dalším vývojovým režimem v Blenderu je Editační režim (Edit mode). Ten obsahuje řadu vlastních funkcí, které jsou určeny zejména pro krabicové a polygonální modelování. Editační režim tak zajišťuje přímý přístup k síti objektu.

Objekt stolu byl vytvořen pomocí krabicového modelování. V objektovém režimu bylo přidáno objektové primitivum krychle. Vytvořená krychle byla škálována do tvaru ploché desky stolu v editačním režimu. Krychle byla několikrát rozřezána funkcí loop cut tak, aby se řezy nacházely uprostřed desky stolu. Tři čtvrtiny desky byly poté odstraněny.

Blender dále podporuje procedurální modelování, kde procedury jsou označovány jako modifikátory (modifier). Na desku stolu byl aplikován modifikátor Mirror, který zrcadlí objekt dle jedné z os. Díky tomu je možné pracovat pouze se čtvrtinou objektu a mít jistotu, že objekt bude symetrický. Také byl použit modifikátor subdivision surface pro zaoblení všech hran. Aby se hrany stolu příliš nedeformovaly byly přidány další řezy do jejich blízkosti. Nohy stolu byly vytvořeny podobným způsobem.



(a) Rozřezaná deska stolu (b) Efekt modifikátorů

(c) Nohy stolu (d) Kompletní stůl

Obrázek 4.10: Objekt stolu

4.4.2 3D skener

Skener se skládá ze tří součástí, které jsou svým tvarem již komplikovanější. Bylo tak využito polygonální modelování. Jedná se o části podstavce, na kterém je objekt skenován, skenovacího čidla a těla skeneru, které vše spojuje dohromady. Polygonální modelování využívá kromě základních transformačních operací posunu, rotace a škálování také další funkce. Níže je jejich základní výčet, nicméně některé z funkcí mají i své alternativní verze, které mají podobné chování.

- Extrude Region vytvoří kopii vybraných vrcholů, hran nebo polygonů a vysune je určitým směrem.
- Bevel zkosí nebo zaoblí vybrané hrany nebo vrcholy.
- Inset Faces vytvoří nové polygony uvnitř existujících polygonů.
- Loop Cut lze aplikovat pouze na čtyřúhelníky nebo smyčky vytvořené ze čtyřúhelníků. Loop Cut umožňuje rozříznout čtyřúhelník na dvě části.
- Knife je nepřesná verze Loop Cut, která může měnit směr a lze ji aplikovat i na jiné polygony než čtyřúhelníky.

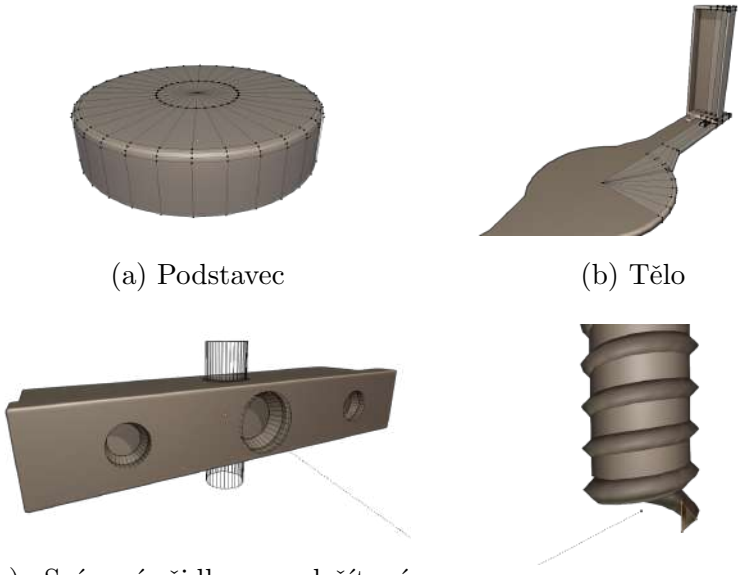
Objekt podstavce byl vytvořen z objektového primitiva kruhu, který byl postupně protahován funkcí Extrude Region v editačním režimu. Podstavec byl vytvořen jako samostatný objekt, protože je tak bude v budoucnu jednodušší animovat.

Hlavní tělo skeneru bylo vytvořeno zejména za užití modifikátoru Mirror pro zrcadlení objektu podle jeho X a Y os. Pro snížení množství polygonů a zároveň snížení složitosti sítě bylo tělo dále rozděleno na podstavu, na které vše stojí a na skříňku, ve které bude později skenovací čidlo. Na celé tělo byl aplikován modifikátor Subdivision Surface, který zaoblil všechny hrany. Proto byly přidány další smyčky hran pomocí funkce Loop Cut, aby hrany nebyly až příliš zaoblené.

Samotné skenovací čidlo využívá již složitější modelovací metody. Hrubý model je opět vytvořen pomocí polygonálního modelování, kde je modelována pouze čtvrtina objektu a zbytek je zrcadlen modifikátorem Mirror na osách X a Z. Na čidlo je dále aplikován modifikátor Boolean, který provádí operaci rozdílu. Jsou odečítány celkově čtyři válcovité objekty, které vytváří prostor pro čočku skenovacího čidla a zároveň pro šroub, po kterém čidlo bude přejíždět.

Posledním objektem, který tvoří tělo 3D skeneru je šroub, po kterém bude snímací čidlo přejíždět nahoru a dolu. Závit šroubu je vytvořen z jednoho trojúhelníku, který je protažen

modifikátorem Screw. Modifikátorem Screw určuje úhel protočení na 360 stupňů, počet závitů na 85 a protažení celého závitu na 2 délkové jednotky. Povrch šroubu pod závitem je pouze protažené objektové primitivum válce.



(a) Podstavec

(b) Tělo

(c) Snímací čidlo a odečítané válce

(d) Závit šroubu

Obrázek 4.11: Objekt 3D skeneru

4.4.3 Notebook

Objekt notebooku byl vytvořen opět pomocí polygonálního modelování. To bylo použito zejména kvůli mnoha křivým povrchům. Výsledný model obsahuje značné množství n-gonů, ale nejedná se o zásadní problém, protože všechny n-gony jsou plně ploché a notebook nebude nijak animován.

Pro tvorbu klávesnice notebooku byla vytvořena jedna jediná klávesa, která byla postupně kopírována po povrchu notebooku. Pro snížení časové náročnosti byl opakovaně použit modifikátor Array. Modifikátor Array vytváří řadu objektů, které jsou identické prvnímu objektu, jedná se tak o nové instance původního objektu. U modifikátoru lze určit vzdálenost mezi jednotlivými instance objektu nebo jejich počet.

Symboly a text umístěn na povrchu některých kláves byl vytvořen pomocí objektového primitiva Text, které je tvořeno z křivek. Objekt Text má podobné vlastnosti jako u

programů pro tvorbu počítačové 2D grafiky. Jedná se tak o dvojrozměrný objekt. Kromě textu, který je zobrazen, lze definovat velikost textu, jeho font nebo zarovnání. Blender dále umožňuje vytáhnout text a dát mu tak hloubku, nebo konvertovat text z křivky na objekt tvořený z sítě polygonů.

Obrázek 4.12: Objekt notebook

4.4.4 Snímací zařízení

Objekt snímacího zařízení je složen z mnoha samostatných objektů, které mají mezi sebou hierarchické vazby. Snímací zařízení lze rozdělit na dva logické celky. Samotné snímací zařízení a stativ, na kterém zařízení stojí.

Stativ opět využívá polygonálního modelování a skládá se z řady samostatných objektů. Průběh modelování je podobný jako u již popsaných objektů, proto nebude podrobně popsán. Za zmínku ovšem stojí objekty nožiček stativu. Byla modelována pouze jedna nožička, na kterou byl aplikován modifikátor Array. Modifikátor Array vytváří dvě nové instance nožičky, ale nožičky nejsou posunuty o nějakou specifickou vzdálenost jako tomu bylo u kláves notebooku. Místo toho jsou nové nožičky posunuty podle transformačních vlastností dalšího objektu, takzvaného objektu Empty. Objekt Empty uchovává své základní transformační informace (svoji lokaci, rotaci a škálování) ale není vykreslován. Pokud je například objekt Empty otočen, vyvolá to změnu na zmíněném modifikátoru Array, a nové instance nožiček se taktéž otočí. Objekt Empty tak byl otočen o 120 stupňů na ose Z, což vyvolalo protočení nožiček stativu okolo hlavního těla stativu.

Samotné snímací zařízení bylo vytvořeno zejména pomocí procedurálního modelování, tedy využití modifikátorů. Základní podoba zařízení byla modelována krabicovým modelováním do velmi triviální podoby. Následně byly aplikovány tři modifikátory Bevel. Dva z nich zaoblují siluetu objektu zatímco třetí zkosí vnější hrany. Posledním použitým modifikátorem byl modifikátor Boolean, který provádí operaci rozdílu. Je odečítána řada kvádrů, čímž lze docílit řady zubů po obvodu zařízení pro lepší uchopitelnost zařízení.

Odečítané kvádry byly taktéž vytvořeny procedurálně. Výchozím objektem je jeden jediný kvádr. Na ten byl aplikován modifikátor Bevel pro zaoblení všech hran kvádru, dále modifikátor Array vytvářející řadu dohromady sedmdesáti kvádrů a poté modifikátor Curve. Modifikátor Curve zakřivil celou řadu kvádrů tak, aby následovala obyčejnou křivku, jejíž tvar byl převzat ze siluety snímacího zařízení. Posledním použitým modifikátorem byl Mirror, který zrcadlil kvádry na obou stranách snímacího zařízení.

Na snímací zařízení bylo posléze přidáno několik dalších objektů, které reprezentují displej, klávesy a objektiv. Paprsek, který bude snímat objekt na pódiu byl vytvořen jako plocha vycházející z objektivu snímacího zařízení.

4.4.5 Škorpión

Pro tvorbu objektu škorpióna byla použita metoda Sculpting, kterou Blender podporuje bohatou zásobou funkcí. Hlavním důvodem pro užití této metody jsou organické tvary škorpióna, které lze jenom těžko vyjádřit pomocí jiných modelovacích metod. Blender obsahuje řadu štětců, které mohou přidávat, odstraňovat nebo transformovat polygony na modelu. Štětce mají zároveň nastavení velikosti, síly působení, směr působení, použité textury nebo nastavení tahu. Lze také využít nástroje symetrie pro vytváření modelů dle os x, y nebo z. Dalším důležitým nástrojem je dyntopo. Tento nástroj umožňuje dynamicky měnit hustotu polygonů v síti objektu. Z pohledu uživatele dyntopo umožňuje vytvářet detaily objektů s vyšší hustotou polygonů a zachovávat nižší hustotu polygonů na zbytku objektu. To zajišťuje úsporu polygonů.

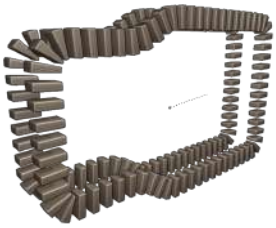
Škorpión byl rozdělen do tří objektů a to do těla včetně hlavy a ocasu, jedné nohy a jednoho klepeta. Toto rozdělení bylo provedeno z důvodu zjednodušení procesu modelování, kdy bylo vytvářeno pouze jedno klepeto a pouze jedna noha, které budou později duplikovány. Během procesu modelování byly opakovaně využívány všechny získané podklady pro vytvoření realistických tvarů škorpióna. Níže je výčet nejčastěji užívaných štětců:

(a) Stativ zařízení

(b) Výchozí model zařízení

(c) Modifikátor Bevel

(d) Výchozí model odečítané krychle



(e) Řada odečítaných krychlí

(f) Výsledné snímací zařízení

Obrázek 4.13: Objekt snímacího zařízení

- Štětce Draw a Draw Sharp vytvářejí nové polygony na objektu.
- Štětec Clay aplikuje nízkou vrstvu nových polygonů na objekt.
- Štětce Inflate a Blob nafukují existující polygony.
- Štětce Layer a Crease vytvářejí ostrou změnu povrchu přídáním respektive ubráním polygonů.
- Štětce Smooth a Flatten vyhlazují respektive zplošťují nerovný povrch.
- Štětec Grab přesouvá polygony.
- Štětec Snake Hook umožňuje chytit existující polygony a protáhnout je určitým směrem.

Během celého modelování škorpióna byl aktivní režim dyntopo. Nástroj byl nastaven na konstantní detail bez ohledu na vzdálenost kamery od objektu, jehož velikost byla ručně nastavována v průběhu modelování. V místech větších prvků objektu, například plátů na horní části těla škorpióna byl použit střední detail 50 px. V místech menších prvků

objektu, například různé hrany a dutiny, byl použit vyšší detail 100 px. Při modelování těla byla zároveň použita symetrie osy X pro modelování pouze jedné poloviny těla. Po vytvoření objektů nohy a klepeta byly tyto objekty duplikovány a zrcadleny. Škorpión tak má osm identických noh a dvě identická klepeta.

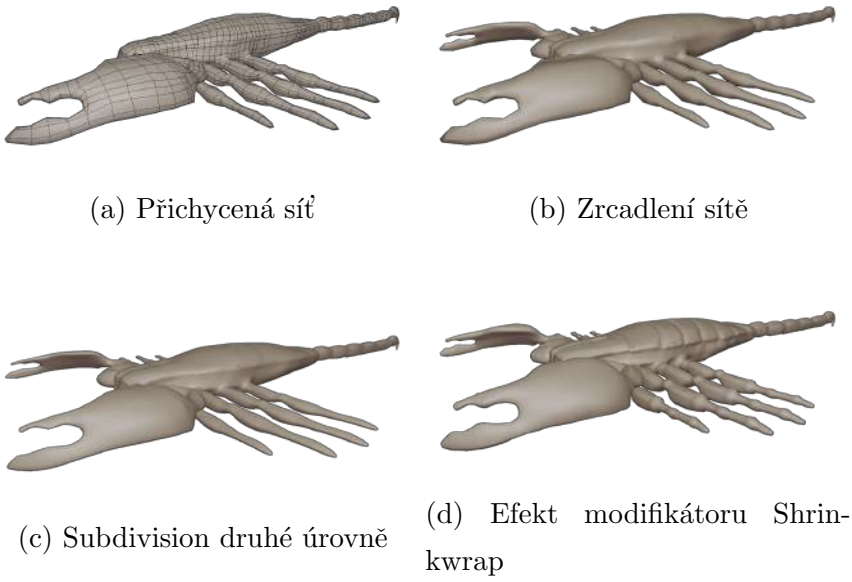
Obrázek 4.14: Objekt škorpióna

Výsledný objekt škorpióna byl vytvořen z více než jednoho miliónu polygonů. Ačkoliv všechny interní renderery Blenderu jsou schopny vykreslovat tak složité objekty, jejich animace může být problematická. Výpočet posunu každého polygonu při užití kosterní animace je výpočetně náročná záležitost, tudíž je vhodné optimalizovat síť škorpióna pro zrychlení procesu vykreslování. K tomu byla použita metoda Retopology.

Pro metodu retopology bylo použito objektové primitivum plocha. Metoda v Blenderu využívá funkci přichycení. Jedná se o funkci, která umožní přichytit vybraný objekt, vrchol, hranu nebo stranu k nějakému jinému prvku. Tato funkce byla použita pro přesun jednotlivých vrcholů objektu plocha a jejich přichycení k povrchu objektu škorpióna. Dále pomocí funkce extrude byla vytvořena síť, která obaluje celé tělo škorpióna. Důraz zde byl kladen na použití výhradně čtyřúhelníků a na hustotu polygonů. Například klouby okolo končetin škorpióna mají vyšší hustotu polygonů, protože se jejich povrch bude v animaci natahovat.

Spolu s tím byly využity modifikátory Mirror pro zrcadlení sítě podle osy X, Subdivision surface pro vyšší míru detailu a Shrinkwrap. Modifikátor Shrinkwrap se snaží obalit jeden objekt druhým přesně tak, jako to bylo popsáno v předchozím odstavci. Rozdíl je v tom, že modifikátor Shrinkwrap je strojový a umožňuje přidat detaily objektu škorpióna po modifikátoru Subdivision surface, který přidává další polygony. Výsledný objekt škorpióna vypadá velice podobně, jako jeho původní verze, ale s tím rozdílem, že nový objekt škorpióna se skládá pouze z devadesáti tisíc polygonů.

Devadesát tisíc polygonů je přesto stále mnoho. Je to způsobeno tím, že modifikátor Subdivision surface byl nastaven na druhou úroveň. Kdyby byl nastaven pouze na první



(a) Přichycená síť

(b) Zrcadlení sítě

(c) Subdivision druhé úrovně

(d) Efekt modifikátoru Shrinkwrap

Obrázek 4.15: Objekt škorpióna po retopology

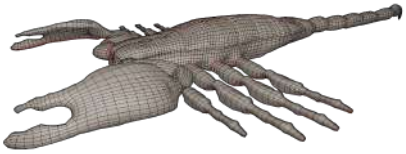
úroveň, činilo by množství polygonů pouhých dvacet dva tisíc. Tím se ale snižuje kvalita detailů. Tuto limitaci lze obejít vytvořením normální mapy. Výše zmíněné modifikátory byly aplikovány, tudíž je již nelze upravit. Jedná se o modifikátory Mirror, Subdivision Surface první úrovně a Shrinkwrap.

Pro tvorbu normální mapy musí být škorpión rozbalen. Objektu byly v editačním režimu označeny některé hrany, které reprezentují místo pro řez. Poté byla použita funkce unwrap, která objekt rozřezala a jeho polygony rozložila do prostoru UV. Pro upečení normální mapy je dále potřeba původní objekt škorpióna tak, jak byl vytvořen metodou Sculpting. Jedná se tak o ten objekt, který se skládá z více než miliónu polygonů. Tento původní objekt lze označit jako high-poly objekt, zatímco objekt po retopology úpravě lze označit za low-poly objekt. Tyto dva objekty se musí překrývat a až poté je možné spustit funkci Texture Bake.

Texture Bake umožňuje takzvaně upéct textury objektů přímo na jejich UV mapu. To probíhá pomocí speciálního ray tracing algoritmu, tudíž je potřeba využít rendereru Cycles. V tomto konkrétním případě je tak upečena normální mapa jejíž rozlišení je $2048 \cdot 2048$. Low-poly objekt je rovnoměrně rozšířen do všech stran o $0,5$ délkových jednotek, čímž lze zajistit, že low-poly objekt bude překrývat high-poly objekt. Z každého texelu

normální mapy (která je mapována na povrch low-poly objektu) je vyslán paprsek o tuto délkovou jednotku, který zkoumá, kde se nachází nejbližší polygon high-poly objektu.

Výsledkem je tak low-poly objekt škorpióna, jehož množství polygonů je dvacet dva tisíc a i přesto přenáší srovnatelnou vizuální hodnotu, jako high-poly objekt složený z jednoho miliónu polygonů. Pro srovnání byly obě verze škorpióna vykresleny samostatně v rendereru Eevee. High-poly verze se vykreslila za osm sekund, zatímco low-poly verze se vykreslila za méně než jednu sekundu.



(a) Vyznačené hrany pro řez

(b) Původní sculpt a nový retopo se překrývají

(c) Upečená normální mapa

(d) Výsledný objekt škorpióna

Obrázek 4.16: Objekt škorpióna s normální mapou

4.4.6 Hologram

Tvorba hologramu se skládá ze dvou dílčích částí. Tvorba samotného hologramu a tvorba projektoru, který bude hologram zobrazovat. Projektor byl vytvořen velmi podobně, jako

pódium 3D skeneru. Jedná se pouze o objektové primitivum kruhu, které bylo protahováno funkcí Extrude Region.

Samotný hologram má reprezentovat skenovaný objekt, tedy škorpióna. Proto byl objekt škorpióna duplikován a práce probíhaly na této kopii. Škorpión byl upraven pomocí modifikátorů Decimate, Wireframe a Build. Modifikátor Decimate spojuje blízké vrcholy sítě do jednoho a snižuje tak množství polygonů. Výsledná síť nemá rovnoměrnou hustotu a skládá se z trojúhelníků, které tak vytvářejí chaotický efekt. Modifikátor Wireframe vytváří efekt sítě na povrchu škorpióna. Na místo každé hrany původního objektu je vytvořen válec. Válce jsou mezi sebou spojeny v místech, kde se nacházely vrcholy původního objektu. Původní objekt je odstraněn a místo něj je vykreslena síť tvořená z mnoha válců. Modifikátor Build se používá pro animaci a vytváří efekt postupného vykreslení objektu.

Obrázek 4.17: Hologram s projektorem

4.4.7 Pozadí scény a skládání scény

Pro pozadí celé scény byly tvořeny jednoduché objekty. Podlaha místnosti je objektové primitivum plochy, které je škálováno aby zabíralo větší prostor. Okolo podlahy byly vytvořeny kvádry reprezentující podlahové lišty.

Byly dále vytvořeny dvě cihlové zdi. Cihly byly narychlo vytvořeny tak, aby měly na první pohled náhodný tvar.

Základem celé stěny je jedna jediná cihla. Ta byla vytvořena z objektového primitiva krychle, které bylo škálováno do podoby kvádru. Na kvádr byl aplikován Subdivision Surface modifikátor, ale nebyl využit aproximační Catmull-Clark algoritmus. Místo toho bylo použito jednoduché interpolační dělení. Na kvádr byl následně použit modifikátor Displacement s nízkou silou. Ten transformuje jednotlivé vrcholy sítě objektu podle nějaké

vstupní textury. Ta byla vytvořena z textury náhodného šumu. Výsledkem je tak kvádr, jehož povrch je nerovný.

Kvádr byl pětkrát duplikován a v režimu Sculpting byly přidány drobné detaily na každý kvádr. Jednalo se o promáčkliny, škrábance nebo zaoblené hrany. Síť každého z pěti kvádrů byla zjednodušena modifikátorem Decimate tak, aby měly něco přes jedno sto polygonů. Z kvádrů byla poté vytvořena stěna jejich náhodným výběrem a jejich duplikací.

Všechny vytvořené objekty byly sestaveny do jedné scény a jejich velikost upravena tak, aby k sobě pasovaly.

(a) Vytvořené varianty cihel (b) Místnost (c) Všechny vymodelované objekty

Obrázek 4.18: Sestavená scéna

4.5 Materiály

Blender umožňuje tvorbu různorodých materiálů prostřednictvím řady interních shaderů i podpory jak klasických textur v podobě obrázků, tak i procedurálních textur. Celý proces tvorby materiálů probíhá v prostředí Shader Editor. Jedná se o dvojrozměrnou plochu, na kterou uživatel vkládá uzly, jejichž parametry následně upravuje dle potřeby. Uživatel tyto uzly navzájem propojuje prostřednictvím jejich vstupních a výstupních hodnot a předává jejich kolektivní informaci koncovému uzlu. Koncový uzel zpracuje všechny své vstupní informace a vykreslí materiál na objekt. Celý postup je podobný vizuálnímu programování. Vložené uzly reprezentují shadery, textury, metody mapování textur na objekty nebo numerické operace.

Součástí Blenderu je řada jednoduchých shaderů, jako je například Diffuse shader nebo Glossy shader.

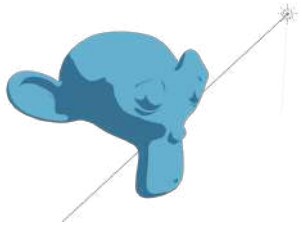
Blender dále podporuje importování existujících textur v podobě rastrových obrázků a jejich využití pro různé účely. Například lze importovat textury základní barvy a normální mapy dřeva a připojit je k diffuse shaderu k vytvoření materiálu dřeva. Je zde také k nalezení sada interních procedurálních textur. Jedná se například o textury cihel, přechodu nebo náhodného šumu.

Eevee bohužel neobsahuje žádný cel shader, který by byl použitelný pro ne-fotorealistické zobrazování. Proto je potřeba shader nejdříve vytvořit. Takový shader musí být univerzální, tj. aplikovatelný na jakýkoliv objekt a zároveň musí být snadné jej upravit.

4.5.1 Cel shader

Základem cel shaderu je diffuse shader, který obsahuje barevnou informaci o povrchu objektu a informaci o přicházejícím světle. Užitím uzlu Shader to RGB je možné tyto informace izolovat a pracovat s nimi samostatně. Pomocí uzlu Color Ramp je možné měnit barevnou informaci povrchu objektu, která je poté předána obyčejnému Emission shaderu, který již informaci vykreslí na povrch objektu.

(a) Použité uzly

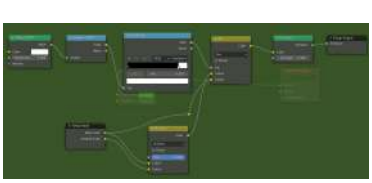


(b) Vykreslený shader

Obrázek 4.19: Nejjednodušší implementace cel shaderu

Výše prezentovaný postup ale představuje dva problémy. Za prvé, rozšíření shaderu vyžaduje implementaci dalších uzlů, které mohou později značně znepřehlednit celou plochu. Za druhé, v souvislosti s prvním bodem, jakákoliv úprava shaderu může vyžadovat rozsáhlé změny napříč všemi uzly. Řešením je vytvoření skupiny uzlů, která bude reprezentovat cel shader. Skupině pak lze definovat vstupní a výstupní údaje. Takový postup má ještě jednu výhodu. Vytvořenou skupinu uzlů lze považovat za třídu a její vstupní údaje za parametry konstruktoru třídy. Každý objekt tak může využívat stejnou třídu, ale každý objekt pracuje se svojí vlastní instancí této třídy. Taková úprava také vyžaduje drobnou

úpravu v logice shaderu. Uzel Color Ramp nemá vstupní parametry, kterými by šlo upravovat barvu. Proto je vhodné používat uzel Color Ramp spíše jako černobílou masku a barvy objektu aplikovat v následujícím uzlu Mix RGB. Černobílý výstup Color Ramp je tak využit jako faktor pro míchání dvou barev. První barva reprezentuje tu část objektu ve stínu, která byla spočtena z násobení základní barvy (base color, například modrá) a barvy stínu (shadow color, například tmavě šedá). Druhou barvou je pak základní barva, která je aplikována na osvětlenou část objektu.



(a) Vnitřní struktura skupiny uzlů

(b) Globální pohled na skupinu uzlů

(c) Vykreslený shader na několika objektech

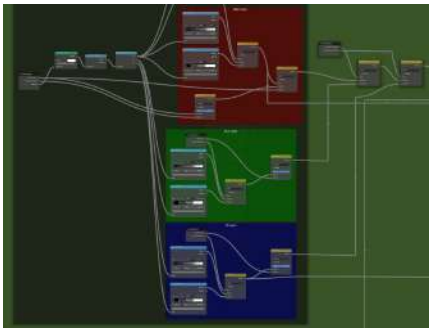
Obrázek 4.20: Rozšíření cel shaderu o skupiny

V tuto chvíli je cel shader plně funkční a je snadné jej rozšiřovat. K lepší přehlednosti je také možné využít panely (frame), které mohou dělit uzly do logických celků.

U přicházejícího světla je zkoumán směr, odkud světlo přichází a jeho intenzita. Nicméně barva světla je cel shaderem přehlížena. Pokud by se ve scéně mělo nacházet více světelných zdrojů, mohlo by to způsobit problémy. Například žluté sluneční světlo v kombinaci s modrým světlem přicházejícím z přilehlé lampy by cel shader vykresloval identicky. Řešením je izolace jednotlivých kanálů RGB přicházejícího světla a práce s každým kanálem samostatně. To vyžaduje aby přicházející světlo bylo plně červené, zelené nebo modré. Červené světlo je tak určeno jako hlavní světlo a na obrázku níže je vidět jeho efekt jak dopadá ze shora na tři objekty. Červené světlo pracuje s základní barvou objektu (base color). Zelená a modrá světla jsou dodatečná světla, která mohou osvítit objekt jinou barvou. Na obrázku níže je zelené světlo, které osvětluje objekty ze zdola. Zelené a modré světlo pracuje s doplňkovou a zadní barvou respektive (fill color a back color).

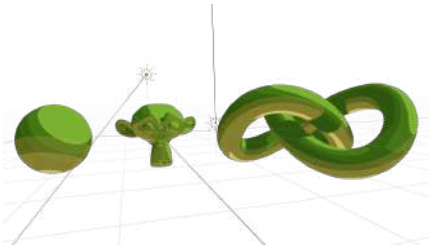
Předchozí sada obrázků prezentovala tónování s ostrým přechodem mezi světlem a stínem. To je způsobeno černobílým uzlem Color Ramp. Jeho rozšířením o odstíny šedé je možné přidat více tónů pro stín. Jemného tónování lze dosáhnout využitím uzlu Color Ramp s lineárním přechodem. Lze tak vytvořit dvě verze tónování ze dvou různých

Color Ramp uzlů pro všechny tři barevné kanály. Přepínání mezi jedním nebo druhým tónováním lze dosáhnout pomocí Mix RGB uzlu.

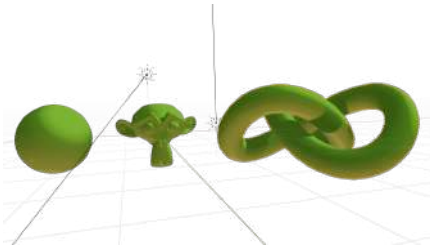


(a) Vnitřní struktura skupiny uzlů

(b) Globální pohled na skupinu uzlů



(c) Osvětlení objektů novým světlem a ostré přechody tónů stínů

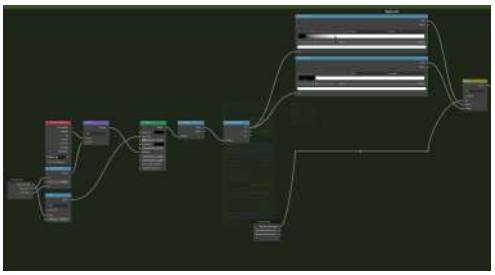


(d) Osvětlení objektů novým světlem a jemné přechody tónů stínů

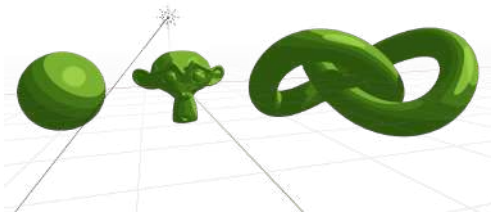
Obrázek 4.21: Izolace kanálů RGB a změna stylu tónování

Pro tvorbu specular odrazu je možné využít Specular shader na který je napojen uzel Shader to RGB. Vykreslení specular odrazu tak funguje stejně, jako u výše popsaného diffuse shaderu, akorát se pracuje pouze s hlavním světlem, které je na červeném kanále. Specular shader je navíc rozšířen o možnost posunu specular odrazu po jeho ose X a Z. Toho je docíleno mapováním specular odrazu na normálu povrchu objektu a k získaným souřadnicím přičítat uživatelem definované souřadnice.

Šrafování lze docílit využitím nějaké textury jako vzorku. V tomto konkrétním případě bylo vytvořeno šrafování, které plně nahrazuje specular odraz objektu za řadu puntíků, jejichž velikost klesá s vzdáleností od středu specular odrazu. Jedná se o plně procedurální texturu. Vzorek byl vytvořen kombinací Gradient Texture a Voronoi Texture uzlů. Taková textura je aplikována na objekt z pohledu kamery, díky čemuž je jistota, že puntíky budou mít vždy stejný tvar a vždy budou směřovat ke kameře. K aplikaci vzorku na specular



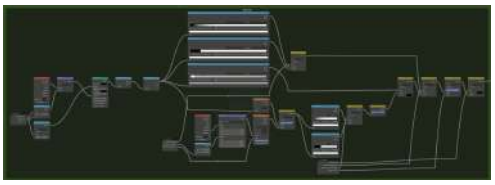
(a) Vnitřní struktura skupiny uzlů



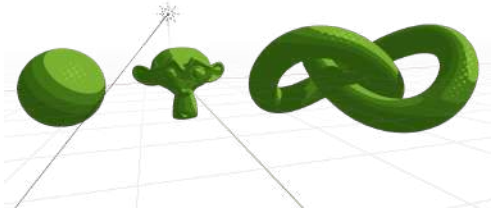
(b) Vykreslený shader na několika objektech

Obrázek 4.22: Rozšíření cel shaderu o specular odraz

odraz je použit nový uzel Color Ramp, který vyžívá odstíny šedé, aby se výsledný vzorek aplikoval ve správné velikosti.



(a) Vnitřní struktura skupiny uzlů

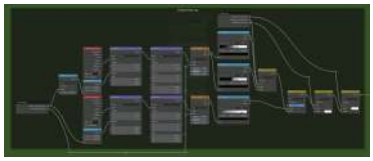


(b) Vykreslený shader na několika objektech

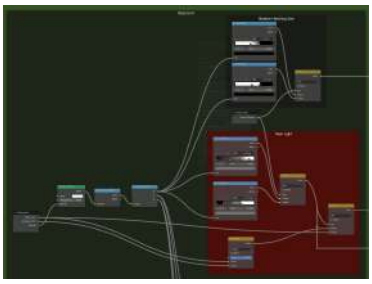
Obrázek 4.23: Rozšíření cel shaderu o specular šrafování

Šrafování lze také aplikovat do stínů objektu. Pro získání místa, kde bude šrafování aplikováno, je použito hlavní červené světlo dopadající na objekt. Pro to je vytvořen nový uzel Color Ramp, který využívá červeného kanálu. Samotný vzorek je pak vytvořen ze dvou uzlů Noise Texture, které vytváří nerovnoměrné čáry po povrchu objektu. Obě textury šumu jsou škálovány po své ose X dvacetkrát aby se protáhly a vytvořili efekt čar a stejně jako šrafování specular odrazu, i zde je textura aplikována z pohledu kamery.

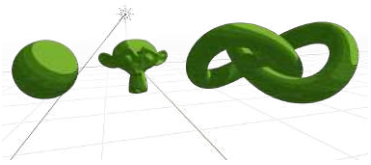
Poslední součástí shaderu je podpora ambient occlusion. Ambient occlusion se obvykle nepoužívá u ne-fotorealistického zobrazování, ale je možné jej přidat. Ambient occlusion lze kdykoliv vypnout v nastavení rendereru Eevee. Pro ovládání velikosti ambient occlusion lze umocnit barvu generovanou tímto efektem. Lze také ambient occlusion kompletně



(a) Vnitřní struktura skupiny uzlů



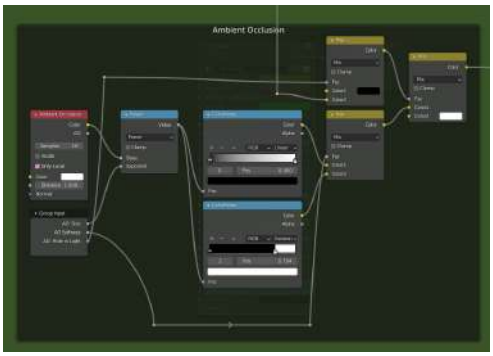
(b) Nové využití červeného kanálu diffuse shaderu



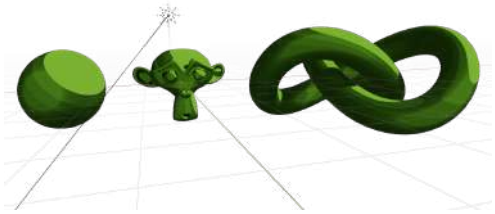
(c) Vykreslený shader na několika objektech

Obrázek 4.24: Rozšíření cel shaderu o šrafování stínů

odstranit z místa kde dopadá světlo na červeném kanálu. Toho je docíleno využití stínu vytvořeného z červeného světla na diffuse shaderu a jeho míchání s plně černou barvou. Výsledná informace je považována za černobílou masku, kterou je rozhodnuto, zda-li bude ambient occlusion použito nebo ne.



(a) Vnitřní struktura skupiny uzlů



(b) Vykreslený shader na několika objektech

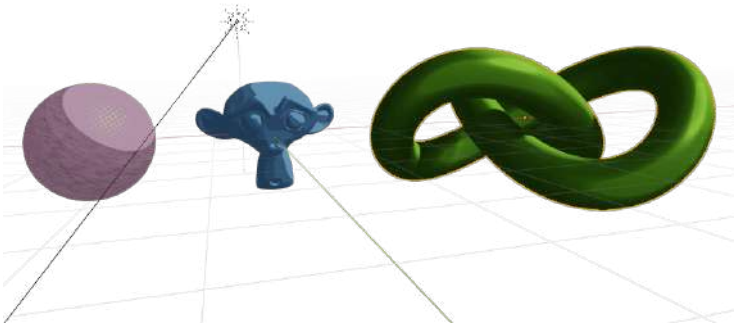
Obrázek 4.25: Rozšíření cel shaderu o podporu ambient occlusion

Součástí shaderu je také řada uzlů Mix RGB a Math, které nebyly popsány dopodrobna. Tyto uzly fungují jako kontrolní uzly, neboť mají přístupné vstupní parametry. Na ty je napojován uzel Group input, který odhaluje tyto parametry. Shader tak lze konfigurovat bez nutnosti vstupovat do jeho složité vnitřní struktury.

Cel shader lze využít pro jakékoliv objekty. Další výhodou je možnost napojení textur do shaderu, ačkoliv jediné dvě použitelné textury, které mají význam napojovat do shaderu jsou textury diffuse a normální mapy.

(a) Hotová logika cel shaderu

(b) Vnější pohled na nastavení cel shaderu



(c) Vykreslený cel shader na několika objektech v různých variantách

Obrázek 4.26: Hotový cel shader

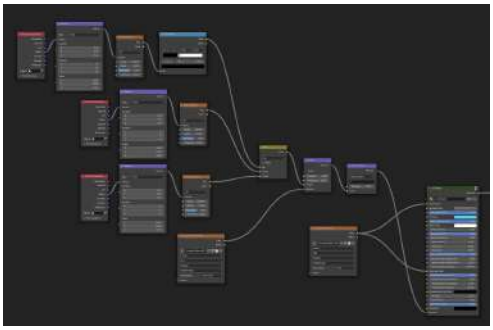
4.5.2 Ruční kreslení textur

Blender dále umožňuje nanášení barev přímo na 3D objekt prostřednictvím různých štětců. K tomu se používá režim Texture Paint, ve kterém jsou tahy štětce po povrchu 3D objektu automaticky zpětně mapovány na rastrový obrázek textury. V režimu Texture paint si lze nastavit velikost štětce, jeho intenzitu, barvu štětce, režim míchání barev nebo režim nanášeného vzorku. Pro použití tohoto postupu je podmínkou mít daný objekt správně rozbalený v UV prostoru a mít vytvořenou a připojenou texturu v Shader Editor.

Výhodou tohoto postupu je to, že uživatel nanáší barvy na povrch trojrozměrného objektu a tato informace se sama od sebe ukládá na rastrový obrázek textury. To umožňuje uživateli se soustředit pouze na samotný 3D objekt a nemusí se starat o UV mapu objektu.

Opačný způsob kreslení textur, tedy kreslení na rastrový obrázek, který je mapován na trojrozměrný objekt, je také možný. Nevýhodou takového postupu je ale obtížná reprezentace hloubky na rastrovém obrázku. Jistou oporou v tu chvíli může být vykreslení UV mapy na texturu před tím, než se na ni začne kreslit. I to ale není ideální a je mnohdy lepší kreslit přímo na 3D objekt.

Tímto způsobem byla vytvořena textura base color pro škorpióna. Jedná se například o zvýraznění přechodu mezi kůží a krunýřem, změna barvy krunýře nebo výrazné zbarvení ostnu na konci ocasu. Výsledná textura je spolu s upečenou normální mapou připojena k cel shaderu. Normální mapa byla dále rozšířena o tři textury náhodného šumu, které vytvářejí drobnou nerovnost celého povrchu těla škorpióna. Jakékoliv další detaily jsou vykresleny právě pomocí cel shaderu, zejména pak šrafováním.



(a) Vytvořené textury v shader editoru

(b) Vykreslený škorpión se všemi texturami

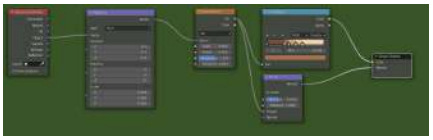
Obrázek 4.27: Vytvořené textury pro škorpióna

4.5.3 Procedurální textury

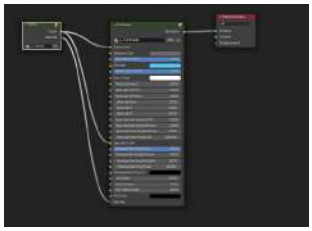
Procedurální textury již byly použity pro tvorbu šrafování pro cel shader. V Blenderu lze vytvořit i složitější textury, které vytváří komplexní povrchovou úpravu pro objekty. Je ale vhodné poznamenat, že kvalita takových procedurálních textur se nikdy nevyrovná kvalitě textur pocházejících z některých jiných programů. Procedurální textury v Blenderu mnohdy nedosahují fotorealistické kvality, ale pro účely ne-fotorealistické animace to

nevadí. Takové textury lze také napojit do cel shaderu, nebo jakéhokoliv jiného shaderu. Pro tuto diplomovou práci bylo vytvořeno několik procedurálních textur.

Pro stůl na kterém stojí všechny modely byla vytvořena procedurální textura dřeva. Taková textura je velice triviální a je založena na jedné textuře náhodného šumu, která je škálována tak, aby byla protažena. Textura je napojena na uzel Color Ramp, který určuje barvy dřeva a zároveň jejich rozložení. Posunutím jedné z ručiček v uzlu Color Ramp lze zvětšit prostor, který barva zabírá. Dále je na texturu šumu připojen uzel Bump, který dokáže vytvořit nerovnostní bump mapu. Celá textura je napojena na cel shader.



(a) Vnitřní struktura textury



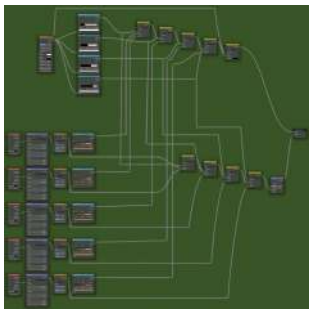
(b) Napojení textury na cel shader

(c) Vykreslená textura

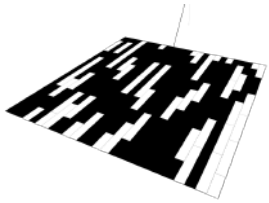
Obrázek 4.28: Procedurální textura dřeva

Podlaha místnost využívá podobnou procedurální texturu, která vypadá jako plovoucí podlaha. Jelikož každé prkno v plovoucí podlaze má jiný vzor, tak je použito pět různých vzorků pro dřevo. Každý z vzorků funguje na stejném principu, jako dřevěná plocha stolu v předchozím odstavci. Rozdíly ve vzorcích jsou vytvořeny posunutím každého vzorku o náhodnou hodnotu po ose X a Y v mapping uzlu. Distribuce vzorků po podlaze je zajištěna uzlem Brick Texture. Napojením uzlu Color Ramp na Brick Texture a přepnutím na konstantní přechod je možné izolovat náhodné cihly v textuře. To je čtyřikrát opakováno pro získání čtyř náhodných skupin cihel. Každé skupině je přiřazen jeden ze vzorků dřeva.

Povrch projektoru hologramu využívá uzel Gradient Texture, který vytváří kruhový přechod jehož střed je uprostřed plochy holoprojektoru. Přechod je použit pro distribuci prvků po povrchu holoprojektoru za pomocí uzlu Color Ramp, který vytváří červené, modré a zelené kruhy. Opět jsou tak izolovány jednotlivé barevné kanály RGB, na které jsou aplikovány prvky textury. Řada modrých kroužků je vytvořena pomocí uzlu Wave Texture nastaveného na tvorbu kruhů. Textura je dále protnuta čtyřmi pruhy. Každý z pruhů opět využívá uzel Wave Texture, ale pomocí uzlu Color Ramp je tloušťka každého pruhu snížena na minimum. Každý pruh je také otočen pomocí Mapping uzlu.



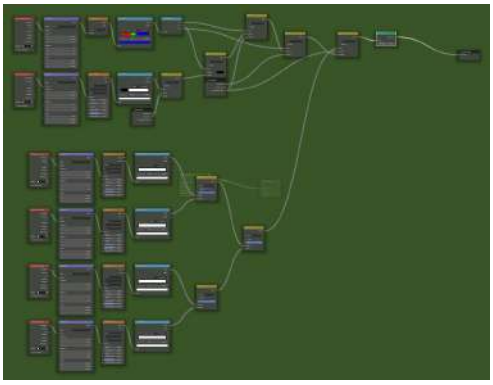
(a) Vnitřní struktura textury



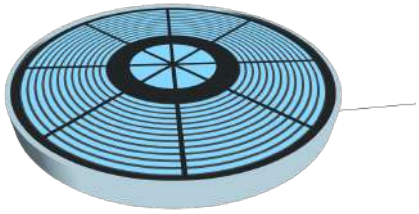
(b) Náhodně vybrané cihly, které budou mít stejný vzorek

(c) Vykreslená textura

Obrázek 4.29: Procedurální textura plovoucí podlahy



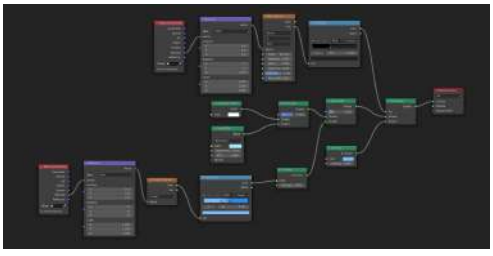
(a) Vnitřní struktura textury



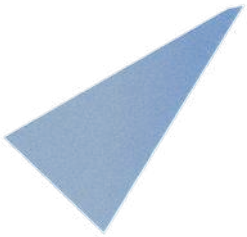
(b) Vykreslená textura

Obrázek 4.30: Procedurální textura holoprojektoru

Poslední procedurální textura byla vytvořena pro snímací plochu. Ta využívá uzly Gradient texture a Wave Texture pro změnu barvy povrchu snímací plochy.



(a) Vnitřní struktura shaderu



(b) Efekt shaderu

Obrázek 4.31: Snímací plocha pro skenování

4.6 Animace

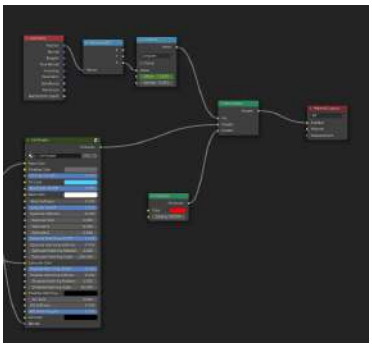
Blender podporuje tvorbu animací prostřednictvím vytváření *keyframe* (klíčových snímků) na vybraných proměnných v určitém čase. Názorným příkladem může být zachycení globálních souřadnic $\{0; 0; 0\}$ objektu krychle do klíčového snímku na snímku 1, přičtení deseti jednotek k souřadnici osy x objektu krychle a druhé zachycení globálních souřadnic $\{10; 0; 0\}$ na snímku 60. Po spuštění animace se krychle posune o 10 jednotek po ose x během prvních šedesáti snímků. Dle výchozího nastavení Blenderu budou snímky 2 až 59 umístěné mezi klíčovými snímky vypočteny pomocí funkce Bézierovy křivky, kterou lze buď upravit, nebo kompletně nahradit jiným druhem interpolace. Další důležitou hodnotou je hodnota snímků za vteřinu. V případě, že by byl výše uvedený příklad vykreslován v 30 snímcích za vteřinu, bude trvat krychli celé dvě sekundy, než se dostane na konec své dráhy. Pokud by bylo video vykreslováno v 60 snímcích za sekundu, bude cesta trvat pouhou vteřinu. Na jakém snímku je vytvořen klíčový snímek je definováno pomocí Timeline editoru. Tento editor umožňuje základní manipulaci s klíčovými snímky, například jejich posouvání v čase. Lze tak například škálovat všechny klíčové snímky najednou o určitou hodnotu, což je vhodné, pokud se mění hodnota snímků za sekundu. V rámci tvořené animace bylo nastaveno vykreslování třiceti snímků za vteřinu a délka videa určena na třicet sekund. Celkový počet snímků tedy činí $30 * 30 = 900$.

Klíčové snímky je možné vytvářet nad většinou proměnných v Blenderu. Lze tak animovat pohyb, rotaci a škálování objektů, míru zkosení hran u modifikátoru Bevel, množství generovaných částic nebo sílu působení normální mapy materiálu. Animace zmíněných hodnot je druhem přímé animace.

Tímto způsobem bylo animováno pódium. Na snímku 130n byl vytvořen klíčový snímek zachycující rotaci pódia $\{0°; 0°; 95, 7°\}$. Pódium bylo následně otočeno o 720° podle osy z a tento nový údaj byl zachycen novým klíčovým snímkem na snímku X v podobě $\{0°; 0°; 816, 7°\}$. Pódium se tedy dvakrát otočí přibližně během dvaceti pěti sekund skutečného času. Zároveň, dle interpolace generované Bézierovou křivkou, bude pódium na začátku své animace pomalu nabírat rychlost než dosáhne maximální rychlosti a na konci své animace bude postupně zpomalovat, dokud se úplně nezastaví. Metodou přímé animace byly dále animovány i další objekty.

Laser přejíždějící přes tělo škorpióna není skutečný zdroj světla, nýbrž pouze součást shaderu. Je zkoumána geometrie objektu škorpióna a je izolována souřadnice Z, po které

lze přesouvat černobílou masku. Ta je použita pro distribuce cel shaderu a červeného Emission shaderu, který funguje jako laserový paprsek.

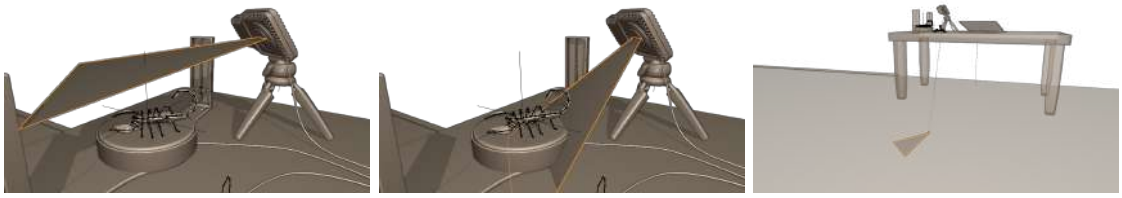


(a) Vnitřní struktura shaderu

(b) Efekt shaderu

Obrázek 4.32: Laser přejíždějící přes škorpióna

Snímací zařízení by mělo provádět skenování škorpióna. K tomu byla vytvořena dříve popsána snímací plocha. Snímací ploše byl přemístěn Origin point, okolo kterého se objekty otáčí. Origin point byl umístěn na místo objektivu snímacího zařízení. Snímací plocha byla následně animována tak, aby se několikrát rychle otočila a prošla skrz pódium a škorpióna. Snímací plocha musí také zmizet, aby nebyla vidět na kameře. K tomu stačí přemístit plochu pod podlahu místnosti.



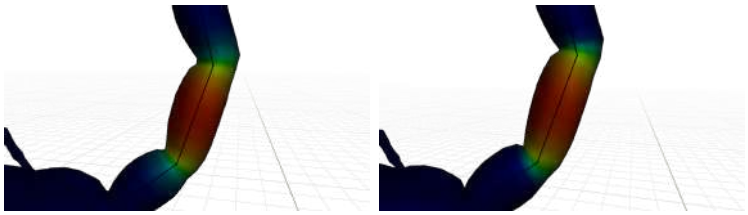
Obrázek 4.33: Snímací plocha přejíždějící přes škorpióna

Hologram byl animován pomocí svých modifikátorů. Modifikátoru Build byl nastaven počáteční snímek jako snímek 650 a trvání modifikátoru na 50 snímků. Modifikátor tak vykreslí škorpióna během padesáti snímků. Byl také animován faktor modifikátoru Decimate. Na snímku 650 je faktor pouhých $0,01$, což znamená, že by se ze škorpióna vykreslila pouze velmi hrubá síť. Na snímku 750 je faktor již $0,1$. V tu chvíli je již hologram vykreslen.

4.6.1 Kosterní animace

V prostředí Blenderu je možné vytvářet kosterní animace za pomocí objektu Armature, který přidá do scény samostatnou kost fungující jako kořenová kost. Další kosti lze přidávat v editačním režimu. Po vytvoření celé kostry je kostra určena jako rodič jiného objektu a při tom jsou vygenerovány skupiny vrcholů spolu s určenými váhami. Tento postup byl použit při animaci škorpióna.

Vytvářené kosti byly umístěny přibližně doprostřed těla škorpióna. Kosti nacházející se uvnitř končetin byly umístěny odděleně od zbytku těla, ale byla zachována dědičnost kostí. Působnost každé kosti na škorpióna je vymezena přidělenou skupinou vrcholů a jejich váhami. Skupiny i váhy jsou automaticky generovány podle relativní vzdálenosti každého vrcholu od kosti. Každý článek těla škorpióna tak má svoji vlastní kost, která by měla ovlivňovat pouze svůj článek. Vygenerované skupiny vrcholů ovšem často zasahují i mimo svůj článek, proto je nutné každou skupinu vrcholů zkontrolovat a případně zmenšit.



(a) Vygenerované váhy skupiny vrcholů

(b) Upravené váhy skupiny vrcholů

Obrázek 4.34: Redukce vah skupiny vrcholů jednoho z článků ocasu

Pro vytvoření animace škorpióna byly animovány jejich jednotlivé kosti pomocí základních transformačních operací. Transformace samostatných kostí se provádí v režimu Pose, kdy se s kostmi pracuje stejným způsobem, jako s jinými objekty.

Z dostupných podkladů lze usoudit, že škorpióni se nehýbou bez důvodu. Škorpióni například nepřešlapují z nohy na nohu a nepohupují ocasem ze strany na stranu jako například někteří savci. Pro demonstraci animací byl tento fakt přehlédnut a byly vytvořeny jednoduché animace, ve kterých škorpión přešlapuje z nohy na nohu, cvaká klepítky, natáčí hlavu a mává ocasem ze strany na stranu. Tyto animace se pouští ve chvíli, kdy škorpión stojí na místě. Dále byla vytvořena animace chůze.

Animace chůze škorpióna byla založena na animaci jedné nohy a kopírování této animace na ostatní nohy. Prvním krokem bylo animování jedné nohy v režimu Pose tak, aby se škorpión zapíral o podlahu. Následovalo kopírování rotace kostí této jedné nohy na všechny ostatní. Dále byla vytvořena knihovna pro uložení transformačních informací vybraných kostí. Póza, ve které škorpión stojí, byla uložena do této knihovny. Je tak možné kdykoliv znovu načíst tuto pózu.

Prvním klíčovým snímkem animace chůze škorpióna byla zachycena rotace levé nohy prvního páru zatímco je noha natočená dopředu. Druhý klíčový snímek se nachází 8 snímků po prvním a zachycuje rotaci nohy, kdy je noha natočena dozadu. Třetí klíčový snímek je zachycen o dalších 8 snímků dále a jedná se o kopii prvního. Třetí klíčový snímek tak zajišťuje, že animace může probíhat opakovaně. Poslední klíčový snímek se nachází mezi druhým a třetím klíčovým snímkem. Rotace nohy tohoto čtvrtého snímku je vypočtena automaticky interpolací sousedních klíčových snímků, ale byla upravena tak, aby to vypadalo, že se noha zvedá. Výsledkem je 16-ti snímková animace ve které se noha škorpióna posune dozadu a poté udělá krok dopředu.

Vytvořená animace levé nohy prvního páru posloužila jako základ pro animaci dalších noh. Pravá noha prvního páru byla animována stejným způsobem s tím rozdílem, že animace byla posunuta o 8 snímků. Animace druhého páru nohou byla také provedena stejným způsobem, ale obě končetiny byly posunuty o dalších 8 snímků. Třetí pár nohou je přesnou kopií prvního páru a čtvrtý pár je přesnou kopií druhého páru. Tak vznikla animace chůze na místě a každé čtyři snímky vytvořené animace byly zaznamenány do knihovny. Namísto obrázků, je v příloze diplomové práce k nahlédnutí krátká animace, demonstrující chůzi škorpióna.

Kostra těla škorpióna byla animována tak, aby se přemístila z místa za notebookem k blízkosti pódia. Jakmile kostra dorazí k pódiu tak zpomalí, zvedne se a nakonec zastaví na pódiu. Do toho byly načítány informace uložené v knihovně póz tak, aby škorpión pohyboval končetinami. Informace obsažené v knihovně ale nestačí k tomu, aby si škorpión vylezl na pódium. V každém snímku, kdy škorpión vylézá na pódium, byly kosti škorpióna upraveny tak, aby končetiny dosahovali na nejbližší povrch a aby škorpión ohýbal své tělo při lezení. To vše bylo doplněno o drobný pohyb klepítek a ocasu škorpióna.

4.7 Kompozice a vykreslení snímků

Kompozice scény je závěrečný proces před vykreslením snímků scény. Jedná se spíše o neformální proces, neboť jeho náplň není pevně daná a může probíhat souběžně s jinými vývojovými fázemi. Tvorba kompozice scény spočívá v umístění všech vytvořených objektů na správné místo, úpravě klíčových snímků v animaci, tvorbě světelných podmínek, úpravě výstupních formátů a nastavení kamery.

Scéna obsahuje dva zdroje světla. Hlavní světlo je plně červené sluneční světlo o síle $5W$. Jedná se o dostačující intenzitu světla pro použitý cel shader. Před obrazovku přivřeného notebooku bylo postaveno další světlo. Jedná se o obdélníkový zdroj světla s zelenou barvou a nízkou silou $10W$. To v kombinaci s cel shaderem vytváří jemné modré světlo na klávesnici notebooku.

Hraniční čáry nebyly v praktické části diplomové práce zatím probírány. Procedurální geometrickou siluetu objektu lze vytvořit v Blenderu užitím modifikátoru Solidify. Jeho šířka by určovala tloušťku hraniční čáry a lze jej nastavit tak, aby otáčel normály ploch a pracoval s materiálem, který vykresluje Emission shader s černou barvou. Metodu stínování normál obrysových hran lze implementovat do cel shaderu. Cel shader by využíval uzel Fresnel a pomocí dalších matematických uzlů by šlo vytvořit nastavitelnou hraniční čáru. Obě metody lze velice snadno implementovat. Z teoretických východisek ovšem vyplývá, že obě metody mají své problémy, proto nebudou použity.

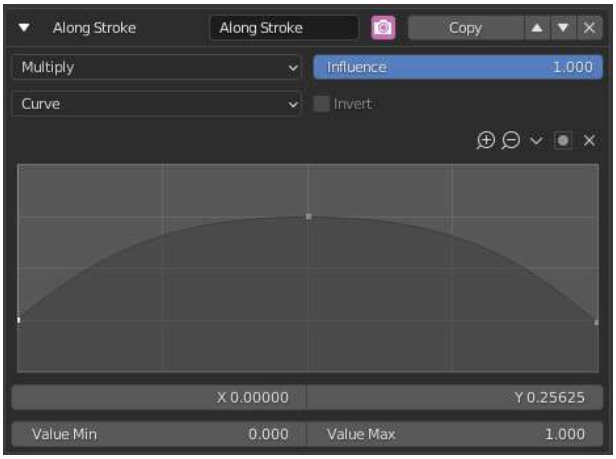
Pro tvorbu hraničních čar bude místo toho použit Freestyle. Jedná se o post-processing efekt dostupný v Blenderu s bohatým nastavením. To znamená, že efekt se spouští až poté, co je snímek vykreslený a kombinuje se s ním. Je také možné vykreslovat Freestyle na samostatné vrstvě, kterou lze dále exportovat do formátu svg. Vykreslované hrany jsou typu Silhouette, Crease, Border, Edge Mark a Material Boundary.

Nastavení Edge Mark umožňuje ručně vybrat některé hrany, které budou mít vždy hraniční čáru. Například objekt notebooku má některé hrany zvýrazněné jak je vidět na obrázku 4.12. Tím lze poznat, že tyto hrany budou vždy mít hraniční čáru. Crease Angle byl nastaven na 150°, což umožní vykreslení všech hran, jejichž přilehlé strany svírají nižší úhel. Všechny objekty byly doposud organizovány do svých kolekcí. Kolekce jsou v podstatě složky, pomocí kterých lze lépe organizovat celou scénu. V nastavení freestyle byla vytvořena nová kolekce, do které byly umístěny takové objekty, které nesmí mít vykreslovanou hraniční čáru. Patří sem hologram škorpióna, text na klávesnici notebooku a snímací plocha. Na samotnou hraniční čáru byla také aplikována procedura Along Stroke.

Ta umožňuje určit tloušťku čáry na jejím začátku, prostředku a konci a přidává tak drobnou variaci na čáře.

Efekt Freestyle je výpočetně náročný, zejména kvůli použitému nastavení. Snímek 887 v animaci byl vykreslen v nízkém rozlišení s a bez efektu Freestyle pro srovnání časové náročnosti. Bez užití Freestyle se snímek vykresloval 7 sekund, zatímco snímek s Freestyle efektem se vykresloval 25 sekund. Čas vykreslení se tím více než ztrojnásobil, nicméně je stále velice nízký.

(a) Obecné nastavení Freestyle



(b) Nastavení procedury Along Stroke

(c) Snímek bez hraničních čar

(d) Snímek s hraničními čárami

Obrázek 4.35: Nastavení Freestyle efektu

Kamera scény dostala objektové omezení Track To. To zajistí, že kamera vždy bude směřovat k určitému cíly. Jako cíl byl vytvořen nový objekt Empty, který je animován tak, aby se přesouval z místa na místo. Kamera jej bude vždy sledovat a vždy se soustředí na místo, kde probíhá nějaká činnost.

Nastavení rendereru Eevee zahrnuje zapnutí Ambient Occlusion efektu a Bloom efektu. Rozlišení stínů bylo zvýšeno na 4096 pixelů, aby stíny nebyly čtverečkované. Snímky jsou vykreslovány na 64 samples, což je plně dostačující a v rozlišení $1920 \cdot 1080$.

Pro srovnání bylo také vykresleno několik snímků užitím rendereru Cycles. Ten byl nastaven na 256 samples a byla zapnuta OpenImageDenoise funkce. Cycles nepodporuje uzel Shader to RGB, který byl využit při tvorbě cel shaderu, proto se cel shader vykresluje velice špatně v Cycles. Místo cel shaderu byl tak použit standardní Principled BSDF shader, ale použité textury byly zachovány. Tabulka níže zachycuje čas vykreslení vybraných snímků v nejvyšší kvalitě. Všech dvacet vykreslených snímků je uloženo v souborové příloze diplomové práce.

Další důležitou informací je cena vykreslení snímků. Snímky byly všechny vykreslovány na osobním počítači na grafické kartě NVIDIA GeForce GTX 960. Jedná se o zastaralou grafickou kartu a na moderní grafické kartě by čas vykreslení mohl být nižší. Je možné využít některou z internetových služeb render farm, které umožňují přístup k moderním grafickým kartám pro vykreslení počítačových animací nebo statických snímků. Takových služeb je mnoho a každá se liší ve své nabídce. Některé render farmy mohou být velice levné nebo dokonce dostupné zdarma, ale negarantují za jak dlouho bude snímek vykreslen nebo kvalitu snímku. Jiné farmy mohou být dražší, ale zase nabízí garanci dodání, anonymitu anebo dlouhodobý plán pro větší podniky. Pro zjištění ceny vykreslení byla použita nezávazná online kalkulačka render farmy RebusFarm. Z průměrného času renderování jednoho snímku a použité grafické karty byla tato kalkulačka schopna spočítat přibližnou cenu vykreslení pro celou animaci pro každý ze dvou rendererů. Ceny jsou uvedeny v přiložené tabulce. Vzhledem k rychlosti Eevee ovšem nemá smysl takovou render farmu používat, neboť animace může být vykreslena na zmíněném osobním počítači za přijatelný čas.

Snímky byly vykreslovány samostatně ve formátu png bez komprese. Bylo tak vykresleno 900 snímků o celkové velikosti $5,22$ GB. Doba vykreslení každého snímku se pohybovala mezi 30 a 55 sekundami. Přibližná doba vykreslení se pohybuje okolo deseti hodin, ale přesný čas nebyl změřen.

Vykreslené snímky byly posléze importovány do nového projektu v prostředí Blenderu a z nich byla vytvořena konečná animace. Na posledních pět vteřin videa byl aplikován efekt Blur, který rozmazal všechny snímky. Zde by se mohlo nalézat reklamní sdělení firmy, pro kterou je animace tvořena. Video sdílí rozlišení i počet snímků za vteřinu s hlavním projektem a je uloženo ve formátu mp4 za použití kodeku H.264. Kvalita snímků

Snímek	**Cycles**	**Eevee**
1	11:15.72	00:35.58
100	10:59.10	00:39.75
200	11:33.55	00:39.26
300	10:22.31	00:41.04
400	10:15.45	00:39.74
500	10:09.51	00:45.80
600	09:28.34	00:37.19
700	09:10.85	00:38.32
800	11:04.05	00:51.62
900	10:51.00	00:49.67
Odhadovaný celkový čas	157 hodin	10 hodin a 30 minut
Odhadovaná cena	2 200 Kč	241 Kč

Tabulka 4.2: Srovnání časů vykreslení dokončené scény

byla nastavena na Perceptualy Lossless. To způsobí kompresi snímků a snížení konečné velikosti videa s minimální změnou kvality obrazu. Konečná animace zabírá pouhých 40 MB a její vykreslení trvalo pár minut.

5. Výsledky a diskuze

Výsledkem praktické části diplomové práce je zejména počítačová animace zachycující škorpióna jak vylézá na pódium, na kterém se posléze otáčí zatímco je snímán dvěma snímacími zařízeními. Animace prezentuje řadu moderních postupů, zejména pak užití nové zobrazovací technologie.

Samotný vývoj počítačové animace probíhal v prostředí programu Blender. Součástí programu Blender je nová technologie Eevee, která značně urychlila vývoj ve všech jeho fázích.

Dílčí výsledky jsou samostatné objekty reprezentující škorpióna, snímací zařízení a další, které byly modelovány s důrazem na jednoduchost jejich sítě. Objekty lze použít znovu u jiných animací v prostředí Blenderu, ale lze je také snadno exportovat do jiných programů pro jiné účely. Lze je například snadno exportovat do herních enginů pro tvorbu počítačových her nebo aplikací virtuální reality.

Pro vykreslení povrchových vlastností objektů byl vytvořen cel shader, který obsahuje řadu nastavení pro tvorbu různorodých povrchů. Lze jej tak snadno využít u jiných projektů v prostředí Blenderu, kde je cílem ne-fotorealistické zpracování scény.

Cel shader je možné rozšířit o další funkcionalitu. Například lze přidat více tónů stínů, jiné šrafování nebo podporu hraničních čar. Cel shader bohužel nejde smysluplně exportovat do jiných programů, nicméně obecné techniky cel shading i techniky prezentované zde vytvořeným shaderem je možné převzít a použít pro tvorbu cel shaderu v jiných programech.

Procedurální textury upravující některé povrchy jsou plně kompatibilní s jakýmkoliv shaderem, ačkoliv byly vytvořeny s důrazem na ne-fotorealistické zobrazování. Lze je tak znovu použít pro jakýkoliv projekt uvnitř Blenderu. Procedurální textury není možné exportovat mimo Blender, nicméně je možné je upéct do podoby rastrového obrázku, který lze již exportovat.

Poslední, ale neméně důležitým výsledkem, je srovnání zobrazovacích schopností rendererů Cycles a Eevee, které jsou oba dostupné jako produkční renderery v Blenderu. Renderer Eevee je v současnosti vzácným rendererem, který je schopný vykreslovat obsah velice rychle a ve vysoké kvalitě.

Z vytvořených testů vyplývá, že Eevee exceluje zejména ve vykreslení povrchových i objemových materiálů. Eevee zároveň podporuje většinu shaderů a dalších funkcí dostuných v Blenderu, které byly původně vytvořeny pro renderer Cycles. Eevee má tak působivou kompatibilitu s existujícím rendererem Cycles. Technologie Eevee ma i své slabé stránky, například nízká podpora průhledných shaderů, složité nastavení světelných sond nebo horší správa paměti způsobující pády programu u větších scén.

Rychlost rendereru Eevee je překvapivě vysoká i pro komplexní scény. Během všech testů byl zaznamenáván čas potřebný pro vykreslení snímku a zatímco renderer Cycles vykresloval snímky přes minutu, renderer Eevee dokázal ten samý snímek vykreslit nanejvýš za několik vteřin.

V závěru práce byl proveden druhý test rychlosti vykreslení, který porovnával oba renderery na již hotové scéně animace. V tomto druhém testu byl Eevee desetkrát rychlejší, než Cycles.

Eevee tak lze doporučit pro vykreslení nízko nákladových počítačových animací a nefotorealistického obsahu. Zda-li je Eevee schopen dosáhnout fotorealistické kvality vykreslení je sporné. V případě, že je scéna závislá na kvalitě materiálů a přímého osvětlení, tak je teoreticky možné dosáhnout na první pohled fotorealistické kvality i za pomoci Eevee. Eevee má tak využití pro tvorbu architektonických snímků exteriérů, některých interiérů a také vizuálních efektů pozadí pro film a televizi.

V porovnání s jinými renderery svého druhu je Eevee velmi mladou technologií. Eevee je tak stále na počátku svého vývoje. Vzhledem k úspěšnosti Blenderu v posledních několika letech a vzhledem k rozsáhlému financování společnosti Blender Foundation lze předpokládat, že vývoj bude velice rychle pokračovat. Eevee má tak ještě šanci se během několika let stát dominantním rendererem pro rychlé vykreslování počítačové 3D grafiky.

6. Závěr

Teoretická část diplomové práce představila rozsáhlou problematikou počítačové 3D grafiky. Pro pochopení problematiky byly vysvětleny základní techniky používané v počítačové 2D grafice a jakým způsobem je počítačové 2D grafika využívána počítačovou 3D grafikou. Dále byly představeny základní techniky počítačové 3D grafiky, konkrétně jak jsou vytvářeny objekty, jejich materiály a jak jsou takové objekty vykreslovány. Součástí tohoto tématu bylo také představení různých metod pro vykreslení 3D obsahu. V závěru teoretické části byl představen program Blender a jeho funkční vybavení.

V praktické části diplomové práce byly využity poznatky získané v teoretické části pro tvorbu krátké počítačové animace v prostředí programu Blender. Animace prezentuje aktuální metody pro tvorbu objektů, povrchových materiálů i kosterních animací. Animace dále využívá nový renderer Eevee, který zrychluje celý vývoj.

Součástí praktické části byla také analýza stávajícího rendereru Cycles dostupného v Blenderu s novým rendererem Eevee. Byla zkoumána kvalita vykreslených snímků za použití obou rendererů a zároveň časová náročnost pro vykreslení. Závěrem analýzy je doporučení užití rendereru Eevee pro nízko nákladové vykreslování a pro ne-fotorealistické zobrazování. Pokud je cílem fotorealismus, Eevee je schopen velmi věrně vykreslovat povrchové materiály, ale má řadu problémů, které znemožňují užití Eevee jako plnou náhradu za Cycles.

7. Seznam použitých zdrojů

1. HUGHES, John F. Computer graphics: principles and practice. 3rd ed. Upper Saddle River: Addison-Wesley, c2014. ISBN 03-213-9952-8.

2. SHIRLEY, Peter, Michael ASHIKHMIN, Michael GLEICHER, Stephen R. MARSCHNER, Erik REINHARD, Kelvin SUNG, William B. THOMPSON a Peter WILLEMSEN. Fundamentals of Computer Graphics. 2nd ed. Wellesley: A K Peters, 2005. ISBN 1-56881-269-8.

3. ŽÁRA, Jiří. Moderní počítačová grafika. 2., přeprac. a rozš. vyd. Brno: Computer Press, 2004. ISBN 80-251-0454-0.

4. 3D Rendering. Mathematical and Computer Programming Techniques for Computer Graphics. London: Springer London, 2006, s. 291–316. ISBN 978-1-84628-292-8. Dostupné z: https://doi.org/10.1007/978-1-84628-292-8_9

5. AKENINE-MÖLLER, Tomas, Eric HAINES, Naty HOFFMAN, Angelo PESCE, Michał IWANICKI a Sébastien HILLAIRE. Real-Time Rendering. Fourth Edition. Boca Raton: CRC Press, 2018. ISBN 978-1-1386-2700-0.

6. MUKUNDAN, R. Advanced methods in computer graphics: with examples in OpenGL. London: Springer, c2012. ISBN 978-1-4471-2339-2.

7. Počítačová grafika. In: Masarykova Univerzita Pedagogická fakulta [online]. Brno: Masarykova univerzita, 2020 [cit. 2020-11-29]. Dostupné z: http://www.ped.muni.cz/wtech/u3v/pspp/u3v-grafika.pdf

8. 9. Rastrová grafika. In: Stránky k výuce informatiky [online]. Vlašim: Gymnázium Vlašim, 2020 [cit. 2020-11-29]. Dostupné z: http://www.ivt.mzf.cz/seminar/9-rastrova-grafika/

9. TIŠNOVSKÝ, Pavel. JPEG - král rastrových grafických formátů? In: ROOT.CZ [online]. Praha: Internet Info, 2020 [cit. 2020-11-29]. Dostupné z: https://www.root.cz/clanky/jpeg-kral-rastrovych-grafickych-formatu/

10. 10. Vektorová grafika. In: Stránky k výuce informatiky [online]. Vlašim: Gymnázium Vlašim, 2020 [cit. 2020-11-29]. Dostupné z: http://www.ivt.mzf.cz/seminar/10-vektorova-grafika/

11. TIŠNOVSKÝ, Pavel. Vektorový grafický formát SVG. In: ROOT.CZ [online]. Praha: Internet Info, 2020 [cit. 2020-11-29]. Dostupné z: https://www.root.cz/clanky/vektorovy-graficky-format-svg/

12. PELIKÁN, Josef. 3D počítačová grafika na PC. In: Computer Graphics Charles University [online]. [cit. 2020-11-05]. Dostupné z: https://cgg.mff.cuni.cz/~pepca/lectures/pdf/Grafika2003.pdf

13. AUTODESK. HELP. Introduction to polygons. In: AUTODESK KNOWLEDGE NETWORK [online]. Autodesk, 2020, Sep 09 2014 [cit. 2020-11-29]. Dostupné z: https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Maya/files/Polygons-overview-Introduction-to-polygons-htm.html

14. HOLDEN, Daniel. Subdivision Modelling. In: The Orange Duck — Daniel Holden [online]. [cit. 2020-11-08]. Dostupné z: http://theorangeduck.com/page/subdivision-modelling

15. Box modeling techniques for getting a perfect 3D Model. In: THEPRO3DSTUDIO [online]. Bengaluru: ThePro3DStudio, 2019, 26 November, 2018 [cit. 2020-11-29]. Dostupné z: https://professional3dservices.com/blog/box-modeling-techniques.html

16. Polygonal 3D Modeling Techniques. In: 3D-Ace [online]. Kharkiv: 3D-Ace, 2020 [cit. 2020-11-29]. Dostupné z: https://3d-ace.com/press-room/articles/polygonal-3d-modeling-techniques

17. NURBS: An Introduction. In: THEPRO3DSTUDIO [online]. Bengaluru: ThePro3DStudio, 2019, 26 November, 2018 [cit. 2020-11-29]. Dostupné z: https://professional3dservices.com/blog/nurbs-modeling.html

18. Boolean Modeling in Maya. In: 3DTutorialZone [online]. [cit. 2020-11-09]. Dostupné z: http://www.3dtutorialzone.com/tutorial?id=57

19. GANGL, Diego. What is a non-manifold mesh and how to fix it. In: Sinestesia [online]. Buenos Aires [cit. 2020-11-30]. Dostupné z: https://sinestesia.co/blog/tutorials/non-manifold-meshes-and-how-to-fix-them/

20. HEGINBOTHAM, Claire. What is 3D Digital Sculpting? In: Concept Art Empire [online]. [cit. 2020-11-09]. Dostupné z: https://conceptartempire.com/what-is-3d-sculpting/

21. PETTY, Josh. What is Retopology? In: Concept Art Empire [online]. [cit. 2020-11-10]. Dostupné z: https://conceptartempire.com/retopology/

22. BYL, Leigh Van Der. PHOTOREALISTIC TEXTURING FOR DUMMIES. In: 3DLinks [online]. Trenton: International Computer [cit. 2020-11-30]. Dostupné z: http://www.3dlinks.com/downloads/texturing.pdf

23. MIXER 2020 [online]. Quixel, c2020 [cit. 2020-11-11]. Dostupné z: https://quixel.com/mixer

24. VIVO, Patricio Gonzalez a Jen LOWE. The Book of Shaders. The Book of Shaders [online]. Copyright 2015 [cit. 2020-11-30]. Dostupné z: https://thebookofshaders.com/

25. PRICE, Andrew. Photorealism Explained. In: Blender Guru [online]. Brisbane, © 2017, May 25, 2016 [cit. 2020-11-30]. Dostupné z: https://www.blenderguru.com/tutorials/photorealism-explained

26. BURLEY, Brent. Physically-Based Shading at Disney. In: Walt Disney Animation Studios [online]. The Walt Disney Company, © 2020, Aug 31, 2012 [cit. 2020-11-30]. Dostupné z: https://www.disneyanimation.com/publications/physically-based-shading-at-disney/

27. Rasterization: a Practical Implementation. In: Scratchapixel 2.0 [online]. Scratchapixel, @ 2009-2016 [cit. 2020-11-30]. Dostupné z: https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation

28. Introduction to Ray Tracing: a Simple Method for Creating 3D Images: Raytracing Algorithm in a Nutshell. In: Scratchapixel 2.0 [online]. Scratchapixel, @ 2009-2016 [cit. 2020-11-30]. Dostupné z: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/raytracing-algorithm-in-a-nutshell

29. Introduction to Ray Tracing: a Simple Method for Creating 3D Images: Adding Reflection and Refraction. In: Scratchapixel 2.0 [online]. Scratchapixel, @ 2009-2016 [cit. 2020-11-30]. Dostupné z: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/adding-reflection-and-refraction

30. GUGICK, Jeffry. IOR List. In: PIXEL and POLY [online]. Pixel and Poly, ©2017 [cit. 2020-11-30]. Dostupné z: https://pixelandpoly.com/ior.html

31. Minecraft Live: The Art of Caves & Cliffs. In: YouTube [online]. 2020 [cit. 2020-11-12]. Dostupné z: https://www.youtube.com/watch?v=4WJ7LmY3mZA&feature=emb_title

32. IS.MENDELU.CZ. Barevné modely. In: Mendelova univerzita v Brně [online]. Brno: Mendelova univerzita v Brně, © 2018 [cit. 2020-11-30]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=772

33. Raster Images vs. Vector Graphics. In: The Printing Connection [online]. Newbury Park: The Printing Connection, 2020 [cit. 2020-11-26]. Dostupné z: https://www.printcnx.com/resources-and-support/addiational-resources/raster-images-vs-vector-graphics/

34. BLENDER FOUNDATION. BoxModelingIntro AddingDetails. In: Wikimedia Commons [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-11-30]. Dostupné z: https://commons.wikimedia.org/wiki/File:BoxModelingIntro_AddingDetails.jpg

35. Modeling » Surfaces » Introduction. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/modeling/surfaces/introduction.html

36. What do you think about my new car render? In: Reddit [online]. [cit. 2020-11-25]. Dostupné z: https://www.reddit.com/r/blender/comments/cdfm1k/what_do_you_think_about_my_new_car_render/

37. JIKZ. Cross Hatch Shader. In: Bnpr [online]. BlenderNPR, November 6, 2012 [cit. 2020-11-30]. Dostupné z: https://blendernpr.org/cross-hatch-shader/

38. HENRIK. Ray trace diagram. In: Wikimedia Commons [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-11-30]. Dostupné z: https://commons.wikimedia.org/wiki/File:Ray_trace_diagram.svg

39. Blender [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-11]. Dostupné z: https://www.blender.org/

40. Modeling » Meshes » Structure. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/modeling/meshes/structure.html

41. Modeling » Meshes » Primitives. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/modeling/meshes/primitives.html

42. Modeling » Curves » Introduction. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/modeling/curves/introduction.html

43. Modeling » Surfaces » Primitives. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/modeling/surfaces/primitives.html

44. Boolean Modifier. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/booleans.html?highlight=boolean

45. Modeling » Modifiers » Introduction. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/modeling/modifiers/introduction.html

46. Sculpting & Painting » Sculpting » Introduction. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/sculpt_paint/sculpting/introduction.html

47. Rendering » Shader Nodes » Introduction. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/render/shader_nodes/introduction.html#shaders

48. Rendering » Shader Nodes » Shader » Principled BSDF. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/principled.html

49. Animation & Rigging » Introduction. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/animation/introduction.html

50. Workbench » Introduction. In: Blender 2.90 Reference Manual [online]. Amsterdam: Blender Foundation, 2020 [cit. 2020-11-29]. Dostupné z: https://docs.blender.org/manual/en/latest/render/workbench/introduction.html

51. Pandinus imperator. In: The Scorpion Files [online]. 2020 [cit. 2020-01-02]. Dostupné z: https://www.ntnu.no/ub/scorpion-files/p_imperator.jpeg

52. Pandinus imperator. In: The Scorpion Files [online]. 2020 [cit. 2020-01-02]. Dostupné z: https://www.ntnu.no/ub/scorpion-files/p_imperator5.jpg

53. Scorpion anatomy. In: The Scorpion Files [online]. 2020 [cit. 2020-01-02]. Dostupné z: https://www.ntnu.no/ub/scorpion-files/scorpion_anatomy.jpg

54. WILLIAMS, Alexander. How to animate a scorpion in motion. In: YouTube [online]. 10. 12. 2012 [cit. 2020-11-30]. Dostupné z: https://www.youtube.com/watch?v=YyMCCw_lPA4

55. Bark. In: BlenderKit [online]. Praha, 2020 [cit. 2020-11-26]. Dostupné z: https://www.blenderkit.com/get-blenderkit/97a6acd7-b0f5-4ff4-85e7-e8ecf3dad8cb/

8. Přílohy

Příloha je k nalezení na přiloženém CD nebo nahraná na informačním systému České zemědělské univerzity. Příloha obsahuje následující soubory:

- Všechny vykreslené snímky vytvořené pro praktickou část diplomové práce.
- Soubory formátu .blend obsahující scénu vytvořené animace, cel shader a testování rendererů.
- Vykreslenou reklamní animaci.