Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Prezentace firmy pomocí 3D modelování

Bc. Miroslav Šimák

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Miroslav Šimák

Systémové inženýrství a informatika
Informatika

Název práce

Prezentace firmy pomocí 3D modelování

Název anglicky

Company presentation using 3D modeling

Cíle práce

Navrhněte vhodný nástroj pro prezentaci firmy formou 3D modelů. Vytvořete samostatně spustitelnou aplikaci, která prezentaci umožní. Běhovým prostředím bude internetový prohlížeč.

Metodika

Prostudujte techniky 3D modelování se zaměřením na:

-Hard-surface modeling

-Sculpting

-Tvroba materiálů

-UV Mapping

-Technologie snímání pro automatické vytváření modelů

-Specifika modelování pro použití mimo statickou scénu

Navrhněte vhodný software a porovnejte jej s dalšími variantami.

Popište možnosti využití herního engine pro prezentace firem.

Doporučený rozsah práce

60-80

Klíčová slova

3D modelování, interaktivní prezentace, hard-surface, herní engine

---

Doporučené zdroje informací

A. Chopine, 3D ART ESSENTIALS: The Fundamentals of 3D Modeling, Texturing, and Animation. Elsevier Inc., 2011, ISBN: 978-0-240-81471-1

T. Akenine-Möller, E. Haines, a N. Hoffman, Real-Time Rendering, 3. vyd. Wellesley, Massachusetts: A K Peters, Ltd., 2008, ISBN: 978-1-56881-424-7

T. Norton, Learning C# by Developing Games with Unity 3D, roč. 2013. Packt Publishing, 2013. ISBN: 978-1-84969-658-6

W. Goldstone, Unity Game Development Essentials, roč. 2009. Packt Publishing, 2009. ISBN: 978-1-84719-818-1

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

Ing. Josef Pavlíček, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 11. 3. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 11. 3. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 20. 03. 2020

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Prezentace firmy pomocí 3D modelování" 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 20.3.2020 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Josefu Pavlíčkovi Ph.D. za vedení diplomové práce, ochotnou spolupráci na konzultacích a pomoc při volbě jejího zaměření.

Prezentace firmy pomocí 3D modelování

Souhrn

Cílem práce je navrhnout a vytvořit prezentaci firmy pomocí 3D modelování. Výsledkem práce je samostatně spustitelná aplikace, která umožňuje firmě USPIN s. r. o. prezentovat vlastní produkt z oblasti strojového vidění. Konkrétně se jedná o implementaci automatizované kontroly kvality polotovarů výfukových trubek. Rešeršní část se zabývá problematikou 3D modelování, se zaměřením na hard-surface modelování a rendering v reálném čase. Nástroje pro modelování a implementaci samotné aplikace byly vybrány na základě požadavků autora na specifické funkce a v neposlední řadě s ohledem na jejich pořizovací náklady.

Praktická část je zaměřena na samotnou implementaci řešení, která zahrnuje modelování jednotlivých objektů, UV mapování, aplikaci materiálů a jejich animaci. Dále je popsána implementace jednotlivých funkcí aplikace. Praktická část také obsahuje záznamy z testování aplikace na funkčnost jednotlivých prvků. Aplikace byla otestována na několika PC sestavách, aby byly zjištěny její systémové požadavky. Výsledky tohoto testování jsou v praktické části zhodnoceny a na jejich základě je provedeno rozhodnutí, zda byly cíle práce úspěšně splněny. Aplikace je spustitelná na platformách Windows, Mac OS a WebGL, poslední jmenovaná umožňuje spouštění aplikace pomocí webových prohlížečů.

Klíčová slova: 3D modelování, prezentace, interaktivní, Unity, WebGL, Blender, rendering, real-time, hard-surface

Company presentation using 3D modeling

Summary

The aim of the thesis is to design and create a company presentation using 3D modeling. The result of this work is a self-executing application that allows USPIN Ltd. to present its own product in the field of machine vision. Specifically, it involves the implementation of automated quality control of exhaust pipe parts. The research part deals with the issue of 3D modeling, with a focus on hard-surface modeling and real-time rendering. Tools for modeling and implementation of the application itself were chosen based on the author's requirements for specific functions and with regard to their purchase costs.

The practical part is focused on the implementation of the solution itself, which includes modeling of individual objects, UV mapping, application of materials and their animation. The implementation of individual functions of the application is described below. The practical part also contains records from application testing for functionality of individual elements. The application has been tested on several PC configurations to determine its system requirements. The results of this testing are evaluated in the practical part and a decision is made, whether the goals of the work were successfully fulfilled. The application is executable on Windows, Mac OS and WebGL platforms, the latter allows running the application using web browsers.

Keywords: 3D modeling, presentation, interactive, Unity, WebGL, Blender, rendering, real-time, hard-surface

Obsah

1 Úvod

Díky rozvoji informačních technologií došlo za posledních 10 let k výraznému zlepšení dostupnosti nástrojů pro 3D vizualizaci a tvorbu interaktivních aplikací pro širokou veřejnost. Společně s rozvojem těchto nástrojů jsou nově k dispozici i technologie, které byly dříve dostupné pouze pro oblast animovaného filmu a často pouze interně ve studiích, kde byly vyvíjeny. Významně byla rozšířena funkcionalita nástrojů dostupných zdarma, ať už se jedná o software pro 3D modelování a rendering, jako je například Blender, nebo nástroje pro projektování interaktivních aplikací, animací, videoher. Výsledné aplikace je již běžně možné spouštět nejen zcela samostatně, ale i z webu. K tomu slouží platforma WebGL založená na Javascriptu, která podporuje vykreslování aplikací pomocí grafické karty na straně klienta a nahrazuje tak zastaralá řešení, jako je používání doplňků Flash a Shockwave.

Práce se zabývá tvorbou interaktivní prezentace produktu společnosti USPIN s.r.o. v oblasti strojového vidění. Dále zkoumá techniky a metody použitelné pro tvorbu takové prezentace. Aplikace bude spustitelná jak samostatně, tak z webu, což umožní společnosti USPIN s.r.o. ukázat vlastní řešení produktu přitažlivou formou na osobních schůzkách s potenciálními klienty, na veletrzích a přehlídkách a v neposlední řadě jako součást online prezentace firmy.

Interaktivní aplikace je novou formou pro prezentaci produktů, která rozšiřuje existující metody vizualizace, jako jsou například 3D renderované statické scény, nebo animace a videa. Práce zkoumá možnosti její implementace pomocí využití vývojového prostředí pro tvorbu her v kombinaci s volně dostupnými nástroji pro 3D modelování. Vývoj použitých technologií a počítačové grafiky jako celku je stručně popsán v prvních kapitolách. Dále jsou v práci popsány techniky, postupy a pojmy související s modelováním, a především renderováním 3D objektů v reálném čase. Vysvětleny jsou také prvky ovlivňující výpočetní složitost renderování. Tyto poznatky jsou pak využity v praktické části práce.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je navrhnout a vytvořit prezentaci firmy pomocí 3D modelování. Výsledkem práce bude vytvořena samostatně spustitelná aplikace, která umožňuje firmě USPIN s. r. o. prezentovat vlastní produkt z oblasti strojového vidění. Aplikace umožní uživateli poznat funkce produktu a částečně je ovlivňovat. Pro prezentaci produktu aplikace bude využívat vlastní 3D modely, jejichž tvorba je součástí práce. Cílovým běhovým prostředím aplikace bude webový prohlížeč.

2.2 Metodika

Rešeršní část DP bude založena na analýze odborných publikací z oblasti 3D modelování a získané poznatky budou synteticky využity v praktické části práce. Rešeršní část se bude zaměřovat na problematiku hard-surface modelování, UV mapování a aplikaci materiálů a textur na 3D modely. Dále se bude zabývat metodami pro renderování, s užším zaměřením na ty, které jsou využívány pro vykreslování 3D modelů v reálném čase. Popsány budou také alternativní metody pro získání 3D modelů. Pro porozumění základním pojmům v oblasti počítačové grafiky a 3D modelování bude část práce věnována historickému vývoji počítačové grafiky.

Na základě poznatků z rešeršní části budou vybrány vhodné nástroje pro tvorbu modelů a aplikace, pomocí kterých bude navržena a realizována praktická část DP. Výsledná aplikace bude testována, aby byla ověřena její funkčnost a dále bude orientačně naměřeny její systémové požadavky pomocí měření dosažené snímkové frekvence na různých PC sestavách.

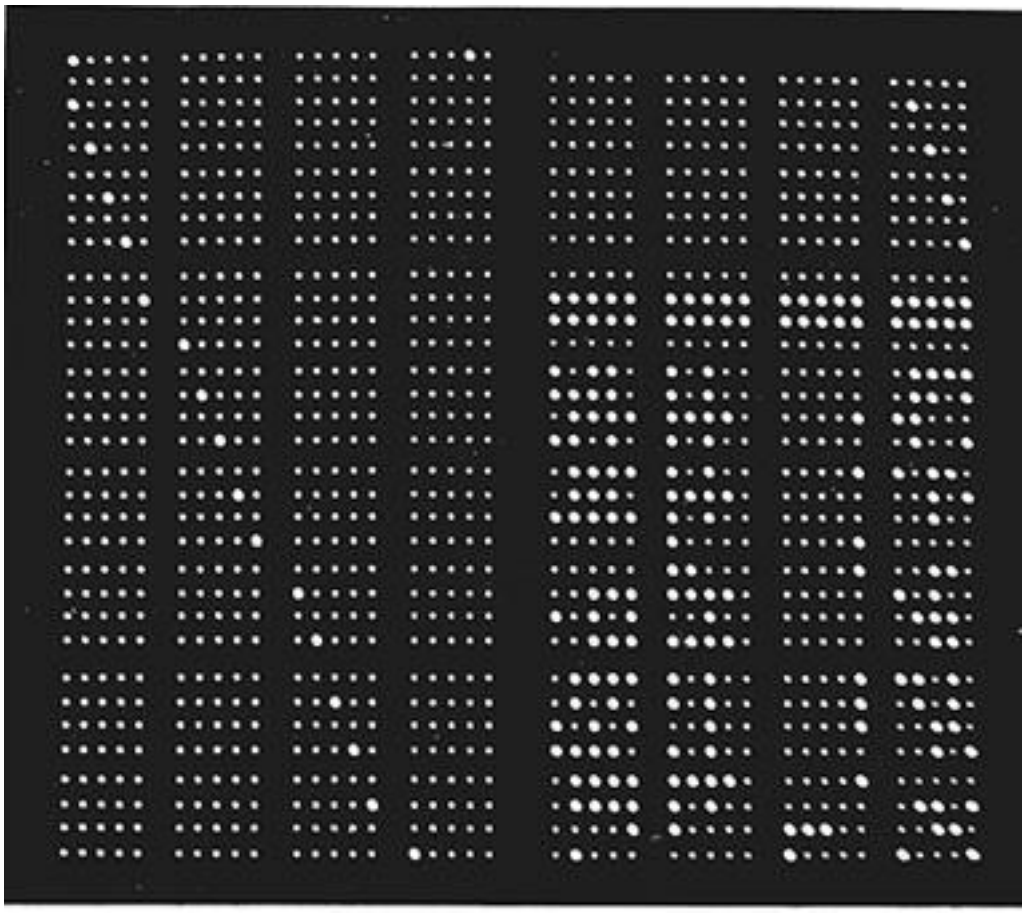
3 Teoretická východiska

3.1 Historie počítačové grafiky a využití 3D modelování

Tato část práce se zabývá stručným popisem vývoje počítačové grafiky a souvisejících technologií. Zavedené pojmy jsou dále využity při popisování problematiky 3D modelování a renderování v současných nástrojích.

3.1.1 Williams Tube

Počátky vývoje počítačové grafiky úzce souvisí s vývojem prvních zobrazovacích zařízení a jejich interaktivity. V roce 1947 Tom Kilburn objevil novou metodu ukládání dat pomocí katodových trubic. Principem této metody bylo uložení náboje do katodové trubice tak, aby představoval kolekci bitů. Náboj byl uložen do pole bodů, kde každý bod představoval podle svého jasu jeden bit. Typ náboje v jednotlivých bodech byl zaznamenáván pomocí kovové záchytné plochy, čímž docházelo ke čtení hodnoty, tedy, zda je bit „aktivní“, nebo „neaktivní“. Samotný náboj se rychle rozptyloval, proto bylo nutné hodnoty neustále číst a obnovovat. Aby nedošlo k narušení z důvodu citlivosti trubic, musely být trubice odstíněny od veškerých zdrojů elektrického rušení. Bylo ale vyžadováno, aby hodnoty byly čitelné pro uživatele, proto se v oblasti obvykle vyskytovala další, zobrazovací trubice, na kterou byla data přenášena, kde nemohlo dojít k přímému narušení z venkovního prostředí a díky tomu bylo možné aktuální stav bitů kontrolovat vizuálně. Katodová trubice je technologií, která je dnes známa pod zkratkou CRT (Cathode-Ray Tube) a položila základ pro většinu zobrazovacích zařízení a s nimi spojený vývoj počítačové grafiky.[1]



Obrázek 1 - Zobrazovací zařízení na počítači Ferranti Mark 1 [2]

Na obrázku 1 je fotografie zobrazovacího zařízení na bázi zobrazovací katodové trubice, konkrétně na počítači Ferranti Mark 1. Světlejší body zastupují jedničky, tmavší body zastupují nuly a vyjadřují tak uloženou hodnotu. Na zařízení je zobrazeno 32 čtyřiceticiferných čísel v binární soustavě. Navíc zařízení zobrazuje další, dvaceticiferný řádek (nahoře). [2]

Využitím této zobrazovací technologie později vzniklo pro počítač Whirlwind jedno z prvních zobrazovacích zařízení, které bylo navíc interaktivní. Pomocí ovladače v podobě "světelného pera" bylo možné se dotknout displeje, což způsobilo v počítači přerušení a ten následně předal další informace o vybraném objektu. [3, 4]

3.1.2 SKETCHPAD pro TX-2

V roce 1963 byl na MIT vyvinut software SKETCHPAD. Jeho autorem byl Ivan Sutherland a vytvořil ho jako součást své disertační práce. Software umožňoval uživateli kreslit tvary a

zajímavostí je, že bylo možné tvořit i 3D objekty. Ovládání bylo zajištěno pomocí „světelného pera“, které uživatel přikládal přímo na obrazovku. Jednalo se o jednu z vůbec prvních možností, jak interagovat s počítačem jinak než pomocí děrných štítků s instrukcemi. Software fungoval na počítači TX-2, umístěném v prostorách MIT. Jednalo se o počítač založený na již zmíněném počítači Whirlwind, konstrukčně se lišil především tak, že využíval tranzistory namísto vakuových trubic. Díky tomu také vykazoval mnohem větší spolehlivost. TX-2 musel být pro použití SKETCHPAD speciálně upraven a pro jakoukoliv jinou činnost musel být navrácen do původního stavu.[3]

3.1.3 První interaktivní hry

Aby se počítače a s nimi spojené obory mohly nadále vyvíjet, muselo dojít k jejich zjednodušení z hlediska uživatelské přístupnosti. Především byla s postupem času zjednodušováno jejich prvotní pevné nastavení a došlo k vytvoření programovacích jazyků, což umožnilo určení jejich konfigurace zadáváním instrukcí pomocí klávesnice.

Jedna z prvních her využívající zobrazovací zařízení pro vykreslování grafických objektů se nazývala „*Spacewar!*“. Byla vytvořena v roce 1962 trojicí Steve Russel, Martin Graetz a Wayne Witaenem. Fungovala na počítačích DEC PDP-1 (komerční verze počítače TX-2 na MIT), k jejímu naprogramování bylo potřeba přibližně 200 hodin práce a šíření obvykle probíhalo formou tištěného zdrojového kódu. Takto se hra dostala ke společnosti DEC (Digital Equipment Corporation), vyrábějící právě zmíněné počítače. Hra byla využita na testování těchto počítačů ve výrobě a později dokonce přikládána ke každému prodanému kusu. Stejně tak jako u software SKETCHPAD se pro zobrazení využívaly CRT obrazovky. 3D objekty složené z polygonů se zobrazovaly jako tzv. wireframes, kde bylo vždy možné vidět i zadní (teoreticky skrytou) stranu objektu.[3]

3.1.4 Vývoj v ARPA

Zájem ARPA (Advanced Research Projects Agency of the US Department of Defense) napomohl dalšímu vývoji počítačové grafiky a souvisejících algoritmů. Agentura poskytla v roce 1973 roční grant 5 milionů dolarů univerzitě v Utahu, kde tak mohlo docházet k dalším průlomům v oblasti realistického zobrazení objektů pomocí počítačů. Jedním z těchto

průlomů byl algoritmus, který dokázal rozpoznat „zadní stranu“ wireframe vynechat její vykreslování, čímž bylo poprvé umožněno realisticky zobrazovat 3D objekty. Během výzkumu došlo také k významnému vývoji v oblastech vědeckých vizualizací, lékařských zobrazovacích metod a CAD (Computer Aided Design). V období, kdy byl výzkum financován z ARPA byly také vyvinuty první shadery, tedy metody stínování.[3]

3.1.5 Phongovo a Blinnovo stínování

Jeden z prvních shaderů, tedy metoda stínování 3D modelů vznikla v 70. letech minulého století. Bui Tuong Phong, na základě zkoumání, jakým způsobem přímé světlo zasahuje reálné objekty, sestavil algoritmus, který tyto jevy simuluje. Společně s druhým algoritmem pro simulaci odlesků tak vytvořil shadery, které jsou ve 3D modelovacích aplikacích využívány dodnes. Poprvé byly vydány v roce 1973 jako součást Phongovy disertační práce a dále ještě ve vědeckém článku z roku 1975, pojednávajícím o výzkumu dosavadních metod pro stínování a vykreslování 3D objektů pomocí počítače.[3, 5]

Phong zde zmiňuje, že před jeho vlastní implementací existovaly dva základní přístupy k vykreslování 3D modelů:[5]

1. Definice povrchu pomocí matematických rovnic
2. Odhad povrchu pomocí planární polygonální mozaiky

Dále popisuje funkce několika systémů pro první přístup (definice pomocí rovnic), pomocí kterých je možné algoritmicky určit, které části 3D objektu mají být skryté. Tyto systémy spočívají v přesných kalkulacích pro matematicky definované zakřivené povrchy, kdy je možno vypočítat polohu každého bodu daného povrchu a patřičně ho skrýt. Phong zmiňuje, že tento přístup vede k nejrealističtějším výsledkům, avšak má i svá omezení. Například je omezen typ povrchů, pro které lze tuto metodu řádně implementovat, kvůli nutné znalosti přesných rovnic pro daný povrch. Phong popisuje tuto metodu jako tak výpočetně náročnou, že je pro složitější objekty někdy zcela nepoužitelná.[5]

Phong dále pro srovnání nabízí druhý přístup, metodu planární aproximace. Ta je založena na odhadu tvaru povrchu pomocí menších polygonů. Jako hlavní výhodu uvádí, že pro její

implementaci není nutné provádět složité matematické výpočty a zmiňuje, že jde o jediný přístup, umožňující snížit náročnost výpočtu skrytých ploch a stínování na přijatelnou mez. Ačkoliv tento přístup nabízí nesrovnatelně nižší výpočetní složitost, přináší s sebou také problémy, související se samotným realistickým vykreslením 3D objektů. Jedním z nich je problém nepřesného obrysu, který vzniká tak, že obrys daného objektu je při použití planární aproximace polygonem a nikoliv křivkou. Hledáním řešení k tomuto problému se zabýval Phong ve spolupráci s Franklinem C. Crowem v dalším vědeckém článku *Improved rendition of polygonal models of curved surfaces* z roku 1975. Druhým zásadním problémem planární aproximace je realistické stínování objektu, který je definován jako zjednodušená verze složená z polygonů. Řešením je právě Phongův algoritmus.[5]

Obrázek 2 – Slavný model konvice na čaj vykreslený pomocí Phongova stínování[3]

Jim Blinn, který stejně jako Phong studoval na Utažské univerzitě, vycházel z Phongovy implementace při vývoji vlastní metody stínování, která se vyznačovala především nižší výpočetní náročností.[3]

3.1.6 Počátky speciálních efektů a 3D animace

Utažská univerzita byla místem, kde probíhala většina rozvoje technologií a metod souvisejících s 3D modelováním. Během 70. let zde byly kromě algoritmů pro stínování také vyvinuty metody pro mapování textur, antialiasing (vyhlazování hran), animaci obličeje a mnoho dalších. [3]

V roce 1974 Ed Catmull společně s týmem z laboratoře počítačové grafiky v New Yorku přispíval k vylepšování technologií pro mapování textur a 3D animace, což upoutalo pozornost George Lucase. Lucas měl o počítačovou grafiku a její využití zájem, proto založil oddělení počítačové grafiky jako součást vlastního studia pro tvorbu speciálních efektů nazvané *Industrial Light and Magic*. Do tohoto oddělení najal tým ze zmíněné laboratoře v New Yorku, včetně Eda Catmulla.[3]

Právě ve studiu ILM (Industrial Light and Magic) vznikla první plně počítačem vytvořená scéna pro celovečerní film. Jednalo se o tzv. *Genesis Effect* ve filmu *Star Trek II: The Wrath of Khan*, který byl vydán v roce 1982. Jednalo se o významný technologický pokrok, protože dříve byla animace pro film vytvářena snímek po snímku a další speciální vizuální efekty byly produkovány pomocí fotografií modelů nebo jinými analogovými metodami. Sekvence ve filmu má stopáž o délce jedné minuty a vyobrazuje transformaci jedné z filmových planet. Na výše zmíněné sekvenci z roku 1982 jsou patrné pokroky v oblasti částicových efektů a využití techniky rozmazání pohybu.[3, 6]

Ve stejném roce vyšel celovečerní film *Tron* od studia Walt Disney Productions. Na spolupráci při natáčení byly najaty tři společnosti zabývající se počítačovou grafikou. Film obsahoval dohromady 40 minut počítačové animace, většinou (na rozdíl od výše zmíněného *Star Trek*) v kombinaci s živým natáčením před černým plátnem. Film byl také natočen z části černobíle a později kolorizován pomocí fotografických technik a rotoskopie. Na filmu se projevila časová náročnost počítačové animace, příprava snímků pro některé scény trvala až šest hodin.[3, 7]

Film *Tron* nezaznamenal přílišné kritické úspěchy, což vedlo v kombinaci s vysokými náklady na provoz grafických studií ke zdržení vývoje 3D animací a efektů pro celovečerní filmy. K obnovení zájmu filmových studií došlo až v roce 1985, kdy vyšel film

The Last Starfigher, hojně využívající 3D modely vesmírných lodí namísto obvyklých fyzických rekvizit. V roce 1986 došlo k oficiálnímu založení studia Pixar, které má i v dnešní době významný podíl na vývoji 3D renderovacích technologií, jako jsou shadery, částicové efekty, simulace vlasů a další.[3, 8]

3.1.7 Rozvoj 3D nástrojů dostupných pro domácnosti

Ačkoliv filmový průmysl od 80. let minulého století vedl většinu pokroku v oblasti využití a zdokonalování 3D technik a speciálních efektů, díky miniaturizaci a vývoji hardware pro běžného spotřebitele bylo nyní možné tyto technologie začít poskytovat i pro širší trh.

V roce 1986 Eric Graham vytvořil krátkou, 24-snímkovou opakující se animaci na svém počítači Amiga společně s primitivním renderovacím software (využívající techniky sledování paprsků) pro vlastní potřeby. Animace ukazovala jednoduchou postavu vytvořenou z několika objektů tvarů koule, žonglující s míčky z reflektivního materiálu. Do této doby bylo renderování 3D objektů a obzvlášť animací výsadou superpočítačů. Když Graham tuto animaci ukázal firmě Commodore, společnost okamžitě koupila práva k jejímu použití pro propagační materiály. Tímto krokem se výrazně zvedl zájem tehdejších nadšenců o „domácí" 3D a Graham byl požádán o vylepšení svého domácího software tak, aby ho bylo možné začít prodávat. Takto vzniknul první software pro tvorbu 3D grafiky, fungující na „běžném" domácím počítači, Sculpt3D. Editor obsahoval většinu funkcí a principů, které lze nalézt i v současných aplikacích, například tzv. primitivní tvary ke skládání komplexních objektů, několik možných pohledů na modelovaný objekt, virtuální kameru a světla. 3D modely byly složeny z mřížek trojúhelníků a počítač jich byl schopen načíst až několik set.[3]

Další vývoj konzumního 3D pokračoval v 90. letech, kdy ceny osobních počítačů klesly dostatečně na to, aby o ně měla zájem i širší veřejnost. 3D modelovací nástroje byly stále nákladné na pořízení, řádově stály tisíce dolarů, což vedlo mimo jiné ke vzniku prvních freeware aplikací napodobujících jejich funkci, jako byl například software POV-Ray. V tomto období byl také poprvé vydán nástroj Blender, který v prvních letech své existence prošel několika licenčními stádii. Jedním z jeho autorů je student průmyslového designu v Eindhovenu, Ton Roosendaal. Původně byl Blender vnitřním nástrojem studia NeoGeo, kromě verze zdarma pro domácí užití později existovaly i prodejné komerční verze od

společnosti Not a Number (NaN). Po neúspěšných prodejích byl vývoj na krátkou dobu ukončen, než byla díky komunitě založena nezisková nadace Blender Foundation.[3, 9]

Díky postupnému šíření Blenderu a dalších nástrojů mezi běžné uživatele po roce 2000 se začal rozvíjet zcela nový směr umění, využívající 3D modelování a rendering. Neustálý růst komunity zajistil, že kromě nástrojů pro tvorbu 3D mohou tvůrci využít také komunitních návodů, předpřipravených modelů, textur, materiálů nebo dokonce celých scén, a to buď za poplatek, nebo dokonce v některých případech zcela zdarma. Novým směrem 3D modelování pro ty, kteří se nezajímají o rendering nebo animace je také 3D tisk, umožňující výrobu a distribuci vlastních výtvorů fyzickou formou. V posledním desetiletí došlo k významnému nárustu poptávky po 3D modelovacích nástrojích a umělcích, kteří se modelováním zabývají. To vedlo k dalšímu snížení cen profesionálního software pro tvorbu 3D grafiky. Filmová animační studia, zodpovědná za většinu pokroků v oblasti 3D, postupně vydávají vlastní nové technologie, díky kterým je možné dosáhnout i v domácím prostředí vizuální kvality na úrovni animovaných filmů z minulých let. Příkladem může být studio Pixar, poskytující přístup k vlastnímu shaderu Renderman, který je zdarma k vyzkoušení pro nekomerční použití.[3, 10]

3.2 3D modelování

3D modelování je v oblasti počítačové grafiky pojem, označující techniku tvorby trojrozměrné digitální reprezentace jakéhokoliv objektu nebo povrchu. K modelování se používá specializovaný software, umožňující pohyb a tvorbu ve virtuálním prostoru.[11]

3D model je podle obecné definice matematickou reprezentací objektu, existujícího ve třech dimenzích. Modely mohou být zkonstruovány matematicky, s tvary definovanými rovnicemi křivek. Tento přístup je považován za výpočetně složitý, proto se místo něj používají aproximační metody, u kterých je typickým základem modelu 3D mřížka, složená z bodů, nazývaných *vertexy*, propojených spojnicemi, jinak nazývanými *okraje*, případně *hrany*. Každý vertex má určenou polohu pomocí souřadnic na třech osách[12, 13]

Mřížka (anglicky *mesh*) je souborem vertexů propojených okraji, dále uspořádanými do polygonů se společnou hranou. Typicky se jedná o soubor čtyřúhelníků, které jsou později

rozděleny na trojúhelníky, některé modelovací aplikace pracují výhradně s trojúhelníky a vykreslování jiných tvarů u nich není podporováno. Teoreticky je umožněno, aby měl polygon v 3D mřížce i více hran, z praktických důvodů se tomuto jevu ale většina modelovacích technik snaží předejít. Důvody jsou popsány v následujících podkapitolách. Metoda zobrazení, zaměřená pouze na znázornění mřížky objektu (bez povrchu) se nazývá *wireframe*.[12, 14]

Modely mohou být tvořeny manuálně, případně existují nástroje pro jejich generování. Možnosti využití 3D modelů jsou široké, typicky v oblastech jako jsou architektura, simulace, medicína, film, ilustrace, reklama, videohry a mnoho dalších. Mohou být exportovány do jiných aplikací pro další vývoj (např. simulace a hry), většina modelovacích aplikací umožňuje generování 2D zachycení modelu pomocí procesu zvaného *3D rendering*. Tento proces je vhodný pro co nejrealističtější vizualizaci modelovaného objektu[11]

V následujících podkapitolách bude rozvedeno dělení modelování na dva základní typy podle přístupu, dále pak budou popsány postupy a techniky používané při digitální tvorbě 3D modelů a dalších souvisejících produktů, jako jsou např. renderované snímky a animace. Tyto dva základní typy modelování se striktně nevylučují a mohou být v moderních nástrojích použity kombinovaně.

3.2.1 Hard-surface modeling

Existuje několik možností, jak definovat hard-surface modeling a hard-surface model jako soubor metod, respektive typ modelu. Jedním z rozlišovacích faktorů je typ modelovaného objektu. Hard-surface modely jsou typicky objekty v reálném světě vytvořené člověkem nebo strojem, sestávající se z plochých nebo lehce zakřivených povrchů a jasně definovaných hran. Reálný objekt, který je pro účely modelování považován za hard-surface se často vyznačuje viditelnou symetrií. Pro uživatele začínající s 3D modelováním je často doporučováno začít s hard-surface modely, z důvodu, že poskytují příležitost naučit se pracovat s jednotlivými prvky v mřížce a pochopit tak základní principy práce v daném SW. Oproti organickému modelování mají hard-surface modely rovnější okraje a méně komplikovaných křivek a tvarů, co potenciálně mohou studium základů 3D komplikovat. [15, 16]

3.2.2 Organic modeling

Organické modelování je poměrně novým přístupem v oblasti digitální tvorby, jeho aplikace je totiž umožněna především existencí specializovaných nástrojů pro proporční editaci mřížek. Soubor technik a metod, jak pracovat s proporčními editačními nástroji, se nazývá *sculpting*. Název „organické modelování“ značí, že tento přístup se především věnuje modelování objektů a povrchů které jsou považovány za „přírodní“. Může se tedy jednat například o zvířata, rostliny nebo lidské postavy. Okruh objektů, které je možné modelovat „organicky“ není pevně určený, zpravidla se ale jedná o objekty nepravidelných tvarů, s komplikovanou strukturou povrchu a nesnadno definovatelným materiálem.[15]

Vzhledem k častějšímu využití organických modelů pro animaci je vhodné, aby měly správnou topologii s mřížkou sestávající se výhradně ze čtyřúhelníkových polygonů. Pro editaci modelů existují specializované softwarové produkty jako např. ZBrush, většina rozsáhlejších 3D modelovacích nástrojů obsahuje pro sculpting vlastní separátní modul.

Princip organického modelování spočívá v úpravě tvaru modelu bez přímého zasahování do jednotlivých prvků mřížky. Modelovací nástroje mřížku zpravidla v běžném režimu nezobrazují a automaticky generují její tvar podle vstupů zadávaných uživatelem. Jedná se tak o jednoduchou metodu pro začátečníky, kde pro samotnou tvorbu není nutná podrobná znalost principů 3D modelování. Nevýhodou organické metody modelování je, že počítačem vygenerovaná topologie je často nepřehledná a modle obsahuje mnohem větší množství polygonů, než kdyby byl vytvořen uživatelem pomocí hard-surface přístupu. V takovém případě je pro další využití modelu nutno provést tzv. *retopologii*.[15, 16]

Vzhledem k relativně jednoduchému ovládání nástrojů pro sculpting lze využít jeho výhod v kombinaci s hard-surface přístupem k tvorbě modelu. Modelovací nástroje umožňují vyexportovat složitou, organickým přístupem vytvořenou, strukturu jako tzv. normálovou mapu, nebo texturu, kterou lze poté aplikovat na model se zjednodušenou topologií a nižším počtem polygonů. Může tak být dosaženo na pohled velice detailního modelu, u kterého je ale významně snížená vykazovaná výpočetní složitost pro vykreslování a další zpracování.[17]

3.3 Nástroje, techniky a pojmy 3D

V následujících podkapitolách jsou popsány základní nástroje, techniky a pojmy související jednak obecně s 3D modelováním, ale především také s praktickou částí této práce. Znalosti z těchto podkapitol budou přímo využity při tvorbě modelů pro výslednou 3D prezentaci, aby bylo dosaženo jejich maximální kvality pro daný účel.

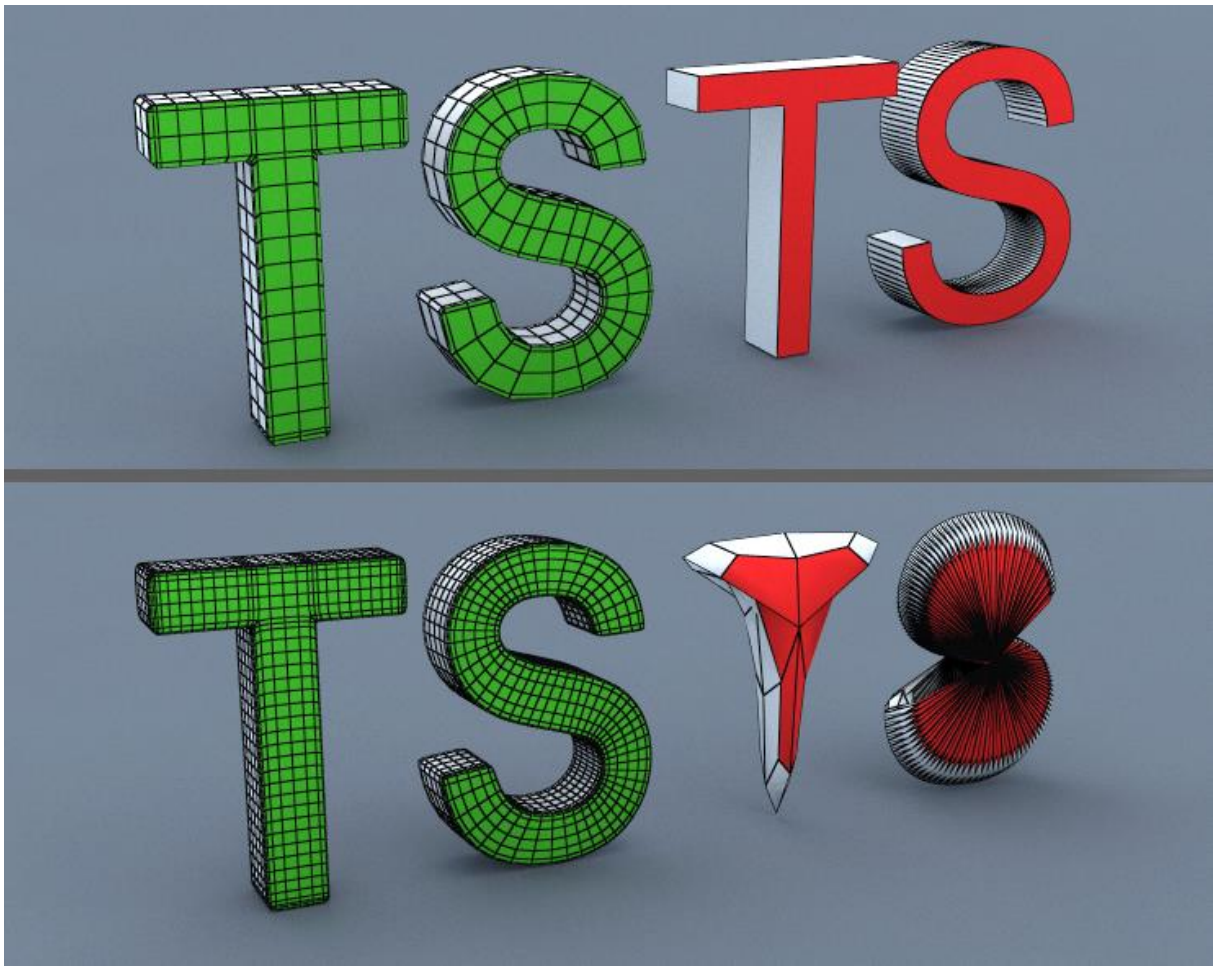
3.3.1 Polygony a jejich dělení

Jak již bylo zmíněno v minulých kapitolách, základem 3D modelu je pro většinu modelovacích nástrojů mřížka z *vertexů* propojených *okraji*. Jednotlivé body, tedy *vertexy* jsou provázány *okraji* tak, že vytváří tzv. *polygony*.

Nejjednodušší formou polygonu je trojúhelník. Tvoří ho 3 body, 3 hrany a plocha mezi nimi. Software produkty, které pracují s hotovými 3D modely, obvykle vlastními algoritmy převádí veškeré polygony v modelu na trojúhelníky.[18]

Nejdůležitější formou polygonu pro digitální tvorbu 3D modelů je čtyřúhelník, tedy čtyřhranný polygon. Většina technik 3D modelování se snaží dosáhnout toho, aby model byl složen výhradně ze čtyřúhelníků, a to z několika důvodů. Prvním je vizuální přehlednost topologie modelu a snadná modifikace mřížky sestávající se výhradně ze čtyřhranných polygonů. Na jeho povrchu jsou všechny zásadní hrany viditelné a snadno se tak předchází narušení jejich přesnosti. Druhým důvodem pro snahu o tvorbu modelů výhradně ze čtyřúhelníků je kompatibilita. Jedná se v tomto případě o zpracovatelnost modelu dalšími algoritmy (např. vyhlazování hran, boolean modifikátor apod.), ale také srozumitelnost modelu pro další software produkty, kterými bude zpracováván. Příkladem může být texturování modelu a redukce počtu polygonů pro použití ve vizualizaci nebo hře, vyvíjené jinou osobou, než je tvůrce modelu.[3, 18]

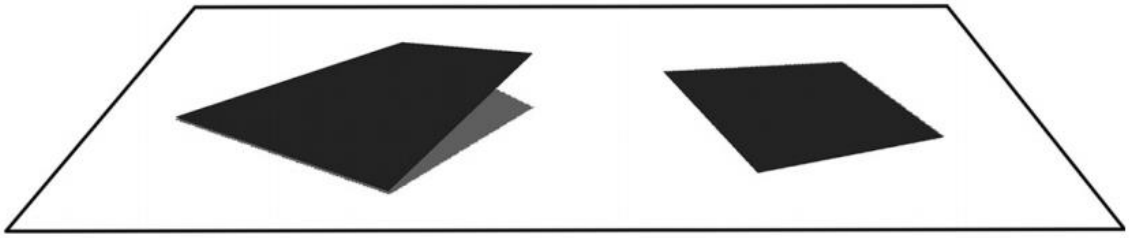
Posledním typem polygonu je *n-gon*. Nazývá se tak polygon s počtem hran a vertexů vyšším než 4. Jeho zpracovávání je výpočetně složitější než u čtyřúhelníků a trojúhelníků a podoba mřížky z *n-gonů* může být vizuálně nepřehledná. Zpracovávání modelu obsahujícího n-gony pomocí algoritmů jako je např. vyhlazování hran může mít neočekávané a potenciálně nežádoucí výsledky, stejně tak jako snaha o importování takového modelu do jiného vývojového prostředí. Vzhledem k těmto nevýhodám a faktu, že každý n-gon lze rozdělit na čtyřúhelníky, jsou polygony s více než čtyřmi hranami považovány v 3D modelování za nežádoucí a měly by být z modelu co nejdříve odstraněny.[3, 18]



Obrázek 3 - Příklad aplikace algoritmu subdivize na objekt ze čtyřúhelníků (vlevo) a n-gonů(vpravo)[18]

Polygony dále můžeme typově dělit na planární a neplanární. Toto rozlišení je možné provést pomocí určení souřadnic jednotlivých vertexů. Pokud jsou souřadnice vertexů takové, že je možné jimi proložit rovinu, pak je polygon planární. Ze své podstaty jsou trojúhelníky jako polygony vždy planární, protože právě jejich 3 body definují protínající rovinu. Podobně jako

n-gony jsou neplanární polygony obvykle ve 3D modelování nežádoucí, protože může dojít k neočekávaným výsledkům při dalším zpracování takto vytvořeného objektu.[3, 19]



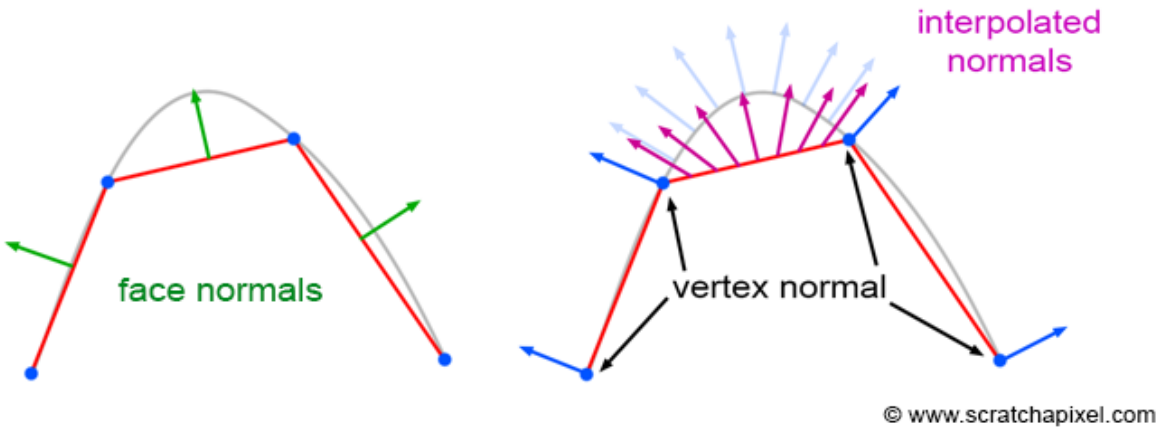
Obrázek 4 - Planární (vpravo) a neplanární (vpravo) polygon[3]

3.3.2 Normálové vektory a stínování

Každý polygon, ať už vygenerovaný nebo manuálně modelovaný má kromě dříve zmíněných vrcholů, hran a plochy také orientaci. Ta je dána tzv. normálovým vektorem polygonu. Normálový vektor, podobně jako v jiných oborech, je vektor takový, který je kolmý ke všem vektorům příslušné roviny, což v tomto případě znamená, že je kolmý k rovině rovnoběžné s povrchem polygonu. Za běžných okolností by měl být pro normálový vektor určen bod, ze kterého vychází, protože se jedná o 3D model, a nikoliv o rovinu. Složitost výpočtu by závisela na komplexitě a tvaru modelu. Protože ale víme, že je každý model sestaven z mřížky polygonů, které jsou zpravidla planární, pro související výpočty lze pro znázornění za počáteční body jednotlivých normálových vektorů považovat jejich těžiště. Také lze předpokládat, že v každém bodě polygonu je možné vztyčit normálový vektor se stejnou orientací a velikostí, jako ve zmíněném těžišti.[20, 21]

Výpočet normálových vektorů, a především jejich orientace má zásadní vliv na stínování a kalkulaci odrazů na povrchu modelu. To, jak povrch polygonu reaguje na světlo je určeno normálovým vektorem, úhlem, pod kterým dopadá na povrch světlo a také vlastnostmi virtuálního materiálu povrchu. Pokud nejsou dány žádné další vlastnosti materiálu, pak povrch odráží množství světla, závislé na úhlu mezi dopadajícím paprskem a normálovým vektorem. Odraz je při běžném výpočtu stejný na celé ploše polygonu, tato metoda se nazývá „*flat shading*“, tedy ploché stínování. V praxi má metoda plochého stínování využití u jednoduchých modelů, kde nezáleží na věrnosti, jinak ovšem naráží na problém spočívající v samotném principu, jak je 3D model vykonstruován. Vzhledem k tomu, že světlo dopadá na

mřížku modelu, u které jsou rozeznatelné její jednotlivé polygony, při použití plochého stínování nebude výsledný obraz působit věrohodně. Pokud je hustota polygonů taková, že je nelze pouhým okem rozeznat, pak je sice ploché stínování nezvýrazní, ze samotného počtu polygonů v takovém modelu ale plyne jeho extrémní výpočetní složitost. Touto problematikou se v minulosti zabýval Henri Gouraud (a také výše zmíněný Bui Tuong Phong) a přišel s metodou, která aplikuje stínování rovnoměrně po celém povrchu modelu a skrývá tak okraje jednotlivých polygonů. Děje se tak pomocí vypočtených normálových vektorů jednotlivých vrcholů, nikoliv ploch polygonů. Výpočet normálových vektorů pro vrcholy modelu probíhá lineární interpolací z normálových vektorů okolních ploch. Metoda se nazývá „smooth shading“, tedy hladké stínování.[20]

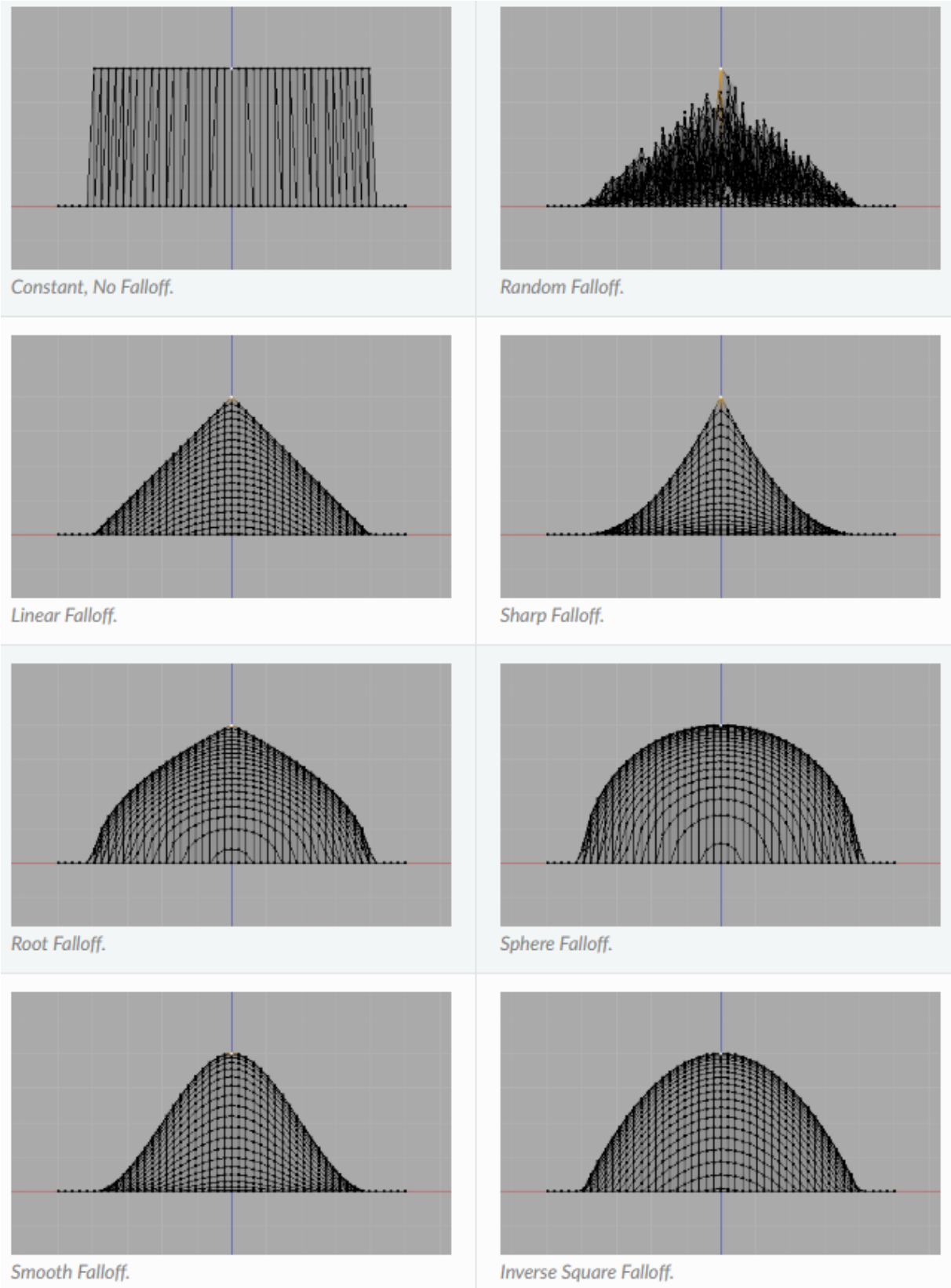


Obrázek 5 - Znázornění výsledku výpočtu interpolace normálových vektorů pro jemné stínování[20]

Díky této metodě může povrch modelu na pohled vypadat jednolitě i přesto, že při pohledu na mřížku modelu by byly patrné jednotlivé polygony a model je relativně výpočetně nenáročný. Způsob stínování ovlivňuje, jak je zobrazen povrch modelu, nikoliv však jeho silueta pro pozorovatele. Proto se může stát, že i při použití jemného stínování bude na okrajích modelu patrná nízká úroveň detailu s jednotlivými rozeznatelnými hranami a vrcholy. V takovém případě je možné použít v dalších kapitolách popsaný algoritmus subdivize, nebo v konfigurovat jemné stínování tak, aby algoritmus počítal s větším okruhem okolních povrchů.

3.3.3 Manipulace s mřížkou modelu

Základní formou editace 3D modelu je pohyb jednotlivými prvky jeho mřížky, tedy bodem (vrcholem), okrajem, nebo plochou polygonu. Je obvyklé, že je možné změnit souřadnice jednoho nebo více prvků pomocí běžných nástrojů pro posun, rotaci a měřítko. Funkce těchto nástrojů jsou z výpočetního hlediska popsány v kapitole 3.5. Editory často také nabízí pro tyto nástroje tzv. měkkou selekci, zajišťující proporcionální manipulaci s okolními prvky selekce tak, aby nedocházelo například k extrémním zlomům v mřížce modelu. Rozměr zóny, kde jsou polygony zasažené proporcionální editací je ve většině případů nutné manuálně nastavit tak, aby odpovídal požadavku uživatele. Například v editačním software Blender lze nastavit kromě rozsahu i tvar křivky, jímž se budou okolní prvky mřížky řídit.[3, 22]



Obrázek 6 - Příklady křivek pro proporcionální editování v nástroji Blender[22]

Pomocí nastavení velkého rozsahu a pseudonáhodného proporcionálního posunu lze modelovat nerovné povrchy objektů, nebo dokonce náhodně vypadající jednoduché krajiny. Metoda proporcionální editace souvisí úzce s dříve zmíněným organickým modelováním a je na ní založena většina nástrojů pro *sculpting*.[22]

3.3.4 Extruze

Pro hard-surface modelování je nejtypičtějším nástrojem extruze. Jedná se o techniku, kdy je zvolený prvek mřížky nejprve duplikován a poté přesunut podle požadavků uživatele. Společně s duplikací jsou také vytvořeny nové prvky propojující nově duplikovanou část a původní objekt. V případě, že je extrudován pouze vrchol polygonu, vygeneruje se kromě duplikátu hrana, jež je spojnicí mezi bodem původním a duplikátem. Pokud je extrudována plocha jednoho nebo více polygonů, pak jsou mezi původním objektem a extrudovanou plochou nově vygenerovány polygony na základě spojnic jednotlivých originálních a duplikovaných bodů. Algoritmus také sám určuje, které plochy se mají vymazat, aby nedošlo k nežádoucímu překrývání a generování neviditelných ploch uvnitř objektu.[23, 24]

3.3.5 Algoritmus subdivize

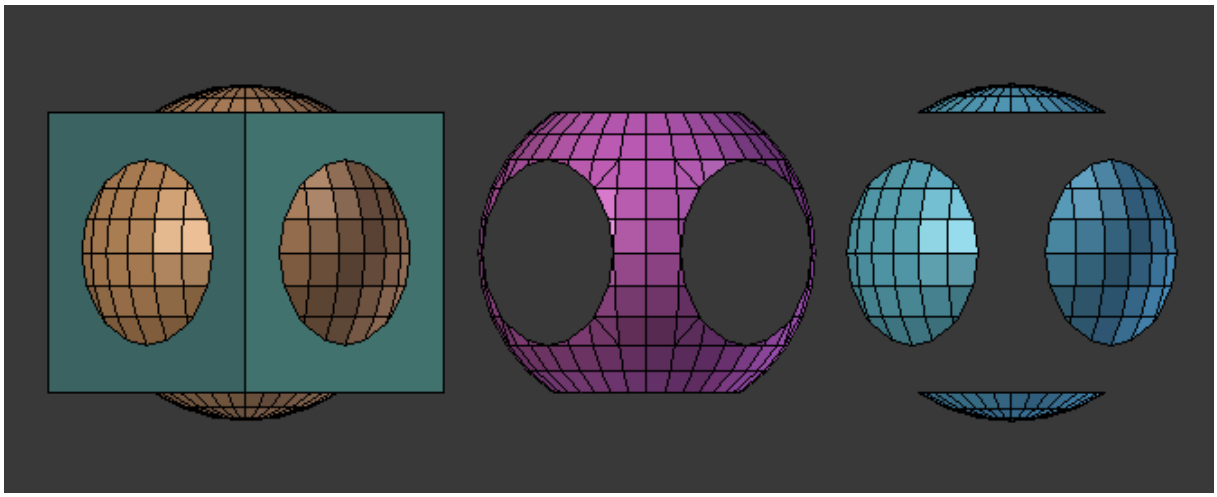
V minulosti probíhaly experimenty s možnostmi, jak dosáhnout ve 3D modelování zdánlivě hladkého povrchu. Řešením byly tzv. NURBS (non-uniform rational Bezier splines), tedy v překladu neuniformní racionální Bézierovy křivky. Jednalo se o kombinaci matematického přístupu k 3D modelování a dnes již standardního polygonálního. Tato metoda byla ve většině aplikací postupně nahrazena algoritmem pro tzv. *subdivizi*. Dnes se jedná o základní nástroj v 3D modelování, který zajišťuje vyhlazování povrchu objektu a teoreticky zvyšuje úroveň detailu. Dává tvůrci modelu možnost pracovat s mnohem větší hustotou polygonů na model, bez toho, aby se jeho editace významně komplikovala. Subdivize v moderních editovacích nástrojích pracuje podle Catmull-Clarkova algoritmu, který zjednodušeně říká, že budou vytvořeny nové vrcholy v polovině každé hrany, následně jsou tyto vrcholy na opačných stranách spojeny hranami a generují se vrcholy v bodech křížení těchto nových hran. Posledním krokem algoritmu je úprava souřadnic bodů objektu. Dochází k posunu původních bodů objektu tak, aby jejich souřadnice byly průměrem ze souřadnic bodů nově vytvořených. [25, 26]

Aplikováním Catmull-Clarkova algoritmu subdivize na objekt dojde k rozdělení každého polygonu na čtyři části, což značí, že čtyřúhelník bude pro tento algoritmus nejvhodnějším typem polygonu. Trojúhelníky, a především pak n-gony, mohou způsobit při subdivizi neočekávané výsledky, často dojde například k úplnému selhání při výpočtu souřadnic pro vyhlazení. Použití algoritmu subdivize pro vyhlazení a zvýšení detailnosti modelu je jednou z hlavních motivací pro správné udržování topologie objektu a sestavování mřížky výhradně ze čtyřhranných polygonů.[25]

3.3.6 Boolean modifikátor a spojování objektů

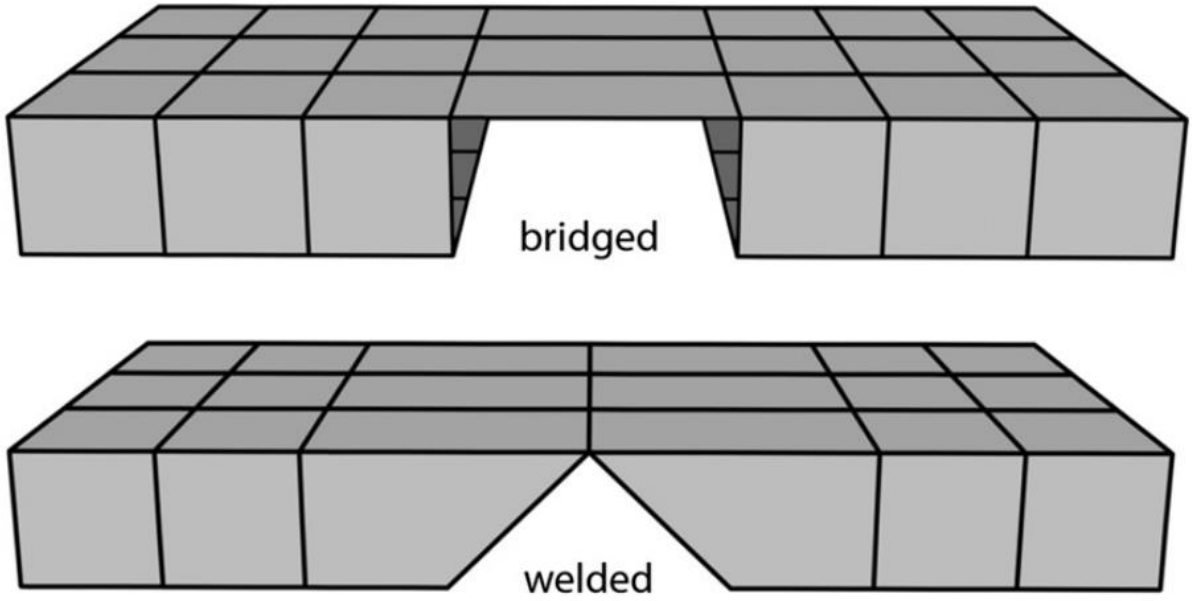
Během procesu modelování komplexního 3D objektu je často vhodné postupovat tak, že jsou jeho části modelovány zvlášť a poté propojeny. To je výhodné nejen z důvodu zjednodušení samotného modelování, ale také tento postup napomáhá při budování vlastní knihovny jednoduchých částí modelů, které jsou znovu použitelné a můžou tak přispět k výraznému urychlení budoucí práce. Jednou z možností jak kombinovat mřížky více objektů je modifikátor boolean, dostupný téměř ve všech moderních SW nástrojích pro 3D modelování, včetně těch specializovaných na modelování organické.[3]

Boolean modifikátor zpravidla podporuje vstupy ve formě dvou objektů, jeden z nich je považován za modifikovaný, druhý za „cílový“. S těmito zvolenými objekty provádí jednu ze třech vybraných operací. První z nich je „spojení“ (union), kdy jsou mřížky obou objektů co nejlépe sloučeny a vzniká jeden nový objekt. Druhou operací je „rozdíl“ (difference). Zde dochází k vyříznutí obsahu cílového objektu z objektu modifikovaného, tedy z původního modifikovaného objektu zůstává takový obsah, který se nepřekrývá s objektem cílovým. Poslední operací je „průnik“ (intersection), kde je z modifikovaného objektu zachována právě ta část, překrývající se s objektem cílovým. Jde tedy v určitém smyslu o opak operace „rozdíl“.[27]



Obrázek 7 - Aplikace modifikátoru boolean: (zleva) spojení, průnik a rozdíl[27]

Použití boolean modifikátoru může vést k nežádoucím efektům na topologii modelu. Pokud mají například dva objekty jinou úroveň detailu (počty polygonů v poměru k rozměru objektů), pak může dojít u výsledného objektu k narušení struktury mřížky. Častými jevy jsou například vytvoření polygonů s plochou uvnitř objektu, zdvojení hran a místy se vyskytující zvýšená hustota polygonů. Tyto jevy mohou zapříčinit potíže při dalších úpravách modelu (např. použití algoritmu subdivize), proto je nutné nevhodnou topologii po použití boolean modifikátoru opravit, aby opět vznikla mřížka s rovnoměrně rozloženými vrcholy a v ideálním případě složená pouze ze čtyřúhelníků. Pro objekty s menším počtem polygonů, nebo objekty, u kterých je nezbytně nutné zachování co největší části původní mřížky se použití boolean modifikátoru pro spojení nedoporučuje, místo toho je možné dvě mřížky spojit manuálně. Operace se provádí výběrem požadovaných spojnicových hran a vrcholů mřížek a provede se jejich kombinace buď přemostěním (vytvoří se nové polygony a jsou zachovány souřadnice vrcholů), nebo sloučením (zprůměrují se souřadnice dvojic vrcholů, dojde ke sloučení, nevznikají nové polygony). Vznikne tak objekt, který je kombinací dvou zvolených a jehož mřížka byla narušena zcela minimálně. Tuto operaci je možné v editorech automatizovat, pokud mají dva objekty stejný počet spojovaných vrcholů a hran.[3, 28]



Obrázek 8 - Znázornění kombinace pomocí přemostění (nahoře) a sloučení (dole)[3]

3.3.7 Vykreslování modelů pomocí GPU

Polygony jsou na hardwarové úrovni zpracovávány grafickým procesorem (GPU). Ten může mít formu dedikované grafické karty, nebo může být integrován do jednotky procesoru. Grafický procesor je specializovaný mikročip, určený k co nejrychlejšímu zpracování vykreslovaných dat.

Jedním z hlavních prvků pro určení výpočetní složitosti modelu je počet polygonů. Obecně lze tvrdit, že čím větší počet polygonů v modelu, tím pomaleji je vykreslován. Hrubý počet ale není jediným parametrem určujícím tuto vlastnost, proto budou níže popsány principy vykreslování polygonů grafickým procesorem, společně s dalšími vlastnostmi modelu schopnými ovlivnit jeho výpočetní náročnost.

Jak již bylo zmíněno, polygon se skládá z vrcholů (vertexů) a hran, tedy vztahy (reference) mezi vrcholy. Ty jsou ukládány jako pole hodnot, kde každý prvek pole reprezentuje jednu ze souřadnic vrcholu a každá trojice tedy definuje jeden vrchol. V druhém poli je pak zaznamenán polygon, a to pomocí identifikace vrcholů, které ho tvoří. GPU považuje každý polygon za trojúhelník, a protože modely vytvořené korektně jsou složeny z čtyřúhelníků (kvůli přehlednosti topologie), musí být před hardwarovým zpracováním převedeny. Při

počítání polygonů v modelu je nezbytné brát tuto konverzi na vědomí a místo polygonů výpočet založit na počtu trojúhelníků.[29, 30]

Při vykreslování polygonu grafický procesor nejprve určí jeho polohu, nazývanou „transform“. Po vypočítání polohy GPU určuje pixely aktuálně obsažené ve vykreslovaném trojúhelníku a následně je vyplňuje podprogramem „fragment shader“, jinak také nazývaným „pixel shader“. Tento proces vykreslování pomocí pixelů se obvykle nazývá rasterizace (Open.GL uvádí použití *fragment shaderu* jako samostatný proces) a z celého procesu výpočtu a vykreslení polygonu zabírá většinu času (více než polovinu). Protože modely jsou tvořeny mřížkou, jejich polygony jsou propojeny. To znamená, že jednotlivé vrcholy se u jednotlivých polygonů opakují (polygony vedle sebe sdílí vrcholy). Aby nedocházelo k redundantnímu výpočtu polohy již známých vrcholů, je zavedena tzv. „Vertex cache“, tedy paměť pro ukládání vrcholů. Díky této paměti je v praxi možné vykreslit čtyřúhelník rozdělený na trojúhelníky v téměř stejném čase, jako by trvalo vykreslit původní čtyřúhelník bez rozdělení. Samozřejmě tento předpoklad není možné zcela obhájit, protože grafický procesor není optimalizován na vykreslování čtyřúhelníků. Paměť pro ukládání vrcholů nemůže být zcela využita, pokud model nemá řádně propojené polygony, proto je vhodné se vyvarovat neúmyslného rozdělování hran modelu.[29, 30]

Kromě počtu trojúhelníků a jejich spojení má na výkon při vykreslování významný vliv také jejich tvar. GPU přiřazuje pro vykreslení polygonu tzv. pipelines a zpracovává čtverce, typicky o hraně s velikostí cca 8 pixelů. Pokud má vykreslovaný polygon (trojúhelník) nerovnoměrný tvar (např. jedna strana výrazně kratší než dvě zbývající), pak musí grafický procesor jeho vykreslení rozdělit na více těchto čtverců, což vede k omezení prostředků použitelných na další paralelní operace. Model s korektní topologií tak může být při stejném počtu polygonů vykazovat mnohem nižší výpočetní náročnost, než jeho navenek identický protějšek.[29, 30]

Posledním zmíněným prvkem ovlivňujícím rychlost vykreslování je „overdrawing“, tedy vícenásobné vykreslení. Tento jev nastává například pokud obsahuje polygony, které se překrývají, obvykle za účelem snížení jejich celkového počtu. Tento postup je považován za nekorektní, protože ačkoliv je počet polygonů nižší, dochází na společných plochách k jejich dvojitému vykreslování, což vede k vyšší výpočetní náročnosti, než kdyby byla topologie

modelu správná. Dalším, častějším případem, kdy může k vícenásobnému vykreslování dojít je při tvorbě scény, kde dva objekty leží na sobě. Typicky se jedná o budovy, prvky terénu a dekorace položené na zemi. Plocha země a spodní strana objektu je v tomto případě částečně sdílená a pro zrychlení výkonu při vykreslování je možné sdílené polygony odstranit. Tento postup je možné aplikovat pouze pokud jsou modely ve scéně statické, nebo nikdy nedojde k jejich pohybu tak, že by bylo možné tuto optimalizaci vizuálně poznat. „Vykrajováním“ děr do jednoduchých objektů (např. rovin) dochází také ke zvyšování počtu polygonů, je tedy nutné individuálně zvažovat, zda se tento typ optimalizace modelu vyplatí.[29, 31]

3.3.8 UV Mapping

Aplikace textur je podstatnou částí procesu 3D modelování. Pokud má model působit realisticky, musí existovat představa o podobě jeho materiálu a barvě. Povrch cílového objektu může mít drobné nerovnosti, specifické barevné přechody a podobně. Jednou z možností, jak přidat na povrch modelu nerovnosti, je jejich přímé modelování pomocí mřížky. To může být podle Wanga enormně obtížné, zvláště jsou-li tyto prvky rozmístěné po celém objektu, mají nepravidelný nebo dokonce nejasný tvar a jejich rozměry jsou mnohonásobně menší než rozměry samotného objektu. Kromě složitosti z pohledu tvůrce je nutné zmínit také složitost výpočetní. Model by musel být (například algoritmem subdivize) zpracován tak, aby byly jednotlivé polygony dostatečně hustě rozmístěné k modelování drobných nerovností. Výpočetní složitost by tak mohla růst až exponenciálně. Proto je pro reprezentaci materiálu povrchu vhodné na model pomocí UV map nanést textury, které budou působit dostatečně věrohodně, a to bez drastického nárustu výpočetní složitosti modelu.[32]

UV mapa představuje dvojrozměrnou reprezentaci povrchu modelu. Písmena „U“ a „V“ v pojmu „UV mapa“ značí horizontální, respektive vertikální osu ve 2D prostoru. Toto označení je použito především z důvodu, že osy X, Y a Z jsou při modelování obsazeny pro reprezentaci 3D prostoru, ve kterém se objekty nachází.[33]

Objekt se do dvojrozměrného prostoru převádí pomocí tzv. rozložení polygonů. Povrch modelu je rozdělen podle hran. Ty mohou být buď určeny počítačem (automatický UV wrapper), nebo ručně. Vznikne tak jeden nebo více již dvojrozměrných povrchů, na které je

již možné aplikovat textury. Z teoretického hlediska optimalizace je doporučováno, aby při „rozbalování“ modelu vznikl minimální počet rozdělených hran, a to z důvodu výpočetní složitosti. Při takové optimalizaci dochází ovšem ke zkreslení tvaru polygonů v UV mapě, což vede k nežádoucím efektům při aplikaci textur. Proto je v praxi nutné více dbát na zamezení zkreslení tvaru polygonů a počet dělených hran se obvykle odvíjí od komplexnosti modelu, a dále kvality a přehlednosti jeho topologie. [33, 34]

Přístupy k UV mapování se mohou lišit podle toho, jestli je model považován za organický nebo hard-surface. Organické modely zpravidla nemají velké množství ostrých okrajů a zkreslení polygonů na nich není do takové míry patrné, proto je jejich mapování často cílené tak, aby vznikl co nejmenší počet dělených hran, které budou navíc pokud možno skryté (v oblastech modelu, které buď nejsou vidět vůbec, nebo kde nebudou na první pohled patrné). U hard-surface modelů je kladen větší důraz na přesnost zobrazení polygonů, proto se model dělí v místech, kde má ostré hrany a jednotlivé sekce jsou pak do UV mapy rozmisťovány s rozestupy, aby nedocházelo k překrývání textur.[34]

3.3.9 Metody aplikace materiálů

Tvorba materiálů se ve 3D považuje za rozšíření systému aplikace textur na objekty. Cílem jejich aplikace je simulace podoby povrchu podle reálných vlastností předlohy. Oproti běžným plochým texturám materiály simulují vlastnosti objektu jako je například míra rozptylu a odrazu světla, průhlednost a odstín světelného odrazu. Díky tomu lze na 3D modelu například rozlišit, zda je modelovaný objekt vyroben z kovu, umělé hmoty, nebo skla. Nástroje pro 3D modelování obvykle obsahují možnost přiřazení tzv. základního materiálu k objektu. Ten je považován za uniformní, jeho vlastnosti jsou tedy stejné na každém polygonu povrchu modelu. Díky tomu je možné takovýto materiál aplikovat na libovolný model, bez nutnosti UV mapování a přiřazování textur. Některé modelovací nástroje dokonce nabízí generátory procedurálních materiálů i s texturou, které mohou až téměř dokonale simulovat všechny vlastnosti povrchu objektu bez nutnosti manuálního mapování. Nevýhodou tohoto typu materiálu je, že je obvykle unikátní pro daný modelovací nástroj a je tedy těžko využitelný mimo jeho vlastní prostředí (např. pro použití ve hře, nebo jiném renderovacím SW).[35]

Materiály, které mají být univerzálně využitelné jsou založené na UV mapách, na rozdíl od jedné ploché textury je ale při tvorbě materiálu vytvořeno vícero dvojrozměrných map, znázorňujících rozdílné vlastnosti povrchu.

Rozlišují se obecně dva přístupy k tvorbě materiálu, tzv. *Specular workflow* a *Metallic workflow*, tedy spekulární a metalický postup/přístup. Materiál vytvořený metalickým přístupem je někdy také označen jako PBR (physical-based rendering) materiál. Tento přístup se totiž vyznačuje zaměřením na základní fyzikální vlastnosti objektu a jejich přímé mapování. Skládá se ze 4 základních map, které mohou být doplněny dalšími dle potřeby. První z map v metalickém přístupu k tvorbě materiálu je Albedo, která vyjadřuje výhradně barvy objektu a nesmí na ní být patrné žádné z jeho jiných vlastností (jako např. drobné stíny nerovností). Mapa Albedo je tvořena v barevném prostoru sRGB. Druhá mapa se nazývá Metallic a popisuje, do jaké míry se části modelu chovají jako kov, což ovlivňuje jejich reakci na světlo. Konkrétně se zde jedná o změnu odstínu odráženého světla podle barvy a typu kovu. Mapa je produkována v lineárním barevném prostoru, v odstínech šedé, kde světlejší odstín znamená, že materiál vlastnostmi více připomíná kov. Pro simulaci reálných kovů bude mít mapa odstíny šedé v rozmezí 75-98 %. Třetí mapou je Roughness, která vyznačuje hrubost materiálu, což se projeví na přesnosti odrazu světla. V praxi se projevuje mírou rozostření zrcadleného obrazu na objektu. Podobně jako mapa Metallic je Roughness mapa v lineárním barevném prostoru, v tomto případě černá (odstín šedé 0 %) představuje dokonale hladký povrch. Poslední mapou v metalickém přístupu je tzv. mapa Ambient Occlusion (AO), někdy překládáno jako ambientní okluze, jindy jako osvětlení okolí. AO mapa obsahuje především informace o stínování nerovností povrchu. Jedná se o měkké stíny vytvořené neexistujícím světelným zdrojem, aby nebyl zřejmý směr, ze kterého byl objekt nasvícen. Tento efekt vyvolává při správném použití dojem, že má model mnohem jemnější a detailnější strukturu než ve skutečnosti, což hraje významnou roli při optimalizaci modelů pro render nebo další aplikace. Metalický přístup má ještě dvě doplňkové mapy, Parallax a Výškovou, které dohromady slouží k realistickému zobrazení nerovností povrchu.[31, 36]

Spekulární přístup k tvorbě materiálu pomocí map je oproti metalickému založen především na viditelných a snáze popsatelných vlastnostech povrchu. První mapou produkovanou v tomto přístupu je Diffuse, která podobně jako Albedo obsahuje především informace

o barvě povrchu v modelu sRGB. Další mapou je Reflection, jindy nazývaná jako Specular. Zahrnuje informace o odrazových vlastnostech povrchu a jeho nerovností. Díky této mapě je možné podobně jako u Ambient Occlusion simulovat detaily, které nejsou reálně na povrchu objektu modelovány. Mapa Specular je tvořena v barevném prostoru sRGB. Třetí hlavní mapou pro spekulární přístup je Glossiness, mapa v lineárním prostoru odstínů šedé, která ukládá informaci o lesklosti materiálu, podobně jako to dělá mapa Roughness. Nedoporučuje se používat v Glossiness mapě odstíny šedé s hodnotou nižší než 50 %. Doplňkovou mapou k tomuto přístupu je Displacement, která umožňuje simulaci výrazných nerovností na povrchu objektu (např. štěrk). Tato mapa se používá především pro rendering statických scén, kde je automaticky převedena na dočasnou modifikaci mřížky objektu.[31, 35, 36]

Materiály mohou být doplněny i dalšími mapami, pokud základní mapy obou přístupů nestačí pro vyjádření všech jejich vlastností. Jednou z častých doplňkových map je například mapa normálová, která usnadňuje simulaci větších nerovností povrchu, nebo mapa vlastního osvětlení, vyjadřující, že části povrchu objektu jsou specifickými zdroji světla. Dále pak existuje speciální mapa pro vyjádření lokální úrovně průhlednosti objektu.[35, 36]

3.3.10 Alternativní způsoby získání modelů

Kromě manuálního modelování existují další způsoby, jak generovat 3D modely například pomocí technik z oblasti fotogrammetrie. Ta se zabývá získáváním informací o objektech pomocí snímání, měření a interpretací fotografických dat pro vytvoření virtuálních modelů. Obecně se metody dělí podle vzdálenosti, ze které jsou cílové objekty pozorovány. Provádí se tak například snímání pro tvorbu 3D map, nebo modelů uměleckých děl. Fotogrammetrie může kromě fotografií zpracovávat také data z laserových snímačů vzdálenosti, GPS a dalších přístrojů, umožňujících kvalitnější zpracování výsledného objektu.[37, 38]

Pro extrakci 3D dat objektu z fotografie jsou teoreticky potřeba alespoň dvě fotografie z různých úhlů. V praxi je ale tento počet mnohem vyšší a pro vygenerování přesné trojrozměrné virtuální reprezentace objektu může být podle jeho parametrů potřeba fotografií až stovky. Pro tento typ zpracování se používá například technika Structure from Motion (SfM), založená na vyhledávání společných bodů v mnoha snímcích, které jsou následně propojeny a je vygenerován odhad 3D topologie sledovaného objektu. Jeho přesnost závisí na

nastavení daného nástroje a na kvalitě pořízených snímků. Nevýhodou techniky SfM je relativní komplexita výsledného modelu z hlediska počtu polygonů. Ten je zpravidla vyšší, než by byl při manuálním modelování stejného objektu tak, aby bylo dosaženo požadované přesnosti.[37]

Pro generování 3D dat můžou být také využity nástroje na principu strojového vidění, které rozpoznají určité typy objektů ve 2D obraze a podle toho je proveden příkaz k jejich vykreslení podle definovaných vzorů. Takto se může provádět například modelování lesů a běžných budov pro trojrozměrné mapy.[37, 38]

3.4 Rendering

Rendering je proces generování dvojrozměrného, nebo třírozměrného obrazu. Obraz je generován z modelu pomocí softwarového nástroje, specifika a použité techniky se mohou lišit např. podle účelu.[39]

Kapitola se zabývá výběrem technik a pojmů souvisejících s renderováním. Zabývá se problematikou z pohledu programového, tedy jak software nástroj dosahuje požadovaných výsledků pomocí specializovaných algoritmů, vycházejících z fyzikálních zákonů. Dále jsou popsány vybrané související procesy a techniky z pohledu uživatele, díky kterým je možné dosáhnout realistických vyobrazení objektů podle předlohy, nebo věrohodných reálně neexistujících objektů. Příkladem může být kompozice, výběr světelných zdrojů a jejich nastavení, nebo pozice virtuální kamery. Výsledkem renderování může být statický obraz, animace, nebo v případě zpracování v reálném čase může být renderování součástí interaktivní aplikace.

3.4.1 Typy renderingu

Nástroje pro rendering 3D objektů a scén lze v praxi dělit na dvě základní kategorie. Tradiční nástroje jsou založeny na precizních výpočtech trasování cest paprsků ze světelných zdrojů. Paprsky reagují na tvar a materiál objektů ve scéně a jsou podle těchto parametrů odráženy nebo pohlcovány. Díky tomu dochází k věrohodné simulaci zrcadlových efektů, stínování je přesně zachyceno podle typu a vzdálenosti světelných zdrojů a celková věrohodnost zobrazení scény záleží na úrovni detailu modelů a kvalitě kompozice. Rendering jednoho snímku

takovým nástrojem může trvat zhruba v rozmezí od několika minut až po několik hodin, v závislosti především na komplexitě scény. Tradiční renderovací nástroje se využívají pro generování statických scén a animací, kde záleží na přesnosti a cílem je maximalizovat výslednou kvalitu obrazu. Proto se tyto nástroje používají například ve filmovém průmyslu, kde se jednotlivé scény navíc dělí na tzv. „průchody“ (např. prostředí, hlavní objekt, sekundární zdroje světel, speciální efekty), které jsou renderovány zvlášť a v pozdějších fázích jsou sestaveny do finální podoby.[40]

Druhou kategorií renderovacích nástrojů jsou takové, co dokážou produkovat finální podobu obrazu „v reálném čase“, tedy jsou schopny produkovat více než 24 snímků za sekundu. Tyto nástroje jsou využívány především pro interaktivní simulace a hry. Dále pak najdou uplatnění pro tvorbu scén a animací, pokud není k dispozici dostatečný výpočetní výkon pro použití tradičního nástroje. Real-time renderovací nástroje jsou optimalizovány tak, aby byl minimalizován čas na vykreslení každého snímku. Mohou být, podobně jako nástroje tradiční, založeny na trasování jednotlivých světelných paprsků, avšak z důvodu výpočetní náročnosti je tento proces zjednodušen, počet sledovaných paprsků je nižší a výsledné nasvícení objektů je určeno pomocí aproximačních algoritmů. [40]

Následující podkapitoly se zabývají principy, na kterých pracují nástroje pro renderování v reálném čase, většina z nic je ale platná i pro nástroje tradiční.

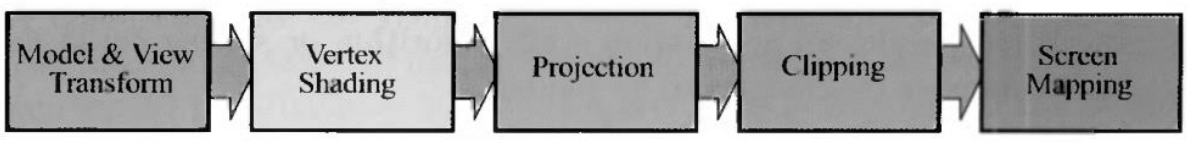
3.4.2 Rendering Pipeline

V dřívější kapitole práce *3.3.7 Vykreslování modelů pomocí GPU* je stručně vysvětlen postup, jakým jsou vykreslovány polygony se zaměřením na optimalizace jejich tvaru, počtu a překrývání. Tato kapitola se zabývá samotným procesem renderování a jeho strukturou, která je neměnná v závislosti na konkrétní specifika modelu. Struktura, jakou se rendering řídí, se nazývá „rendering pipeline“. Pojem pipeline se v oboru překládá jako „roura“ nebo „potrubí“, častější je ale použití původního anglického pojmu. Rendering pipeline popisuje a dělí podprocesy, které vedou k výslednému zpracování trojrozměrné scény do podoby dvojrozměrného obrazu vykresleného na obrazovce. Dělí se na tři základní fáze: aplikační (angl. „Application“), geometrickou (angl. „Geometry“) a rasterizační (angl. „Rasterizer). Z těchto abstraktních fází může každá obsahovat vlastní pipeline nižší úrovně, např. v případě

rasterizační a geometrické fáze můžou být některé kroky v pipeline nižší úrovně zpracovávány paralelně. Paralelní zpracování probíhá i na nejvyšší úrovni, rasterizační fáze objektu probíhá současně s geometrickou fází dalšího objektu nebo součásti, zároveň již probíhají výpočty v aplikační fázi pro ostatní části scény. Platí zde teoretický princip pipeline, který říká, že rychlost zpracování celku se rovná rychlosti zpracování nejpomalejšího kroku nebo fáze. V následujících odstavcích budou stručně popsány tři hlavní fáze renderovací pipeline.[41, 42]

Aplikační fáze renderovací pipeline je z velké části řízena přímo vývojářem, který určuje její implementaci a má tak možnost docílit nárustu výkonu díky vlastní optimalizaci. Změny v aplikační fázi mohou ovlivnit rychlost zpracování v dalších fázích pipeline, publikace Real-time Rendering uvádí jako příklad, že v aplikační fázi je možné provést minimalizaci potřebných trojúhelníků k vykreslení, což povede ke zvýšení vykreslovacího výkonu. Obsah aplikační fáze je obvykle zpracováván procesorem namísto GPU. Důsledkem toho, že je tato fáze implementována na softwarové úrovni, není rozdělena na podprocesy a z teoretického hlediska je zpracovávána jako jeden souvislý proces. Díky konstrukci procesoru je ale v praxi tento proces zpracováván paralelně ve vícero vláknech. V aplikační fázi jsou implementovány například procesy animací objektů a textur, nebo zpracování signálů ze vstupních zařízení. Dále je často v aplikační fázi implementována detekce kolizí. Pokud je výpočtem zjištěna kolize dvou objektů ve scéně, může být naprogramována reakce, např. signál pro dané objekty k zastavení pohybu nebo změně směru. Výstupem aplikační fáze jsou geometrická data v podobě bodů (vertexů), hran a trojúhelníků.[41]

Druhou hlavní částí renderovací pipeline je geometrická fáze. Ta je obecně zaměřena na operace, prováděné na úrovni polygonů a vrcholů. Podle publikace Real-time Rendering je tuto fázi možné dělit celkem na pět podprocesů: Transformace modelu, Vertex Shading, projekce, clipping a screen mapping.[41]



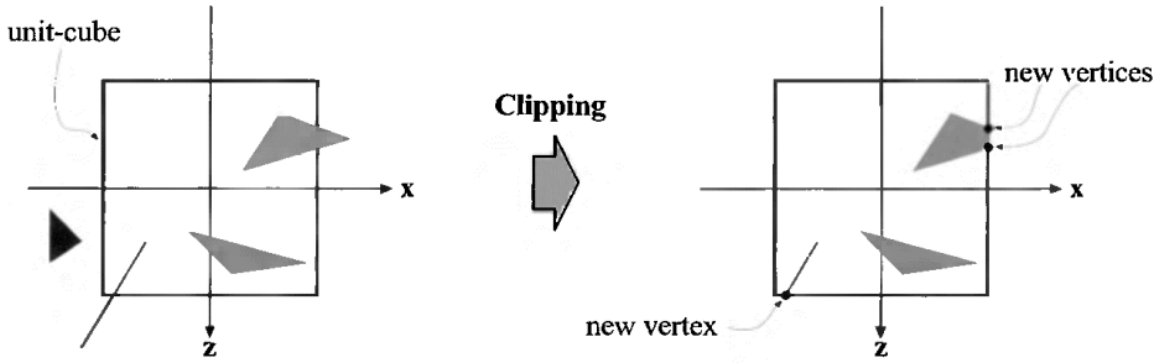
Obrázek 9 - Geometrická fáze znázorněná jako pipeline podprocesů[41]

Subfáze Transformace modelu se zabývá určením, jestli je každý daný model ve scéně viditelný a z jakého úhlu by byl pozorován. Každý model může být ve scéně zobrazen několikrát, může tedy mít i několik variant polohových souřadnic. To vede k vylepšení výkonu, protože není nutné zpracovávat geometrii pro každou kopii (instanci) zvlášť, ale stačí pouze aplikovat souřadnice polohy a otočení. Modely jsou postupně přeneseny do scény, přičemž jsou přepočítány jejich souřadnice podle nového nulového bodu, který se přesouvá do ohniska objektu virtuální kamery. [31, 41]

Druhý podproces „Vertex Shading“ se zabývá výpočtem stínování objektů a aplikací materiálů. V této fázi jsou výpočty prováděny na úrovni vrcholů polygonů, v rasterizační fázi pipeline probíhají další výpočty podobného charakteru na úrovni pixelů výsledného obrazu.[41, 42]

Třetí částí geometrické fáze renderovací pipeline je tzv. projekce. V této části probíhá transformace objemu a „tvaru“ záběru kamery na jednotkovou kostku. Aplikuje se tak jedna z projekčních metod. Dvě nejpoužívanější metody projekce jsou ortografická a perspektivní. Ortografická projekce se vyznačuje zachováním rovnoběžek a rozměrů objektu bez ohledu na vzdálenost od kamery. Záběr kamery je tedy tvarován jako krychle. V perspektivní projekci je tvarem záběru tzv. frustum, tedy čtyřstranný jehlan se seříznutou horní částí. Objekty vzdálenější od kamery se jeví jako menší a rovnoběžné hrany jsou v projekci zobrazeny jako konvergující.[41, 43]

Podproces Clipping má za úkol určit, které primitivní objekty (hrany, polygony atd.) budou předány do rasterizační fáze na základě toho, zda jsou alespoň částečně obsaženy v záběru kamery. Objekty zcela mimo záběr kamery jsou v této fázi vyřazeny, protože nedochází k jejich vykreslování. Objekty, které jsou v záběru zachyceny jen z části jsou v této fázi seříznuty přesně podle okrajů záběru. [41]



Obrázek 10 - Znázornění průběhu podprocesu Clipping[41]

Posledním podprocesem geometrické části renderovací pipeline je Screen Mapping. Tato fáze má na vstupu trojrozměrné souřadnice výsledných objektů po provedení clippingu. Screen Mapping je proces, kterým jsou tyto souřadnice převáděny na dvojrozměrné, které zároveň odpovídají souřadnicím výstupního okna na obrazovce. Způsob výpočtu těchto souřadnic se může lišit podle typu a verze grafického programovacího rozhraní (API), rozdíly jsou především v přístupu k přepočtu desetinných hodnot souřadnic a dělení výsledných pixelů.[30, 41]

Jako poslední fáze renderovací pipeline je prováděna rasterizace. Fáze rasterizace má obecně za úkol určit barvu pixelů, dohromady pak tvořících obraz objektu. Obecně jde o opětovný převod souřadnic bodů v poli záběru, aplikaci stínování a dalších metod, jejichž výsledkem jsou konkrétní pixely na zobrazovacím zařízení, tedy dochází k vykreslení výsledného obrazu. Podobně jako v geometrické fázi i tuto část lze dělit na několik podprocesů.[30, 41] [42]

První podproces této fáze, Triangle Setup, připravuje data o povrchu polygonů, která budou dále využita například pro interpolaci dat stínování a samozřejmě také samotnou konverzi povrchu na výsledný obraz. Tato operace je obvykle prováděna na vyhrazené části grafické karty nebo integrovaného grafického čipu v procesoru. Druhý podproces, Triangle Traversal, zjišťuje, které výsledné pixely budou aktivní součástí vykreslení polygonu (trojúhelníku). Protože pixel nemusí plochu trojúhelníku zcela překrývat, je generován fragment trojúhelníku pomocí interpolace dat z jeho vrcholů. Tento podproces také aplikuje některá data o stínování z geometrické fáze pipeline.[41]

Třetím podprocesem je Pixel Shading. Jak již název napovídá, součástí tohoto podprocesu jsou veškeré operace na úrovni jednotlivých pixelů. Kvůli různorodosti operací je obvykle využíváno programovatelných jader grafického procesoru, na rozdíl od pevně nastavených hardwarových řešení určených pro podprocesy Triangle Setup a Triangle Traversal. Jednou z nejdůležitějších operací tohoto podprocesu je aplikace textur. Způsob jejich tvorby je podrobněji popsán v kapitole 3.3.8 *UV Mapping* a 3.3.9 *Metody aplikace materiálů*. Podproces používá interpolovaná data o stínování jako vstup a výsledkem podprocesu Pixel Shading je jedna nebo více barevných kombinací pro každý pixel.[41, 42]

Poslední fází rasterizace je podproces sloučení. Vypočtená data pro jednotlivé pixely jsou uložena v tzv. zásobníku barev (color buffer), který má formu barevného pole (prvky červená, zelená a modrá podle barevného modelu RGB). Hlavním úkolem podprocesu je zkombinovat tyto prvky uložené v zásobníku s těmi, které přichází z minulých fází pipeline. Dále podproces řeší problematiku viditelnosti objektů, obvykle pomocí tzv. Z-buffer, tedy zásobníku hloubky. Z-buffer funguje na podobném principu jako color buffer, jedná se o pole stejných rozměrů. V poli jsou obsaženy hodnoty Z, určující vzdálenost od ohniska virtuální kamery k nejbližšímu objektu (polygonu). Při každém průchodu vykreslování je pro každý pixel vypočtena tato hodnota a porovnána s předchozí. Pokud je nově vypočtená hodnota nižší (objekt je v záběru blíže a překrývá původní), pak je původní pixel nahrazen novým, tedy je přepsána hodnota color buffer a Z-buffer. V opačném případě je původní hodnota v obou zásobnících zachována. Se zásobníkem barev je úzce spojen alpha kanál, který se zabývá průhledností na úrovni pixelů. Dalším zásobníkem je tzv. stencil buffer, ukládající polohy renderovaných objektů (součástí) do osmibitových hodnot na každý pixel. Jednoduché objekty mohou být renderovány do tohoto zásobníku a z něj je dále možné vykreslovat do zásobníku barev a hloubky. Stencil buffer je vhodným nástrojem pro vytváření speciálních efektů.[41, 42]

Výsledný obraz, tedy obsah všech zásobníků, je ukládán do zásobníku snímků (frame buffer). Protože obsah tohoto zásobníku by byl přímo viditelný na obrazovce a není generován v nulovém čase, používá se pro finální vykreslování technika double buffering. Vykreslování popsané v minulých odstavcích probíhá na pozadí a poté je přeneseno do popředí, zatímco v pozadí je zahájeno vykreslování dalšího snímku.[30, 41, 42]

3.4.3 Osvětlení objektů a prostředí

Tato kapitola pojednává o principech osvětlování objektů a scén ve 3D prostředí z jak z hlediska technického, tak i z hlediska kompozice a principů například z oblasti fotografie. Hlavním prvkem v osvětlení scény jsou samotné zdroje světla. Ty mají v reálném světě specifické charakteristiky, díky kterým je například možné z fotografie rozeznat, v jakém prostředí se sledované objekty nachází a pomocí kompozice scény je možné rozlišit, který objekt je považován za hlavní a co má daná scéna pozorovateli ukázat, či sdělit. Při generování virtuální scény pomocí počítače jsou zdroje světla obvykle plně konfigurovatelné a lze rozlišit níže popsané kvantifikovatelné vlastnosti, jejichž správnou kombinací lze docílit věrné simulace reálného požadovaného zdroje světla.

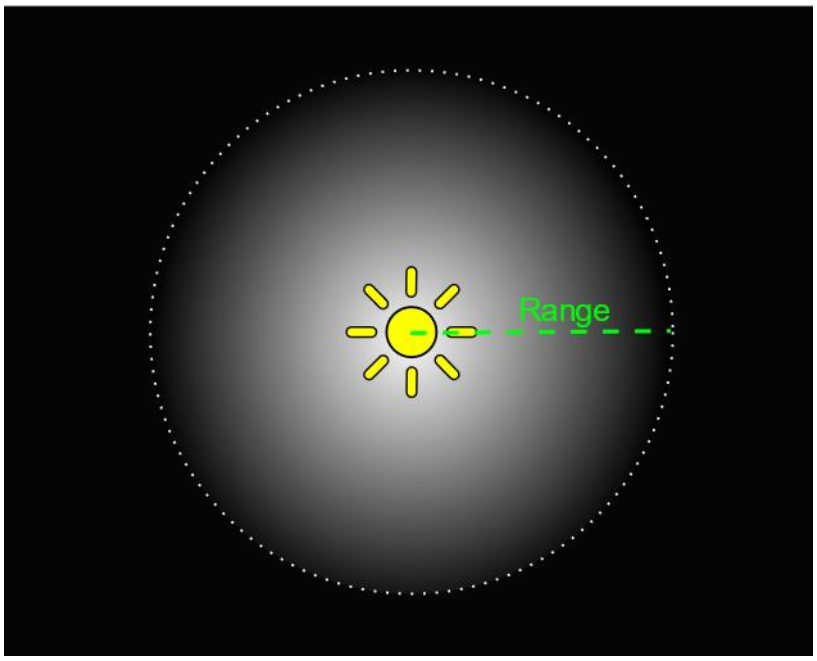
První z nejdůležitějších vlastností zdroje světla je jeho odstín. Ten se udává hodnotou tzv. teploty barev v Kelvinech. Například světlo svíčky má uváděnou teplotu barev zhruba 1900 K, zatímco přímý sluneční svit se nachází v rozmezí od 6000 K do 7500 K. Monitory využívající barevný model RGB (aditivní barevný model stanovený normou ISO) mají při zobrazení bílé barevnou teplotu cca 6500 K. Tyto teploty jsou ve virtuálním prostředí simulovány jako vektory záření v barevném modelu RGB, skládající se z vyzařovaných intenzit na jednotlivých kanálech.[31, 41]

Reálné i virtuální zdroje světla mají dále definovaný jas, určující intenzitu, jakou světlo působí na okolní objekty. Výpočet intenzity záření vychází z radiometrie, kde se v tomto případě měří energie procházející jednotkovou plochou, která je kolmá na směrový vektor světla. Tato naměřená energie je pak sumou energie všech fotonů, procházejících jednotkovou plochou za sekundu. Intenzitu lze dále počítat například z vlnové délky (za předpokladu, že je světlo považováno za vlnu). Intenzita světelného zdroje je zpětně vyrovnávána pomocí nastavení clony na kameře, ať už virtuální nebo reálné.[31, 41]

3.4.4 Typy světel

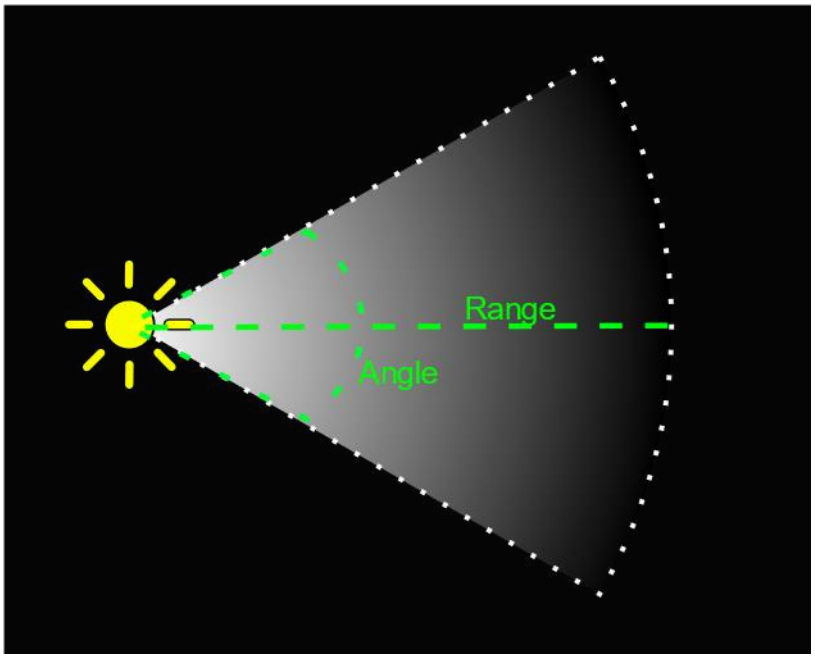
Ve 3D aplikacích rozlišujeme několik základních typů světelných zdrojů, rozlišených podle tvaru, vlastností generovaného světla a s tím související výpočetní náročností.

Prvním typem je světlo bodové, též nazývané jako *omnidirekcionální* (všesměrové). Tento zdroj generuje světlo do všech okolních směrů z právě jednoho bodu v prostoru. Generované světlo se dá přirovnat k žárovce, která má ale nekonečně malé rozměry. Samotné světlo je výpočetně nenáročné, kromě osvětlení virtuálního prostředí pomocí statických zdrojů je vhodné také pro osvětlení v rámci částicových efektů. Intenzita klesá se vzdáleností od zdroje a dosahuje nuly na určené vzdálenosti.[31, 44]



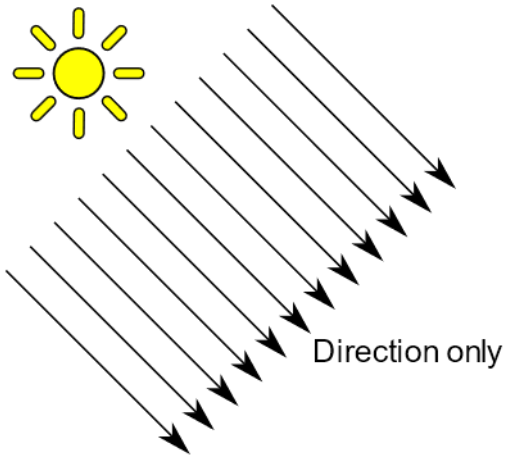
Obrázek 11 - Bodové světlo definované v prostředí Unity[44]

Reflektorový zdroj světla (*spotlight*) je v 3D aplikacích obecně jedním z nejpoužívanějších, především z důvodu přizpůsobitelnosti. Podobně jako světlo všesměrové vychází z jednoho bodu v prostoru, generuje ale světelný kužel, jehož vlastnosti je možné dále upravovat. Obvykle lze například světlo nastavit tak, aby vždy dopadalo na cílový objekt zájmu. Reflektor má vlastnosti inspirované reálnými studiovými světly, a navíc nabízí i další specifická nastavení, jako je například rozptyl světla kolem okraje kuželu. Díky tomu je možné téměř každou statickou 3D scénu nasvítit pomocí kombinace několika reflektorů. Reflektorové zdroje se obvykle využívají pro simulaci umělých zdrojů světla, jako jsou například svítilny nebo lampy.[31, 44]



Obrázek 12 - Reflektorové světlo definované v prostředí Unity[44]

Třetím typem světelných zdrojů jsou tzv. směrová světla. Jedná se o světla, která mají určený pouze směr a intenzitu. Jejich paprsky dopadají paralelně na všechny objekty a vytváří tak ostré stíny. Kvůli těmto vlastnostem jsou často používány pro simulaci slunečního světla ve scéně a díky jednoduchosti výpočtu je tento typ světla často používán pro renderování v reálném čase.[31]



Obrázek 13 - Směrové světlo definované v prostředí Unity[44]

Posledním ze základních zdrojů světla je světlo plošné. To je využíváno především pro renderování statických scén, protože dokáže generovat měkké věrohodné stíny objektů a simulovat tak skutečné rozptýlené denní světlo. V dynamických scénách je toto generování

stínů v reálném čase považováno za výpočetně náročné. U tohoto zdroje lze oproti ostatním nastavit rozměry a tvar plochy. [31, 44]

3.4.5 Teorie stínování

Výpočty stínování ve 3D aplikacích vychází z abstrakce poznatků z pozorování reálného světa. Nejdůležitějšími zdroji těchto poznatků jsou podobory optiky, konkrétně radiometrie, fotometrie a kolorimetrie.

Radiometrie se zabývá měřením elektromagnetického záření, vnímaného jako proud fotonů. Ty jsou ve většinou vnímány jako částice, s výjimkami, jako je například výpočet ozáření (*irradiance*), tedy intenzity záření zdroje světla. Ta potřebuje pro výpočet znát zářivý tok, který je závislý na vlnové délce. Ozáření vyjadřuje intenzitu světla dopadající na určenou plochu s jednotkou W/m^2. Tato veličina také popisuje dříve zmíněný princip slábnutí intenzity světla se vzdáleností.[41]

Fotometrie zkoumá stejné veličiny jako radiometrie, rozdílem však je, že ve fotometrie navíc bere v úvahu citlivost lidského oka. Konverze jednotek na fotometrické je prováděna pomocí vynásobení tzv. fotometrickou křivkou. Jedná se o Gaussovu (Bellovu) křivku intenzity vnímání světla v závislosti na vlnové délce. Střední hodnota vnímané vlnové délky je 550nm. Mezi veličiny měřené ve fotometrii patří například jas, intenzita osvětlení nebo světelný tok. Tyto veličiny se v reálném světě používají pro popis vnímání světla a jejich přepočet ve virtuálním prostředí může sloužit jako reference při tvorbě realistických scén.[41]

Kolorimetrie je podoborem optiky a zabývá se sledováním barevného spektra vnímaného světla. Barevné spektrum světla vychází z distribuce fotonů o různých vlnových délkách z daného zdroje. Lidské oko dokáže rozlišit zhruba 10 milionů odstínů a k rozeznávání používá tři oční čípky, každý zachycuje fotony s určitým rozsahem vlnové délky. Díky tomu lze říct, že každý vnímaný barevný odstín spektra lze teoreticky simulovat pomocí pouze třech hodnot. Tento poznatek vedl k vytvoření tzv. barevných modelů, sloužících právě k této simulaci. Pro 3D aplikace je nejdůležitější barevný model RGB, který využívají zobrazovací zařízení produkující světlo. Je nutné podotknout, že zobrazovací zařízení jsou omezena maximálním jasem a sytostí, proto na nich nelze nasimulovat veškeré teoretické odstíny vnímatelného barevného spektra.[41]

3.4.6 Optimalizace renderingu

Algoritmy pro generování realistického obrazu ve 3D prostředí mají výpočetní složitost, která roste úměrně komplexitě použitých modelů a množství světelných zdrojů. Autoři publikace *Real-Time Rendering* uvádí 4 základní parametry hodnotící kvalitu renderované scény. Při optimalizaci je teoretickým cílem tyto parametry maximalizovat a dosáhnout jejich vyvážení tak, aby žádný z nich neklesl pod minimální požadovanou hodnotu. Uvedenými parametry jsou:[41]

- Snímková frekvence
- Rozlišení obrazu a vzorkovací frekvence
- Vizuální kvalita materiálů a stínování
- Míra komplexity scény

Snímková frekvence by neměla nikdy dosahovat nižší hodnoty než 30, důležitá je její stabilita a za dostatečný se považuje rozsah hodnot 60-85, což odpovídá obnovovací frekvenci většiny moderních zobrazovacích zařízení (LCD panelů). V současné době jsou běžně na trhu k dispozici panely s obnovovací frekvencí 60-240 Hz, v lednu 2020 byl společností Asus představen nový FullHD panel s obnovovací frekvencí 360 Hz. Snímková frekvence má zásadní vliv na latenci vnímanou uživatelem, který s interaktivní scénou pracuje a je tedy hlavním omezujícím parametrem při její optimalizaci. Rozlišení (počet vykreslovaných bodů) má na výkon zásadní vliv, jeho snížení má ale podstatně zhoršuje uživatelem vnímanou kvalitu zobrazení. Kvalita materiálů a stínování může podobně jako rozlišení výrazně ovlivnit vizuální dojem z modelované scény. Posledním parametrem je míra komplexity scény. Zjednodušeně ji můžeme vnímat jako součet polygonů nutný k vykreslení všech modelů. Komplexitu scény nelze jednoduše změnit, počet požadovaných objektů ve scéně je často jednoznačně určen, stejně tak jako jejich úroveň detailu. Jak již bylo zmíněno, teoretickým cílem by bylo všechny parametry maximalizovat, avšak v praxi je cílem optimalizace renderingu vykreslit danou scénu (určující komplexitu) v co nejlepší kvalitě stínování a materiálů a dosáhnout při tom rozlišení odpovídajícího zobrazovacímu zařízení, a hlavně dostatečné snímkové frekvence. Důvodem pro tento přístup je, že ačkoliv lze snímkovou

frekvenci, rozlišení a kvalitu stínování teoreticky nekonečně zvyšovat, uživatelem vnímané zlepšení vizuální kvality po dosažení požadovaných hodnot (tedy například snímkové frekvence odpovídající zobrazovacímu zařízení uživatele) klesá.[41]

Aplikace většiny optimalizačních metod je prováděna automaticky pomocí nástrojů pro 3D rendering. Jedná se především o metody urychlující přístup k jednotlivým objektům ve scéně, datové struktury pro organizaci modelů a urychlení výpočtu jejich překrytí a metody využívající cache paměť například pro uložení výpočtů pro více kopií objektu. Dále pak jde o metody pro tzv. *culling*, které se zabývají dočasným odstraněním objektů a prvků, které jsou ve scéně v daný moment neviditelné. Pro zjednodušení renderingu modelů s velkým počtem polygonů se používá metoda *LoD* (Level of Detail). Jejím principem je uložení několika verzí daného modelu v různých úrovních komplexity do paměti, odkud jsou poté načítány a vykreslovány do scény v závislosti na vzdálenosti objektu od virtuální kamery. Tato metoda může výrazně přispět ke zlepšení výkonu bez toho, aby byla omezena kvalita zobrazení vnímaná uživatelem.[41, 45]

3.4.7 Algoritmy detekce kolizí

Výpočet kolizí mezi objekty je klíčovou součástí interaktivních 3D aplikací, obsahujících simulaci nebo alespoň znázornění reálných fyzikálních jevů. Ve většině případů se jedná o působení vybraných sil na objekty, vedoucí ke změně v jejich pohybu. Samotná detekce kolizí je součástí celku činností řízení kolizí. Ten kromě detekce kolize dále obsahuje její určení a reakci (výsledek). Výsledkem detekce kolizí je zpravidla odpověď, zda ke kolizi dochází, výsledkem určení kolize jsou místa kolize objektů a reakce určí akce, které mají být při takto určené kolizi vykonány. Protože objekty mohou být značně komplexní (skládající se z velkého počtu polygonů), detekce kolizí by při použití naivních metod byla na této komplexitě závislá, a tedy potenciálně extrémně výpočetně náročná. Algoritmy detekce kolizí jsou proto optimalizovány pomocí aproximačních metod, kde jsou mřížky modelu obvykle substituovány jednoduchými tvary, přibližně odpovídajícími tvaru celkového modelu. Objekty jsou uloženy v hierarchických datových strukturách, což vede k lepší přístupnosti, jak již bylo zmíněno v sekci *Optimalizace renderingu*. Algoritmy pro detekci kolizí jsou neustále ve vývoji z důvodu jejich současné nedokonalosti a širokých možností využití v oborech jako jsou Virtual Manufacturing, animace, fyzikální simulace, virtuální a augmentovaná realita

a mnoho dalších. Podle publikace *Real-Time Rendering* by algoritmus detekce kolizí měl splňovat tyto požadavky:[41, 46]

- Je schopen efektivně pracovat s komplexními modely s velkými počty polygonů, a to bez ohledu na vzdálenost mezi takovými objekty.
- Dokáže pracovat tzv. *polygon soups*, tedy s mřížkami neorganizovaných polygonů (například mřížky, kde nejsou vypočteny nebo nejsou dostupné informace o sousedních polygonech).
- Předpokládá, že modely mohou být v pohybu na všech dostupných osách, tedy rotačních i směrových, nebo mohou být dokonce deformovány.
- Provádí výpočet hranic objektů libovolným způsobem. Jejich vytvoření by ale mělo být co nejrychlejší a tvar co možná nejjednodušší pro zrychlení samotného výpočtu kolize.

V současné době neexistuje algoritmus, který by se dal označit jako optimální pro všechny možné případy užití. Příkladem používaného algoritmu splňujícího výše uvedené požadavky může být algoritmus OBBTree, využívající pro výpočet kolize objektu tzv. orientovanou hraniční kostku (OBB = Oriented Bounding Box). Algoritmus OBBTree je obzvlášť vhodný pro detekci kolize objektů, pohybujících se téměř paralelně a v těsné blízkosti.[41]

3.5 Geometrické transformace ve 3D modelování

Geometrické transformace jsou základním nástrojem pro manipulaci objektu ve 3D prostředí. V této kapitole je podrobněji popsán způsob zpracování objektů ve 3D prostředí, který je relevantní pro většinu grafických API (rozhraní pro programování aplikací).

3.5.1 Posun

Posun objektu v trojrozměrném prostředí se dá považovat za transpozici pomocí vektoru **t**. Jeho vyjádření je následující:

$$\boldsymbol{t = (t_x, t_y, t_z)}$$

Každá ze souřadnic vektorů vyjadřuje pohyb po příslušné ose v prostoru. Transpoziční matice je definována takto:

$$\mathbf{T}(\mathbf{t}) = \mathbf{T}(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Obrázek 14 –Matice transpozice[41]

Inverzní maticí $\mathbf{T^{-1}}$ k této matici je matice **T(-t)**, tedy matice transpozice negací vektoru **t**. Transformace objektu posunem je afinní, při posunu objektu nedochází k deformaci. [41]

3.5.2 Rotace

Podobně jako u posunu se i při rotaci objektu provádí afinní transformace. Tentokrát však jde o otočení objektu podle jedné nebo více os. K tomu slouží matice orientace objektu v prostoru a dále pak samotné rotační matice pro jednotlivé osy, definovány podle obrázku.

$$\mathbf{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_y(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Obrázek 15 - Rovnice rotace objektu[41]

Pro všechny rotační matice platí, že mají determinant roven jedné a jsou ortogonální, jedná se tedy o čtvercové matice a jejich transponované matice jsou zároveň jejich matice inverzní. V případě rotace pomocí libovolné osy (takové, co není rovnoběžná z některou z os souřadnic) probíhá výpočet otáčení podle jednotlivých os zvlášť, takže na něj lze nahlížet jako na 3 transformace podle výše definovaných matic provedené postupně. Pro výpočet rotace probíhá dočasný posun objektu tak, aby střed otáčení ležel v nulovém bodě soustavy souřadnic prostoru. Po dokončení rotace je proveden zpětný posun pomocí opačného vektoru.[41, 47]

3.5.3 Transformace rozměru

Pro úpravu rozměru objektu je definována následující matice:

$$\mathbf{S(s)} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Obrázek 16 - Rovnice změny rozměru objektu[41]

Matice rozměrů je v původní formě jednotková a transformace rozměru vzniká ve chvíli, kdy je na jednu z pozic ***s*** dosazena jiná hodnota. Pokud je absolutní hodnota větší než jedna, dochází ke zvětšení objektu v daném směru *x, y, z*. Pokud je hodnota v intervalu od nuly do jedné, dochází ke zmenšení objektu. Hodnota 1 zachovává původní rozměr v daném směru a při dosazení záporné hodnoty dochází kromě zvětšení nebo zmenšení také k zrcadlení objektu podle příslušné osy (*x, y* nebo *z*).[41, 47]

Změna rozměru je považována za uniformní, pokud platí:

$$\boldsymbol{s_x = s_y = s_z}$$

Uniformní a neuniformní transformace jsou někdy nazývány také jako *isotropické*, resp. *anisotropické*. Pokud je požadována změna rozměru v jiných směrech, než jsou základní osy

soustavy souřadnic, je prováděna doplňková transformace pomocí matice F, definované na obrázku.[41]

$$\mathbf{F} = \begin{pmatrix} \mathbf{f}^x & \mathbf{f}^y & \mathbf{f}^z & \mathbf{0} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Obrázek 17 – Matice F pro doplňkovou transformaci objektu[41]

Tento postup je založený na transformaci objektu do soustavy souřadnic, ve které je poté aplikovatelná výše definovaná matice pro změnu rozměru. Když je nový rozměr vypočten, objekt je transformován zpět do původní soustavy souřadnic.[41, 47]

Transformace má následující rovnici:[41]

$$\boldsymbol{X = FS(s)F^T}$$

3.6 Technologie a metody v 3D aplikacích

Tato část práce se zabývá způsoby, jak lze ve 3D aplikacích dosáhnout věrohodnějšího vykreslení cílových scén, odstranit některé nedostatky vycházející ze samotného způsobu renderování. Jedná se o doplňkové technologie a postupy, které lze zpravidla aplikovat na libovolnou 3D scénu, nejsou ale na rozdíl od obsahu dřívějších kapitol klíčové pro samotné vykreslování.

3.6.1 Anti-aliasing

Anti-aliasing, překládaný obvykle jako vyhlazování hran, je jednou z nejdůležitějších a nejpoužívanějších metod pro vylepšení vizuálního dojmu z vykresleného obrazu. Aliasing je efekt vyskytující se kromě 3D renderingu také v oblasti digitální fotografie a digitálního zpracování zvuku. Vzniká tam, kde dochází k tzv. překrytí signálů v jednom místě tak, že není možné jednoznačně určit, který z nich by měl být do daného místa vykreslen. V oblasti 3D renderování se zpravidla jedná o předěly objektů nebo detaily na texturách, kde je nutné pro jednotlivé pixely vybrat, který ze signálů budou svojí podobou reprezentovat. Výsledkem jsou pak ostré „zubaté“ hrany působící nerealistickým dojmem, proto je aliasing považován za

nežádoucí. Ve 3D aplikacích existuje několik metod vyhlazování hran, jejich výsledkem je vždy určitá forma přepočtu hodnot barev pro jednotlivé pixely tak, aby vizuálně korektně vyjadřovaly přechod mezi jednotlivými signály, tedy například hodnotou pro vykreslení hrany objektu a hodnotou pro vykreslení části okolního prostředí. Jednotlivé metody se liší výpočetní náročností a přesností (kvalitou výsledného obrazu).[48]

Nejstarší metodou vyhlazování hran je Supersample Anti-Aliasing (SSAA), někdy také nazývaný jako FSAA (Full-Screen Anti-Aliasing). Jedná se o výpočetně nejnáročnější metodu vyhlazování, založenou na renderování scény ve vyšším rozlišení. Díky tomu dojde k samostatnému výpočtu přechodových hodnot, které jsou poté převedeny na výsledné pomocí downsamplingu. Downsampling je proces, při kterém dochází ke snížení rozlišení obrazu při zachování stejné vizuální podoby. Metoda SSAA je považována za nejpřesnější, její použití v dynamických a interaktivních scénách se ale z důvodu výpočetní náročnosti nedoporučuje.[49, 50]

Druhou z metod je Multi-Sample Anti-Aliasing (MSAA). Tato metoda pracuje podobně jako SSAA, s tím rozdílem, že je místo zpracování celého obrazu zaměřena pouze na samotné objekty. Vynecháním zpracování efektů, textur a dalších vedlejších prvků finálního obrazu dochází k výraznému snížení výpočetní náročnosti oproti SSAA, spolu se zanedbatelným poklesem kvality. MSAA je nejpřesnější běžně používanou metodou vyhlazování hran v dynamických a interaktivních scénách.

Jedním z nejčastějších algoritmů pro vyhlazování hran je Fast Approximate Anti-Aliasing (FXAA). Jak již název napovídá, jedná se o aproximační metodu vyhlazování hran. FXAA pracuje s hotovým vykresleným obrazem a aplikuje na něj jednoprůchodový filtr, který rozeznává ostré hrany a pokouší se je vyhladit. Při použití FXAA dochází obvykle k mírnému rozmazání obrazu například na texturách, u kterých algoritmus nedokáže určit, zda se nejedná o hranu objektu, která má být vyhlazena. Výhodou FXAA je nízká výpočetní náročnost. Implementace od společnosti Nvidia udává zdržení vykreslování o 0.39 ms na jeden snímek při rozlišení 1920x1080 a 60 Hz na grafické kartě GTX 560. Zmíněná karta je z roku 2011, moderní čipy dokážou FXAA zpracovat ještě rychleji, proto je výpočetní náročnost této metody možné považovat za zanedbatelnou.[50, 51]

Jednou z nově implementovaných technologií pro vyhlazování hran je Temporal Anti-Aliasing (TXAA). Princip této metody je známý z 80. let minulého století, současnou nejpoužívanější implementací je ta od společnosti Nvidia. Metoda je kombinací filtrování obrazu (podobně jako FXAA), hardwarového vyhlazování (např. MSAA) a dalších postupů používaných například pro vyhlazování hran v animovaných filmech. Pro vyhlazení hrany metoda využívá několika vzorků z přepočítávaného pixelu a jeho okolí. Dále pak využívá předchozích uložených, již dříve vykreslených, snímků, pro zpřesnění výpočtu výsledného obrazu. Podle společnosti Nvidia dosahuje TXAA lepší přesnosti vyhlazení hran drobných objektů než MSAA. Výpočetní náročnost je v praxi nižší než u MSAA, celková přesnost je díky přidanému filtrování a vyhodnocení přechozích snímků lepší. Metoda TXAA je podporována pouze na grafických kartách Nvidia řady 600 (vydány 2012) a vyšších.[52]

3.6.2 Global Illumination

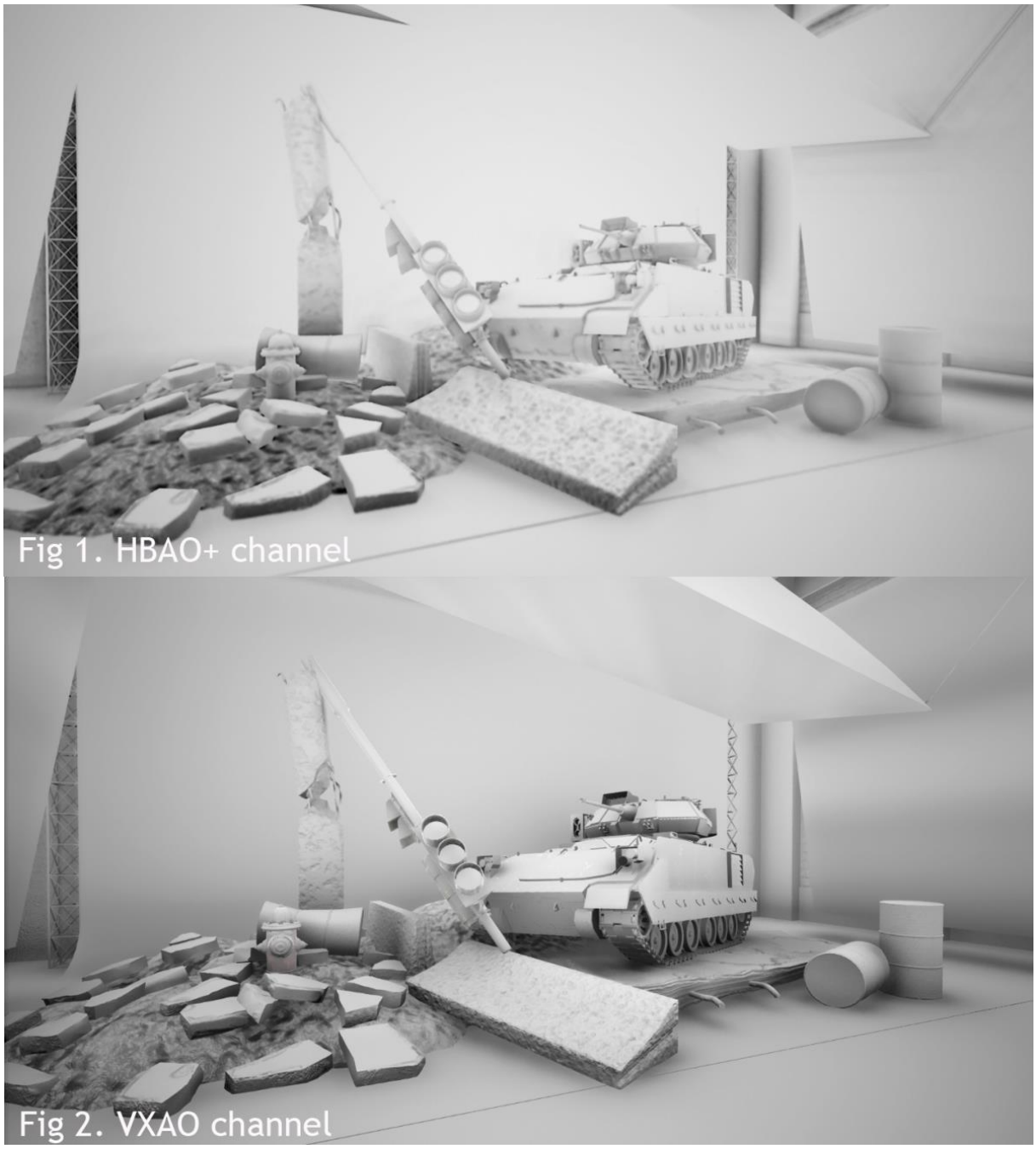
Aplikace využívající renderování v reálném čase byly v minulosti omezeny výhradně na stínování pomocí přímých zdrojů světla. Global Illumination je obecně systém, který zajišťuje modelování působení světla, jehož paprsky byly odraženy některým z povrchů. Simulace tohoto nepřímého působení světla zlepšuje vnímanou věrohodnost obrazu, protože se jedná o jev, který uživatel zná z reálného světa a (někdy podvědomě) očekává jeho přítomnost. Protože modelování nepřímého světla v reálném čase je zpravidla výpočetně náročné, provádí se obvykle pouze u statických objektů ve scéně, kde může dojít ke kalkulaci všech možností odrazu světla předem. Kvůli výpočetní náročnosti může být simulace Global Illumination úplně vynechána a nahrazena jinými technikami, případně je možné provést pouze simulaci částečnou pro vybrané objekty, na které se pak aplikují tzv. světelné mapy, kde jsou uložené efekty nepřímého světla. V současnosti se na trhu objevují nové technologie typu „real-time raytracing“, umožňující v reálném čase mimo jiné téměř dokonale simulovat Global Illumination, odlesky a ostatní jevy související s nasvícením. Jedná se o technologie vyžadující vlastní HW podporu pro urychlení výpočtů dopadu světla, například společnost Nvidia pro tyto účely vydává řadu grafických karet *RTX*.[53, 54]

3.6.3 Ambient Occlusion

Zastínění okolí, známé jako Ambient Occlusion je technika především pro real-time zpracování obrazu, používaná pro vykreslení realističtějších doplňkových stínů, obvykle v těsné blízkosti hran objektů. Jedná se o výpočet stínů, které by jinak nebyly vykresleny z důvodu nedokonalé simulace zdrojů světel. Tyto nedokonalosti se při renderování v reálném čase vyskytují z důvodu ušetření výpočetní složitosti, aby bylo dosaženo požadované snímkové frekvence. Ambient Occlusion má za úkol co možná nejvíce potlačit jeden z hlavních projevů těchto nedostatků, kterým jsou chybně vykreslené stíny. Ambient Occlusion úzce souvisí s technikou Global Illumination, používá se buď jako její náhrada, nebo jako doplněk, aby mohla být simulace osvětlení okolí zjednodušena. V praxi platí, že Global Illumination dodá renderované scéně potřebnou světlost, zatímco stíny z Ambient Occlusion zajistí potřebnou hloubku a kontrast.[55, 56]

Algoritmy pro Ambient Occlusion jsou stále ve vývoji, jedním z nejstarších je tzv. Screen-Space Ambient Occlusion (SSAO). Poprvé se (vykreslován v reálném čase) objevil v počítačové hře Crysis z roku 2007. Není závislý na komplexitě scény a je tak možné ho aplikovat na libovolnou 3D scénu. Nevýhodou SSAO je relativně nízká přesnost oproti modernějším algoritmům, jako je Horizon-Based Ambient Occlusion, nebo Voxel Ambient Occlusion (VXAO).[55, 57]

VXAO se dá považovat za jednu z nejmodernějších implementací Ambient Occlusion od společnosti Nvidia. Oproti dříve zmíněným SSAO a HBAO technikám je vykazuje až čtyřikrát vyšší výpočetní náročnost, dosahuje výrazně lepších (přesnějších) výsledků. Zároveň i přes vyšší výpočetní náročnost je tento algoritmus dvakrát až desetkrát rychlejší při vykreslování snímků, než by byla plnohodnotná simulace typu Global Illumination. VXAO je implementováno například v Unreal Engine 4, jednom z nejmodernějších vývojových prostředí pro hry a interaktivní aplikace současnosti. Pro integraci do jiných prostředí nabízí společnost Nvidia distribuční balíček zcela zdarma.[56]



Obrázek 18 - Ukázka rozdílu mezi HBAO+ (nahoře) a VXAO (dole) na scénu bez aplikovaných materiálů [56]

4 Aplikace pro prezentaci firmy

Praktická část diplomové práce se zabývá tvorbou interaktivní prezentace pomocí 3D modelování.

4.1 Představení společnosti

Prezentace je vyvíjena pro společnost USPIN s.r.o., sídlící v Praze. Firma byla založena v roce 2017 a zabývá se návrhem a dodávkou řešení, především v oblasti počítačového vidění a zpracování obrazu. Nabízí zde komplexní služby a poradenství skládající se mimo jiné z návrhu a realizace systémů pro snímání obrazu, analýzy obchodních procesů klienta, návrhu optimálního řešení pro daný problém, návrhu a implementace software a návrhu a testování postupů a algoritmů pro rozpoznání objektů v obraze. Společnost je autorem frameworku pro zpracování obrazu a rozpoznání objektů „USPIN Hightech Object Recognition (HOR)“.[58]

Mezi další nabízené oblasti služeb patří:

- Návrhy uživatelských rozhraní
- Zakázkový vývoj software
- Modelování obchodních procesů
- Umělá inteligence
- Bezpečnostní řešení
- Vědecký výzkum

Společnost se aktivně podílí na projektech v akademickém prostředí, příkladem může být například projekt Ptáci Online, který využívá chytré ptačí budky a strojové vidění implementované společností USPIN například pro rozpoznávání počtu vajíček v hnízdě, počtu zobáčků ptáčat, nebo příletů a odletů dospělých ptáků.[59]

Mezi komerční projekty společnosti USPIN patří například implementace automatizované kontroly kvality těsnících prvků spalovacích motorů pro společnost Piston Rings Komarov

s.r.o., využívající roboticko-optickou jednotku pro snímání a měření potenciálních vad s přesností 0,005 mm.[60]

4.2 Účel a popis aplikace

Produkt vybraný pro prezentaci, je zaměřený na automatizaci kontroly kvality pomocí implementace strojového vidění a technologie USPIN Hightech Object Recognition, podobně, jako zmíněný referenční projekt. Prezentace má interaktivní formou klientovi přiblížit podobu a způsob, jakým automatická kontrola kvality probíhá a jaké jsou její výhody. Předmětem prezentace je model virtuální místnosti, kde je proces automatické kontroly realizován. V této místnosti se nachází hlavní jednotka osazená kamerami pro snímání produktu ze všech potřebných úhlů, která celý proces kontroly kvality řídí. Dále je vymodelován transportní pás, pomocí kterého do procesu vstupují hotové výrobky nebo polotovary, kontrolované zmíněnou hlavní jednotkou. Výrobky jsou podle výsledku kontroly kvality roztříděny a opouští místnost na jednom z dalších dvou pásů, podle toho, zda byla nalezena závada. K přesunu mezi jednotlivými pásy a řídicí jednotkou slouží robotický podavač. Samotný proces kontroly kvality je pro účely prezentace zjednodušen, výstup kontroly je zobrazen na hlavní jednotce formou jednoduchých sdělení o tom, zda byla nalezena závada.

Uživatel má možnost se po virtuální místnosti volně pohybovat z pohledu první osoby a sledovat průběh procesu. Dále má možnost ovlivnit procento příchozích produktů, které mají nějakou závadu. Díky tomu je například možné názorně předvést, jak by situace vypadala, pokud by ve výrobním procesu došlo k chybě. Sledovanými produkty jsou části výfukových trubek nákladních automobilů, vyráběné klientem společnosti USPIN s.r.o.

4.3 Výběr prostředí

Tato kapitola popisuje proces výběru prostředí pro vývoj 3D prezentace na základě požadavků stanovených autorem a kritérií v zadání práce. Pro úspěšnou realizaci práce bylo nutné vybrat vhodné prostředí pro modelování požadovaných objektů a dále pak prostředí, ve kterém bude realizována samotná prezentace.

4.3.1 Požadavky na nástroj pro 3D modelování

Prostředí pro samotnou tvorbu 3D modelů musí splňovat tyto požadavky:

- Podporuje hard-surface modelovací techniky
- Podporuje techniky pro tvorbu animací modelů, především „rigging“, tedy proces vytváření pohyblivé kostry modelu
- Umožňuje provádět UV Mapping a přiřazení materiálů na jednotlivé modely a jejich části
- Podporuje funkce pro export do běžných formátů pro přenos do jiných vývojových prostředí
- Umožňuje komerční využití v něm vytvořených modelů při minimální výši licenčního poplatku

Z hlediska funkčnosti výběru odpovídají například produkty od společnosti Autodesk, jako je Maya a 3ds Max. Dále pak Cinema 4D od společnosti Maxon a Blender od Blender Foundation. Z výběru byly vyřazeny CAD nástroje určené především pro tvorbu technických výkresů a jejich vizualizaci. Dále pak byly vyřazeny produkty jako je SketchUp a Paint3D, z důvodu nesplnění požadavků na funkce. Splnění požadavků bylo testováno přímo autorem, pro vybrané produkty byla stažena jejich testovací verze, která byla subjektivně ohodnocena, podle dostupnosti jednotlivých funkcí. Ze zmíněného užšího výběru byl zvolen 3D modelovací software Blender od Blender Foundation. Hlavním důvodem pro volbu byla cena licencí ostatních produktů pohybující se od 2 150 Kč měsíčně (Cinema 4D [61]) do 4 450 Kč měsíčně nebo 95 550 Kč při zakoupení licence na 4 roky (oba produkty společnosti Autodesk [62, 63]). Blender byl z hlediska funkcí zhodnocen jako zcela dostačující, navíc jeho výhodou je snadno dostupná a srozumitelná dokumentace a rozsáhlá podpora online ze strany komunity. V roce 2019 vyšla verze 2.8 a nástroj je dostupný jako open source zcela zdarma i pro komerční využití. [64]

4.3.2 Požadavky na nástroj pro vývoj prezentace

Tvorba 3D interaktivní prezentace pro daný účel vyžaduje:

- Podporu práce ve 3D prostředí
- Možnost výběru běhového prostředí pro produkt, včetně podpory webového přehrávače
- Podporu importu 3D modelů a dalších prvků z jiných prostředí
- Uživatelskou přívětivost prostředí a srozumitelnost pro začínající vývojáře
- Co nejvýhodnější licenční podmínky
- Co nejrozsáhlejší online dokumentaci a komunitní podporu

Z výběru byla vyřazena prostředí neodpovídající základním požadavkům, tedy taková, kde nelze vyvíjet produkty obsahující 3D grafiku, nebo která neumožňují vývoj pro běhové prostředí webového prohlížeče (např. CryEngine[65]). V dalším kroku byla vyřazena prostředí, kde by bylo nutné za samotný vývoj hradit licenční poplatek. Do užšího výběru postoupila vývojová prostředí Unity a Unreal Engine.

Hlavní výhodou Unreal Engine od společnosti Epic Games je široká nabídka nástrojů a podpora technologií pro tvorbu produktů s nejvyšší kvalitou grafického zpracování na trhu. Unreal Engine dokáže například efektivně pracovat s až miliony částic a podporuje nejmodernější technologie dynamického nasvícení scén. Nevýhodou jsou relativně vyšší systémové požadavky na zařízení, na kterém se má produkt spouštět. Prostředí Unreal Engine je vzhledem k jeho možnostem poměrně složité a pro vývoj zadaného produktu by většina funkcí zůstala nevyužita. Unreal Engine je dostupný zdarma pro vývoj, pro prodej vyvíjených produktů je však nutné hradit tzv. royalty fee ve výši 50 % z celkových prodejů produktu.[66]

Unity je vývojové prostředí pro tvorbu videoher, prezentací a dalších interaktivních produktů od společnosti Unity Technologies. Nabízí širokou škálu nástrojů pro tvorbu ve 2D i 3D prostředí. Hlavní výhodou prostředí je jeho uživatelská přívětivost a rozsáhlá komunitní podpora. Prostředí je tedy vhodné pro začínající vývojáře. Produkty je možné snadno

optimalizovat i pro zařízení s nižším výpočetním výkonem, jako jsou mobilní telefony a tablety. Unity také nabízí možnost vývoje pro běh ve webovém prohlížeči s pomocí doplňku instalovaného na straně klienta. Oproti Unreal Engine nedisponuje Unity tak širokou podporou nejmodernějších technologií, pro účely tvorby zadané 3D prezentace je však výběr nástrojů více než dostatečný. V oblasti poplatků Unity nabízí plán zdarma a bez dalších poplatků, určený pro jednotlivce nebo malé společnosti, jejichž roční příjem nepřesahuje v přepočtu 2 289 700 Kč.[67]

Pro vývoj prezentace bylo vybráno vývojové prostředí Unity, z důvodů dostupnosti komunitní podpory, srozumitelnosti pro začínajícího vývojáře a absence licenčních poplatků. Aplikace bude vyvíjena jako samostatně spustitelná pro platformy Windows (x86), MacOS, dále pak bude proveden export pro platformu WebGL (instalace na serveru a následný běh ve webovém prohlížeči s instalovaným doplňkem).

4.4 3D Modelování

Tato kapitola stručně popisuje jednotlivé 3D modely v prezentaci, jejich účel, a metody, pomocí kterých byly vytvořeny. Všechny modely byly vytvořeny manuálně v modelovacím software Blender ve verzi 2.81. Některé materiály a textury byly získány z webů CC0Textures.com[68] a cgBookCase.com[69]. Tyto materiály jsou distribuovány pod licencí Creative Commons Public Domain (CC0). Jedná se tedy podle definice Creative Commons o díla, kde osoba, která k dílu přiložila toto prohlášení, souhlasila s vystavením tohoto díla jako díla volného a celosvětově se vzdala všech svých autorských práv k dílu, včetně všech práv souvisejících a práv příbuzných v rozsahu, který je povolen zákonem. Takovéto dílo dílo je možné kopírovat, upravovat, distribuovat a zpracovávat, a to i pro komerční účely, bez získávání dalšího souhlasu.[70]

Ostatní materiály použité v prezentaci byly vytvořeny autorem v prostředí Unity3D.

4.4.1 Model robotického podavače

Robotický podavač je nejsložitějším modelem použitým v prezentaci. Kromě samotné implementace byl využit také ke studiu principů UV mapování, animace a s ní související tvorby pohyblivé kostry modelu. Podoba modelu byla oproti reálným příkladům částečně

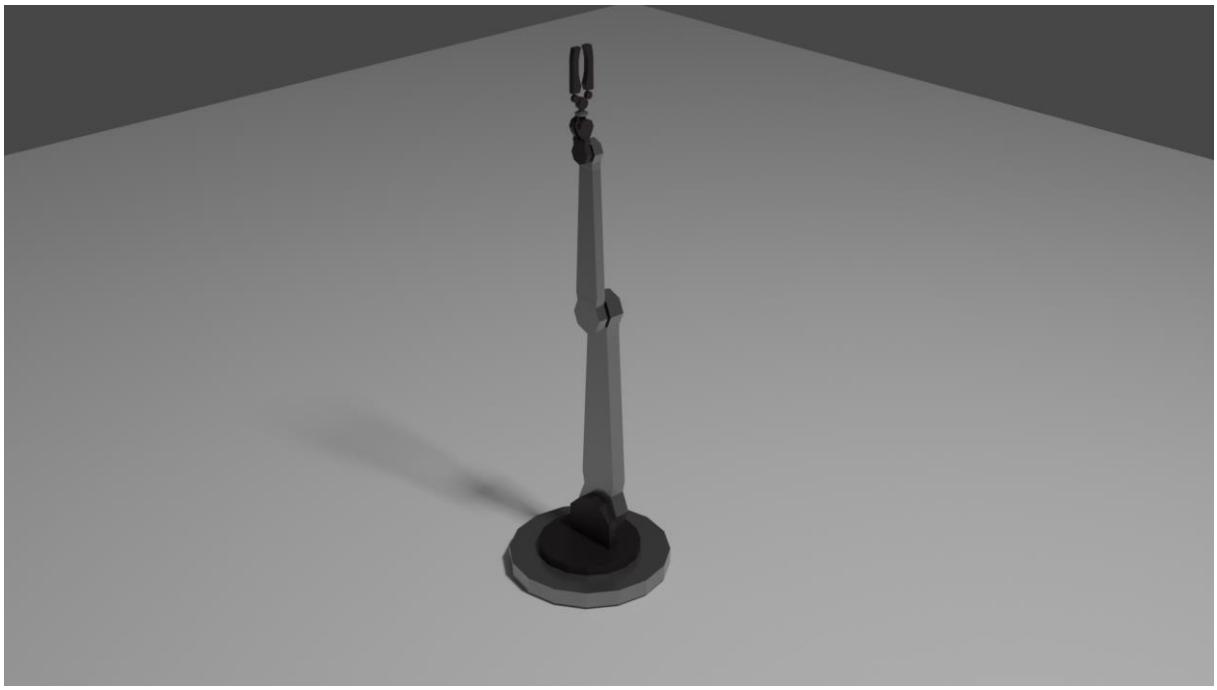
modernizována a zjednodušena tak, aby byly zachovány základní prvky a funkčnost. zároveň byl snížen počet modelovaných prvků kvůli časové a výpočetní náročnosti. Počet modelovaných prvků byl tak snížen, jednak aby bylo modelování o něco méně časově náročné, ale hlavně aby byl výsledný model použitelný v interaktivním prostředí s ohledem na požadovaný výpočetní výkon cílových zařízení.

4.4.1.1 Hard-surface modelování tvaru podavače

Model byl nejprve navržen společně s rozložením scény pomocí skic na papíře a následně byla v prostředí Blender sestavena jeho základní struktura. Protože se jedná o objekt s pohyblivými částmi, skládá se z několika podstruktur. Jednotlivé části jsou modelu jsou následující:

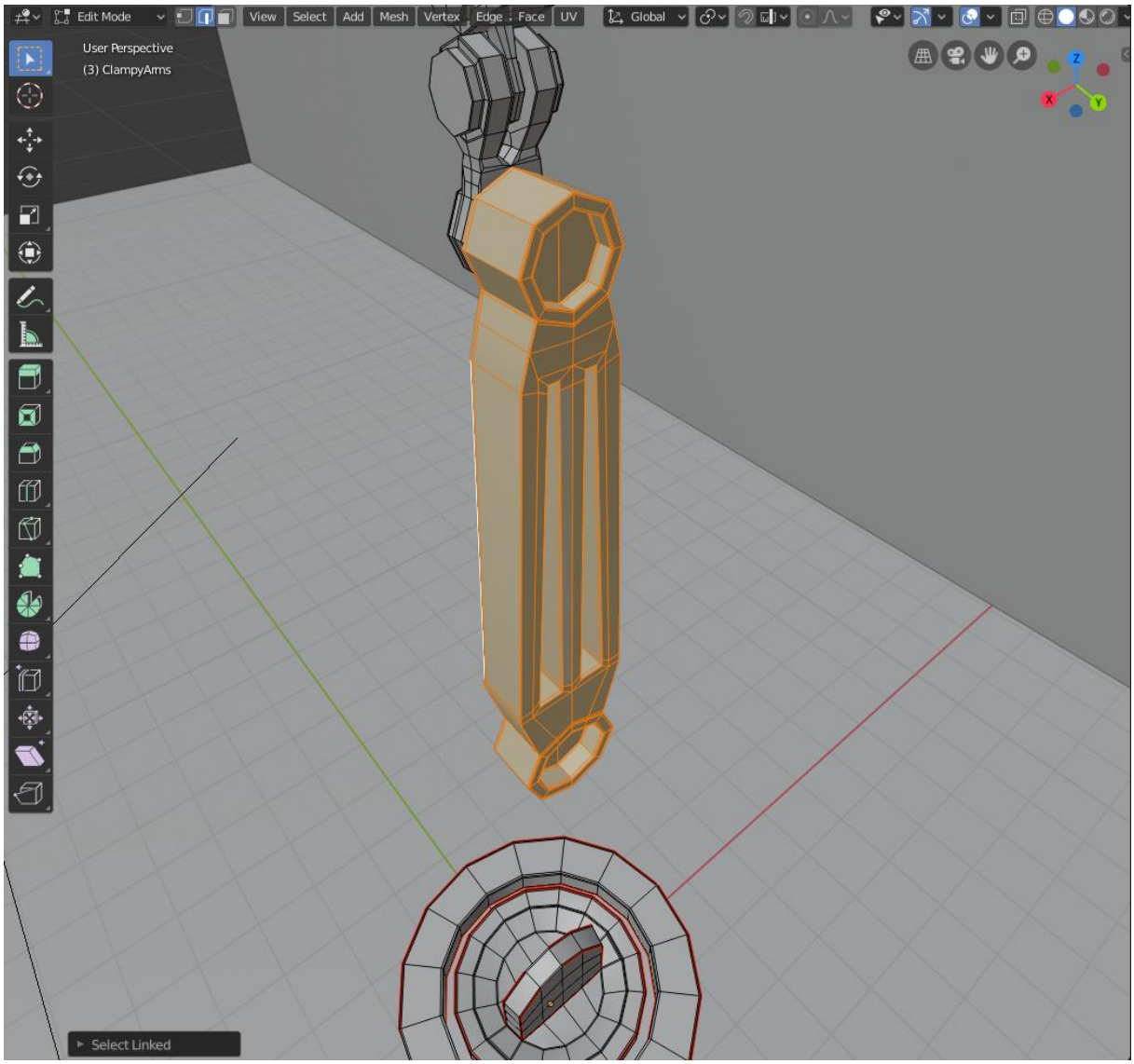
- Pevná základna, vymodelována pomocí primitivního objektu typu „válec“
- Rotační díl, propojený s pevnou základnou, vytvořený z primitivního objektu typu „válec“
- Primární a sekundární rameno (primární připojeno k rotačnímu dílu a sekundární připojeno k primárnímu), obě vytvořeny pomocí modifikace a spojení primitivních objektů typu „kvádr“ a „válec“
- Ruka podavače, skládající se ze čtyř samostatných struktur, vymodelovaných modifikací primitivních objektů různých druhů, zajišťujících realistickou vizualizaci pohybu
- Úchopné „prsty“ podavače, vymodelované úpravou primitivních objektů typu „kvádr“, přichycené na ruku podavače

Pro tvorbu modelu byly použity techniky z oblasti hard-surface modelování. Pro každou podstrukturu byl v nástroji Blender primitivní objekt, který byl dále modifikován pomocí zvýšení hustoty polygonů, pohybu a změny velikosti jednotlivých částí mřížky a extruze.



Obrázek 19 - První zjednodušená verze modelu podavače na ukázkovém renderu (vlastní zpracování)

Příkladem postupu při modelování může být tvorba sekundárního ramena podavače. Nejprve byl vytvořen primitivní objekt typu „válec“, který byl zvětšen na odpovídající rozměr a dále duplikován a posunut, čímž byly vytvořeny koncové klouby struktury. Dále byl generován primitivní objekt typu „kvádr“, jehož hustota polygonů byla pomocí generování okrajů navýšena tak, aby mohl být propojen s připravenými klouby. Pro připojení bylo nutné odstranit z kloubů a válce polygony v místech, kde měly být objekty sloučeny. Sloučení bylo dosaženo díky vygenerování nových polygonů, vždy pomocí zadání dvou protilehlých hran. Dále byly ve výsledném objektu vytvořeny díry, a to pomocí smazání povrchů protilehlých polygonů a vytvoření nových vazeb mezi vzniklými volnými protilehlými hranami. Další tvorba detailu spočívala v extruzi částí kloubů pro vytvoření jejich vystupujících okrajů v místech, kde je bude objekt propojen s jinou strukturou.



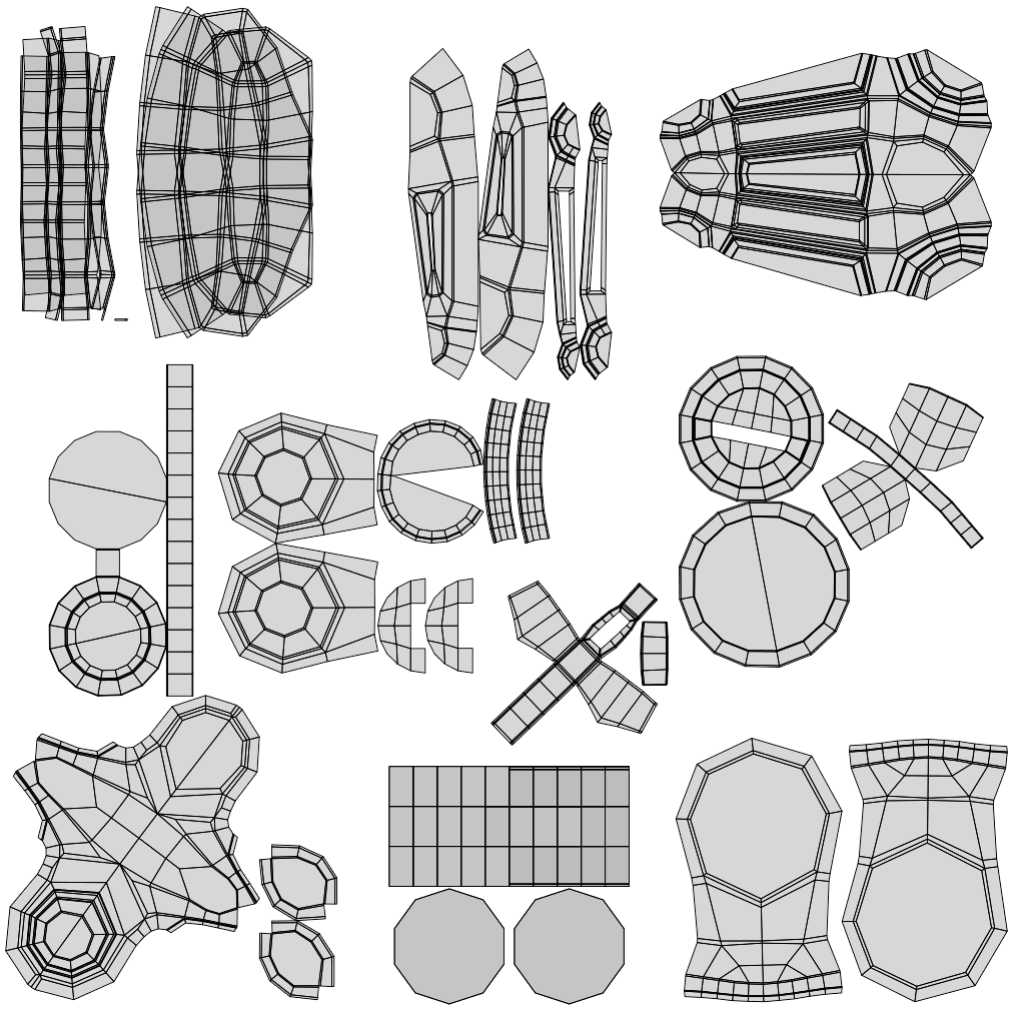
Obrázek 20 - Průběh modelování ramena podavače (vlastní zpracování)

Obdobným způsobem byly tvořeny všechny jmenované podstruktury modelu. U „ruky“ podavače bylo vytvořeno několik iterací návrhu na papíře, než bylo rozhodnuto o její finální podobě.

Pro zvýšení úrovně detailu celého modelu byl aplikován modifikátor subdivize. Kvůli tomu musely být do modelu přidány pomocné hrany, tzv. „edge loops“, v místech, kde má objekt definované okraje. Tyto hrany nemají bez použití modifikátoru na podobu modelu vliv, bez nich by ale při aplikaci subdivize došlo k nežádoucímu zkreslení tvaru.

4.4.1.2 UV mapování a aplikace materiálů

Po dokončení samotného modelování bylo provedeno jeho UV mapování a aplikace základních materiálů. Účelem mapování je rozložení jednotlivých povrchů modelů tak, aby mohly být reprezentování ve dvojrozměrném prostoru, a tedy na ně mohly být aplikovány textury a nedošlo k jejich zkreslení. Pomocí nástroje Blender byly určeny tzv. „seams“, tedy hrany, kde jsou povrchy virtuálně odděleny. Model se tak pro potřeby UV mapování dělí na další podčásti, které jsou reprezentovány ve 2D prostoru a rozloženy do plochy ve tvaru čtverce. Tento čtverec reprezentuje UV mapu objektu, která je přenositelná mezi prostředími a lze na ni aplikovat textury a materiály. Informace uložené v UV mapě včetně vybraných dělicích hran jsou zachovány pro export modelu pro práci v jiných prostředích.



Obrázek 21 - UV mapa robotického podavače (vlastní zpracování)

Pro model podavače byly vybrány tyto materiály:

- Černě lakovaný kov pro většinu částí modelu
- Nelakovaný kov, použitý pro rotační díl hlavní část „prstů“ podavače
- Textura plastu, později upravena na oranžovou barvu, použitá na částech „prstů“ podavače, sloužících k úchopu

Obrázek 22 - Render finálního modelu robotického podavače s aplikovanými materiály (vlastní zpracování)

Aplikací materiálů byl proces samotného modelování podavače dokončen a v dalším kroku byla vytvářena jeho pohyblivá kostra.

4.4.1.3 Tvorba pohyblivé kostry modelu podavače

Kostra pro animaci podavače byla vytvořena v nástroji Blender. Slouží k tomu speciální objekty, které označujeme jako „kosti“. Místa, kde se tyto objekty spojují se nazývají „klouby“. Kostra byla vytvářena jako samostatný objekt, který byl následně propojen s hotovým modelem, přičemž došlo k přiřazení jednotlivých částí modelu k prvkům kostry. Prvky kostry byly navrženy tak, aby odpovídaly pozicí a rozměry jednotlivým částem

modelu, s výjimkou tzv. kořenové kosti, která slouží pouze pro referenci polohy a ostatní prvky jsou na ni napojeny.

Po přiřazení jednotlivých částí modelu k vytvořené kostře byla otestována funkčnost pomocí „režimu pózy“ v nástroji Blender a následně byla vygenerována testovací animace modelu tak, aby se na ní projevily všechny možné varianty pohybu objektu.

4.4.2 Model transportního pásu

Transportní pásy v prezentaci slouží k přesunu zkoumaných polotovarů trubek. Celkově se ve scéně vyskytují tři kopie, jedna sloužící pro přesun směrem k podavači, další dvě pro transport zkontrolovaných polotovarů mimo scénu. Samotná mřížka modelu je zjednodušena pro snížení výpočetní náročnosti modelu.

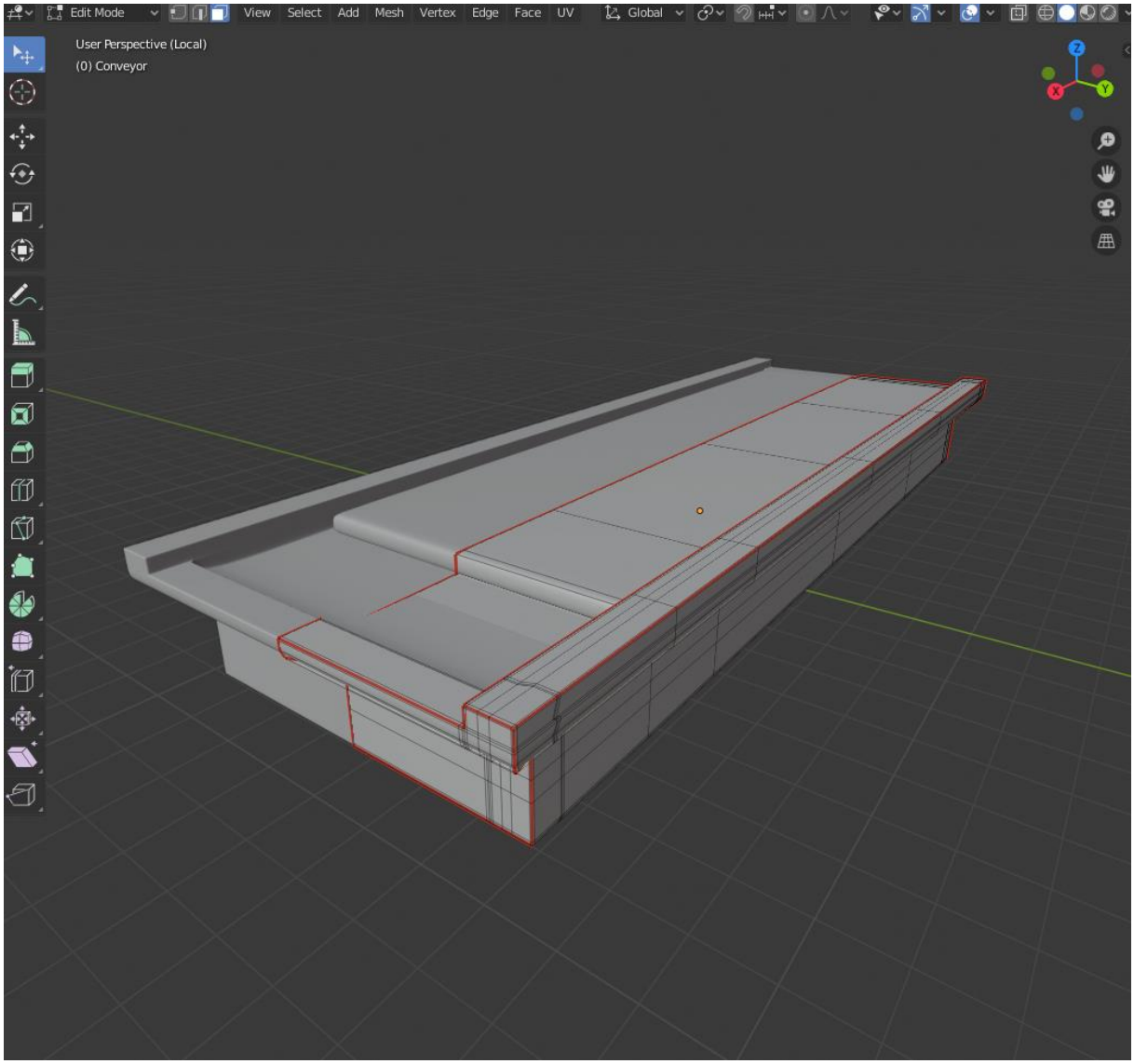
4.4.2.1 Hard-surface modelování tvaru podavače

Model se skládá ze dvou částí, první z nich je pomocná konstrukce nesoucí samotný pás. Ta byla vytvořena z primitivního objektu typu „kvádr“, bylo zde použito mnohonásobné dělení kvádru pomocí středových hran a následná extruze a změny rozměrů částí mřížky, aby bylo dosaženo požadovaného tvaru. Takto byla vymodelována horní část konstrukce, zahrnující obdélníkový výřez pro uložení pásu včetně oblasti, kde budou polotovary po přesunu zastaveny. Pás byl vymodelován pomocí úpravy rozměrů primitivního objektu „kvádr“ tak, aby odpovídal požadovaným rozměrům. Pro zjednodušení modelu byl pás modelován pouze jako jeden objekt, jeho pohyblivost bude vyřešena pomocí nástrojů v prostředí Unity. Do celého modelu byly přidány pomocné hrany, podobně jako u modelu podavače. Dále byl aplikován modifikátor subdivize, díky kterému bylo možné poměrně snadno zvýšit vnímanou úroveň detailu modelu.

Verze transportního pásu pro přesun polotovarů k podavači obsahuje ještě třetí část, kterou je vyvýšená plocha, umístěná na konci pásu. Tato plocha slouží k zastavení polotovarů tak, aby je mohl podavač co nejsnáze uchopit. Plocha byla přidána až v prostředí Unity jako nový objekt přiřazený do struktury daného pásu.

4.4.2.2 UV mapování a přiřazení materiálů pro transportní pás

Stejně jako u modelu podavače i zde bylo provedeno UV mapování jednotlivých částí modelu. Oproti podavači, kde se z velké části jednalo o automatizovaný proces, zde byla zvolena převážné manuální metoda postupné projekce vybraných ploch. Mapování bylo stejně jako modelování provedeno v prostředí Blender, byly vybrány oddělující hrany a následně byla provedena projekce vybraných ploch do 2D prostoru z pevně určených pohledů. Tyto plochy byly pak stejně jako u modelu podavače rozmístěny do konkrétní UV mapy.

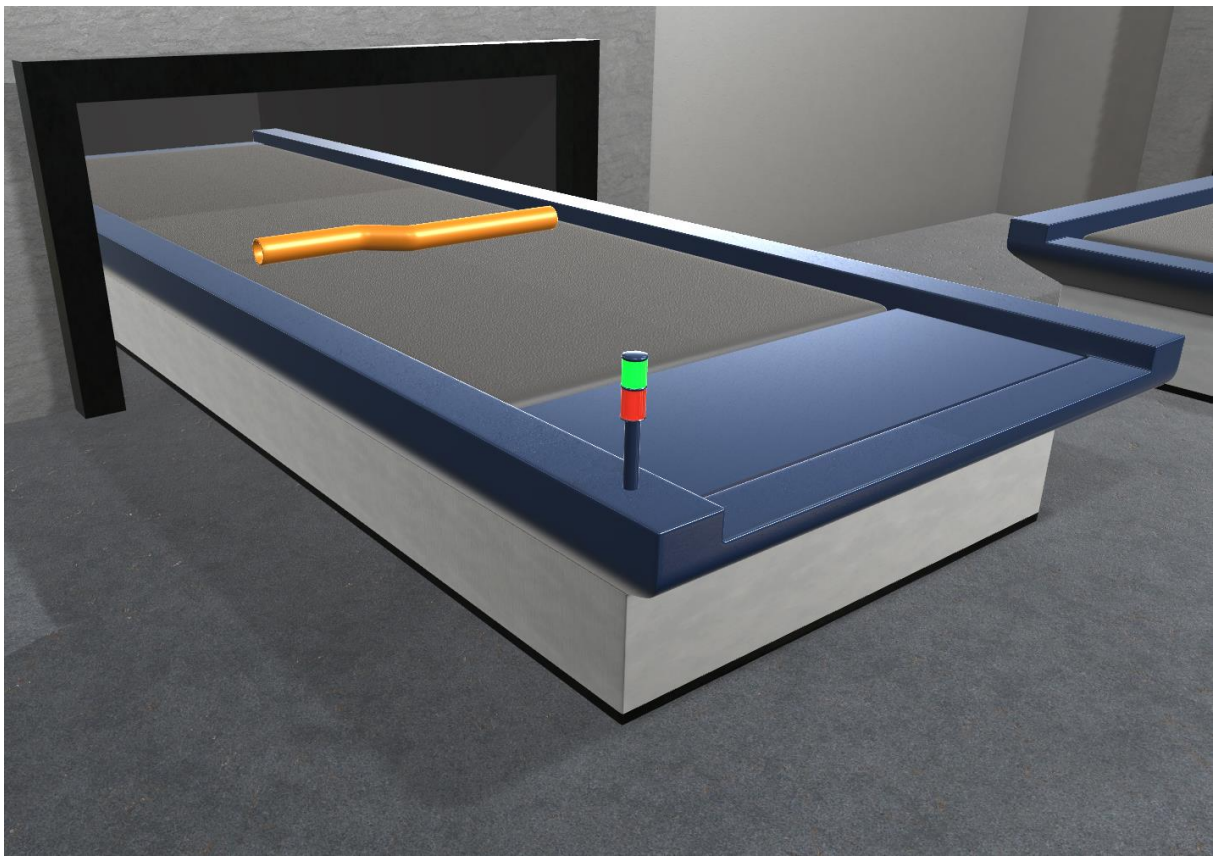


Obrázek 23 - Model transportního pásu v nástroji Blender (vlastní zpracování)

Ve fázi UV mapování byly vybraným plochám přiřazeny testovací materiály, aby mohl být model úspěšně exportován. V prostředí Unity pak byly vytvořeny přiřazeny následující finální materiály:

- Modře lakovaný kov pro většinu modelu
- Broušený nelakovaný kov ve spodní části konstrukce
- Černý hrubý plast pro povrch samotného pásu

Model neobsahuje pohyblivé části, animace pásu je řešena pomocí nástrojů prostředí Unity.



Obrázek 24 - Finální model transportního pásu v prostředí Unity (vlastní zpracování)

4.4.3 Model hlavní jednotky pro kontrolu kvality

Hlavní jednotka pro kontrolu kvality polotovarů je vymodelována jako skříň osazená kamerami pro rozpoznávání závad. Dále zahrnuje obrazovku, na které bude zobrazován aktuální stav kontroly.

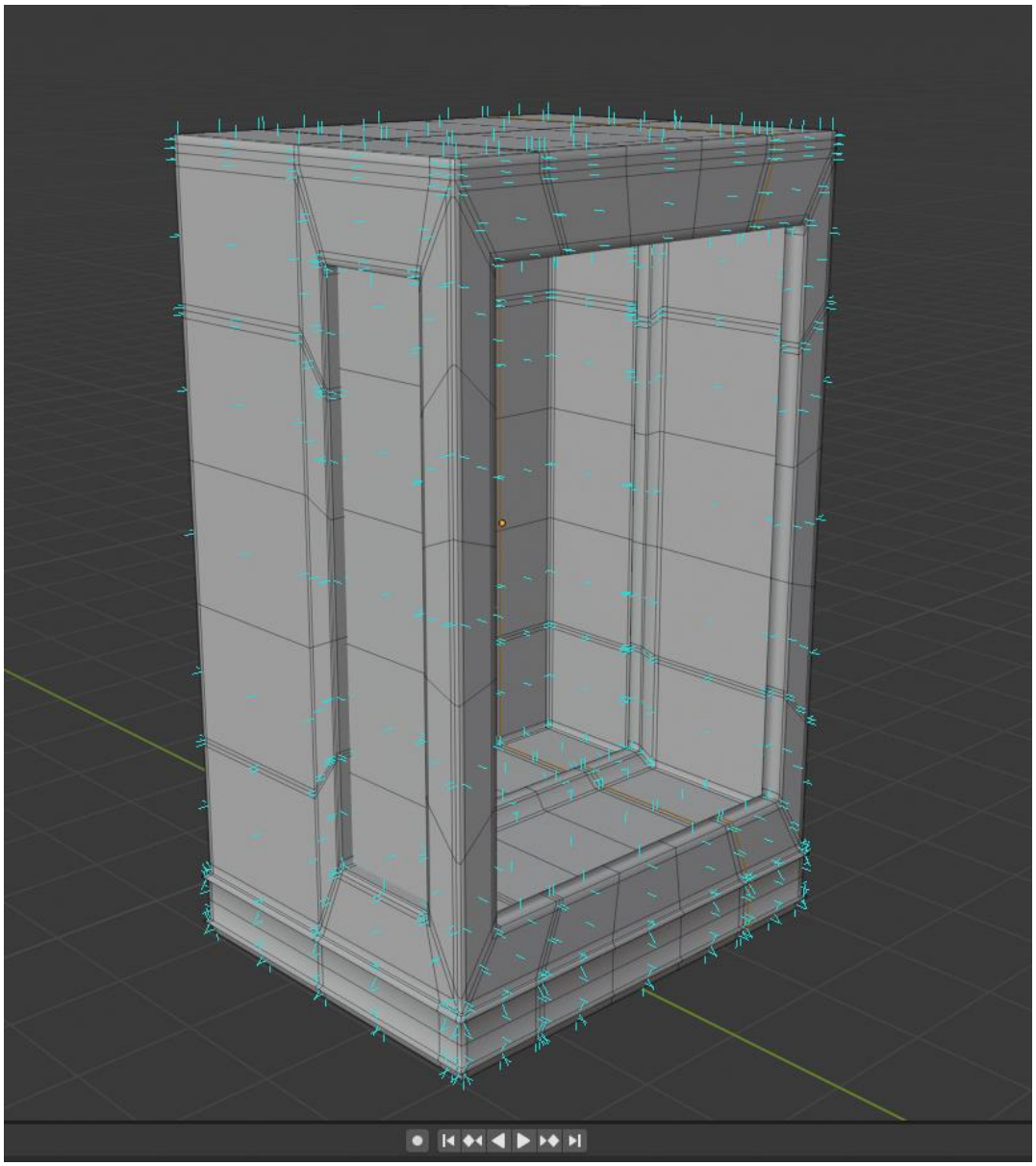
4.4.3.1 Hard-surface modelování prvků jednotky

Hlavní konstrukční část jednotky je podobná skříni, proto byl jako základní primitivní objekt vybrán kvádr. Pomocí extruze byl do kvádru vytvořen otvor, ve kterém byla dále vymodelována lišta na osazení kamer. Do mřížky modelu byly na vybraných místech vloženy další hrany, aby bylo možné lištu vytvarovat. Dále bylo extruzí vymodelováno okno na jedné straně skříně. Na zadní straně skříně byla vytvořena výseč, aby objekt nepůsobil jednotvárně. Stejně jako u ostatních objektů i zde byly přidány pomocné hrany a byl aplikován modifikátor subdivize.

V dalším kroku byla vymodelována kamera. Jedná se o samostatný objekt, který je sice součástí hlavní jednotky, ale jeho kopie jsou umístěny až po importu do prostředí Unity. Kamera se skládá ze dvou objektů, kostky a válce, vytvarovaných tak, aby připomínaly běžné reálné průmyslové kamery. Úroveň detailu základního modelu je velice nízká. Důvodem je, že kamery nebudou ve scéně dobře viditelné kvůli jejich rozměrům a místě osazení. Do modelu byly přidány pomocné hrany a byl aplikován modifikátor subdivize.

Poslední částí jednotky je obrazovka stavu kontroly. Byla vytvořena extruzí a úpravou rozměrů z primitivního objektu „kvádr“. Dále pak bylo vytvořeno její sklo z primitivního objektu „plocha“, které bylo umístěno na okraj rámečku obrazovky. Plocha pro vykreslení obrazu byla přidána až v prostředí Unity a je umístěna v úrovni těsně za sklem obrazovky.

Pro všechny objekty byly vybrány z mřížky části pro přiřazení jednotlivých materiálů.



Obrázek 25 - Model hlavní jednotky vč. zobrazených normálových vektorů (vlastní zpracování)

4.4.3.2 UV mapování hlavní jednotky

Na rozdíl od přechozích modelů bylo mapování hlavní jednotky provedeno pomocí „Smart UV Project“ algoritmu v prostředí Blender. Tento algoritmus se snaží vytvořit projekci ploch tak, aby nedošlo k jejich zkreslení. Výsledná UV mapa obsahuje překrývání ploch, což by mohlo způsobit problémy při použití pokročilejších metod nasvícení v prostředí Unity. Protože se ale jedná o poměrně jednoduchý model a tyto metody nebudou použity, vygenerovaná UV mapa byla zachována a stejným způsobem bylo provedeno mapování i pro modely kamer a obrazovky.

Obrázek 26 - Finální model hlavní jednotky v prostředí Unity (vlastní zpracování)

Stejně jako u transportního pásu i zde byly přiřazeny pouze testovací materiály, které byly následně nahrazeny po přenosu modelů do prostředí Unity. Finální použité materiály jsou následující:

- Modře lakovaný kov pro většinu modelu
- Broušený nelakovaný kov ve spodní části konstrukce a zadní výseči
- Černý hrubý plast pro vnitřní část hlavní jednotky a povrch kamery

- Sklo vytvořené v prostředí Unity pro přední část obrazovky, čočku kamery a okno hlavní jednotky

- Kovový hladký materiál vytvořený v prostředí Unity pro rám obrazovky

Hlavní jednotka neobsahuje pohyblivé části, s výjimkou části obrazovky pro vykreslení stavu, která byla implementována až po přenosu do prostředí Unity.

4.4.4 Ostatní modely

Pro vytvoření scény v prostředí Unity byly kromě dříve zmíněných hlavních modelů vytvořeny navíc následující doplňující objekty:

- Stěna s otvorem pro transportní pás

- Stavové osvětlení pro umístění na průmyslové stroje (hlavní jednotka a transportní pásy)

- Stropní svítidlo

- Dveře do virtuální místnosti

- Okna virtuální místnosti

- Polotovar výfukové trubky

Tyto doplňkové modely byly vytvořeny stejným způsobem, jako hlavní modely, pomocí primitivních objektů. Dále prošly postupnými úpravami a aktualizacemi po jejich přenosu do prostředí Unity. Modely slouží k sestavení scény virtuální místnosti, do které jsou vloženy hlavní modely pro prezentaci automatizované kontroly kvality. Polotovar výfukové trubky bude v prostředí Unity duplikován na dvě verze, které se liší použitým materiálem. Tím bude zjednodušeně vizuálně rozlišeno, zda polotovar vykazuje závadu.

4.5 Přenos modelů do prostředí Unity

Pro přenos 3D modelů z prostředí Blender do prostředí Unity lze využít několik souborových formátů. Jako nejvhodnější byl zvolen formát FilmBoX (FBX), obvykle používaný produkty společnosti Autodesk. Druhou možnou variantou byl formát OBJ, který je určený pro obecnou reprezentaci 3D objektů. Formát FBX byl upřednostněn, protože se jedná o formát doporučený pro import do prostředí Unity, hlavním důvodem ale byla přenositelnost pokročilých vlastností objektů. Při přenosu pomocí formátu OBJ nebyly správně přeneseny informace o pohyblivé kostře podavače a některé UV mapy modelů neodpovídaly podobě, jakou měly v nástroji Blender.

Všechny modely byly do prostředí Unity importovány jako tzv. prefabrikáty, které byly později rozloženy na dílčí části tak, aby s nimi bylo možné v prostředí Unity dále pracovat. Objekty podavače, transportního pásu, hlavní jednotky a kamery byly importovány ve třech verzích s různou úrovní detailu. Verze se liší především počtem průchodů modifikátorem subdivize v prostředí Blender. V Unity pak byly tyto objekty vloženy do scény a pomocí komponentu LOD (Level of Detail) bylo zajištěno, aby se jednotlivé úrovně detailu zaměňovaly podle vzdálenosti od kamery (avatara uživatele).

Prvním přeneseným objektem byl model podavače, na kterém bylo nutné otestovat funkčnost pohyblivé kostry ve vztahu k animačnímu nástroji prostředí Unity. Dále bylo na modelu podavače testováno přichycení dalších objektů. Implementace této funkce bude popsána v dalších kapitolách. Model musel být během testování upraven, protože prostředí Unity vyžaduje pro animace základnu pohybu vyjádřenou jako součást kostry, nikoliv samotného objektu. Dále byla k objektu přichycena prázdná entita, umístěná mezi úchopové „prsty“ podavače. Tato entita bude sloužit k referenci polohy přichycených objektů. Po úspěšném otestování funkce podavače mohly být do scény postupně vloženy i ostatní objekty.

Vloženým objektům byly v prostředí Unity přiřazeny nové materiály tak, aby bylo možné je během implementace dále upravovat. Z nástroje Blender tedy byly importovány z hlediska materiálů pouze UV mapy a přiřazení jednotlivých částí objektů. Scéna byla sestavována postupně během implementace jednotlivých funkcí. Kromě objektů jmenovaných v minulých kapitolách byly do scény přiřazeny ještě další primitivní objekty, většinou typu „plocha“,

vygenerované přímo v Unity. Tyto objekty slouží k dotvoření modelu místnosti a jsou zobrazeny jako stěny, podlaha, strop, případně jako další pomocné struktury, například dříve zmíněná vyvýšená plocha pro zastavení polotovarů na konci transportního pásu. Hlavní výhodou této metody sestavení scény je modulárnost, tedy možnost přizpůsobit finální podobu přímo v prostředí Unity podle potřeby další implementace, případně podle připomínek zadavatele.

Před dokončením sestavení scény bylo nutné změnit rozměry několika objektů. Hlavní jednotka byla zvětšena v prostředí Unity tak, aby dokázala pojmout připravený model polotovaru a aby ve scéně nezanikala oproti ostatním modelům. Model transportního pásu byl během implementace několikrát upraven, především byla snížena jeho poloha, aby bylo možné jeho funkci z pohledu uživatele snáz sledovat.

4.6 Implementace funkcí v Unity

Tato kapitola popisuje postupy, použité v prostředí Unity pro vytvoření samotné prezentace. Funkce byly implementovány postupně během fáze přidávání objektů do scény, aby bylo možné včas identifikovat problémy v příslušných modelech a mohlo dojít k jejich opravě. Pro získání potřebných znalostí k vývoji aplikace byla využita online dokumentace Unity v kombinaci s publikacemi *Unity Game Development Essentials*[71] a *Learning C# by Developing Games with Unity 3D*.[72]

4.6.1 Pohyb uživatele

Uživatel se v aplikaci pohybuje pomocí klávesnice a myši z pohledu první osoby. Kamera je tedy upevněna přímo na virtuálním „těle“ uživatele. K zajištění pohybu těla uživatele a rotace kamery slouží dva oddělené skripty v jazyce C#, využívající především nativní funkce z knihoven prostředí Unity.

Pohyb těla uživatele je zajištěn skriptem „*FPSControl.cs*“, který byl v Unity přiřazen jako komponent objektu těla uživatele. Ve skriptu je volán komponent *CharacterController*, pro který jsou definovány mimo jiné i funkce pohybu na všech dostupných osách. Získání vstupních hodnot a provedení pohybu obstarává funkce *PlayerMovement*, která je implementována následovně:

```
private void PlayerMovement()
    {


        float horizInp = Input.GetAxis(horizontalInputName) * calcSpeed;
        float vertInp = Input.GetAxis(verticalInputName) * calcSpeed;

        Vector3 forwardMovement = transform.forward * vertInp;
        Vector3 rightMovement = transform.right * horizInp;
        charController.SimpleMove(forwardMovement + rightMovement);

        JumpInput();
        SprintSneakInput();

    }
```

Funkce získává data ze vstupů, které prostředí Unity určí automaticky podle platformy. V případě osobních počítačů je pohyb možné ovládat pomocí směrových šipek, nebo kláves W, S, A a D. Funkce tyto vstupy přepočítá pomocí proměnné obsahující aktuální koeficient rychlosti pohybu a následně hodnoty převede na vektory transformace objektu. Tyto vektory jsou pak zadány do funkce *SimpleMove* komponentu *CharacterController*, čímž je dosaženo požadovaného pohybu uživatele. V posledním kroku funkce volá podfunkce *JumpInput* a *SprintSneakInput*. Funkce *JumpInput* vyvolá, pokud je na klávesnici stisknutý mezerník, „skok“ uživatele pomocí předdefinovaného vektoru pohybu. Funkce *SprintSneakInput* mění koeficient rychlosti pohybu podle toho, zda je stisknuta klávesa SHIFT (pro zrychlení), nebo CTRL (pro zpomalení). Funkce *PlayerMovement* je volána metodou *Update*, která je automaticky aktivována posunem interního časovače v prostředí Unity.

Pohyb kamery a rotace těla uživatele jsou zajištěny skriptem „*MouseView*.cs“, připojeného jako komponent objektu virtuální kamery. Skript získává vstupní data podobným způsobem, jako při pohybu těla uživatele. Hodnoty pohybu myši po dvou osách jsou modifikovány pomocí proměnné určující citlivost. Vertikální pohyb myši má vliv na rotaci kamery podle osy X, která je omezena na interval od -90 do 90 stupňů. Horizontální pohyb myši řídí rotaci těla hráče bez omezení rozsahu. Skript dále obsahuje funkci pro přiblížení, která při stisku pravého tlačítka myši zmenší zorné pole kamery o 15 stupňů. Funkce rotace kamery a funkce pro přiblížení jsou volány metodou *Update*. Při inicializaci skriptu (metoda Awake) je navíc uzamčen pohyb kurzoru myši a je vyvoláno jeho skrytí. Přiřazení objektu pro rotaci a os pro

získání hodnot pohybu je provedeno pomocí privátních proměnných, které jsou přiřazeny v prostředí Unity, je tedy možné je zaměnit bez zásahu do samotného skriptu.

4.6.2 Animace objektů

Hlavním animovaným objektem ve scéně je robotický podavač sloužící pro přesun polotovarů mezi transportními pásy a hlavní jednotkou pro kontrolu kvality. Animace podavače je implementována pomocí komponenty *Animator* v prostředí Unity. Ta pracuje na principu stavů, mezi kterými objekt podle definovaných pravidel přechází a ke kterým může být přiřazena konkrétní animace. Při přechodu mezi stavy může nastat situace, kdy je generována tzv. přechodová animace, která je kombinací animací dvou stavů, mezi kterými dochází k přechodu. Podavač má celkem 11 definovaných stavů, z toho 9 je aktivně využíváno a jeden je tzv. vstupní, který definuje, do kterého stavu podavač přechází v momentě inicializace scény. Všechny stavy obsahují animaci vytvořenou pomocí rotací jednotlivých propojených částí podavače. Rychlost pohybu v každém momentě animace je určena pomocí křivek, aby bylo dosaženo plynulých přechodů mezi jednotlivými stavy. Animace jsou spouštěny buď automaticky po dokončení pohybu v předchozím stavu, nebo pomocí přepínačů, které mohou být aktivovány pomocí skriptů popsaných v implementaci kontroly kvality. Součástí některých stavů a animací jsou tzv. události, které při dosažení bodu v animaci způsobí volání některé z funkcí v přiřazených skriptech. Průběh animace a přechodů mezi stavy podavače je popsán v následující kapitole jako součást implementace kontroly kvality polotovarů. Nepoužívané stavy jsou určeny pro testování funkčnosti animací, pokud by došlo k dodatečné úpravě modelu podavače.

Druhým animovaným objektem je model transportního pásu a s tím související pohyb kontrolovaných polotovarů. Implementace jeho pohybu je rozdělena na dvě části. První z nich je vizuální pohyb povrchu pásu, který je implementován pomocí posouvání materiálu. K tomu slouží pomocný skript *AnimatedTexture.cs*. V tomto skriptu je definována rychlost posunu a ta je aplikována pro každý posun v čase díky volání metody *LateUpdate* v kombinaci s nativní komponentou *Time* prostředí Unity. Při každém volání dochází k posunu textury (materiálu) po povrchu objektu v požadovaném směru. Díky tomu se pás viditelně pohybuje bez nutnosti implementace mnoha fyzických pohyblivých prvků.

Druhou částí implementace pohybu transportního pásu je část fyzická, která zajišťuje, že objekty, které na pásu leží, se budou pohybovat definovanou rychlostí směrem k jeho konci. Tato funkce je zajištěna pomocí neviditelného pomocného fyzického objektu ve scéně, který má přiřazený skript *BeltPhysics.cs*. Tento jednoduchý skript pracuje s komponentou *Rigidbody*, která v Unity zajišťuje simulaci působení fyzikálních sil. Pracuje ve dvou krocích při každém obnovení, podobně jako ve skriptu pro animaci vizuálního pohybu. V prvním kroku se neviditelný objekt okamžitě přesune směrem dozadu (ve vztahu k poloze pásu) bez aplikace simulace fyzikálních sil. V druhém kroku se pomocí nativní metody *MovePosition* pro komponent *Rigidbody* posouvá zpět na původní pozici, čímž dochází k simulaci fyzického pohybu pásu. Objekty ležící na pásu jsou tak při každém obnovení posunuty směrem k jeho konci, rychlost je určena konstantou tak, aby odpovídala rychlosti vizuálního pohybu povrchu pásu.

Posledním objektem s implementovanou změnou stavu je obrazovka hlavní jednotky, vykreslující aktuální stav kontroly kvality. Změny na této obrazovce jsou implementovány pomocí funkce pro změnu materiálu objektu. Objekt obrazovky tak zůstává zachován a při změnách je pouze nahrazen materiál přiřazený k její vnitřní části. Tyto změny jsou vyvolávány pomocí událostí v animaci podavače.

4.6.3 Kontrola kvality

Proces vizualizace automatizované kontroly kvality se skládá z několika kroků a je zajištěn především hlavním řídicím skriptem *ClampyControl.cs*. Ten byl připojen jako komponent objektu robotického podavače. Důvodem pro toto připojení (oproti připojení například k objektu hlavní jednotky) je snazší implementace vyvolání funkcí z komponentu *Animator* stejného objektu. Akce jsou z velké části vyvolávány právě pomocí této komponenty, proto je vhodné na stejný objekt vázat i řídicí skript.

Vizualizace probíhá v následujících krocích, které mohou být přiřazeny k jednotlivým stavům komponenty *Animator* robotického podavače. V následujících podkapitolách budou stručně vysvětleny probíhající aktivity a jejich implementace v chronologickém pořadí.

4.6.3.1 Generování nového polotovaru

Polotovary jsou uloženy v programu jako tzv. prefabrikáty. Existuje verze pro polotovar vykazující závadu a polotovar bez závad. Podoba těchto polotovarů je na první pohled rozlišena pomocí použitých materiálů (polotovar vykazující závadu je znázorněn pomocí oranžové textury kovu).

Rozhodnutí o výběru polotovaru je provedeno pomocí generování pseudonáhodného celého čísla z intervalu 0–100. Toto číslo je dále porovnáno s uživatelem nastavenou šancí na generování vadného polotovaru, kterou lze za běhu kdykoliv měnit pomocí kláves popsaných v uživatelském rozhraní. Pro nastavení šance jsou určeny tyto klávesy:

- **B**: šance na závadu 20 % (základní hodnota pro prezentaci funkce hlavní jednotky)
- **M**: šance na závadu 90 % (simulace fatální chyby ve výrobě)
- **N**: šance na závadu 2 % (simulace dobře fungujícího procesu výroby)

Na základě porovnání hodnoty šance na závadu a vygenerované hodnoty proměnné *DiceRoll* je rozhodnuto, který z prefabrikátů bude použit pro vytvoření nového polotovaru. Výsledek rozhodnutí je uložen do pomocné proměnné *diceRollResult*, která v dalších krocích rozhoduje o tom, jaký objekt má být uchycen k robotickému podavači a o budoucím výsledku virtuální kontroly kvality. Protože k vytvoření dalšího polotovaru probíhá již během přesunu prvního polotovaru k hlavní jednotce, hodnota *diceRollResult* je při generování každého polotovaru (kromě prvního) přenesena do proměnné *diceRollResLast*, aby nedošlo k její ztrátě.

Generování objektů v prostředí Unity, stejně jako jejich destrukce, jsou považovány za výpočetně náročné operace. Pro celkovou optimalizaci aplikace bylo tedy nutné množství těchto operací minimalizovat. Aplikace proto ukládá existující polotovary do datových struktur typu „kruhové pole“, aby k destrukcím nemuselo vůbec docházet. Pro tato pole byl určen pevný rozměr, tedy počet zároveň existujících objektů stejného typu (polotovar bez závad a se závadami). Kromě kruhového pole bylo zváženo i využití datové struktury typu „fronta“, kruhové pole bylo vybráno z důvodu údajně nižší výpočetní náročnosti pro přístup k prvkům (dle dokumentace Unity). Funkce *SpawnPipe*, která generování polotovarů řídí,

před vytvářením nového objektu kontroluje, zda již na dané pozici v poli není jiný objekt uložen. Pokud tomu tak je, dochází pouze k jeho výpočetně jednoduchému přesunu na dané souřadnice a odpovídající rotaci. Jako příklad může být uvedena implementace generování polotovaru bez závad, která vypadá následovně:

```
if (diceRoll > failChancePercent)
   {
    animator.SetBool("PipeIsGood", true);
  // nastavení parametru pro Animator
    diceRollResult = true;
    if (pipeGList[curPosG])
        {
            pipeGList[curPosG].transform.position
         = spawnP.transform.position;
            pipeGList[curPosG].transform.localRotation
         = spawnP.transform.rotation;
        } else {
            pipeGList[curPosG] = Instantiate(PipeGprefab,
           spawnP.transform.position, spawnP.transform.rotation);
        }
    } else {
        ***
        }
```

Generování polotovaru se závadou je provedeno obdobným způsobem. Polotovary se po vytvoření přesouvají na konec transportního pásu, kde dochází k přechodu do dalšího kroku.

4.6.3.2 Uchopení polotovaru a jeho přesun

Rozpoznání, že je polotovar připraven k přesunu pomocí podavače a následné kontrole je zajištěno pomocí systému detekce kolizí v prostředí Unity. Pro tento účel je definován neviditelný objekt na konci transportního pásu, který slouží jako tzv. „trigger“. Když model polotovaru koliduje s tímto objektem, je v komponentě *Animator* robotického podavače nastaven parametr *PipeIsReady* a podavač se automaticky přesouvá do pozice k úchopu připraveného polotovaru. Podavač v tomto momentě přechází do stavu, kdy je animován úchop polotovaru a s tím související událost přichycení polotovaru k objektu podavače. To je provedeno pomocí funkce *AttachObject*, která přiřadí objekt polotovaru na konec struktury objektu podavače. Jeho souřadnice polohy jsou synchronizovány s úchopným bodem podavače a je deaktivována simulace fyzikálních sil pro polotovar.

Na konci tohoto kroku je provedeno v komponentě *Animator* podavače rozhodnutí o dalším pohybu, podle toho, zda byl přichycen polotovar se závadou, či nikoliv. K tomuto rozhodnutí je použita hodnota parametru *PipeIsGood*, který byl nastaven v momentě generování polotovaru.

Obrázek 27 - Hlavní scéna, kde probíhá kontrola kvality (vlastní zpracování)

Podavač dále provádí animaci přesunu polotovaru do hlavní jednotky, kde dojde k vizuální simulaci kontroly kvality. Během tohoto přesunu je vyvoláno generování dalšího polotovaru, popsané v předchozí podkapitole.

4.6.3.3 Kontrola kvality polotovaru

Polotovar je přesunut pomocí podavače do hlavní jednotky pro kontrolu kvality. Animace je zde zastavena a dochází k vyvolání funkce pro zobrazení výsledku kontroly na obrazovce hlavní jednotky. Pro tento účel bylo vytvořeno několik materiálů pro vnitřní část obrazovky, které reprezentují jednotlivé varianty stavu kontroly. Tyto materiály jsou přiřazeny pomocí funkce *SendToScreen*, která rozhodne o výsledku kontroly podle hodnoty proměnné *diceRollResLast*, určené při generování polotovaru. Výsledek je v případě polotovaru bez závad reprezentován pomocí zelené obrazovky s nápisem „Succes“, jinak se pozadí obrazovky změní na červenou s nápisem „Failed“. Pokud ke kontrole ještě nedošlo, nebo byl stav kontroly obnoven, zůstává obrazovka černá s nápisem „Waiting/Processing…“. Po vykreslení výsledku kontroly kvality na obrazovku komponenta *Animator* pokračuje dalším krokem, kterým je přesun polotovaru na cílový pás.

4.6.3.4 Přesun polotovaru na cílový pás a opakování cyklu

Animace podavače pokračuje přesunem na jeden z cílových transportních pásů, podle výsledku kontroly kvality. Rozhodnutí o tom, kde bude polotovar umístěn, bylo provedeno již při prvním úchopu po jeho vygenerování. Během přesunu na cílový pás je vyvolána funkce *ResetScreen*, která na obrazovku hlavní jednotky přiřadí původní materiál (černá obrazovka s nápisem „Waiting/Processing“. Díky tomu je vyvolán dojem, že skutečně došlo ke kontrole kvality a hlavní jednotka nyní čeká na vložení dalšího polotovaru.

Po přesunu polotovaru pomocí podavače do finální pozice je vyvolána animace uvolnění, společně s funkcí *DetachObject*. Tato funkce odpojí objekt polotovaru od struktury podavače a zároveň pro něj znovu aktivuje simulaci fyzikálních sil. Dochází tak k přirozenému dopadu polotovaru na transportní pás, po kterém se polotovar přesouvá na jeho konec. Polotovar je společně s ostatními uložen v prohlubni na konci transportního pásu, kde čeká na opětovný přesun do startovní pozice, který je vyvolán funkcí *SpawnPipe*, pokud bylo dosaženo maximálního určeného počtu polotovarů ve scéně.

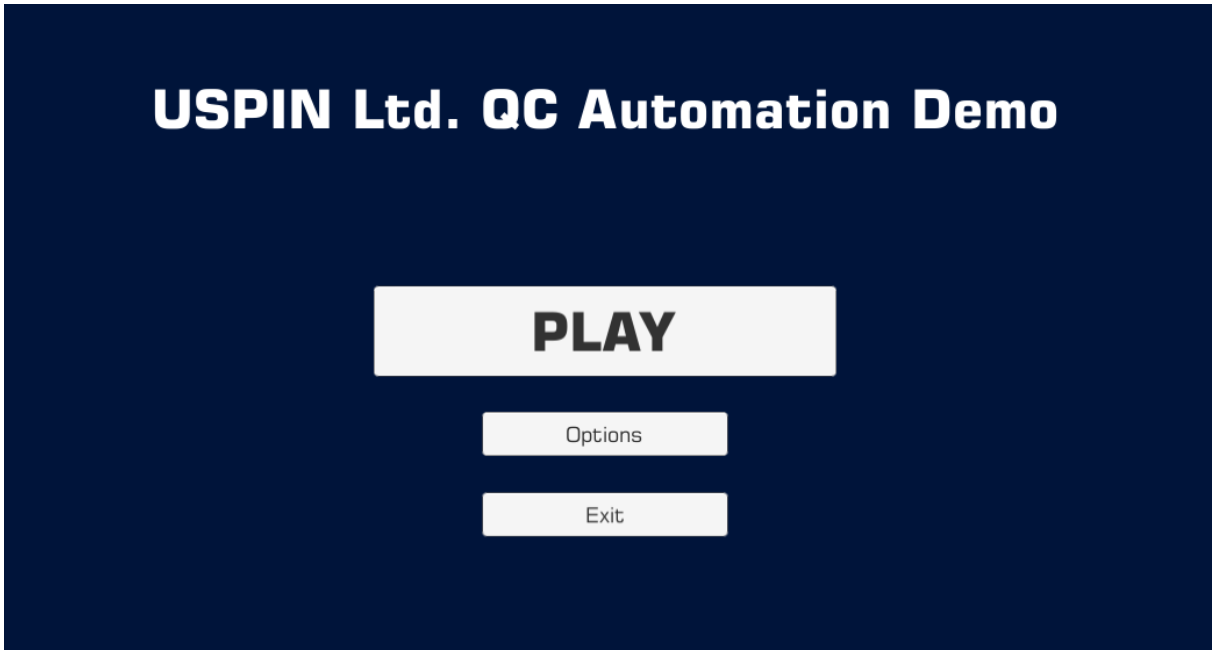
Podavač se přesouvá do základní pozice, kde čeká na změnu parametru *PipeIsReady,* aby mohlo dojít k opakování celého cyklu kontroly pro další polotovar. Pokud je další polotovar připraven již v momentě, kdy je původní odpojen, podavač plynule přechází do stavu úchopu dalšího polotovaru a animace pro návrat do základní pozice je v tomto případě zkombinována s animací pro přesun do této polohy. Kombinace dvou současných animací je provedena automaticky díky nastavení přesahu animací v komponentě *Animator*.

Limit na počet polotovarů každého druhu (se závadou a bez závad) je určen na 5 kusů, aby byla jistota, že se polotovar přesouvaný při volání funkce *SpawnPipe* nachází v příslušném úložném prostoru na konci jednoho z cílových transportních pásů.

4.6.4 Menu a nastavení

Při spuštění aplikace je uživatel přesměrován nejdříve do hlavního menu, kde může uživatel prezentaci spustit, opustit, nebo přejít do jejího nastavení. Menu je vytvořeno ze základních komponent *Unity UI*, rozložení je konfigurováno tak, aby jednotlivé položky měly pevnou reálnou velikost, která je vypočítána z údaje DPI (dots per inch) přijatého ze zobrazovacího

zařízení. Díky tomu může aplikace fungovat na téměř libovolném displeji, přičemž je zachována čitelnost položek v menu a UI panelu v hlavní scéně. V menu nastavení, které je přístupné po kliknutí na tlačítko *Options* v hlavní nabídce, je možné nastavit několik parametrů ovlivňujících vykreslování scény.



Obrázek 28 - Hlavní menu aplikace (vlastní zpracování)

Nejdůležitějším parametrem je nastavení zobrazované kvality. Aby bylo nastavení srozumitelné i pro nezkušeného uživatele, byla nastavení kvality zobrazení zjednodušena na výběr předpřipravených konfigurací z rolovacího menu. Těchto konfigurací nabízí aplikace celkem šest. Konfigurace mění především počet v reálném čase simulovaných světelných zdrojů, rozlišení stínů a vzdálenost na kterou jsou vykreslovány. Každá konfigurace dále určuje koeficient mezní vzdálenosti pro vykreslování jednotlivých úrovní detailu vybraných modelů. Konfigurace „*Low*“ a „*Very Low*“ navíc omezují maximální úroveň detailu modelů na střední, resp. nízkou. V konfiguraci „*Very Low*“ je navíc zcela deaktivováno vykreslování stínů v reálném čase. Varianty konfigurace „*Very High*“ a „*Ultra*“ kromě výše zmíněných efektů aktivují metodu vyhlazování hran MSAA a jsou tak určeny pro běh na relativně výkonnějších počítačích. Při prvním spuštění je zvolena varianta „*High*“, která by měla bez problémů fungovat na běžných počítačích s moderní dedikovanou grafickou kartou. Testování vlivu výpočetního výkonu na běh aplikace je popsáno v následující kapitole.

V menu nastavení lze dále konfigurovat parametr *Field of View*, tedy vertikální zorné pole ovládané kamery. Toto nastavení závisí především na preferenci uživatele, udává se ve stupních a základní hodnota byla na 60°. Vyšší hodnoty zorného pole jsou určeny pro uživatele, kteří trpí nevolností při ovládání aplikací z pohledu první osoby. Pro takové uživatele je doporučeno nastavit hodnotu zorného pole alespoň na 90°.

Posledním nastavitelným parametrem je citlivost kamery. Parametr určuje relativní rychlost otáčení kamery při pohybu myší. Nastavení zlepšuje přístupnost aplikace pro různé citlivosti sensorů na dostupných myších a dává uživateli možnost úpravy tak, aby ovládání aplikace bylo co možná nejpohodlnější. Nastavení neovlivňuje rychlost kurzoru v hlavním menu, která je závislá na nastavení v konkrétním operačním systému, kde je aplikace spuštěna. Základní hodnota byla nastavena na 150 %.

Obrázek 29 - Menu nastavení aplikace (vlastní zpracování)

Uživatelem nastavené parametry se nachází ve skrytém souboru *PlayerPrefs*, ze kterého jsou načteny při každém spuštění aplikace a do kterého jsou ukládány při každé jejich změně.

Soubor *PlayerPrefs* nelze manuálně upravovat a jeho lokace je závislá na operačním systému počítače, kde je aplikace spuštěna.

4.7 Testování a hodnocení aplikace

Aplikace byla během vývoje průběžně testována autorem, aby mohlo dojít k odstranění chyb v samotné funkcionalitě. Navíc bylo prováděno průběžné měření vytížení systému, na kterém byla aplikace takto testována a součástí vývoje bylo také sestavování konfigurací nastavení kvality zobrazení. Po dokončení vývoje byla aplikace dále otestována na několika sestavách různými uživateli, kteří měli za úkol vyplnit testovací protokol obsahující nalezené chyby a informace o naměřeném vytížení systému při spuštění aplikace.

4.7.1 Testování

Testování finální aplikace bylo rozděleno na dvě části. V první části byly autorem testovány interaktivní prvky aplikace. Mezi tyto prvky patří:

- Pohyb uživatele a nastavení hranic scény
- Nastavení šance na vadný polotovar pomocí klávesnice
- Správné zobrazení šance na vadný polotovar na hlavní obrazovce
- Funkce nastavení zorného pole a citlivosti kamery
- Funkce nastavení kvality zobrazení
- Přepínání mezi hlavní scénou a menu, ukončení aplikace pomocí tlačítka *Exit*

Během tohoto testování nebyly odhaleny žádné chyby, které nebyly opraveny již ve fázi vývoje. Dále byla v této části ověřena funkčnost aplikace na cílové platformě, tedy při spuštění pomocí webového prohlížeče a technologie WebGL. Ani v tomto případě aplikace nevykazovala žádné chyby ve funkčnosti. Protokol z testování je součástí příloh práce.

Aplikace byla ve druhé části záměrně testována v samostatně spustitelné formě pro OS Windows 10, aby měření výkonu nebylo ovlivněno například připojením k internetu,

vytížením serveru a výběrem prohlížeče, nebo dalšími v něm otevřenými záložkami. Díky měření v samostatně spustitelné verzi aplikace bylo možné využít nástroj MSI Afterburner pro získání přesných dat o vytížení systému. Test byl spuštěn pro každou sestavu dvakrát po dobu 10 minut, jednou pro nastavení kvality „Very Low“ a jednou pro nastavení „Very High“. Hlavním měřeným parametrem byla dosažená snímková frekvence, která by měla dosáhnout minimální hodnoty 30, aby aplikace byla považována za uspokojivě ovladatelnou. Aplikace je zhodnocena jako zcela bez problémů ovladatelná, pokud bylo dosaženo snímkové frekvence vyšší než 60. Testování proběhlo na verzi aplikace, která se od finální liší tím, že snímková frekvence není uzamčena na hodnotu obnovovací frekvence monitoru (funkce Vsync je deaktivována). Aplikace byla testována na monitorech s rozlišením 1920x1080 a obnovovací frekvencí 60 Hz, s výjimkou protokolu V2, kde aplikace běžela na monitoru s rozlišením 2580x1080 a obnovovací frekvencí 75 Hz. Vzorový testovací protokol je součástí příloh práce, výsledky byly zpracovány do níže uvedených tabulek.

Výsledky testování při nastavení kvality zobrazení na „*Very High*“ jsou následující:

ID	Model CPU	Model GPU	Kapacita RAM	FPS prům.	Vytížení CPU	Vytížení GPU	Δ Využití RAM
V1	Intel i5-8250U	Nvidia MX150	8 GB	70	11 %	99 %	225 MB
V2	Intel i5-3570K	Nvidia GTX 1070	12 GB	332	36 %	97 %	221 MB
V3	Intel i5-3210M	Nvidia GTX 660M	8 GB	**38**	13 %	99 %	123 MB
V4	Intel i5-3210M	Intel HD4000 (iGPU)	8 GB	**7**	8 %	100 %	492 MB
V5	Intel i7-7700HQ	Nvidia GTX 1050 + iGPU	8 GB	125,1	18 %	98 % + 29 %	223 MB
V6	Intel i5-8265U	Intel UHD 620 (iGPU)	16 GB	**19,5**	7 %	100 %	260 MB

Tabulka 1 - Výsledky testování konfigurace "Very High" (vlastní zpracování)

Při nastavení konfigurace kvality zobrazení na „*Very High*“ dva z šesti systémů nedosáhly uspokojivé obnovovací frekvence. Jedná se o systémy V4 a V6, které pracují pouze s grafickým čipem integrovaným v procesoru. Zvláště špatného výsledku dosáhl systém V4, což splňuje očekávání, protože se jedná o notebook střední z roku 2012 třídy a lze ho tedy považovat za zastaralý. Systém V6 je naopak novějším notebookem nižší střední třídy a vzhledem k jeho výsledku by měl bez problému dosáhnout snímkové frekvence vyšší než

60 při výběru nižšího nastavení kvality zobrazení. Stejně tomu je i u systému V3, kde se jedná o obdobný systém, jako je V4, avšak v tomto případě s aktivovaným dedikovaným grafickým čipem. Nejlépe v testu dopadl systém V2, který dosáhl snímkové frekvence 332 i přesto, že oproti ostatním vykresloval na zobrazovací zařízení s vyšším rozlišením a zároveň disponuje oproti většině testovaných systémů jedním ze slabších procesorů. Důvodem pro tento výsledek je grafická karta GTX 1070, která je z uvedených zdaleka nejvýkonnější. Z výsledků je zřejmé, že právě grafická karta systému má zdaleka největší vliv na výslednou snímkovou frekvenci. Systém V5 má uvedeny hodnoty pro dvě grafické karty (GTX 1050 a iGPU), které byly při testu využívány současně.

Výsledky testování při nastavení kvality zobrazení na „Very Low“ vypadají takto:

ID	Model CPU	Model GPU	Kapacita RAM	FPS prům.	Vytížení CPU	Vytížení GPU	Δ Využití RAM
V1	Intel i5-8250U	Nvidia MX150	8 GB	198	11 %	99 %	255 MB
V2	Intel i5-3570K	Nvidia GTX 1070	12 GB	1099	40 %	72 %	180 MB
V3	Intel i5-3210M	Nvidia GTX 660M	8 GB	192	35 %	96 %	98 MB
V4	Intel i5-3210M	Intel HD4000 (iGPU)	8 GB	48	15 %	97 %	245 MB
V5	Intel i7-7700HQ	Nvidia GTX 1050 + iGPU	8 GB	280,6	40 %	70 % + 85 %	200 MB
V6	Intel i5-8265U	Intel UHD 620 (iGPU)	16 GB	107,8	7 %	98 %	210 MB

Tabulka 2 - Výsledky testování konfigurace "Very Low" (vlastní zpracování)

Při konfiguraci kvality zobrazení „*Very Low*“ všechny testované systémy dosáhly uspokojivé snímkové frekvence vyšší než 30 a všechny kromě systému V4 dosáhly frekvence výrazně převyšující hodnotu 60. Oproti nastavení „*Very High*“ byl zaznamenán nárust snímkové frekvence na více než dvojnásobek u všech systémů. Například u systému V4 byla naměřená snímková frekvence téměř 7x vyšší než u předchozí konfigurace. U systému V2 byla zaznamenána frekvence téměř 1100 snímků za sekundu při využití GPU 72 %, což značí, že bylo dosaženo maximální snímkové frekvence, jakou grafická karta GTX 1070 dokáže vykreslit. Vytížení procesoru je oproti nastavení „*Very High*“ vyšší, což je přímým následkem vyššího počtu vykreslovaných snímků za sekundu. Pokud by byla snímková frekvence uzamčena na pevnou hodnotu (jako je tomu u finální verze aplikace), vytížení procesoru

i grafických čipů by bylo výrazně nižší. Systém V6, který je notebookem nižší střední třídy s pouze integrovaným grafickým čipem bez problému dosáhl snímkové frekvence vyšší než 60, a systém V4 dosáhl snímkové frekvence 48, což ukazuje, že aplikace je bez problému spustitelná i na starších zařízeních s nižším výkonem a při nasazení na web pomocí platformy WebGL by tak neměl u většiny uživatelů nastat problém s nedostatkem výpočetního výkonu.

4.7.2 Vyhodnocení splnění cíle

Z hlediska funkcí aplikace splnila požadavky zadavatele a je funkční prezentací produktu společnosti USPIN s.r.o. v oblasti strojového vidění a automatizované kontroly kvality. Uživatel se může po virtuální místnosti volně pohybovat, sledovat průběh kontroly kvality a pro účely prezentace je mu umožněno měnit šanci generování vadného polotovaru a simulovat tak například kritickou chybu ve výrobě.

Pro modelování byl vybrán open-source nástroj Blender z důvodů popsaných v minulých kapitolách práce. Aplikace byla vytvořena v prostředí Unity pomocí skriptů v jazyce C# s využitím knihoven, které jsou součástí uvedeného prostředí.

Aplikace je spustitelná samostatně na platformách Windows a Mac OS a dále může být uložena na serveru a pomocí platformy WebGL spouštěna na většině prohlížečů. Z hlediska zobrazovacích zařízení aplikace podporuje libovolné poměry stran a rozlišení, autor však doporučuje minimální rozlišení 1280x720, aby nebyla narušena čitelnost textu. Při vyšších rozlišeních je třeba dbát na s nimi spojené vyšší nároky na výkon systému. Při nedostatečné snímkové frekvenci je možné v aplikaci snížit úroveň kvality zobrazení až na konfiguraci „*Very Low*“, u které se podařilo dosáhnout uspokojivé snímkové frekvence na všech testovaných systémech. Naopak pro výkonné systémy je možné nastavit konfiguraci „*Ultra*“, při které je dosaženo maximální úrovně detailu modelů, nejlepší kvality osvětlení a stínování, a navíc je aktivována metoda vyhlazování hran MSAAx8. Aplikace z hlediska přístupnosti obsahuje nastavení zorného pole kamery, což může pomoci především uživatelům trpícím nevolností při ovládání aplikací z pohledu první osoby. Dále je možné nastavit citlivost ovládání kamery.

Cíle v zadání práce byly úspěšně splněny a aplikace bude zavedena jako součást prezentace společnosti USPIN s.r.o.

5 Závěr

Cílem práce bylo navrhnout a vytvořit prezentaci firmy pomocí 3D modelování. Byla vytvořena samostatně spustitelná aplikace, která umožňuje firmě USPIN s. r. o. prezentovat vlastní produkt z oblasti strojového vidění. Aplikace je interaktivní vizualizací funkce produktu, který by bylo obtížné prezentovat jinou formou. Uživatel se může volně pohybovat po virtuální místnosti, kde je řešení zahrnující produkt implementováno a získává tak lepší přehled o jeho funkcích. Toto řešení se zabývá automatizovanou kontrolou kvality polotovarů výfukových trubek a díky interaktivní prezentaci může firma předvést vlastní řešení klientovi tak, aby byla zřejmá jeho efektivita oproti implementaci manuální kontroly. Vizualizace řešení je částečně zjednodušena tak, aby byla jeho funkce srozumitelná i pro klienty, kteří nemají s oblastí strojového vidění mnoho zkušeností. Aplikace je dostupná ve verzích pro Windows, Mac OS a dále WebGL, což umožní její spuštění ve webovém prohlížeči.

Rešeršní část se zabývá problematikou 3D modelování, se zaměřením na hard-surface modelování a rendering v reálném čase. Pro porozumění základním pojmům v těchto oblastech je také v rešeršní části stručně popsán dosavadní vývoj v oblasti počítačové grafiky a 3D modelování. Nástroje pro modelování a implementaci samotné aplikace byly vybrány na základě požadavků autora na specifické funkce a v neposlední řadě s ohledem na jejich pořizovací náklady. Pro modelování byl vybrán nástroj Blender a pro implementaci aplikace prostředí Unity.

Výsledná aplikace funguje na platformách Windows a Mac OS a dále může být uložena na serveru a pomocí platformy WebGL spouštěna ve webových prohlížečích. Během testování byla potvrzena funkčnost všech implementovaných prvků. Aplikace byla dále otestována na šesti PC sestavách a bylo zjištěno, že největší vliv na dosaženou snímkovou frekvenci má grafická karta dané sestavy. Během tohoto měření výkonu bylo dosaženo uspokojivé snímkové frekvence na všech sestavách, včetně notebooku s pouze integrovaným grafickým čipem z roku 2012. Je tedy možné tvrdit, že aplikace by měla bez problémů fungovat na většině cílových zařízení. Cíle práce tak byly splněny a aplikace se stane součástí prezentace reálného produktu společnosti USPIN s. r. o.

6 Seznam použitých zdrojů

[1] B. Napper, „The Williams Tube", *Computer 50*, 1998. [Online]. Dostupné z: http://curation.cs.manchester.ac.uk/computer50/www.computer50.org/kgill/williams/williams.html. [Citován: 26. 7. 2019].

[2] B. Napper, „A Display for a Williams-Kilburn CRT Store", *Computer 50*, 1998. [Online]. Dostupné z: http://curation.cs.manchester.ac.uk/computer50/www.computer50.org/kgill/williams/display.html. [Citován: 25. 6. 2019].

[3] A. Chopine, *3D ART ESSENTIALS: The Fundamentals of 3D Modeling, Texturing, and Animation*. Elsevier Inc., 2011.

[4] W. E. Carlson, *Computer Graphics and Computer Animation: A Retrospective Overview*. The Ohio State University, 2017.

[5] B. T. Phong, „Illumination for Computer Generated Pictures", *Commun. ACM*, roč. 1975, č. 6, 1975.

[6] J. Norman, „Star Trek II Includes the First Completely Computer-Generated (CGI) Cinematic Image Sequence in a Feature Film", *History of Information*. [Online]. Dostupné z: http://www.historyofinformation.com/detail.php?entryid=3584. [Citován: 29. 7. 2019].

[7] E. Soares, „Tron (1982)", *Turner Classic Movies*, 2005. [Online]. Dostupné z: http://www.tcm.com/tcmdb/title/93956/Tron/articles.html. [Citován: 29. 7. 2019].

[8] Disney a Pixar, „Our story", *Pixar Animat. Studio*.

[9] Blender Foundation, „History", *blender.org*, 2013. [Online]. Dostupné z: https://www.blender.org/foundation/history/. [Citován: 30. 7. 2019].

[10] Pixar, „RenderMan - Choose the right solution for you", *Pixar's Renderman*, 2019. [Online]. Dostupné z: https://renderman.pixar.com/store. [Citován: 30. 7. 2019].

[11] Josh Petty, „What is 3D Modeling & What's It Used For?", *Concept Art Empire*, 2019. [Online]. Dostupné z: https://conceptartempire.com/what-is-3d-modeling/. [Citován: 1. 8. 2019].

[12] Haresh Khemani, „What is Solid Modeling? 3D CAD Software. Applications of Solid Modeling.", *Bright Hub Engineering*, 2019. [Online]. Dostupné z: https://www.brighthubengineering.com/cad-autocad-reviews-tips/19623-applications-of-cad-software-what-is-solid-modeling/. [Citován: 1. 8. 2019].

[13] Margaret Rouse, „3D model", *TechTarget: WhatIs.com*, 2016. [Online]. Dostupné z: https://whatis.techtarget.com/definition/3D-model. [Citován: 1. 8. 2019].

[14] Margaret Rouse, „3D mesh", *TechTarget: WhatIs.com*, 2016. [Online]. Dostupné z: https://whatis.techtarget.com/definition/3D-mesh. [Citován: 1. 8. 2019].

[15] Pluralsight, „What's the difference between hard surface and organic modeling?", *Pluralsight*, 16. 6. 2015. [Online]. Dostupné z: https://www.pluralsight.com/blog/film-

games/whats-the-difference-between-hard-surface-and-organic-models. [Citován: 11. 10. 2019].

[16] Glen Southern, „Tips and tricks for organic modelling", *Creative Bloq,* 18. 7. 2012. [Online]. Dostupné z: https://www.creativebloq.com/tips-and-tricks-organic-modelling-7123070. [Citován: 14. 10. 2019].

[17] Thomas Denham, „What is 3D Hard Surface & Organic Modeling?", *Concept Art Empire*, 2019. [Online]. Dostupné z: https://conceptartempire.com/hard-surface-organic-modeling/. [Citován: 11. 10. 2019].

[18] TurboSquid, „Tris, Quads & N-Gons", *Turbosquid 3D Resources,* 2017. [Online]. Dostupné z: https://www.turbosquid.com/3d-modeling/training/modeling/tris-quads-n-gons/. [Citován: 11. 10. 2019].

[19] Autodesk, „Planar and non-planar polygons", *Autodesk Knowledge Network,* 23. 1. 2015. [Online]. Dostupné z: https://knowledge.autodesk.com/support/maya-lt/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/MayaLT/files/Polygons-overview-Planar-and-nonplanar-polygons-htm.html. [Citován: 11. 10. 2019].

[20] Scratchapixel, „Introduction to Shading", *Scratchapixel 2.0*, 2016. [Online]. Dostupné z: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/shading-normals. [Citován: 21. 10. 2019].

[21] Eric W. Weisstein, „Normal Vector", *MathWorld--A Wolfram Web Resource,* 2002. [Online]. Dostupné z: http://mathworld.wolfram.com/NormalVector.html. [Citován: 21. 10. 2019].

[22] Blender Foundation, „Proportional Edit", *Blender 2.79 Manual,* 2019. [Online]. Dostupné z: https://docs.blender.org/manual/en/2.79/editors/3dview/object/editing/transform/control/proportional_edit.html. [Citován: 14. 10. 2019].

[23] Blender Foundation, „Extrude", *Blender 2.79 Manual*, 2019. [Online]. Dostupné z: https://docs.blender.org/manual/en/2.79/modeling/meshes/editing/duplicating/extrude.html. [Citován: 14. 10. 2019].

[24] 3D-Ace, „Polygonal 3D Modeling Techniques", *3D-Ace.com*, 2018. [Online]. Dostupné z: https://3d-ace.com/press-room/articles/polygonal-3d-modeling-techniques. [Citován: 11. 10. 2019].

[25] Pixar, „Modeling Tips", *OpenSubdiv Documentation*, 2017. [Online]. Dostupné z: https://graphics.pixar.com/opensubdiv/docs/mod_notes.html. [Citován: 14. 10. 2019].

[26] Autodesk, „Subdivision Surface Modeling". Autodesk, Inc, 2011.

[27] Blender Foundation, „Boolean Modifier", *Blender 2.80/ Manual,* 2019. [Online]. Dostupné z: https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/booleans.html. [Citován: 16. 10. 2019].

[28] Autodesk, „Combining polygon meshes", *Autodesk Knowledge Network,* 21. 4. 2018. [Online]. Dostupné z: https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/Maya-Modeling/files/GUID-79B4AF50-94D0-424F-BCB9-1DFCCCE093AD-htm.html. [Citován: 16. 10. 2019].

[29] Eskil Steenberg, „Model for Real Time—Beyond Counting Polygons", *Intel Developer Zone*, 16. 11. 2018. [Online]. Dostupné z: https://software.intel.com/en-us/articles/model-for-real-time-beyond-counting-polygons. [Citován: 23. 10. 2019].

[30] OpenGL, „The graphics pipeline". [Online]. Dostupné z: https://open.gl/drawing.

[31] J. Birn, *Digital Lighting and Rendering, Third Edition*. New Riders, 2014.

[32] Huamin Wang, „Texture Mapping", prezentováno v Computer Science Education, Ohio State University, 2013.

[33] Thomas Denham, „What is UV Mapping & Unwrapping?", *Concept Art Empire*, 2019. [Online]. Dostupné z: https://conceptartempire.com/uv-mapping-unwrapping/. [Citován: 30. 10. 2019].

[34] Cirstyn Bech-Yagher, „UV mapping for beginners", *Creative Bloq*, 23. 10. 2018. [Online]. Dostupné z: https://www.creativebloq.com/features/uv-mapping-for-beginners. [Citován: 30. 10. 2019].

[35] Blender Foundation, „Materials", *Blender 2.79 Manual*, 2019. [Online]. Dostupné z: https://docs.blender.org/manual/en/2.79/render/blender_render/materials/introduction.html. [Citován: 1. 11. 2019].

[36] TurboSquid, „Materials & Texturing", *Turbosquid 3D Resources*, 2019. [Online]. Dostupné z: https://www.turbosquid.com/3d-modeling/materials-texturing/. [Citován: 1. 11. 2019].

[37] D. G. Chaudhary, R. D. Gore, a B. W. Gawali, „Inspection of 3D Modeling Techniques for Digitization", *Int. J. Comput. Sci. Inf. Secur.*, roč. 2018, č. 7, s. 13.

[38] J. S. Aber, I. Marzolff, a J. B. Ries, *Small-Format Aerial Photography*, roč. 2010. Elsevier Inc.

[39] Techopedia, „Rendering", *Techopedia*, 2019. [Online]. Dostupné z: https://www.techopedia.com/definition/9163/rendering. [Citován: 4. 11. 2019].

[40] A. Baresgyan, „Unity and Unreal Engine: Real-time Rendering VS Traditional 3DCG Rendering Approach", *CGI Coffee*, 27. 1. 2018. [Online]. Dostupné z: http://cgicoffee.com/blog/2018/01/unity-real-time-rendering-vs-offline-cgi. [Citován: 8. 11. 2019].

[41] T. Akenine-Möller, E. Haines, a N. Hoffman, *Real-Time Rendering*, 3. vyd. Wellesley, Massachusetts: A K Peters, Ltd., 2008.

[42] Microsoft, „Graphics Pipeline", *Windows Dev Center*, 31. 5. 2018. [Online]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/direct3d11/overviews-direct3d-11-graphics-pipeline?redirectedfrom=MSDN. [Citován: 13. 11. 2019].

[43] J. O'Connor, *Mastering mental ray: Rendering Techniques for 3D & CAD Professionals*. Indianapolis, Indiana USA: Wiley Publishing, Inc., 2010.

[44] Unity Technologies, „Types of light", *Unity Manual 2019.2-003J*, 2019. [Online]. Dostupné z: https://docs.unity3d.com/Manual/Lighting.html. [Citován: 2. 1. 2020].

[45] J. Ribelles, A. López, a O. Belmonte, „An Improved Discrete Level of Detail Model Through an Incremental Representation". The Eurographics Association, 2010.

[46] N. Souto, „Video Game Physics Tutorial - Part II: Collision Detection for Solid Objects", *Toptal Technology*, 2015. [Online]. Dostupné z: https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects. [Citován: 8. 1. 2020].

[47] OpenGL, „Tutorial 3 : Matrices", *OpenGL Tutorial*. [Online]. Dostupné z: Tutorial 3 : Matrices. [Citován: 18. 11. 2019].

[48] R. Leadbetter, „Digital Foundry: The Future of Anti-Aliasing", *Digital Foundry*, 16. 7. 2011. [Online]. Dostupné z: https://www.eurogamer.net/articles/digital-foundry-future-of-anti-aliasing. [Citován: 13. 1. 2020].

[49] A. Youseff, „Image Downsampling and Upsampling Methods". Department of EECS - The George Washington University.

[50] S. Conroy, „What is Anti-Aliasing?", *WePC*, 10. 12. 2019. [Online]. Dostupné z: https://www.wepc.com/tips/what-is-anti-aliasing/. [Citován: 13. 1. 2020].

[51] NVIDIA Corporation, „FXAA Sample", 2019. [Online]. Dostupné z: https://docs.nvidia.com/gameworks/content/gameworkslibrary/graphicssamples/opengl_samples/fxaa.htm. [Citován: 13. 1. 2020].

[52] NVIDIA Corporation, „Temporal Anti-Aliasing", *GeForce Technology*, 2019. [Online]. Dostupné z: https://www.geforce.com/hardware/technology/txaa/technology. [Citován: 13. 1. 2020].

[53] Unity Technologies, „Global Illumination", *Unity Manual 2020.1-003K*, 13. 1. 2020. [Online]. Dostupné z: https://docs.unity3d.com/2020.1/Documentation/Manual/GIIntro.html. [Citován: 17. 1. 2020].

[54] NVIDIA Corporation, „NVIDIA RTX Ray Tracing", *NVIDIA Developer*, 2018. [Online]. Dostupné z: https://developer.nvidia.com/rtx/raytracing. [Citován: 17. 1. 2020].

[55] A. P. Andrei Tatarinov, „Advanced Ambient Occlusion Methods for Modern Games", prezentováno v Game Developers Conference, 26. 4. 2017.

[56] Alexey Panteleev, „VXAO: Voxel Ambient Occlusion", *NVIDIA Developer*, 23. 3. 2016. [Online]. Dostupné z: https://developer.nvidia.com/vxao-voxel-ambient-occlusion. [Citován: 20. 1. 2020].

[57] Evelina Fairclough, „What is Ambient Occlusion?", *The WiredShopper*, 15. 12. 2019. [Online]. Dostupné z: https://thewiredshopper.com/ambient-occlusion/. [Citován: 20. 1. 2020].

[58] USPIN s.r.o., „O společnosti USPIN s.r.o.", *USPIN.cz*. [Online]. Dostupné z: https://www.uspin.cz/cz/o-spolecnosti. [Citován: 27. 1. 2020].

[59] USPIN s.r.o., „Vědecký výzkum", *USPIN.cz*. [Online]. Dostupné z: https://www.uspin.cz/cz/vedecky-vyzkum. [Citován: 27. 1. 2020].

[60] USPIN s.r.o., „Referenční projekty", *USPIN.cz*. [Online]. Dostupné z: https://www.uspin.cz/cz/referencni-projekty. [Citován: 27. 1. 2020].

[61] Maxon Inc., „Cinema 4D", *Maxon.net*, 2020. [Online]. Dostupné z: https://www.maxon.net/en-us/products/cinema-4d/overview/. [Citován: 29. 1. 2020].

[62] Autodesk, „Maya Overview", *Autodesk.com*, 2020. [Online]. Dostupné z: https://www.autodesk.com/products/maya/overview. [Citován: 29. 1. 2020].

[63] Autodesk, „3ds Max Overview", *Autodesk.com*, 2020. [Online]. Dostupné z: https://www.autodesk.com/products/3ds-max/overview. [Citován: 29. 1. 2020].

[64] Blender Foundation, „About", *blender.org*, 2020. [Online]. Dostupné z: https://www.blender.org/about/. [Citován: 29. 1. 2020].

[65] Crytek GmbH., „CryEngine", *CryEngine*, 2019. [Online]. Dostupné z: https://www.cryengine.com/. [Citován: 31. 1. 2020].

[66] Epic Games, „Unreal Engine", *Unreal Engine*, 2019. [Online]. Dostupné z: https://www.unrealengine.com/en-US/. [Citován: 31. 1. 2020].

[67] Unity Technologies, „Unity 3D", *Unity*, 2019. [Online]. Dostupné z: https://unity.com/solutions/game. [Citován: 31. 1. 2020].

[68] Lennart Demes, „CC0Textures", *CC0Textures.com*, 2019. [Online]. Dostupné z: https://cc0textures.com/. [Citován: 20. 10. 2019].

[69] Dorian Zgraggen, „cgBookCase", *cgBookCase.com*, 2019. [Online]. Dostupné z: https://www.cgbookcase.com/. [Citován: 20. 10. 2019].

[70] Creative Commons, „CC0 1.0 Univerzální (CC0 1.0) Potvrzení o statusu volného díla", *CreativeCommons.org*, 2020. [Online]. Dostupné z: https://creativecommons.org/publicdomain/zero/1.0/deed.cs. [Citován: 3. 2. 2020].

[71] W. Goldstone, *Unity Game Development Essentials*, roč. 2009. Packt Publishing.

[72] T. Norton, *Learning C# by Developing Games with Unity 3D*, roč. 2013. Packt Publishing, 2013.

7 Přílohy

7.1 Příloha 1: Testovací protokol – vytížení systému

Diplomová práce - Prezentace firmy pomocí 3D modelování							
Autor: Miroslav Šimák							
TESTOVACÍ PROTOKOL							
Vytížení systému							
ID protokolu: V1							
Datum	26.02.2020						
Informace o PC							
Model CPU							
Model GPU							
Kapacita RAM							
Instrukce pro testování:							
1. Spusťte nástroj MSI Afterburner nastavte klávesy pro *Benchmarking* a *On-Screen Display* (menu *nastavení -> Benchmarking* a *nastavení->On-Sceen Display*)							
2. Spusťte aplikaci QC Automation.exe a v nastavení vyberte konfiguraci kvality							
3. V hlavním menu klikněte na tlačítko PLAY a zapněte měření v MSI Afterburner							
4. Nechte aplikaci běžet 10 minut, z On-Sceen Display MSI Afterburner zaznamenejte vytížení systému a poté vypněte měření							
5. Ze souboru Benchmark.txt vytvořeného pomocí MSI Afterburner zaznamenejte výsledné hodnoty testu do níže uvedené tabulky							
6. Opakujte kroky 2-5 pro nastavení kvality "Very High"							
7. Problémy při testování zaznamenejte do níže uvedeného pole "Poznámka"							
Výsledky testování "Very Low"			**Výsledky testování "Very High"**				
FPS (průměr)			FPS (průměr)				
Vytížení CPU (%)			Vytížení CPU (%)				
Vytížení GPU (%)			Vytížení GPU (%)				
Využití RAM (MB)			Využití RAM (MB)				
Poznámka:							
zde vložte poznámky (například problémy vzniklé při testování)							

7.2 Příloha 2: Testovací protokol – funkce aplikace

Diplomová práce - Prezentace firmy pomocí 3D modelování

Autor: Miroslav Šimák

TESTOVACÍ PROTOKOL

Funkce aplikace

ID protokolu: F1

Datum	26.02.2020
Informace o PC	

Model CPU	Intel i5 3570K @ 4.2 GHz
Model GPU	Nvidia GeForce GTX 1070
Kapacita RAM	12 GB

Test funkcí

Funkce	**Výsledek**
Pohyb uživatele a nastavení hranic scény	OK
Nastavení šance na vadný polotovar pomocí klávesnice	OK
Správné zobrazení šance na vadný polotovar na hlavní obrazovce	OK
Funkce nastavení zorného pole a citlivosti kamery	OK
Funkce nastavení kvality zobrazení	OK
Přepínání mezi hlavní scénou a menu, ukončení aplikace pomocí tlačítka *Exit*	OK

Poznámka:

Testování proběhlo v pořádku

7.3 Příloha 3: Výsledná aplikace (elektronicky + DVD)

Výsledná aplikace je nahrána jako součást příloh v elektronické verzi práce ve variantách pro Windows a WebGL. Soubory jsou uloženy o archivu *XSIMM031_DP_Aplikace.zip*.

Na DVD je přiložen navíc zdrojový projekt v prostředí Unity a soubor *.blend* obsahující všechny použité modely vytvořené v nástroji Blender. Dále DVD obsahuje *.fbx* soubory jednotlivých modelů, které byly využity pro přenos objektů mezi prostředími Blender a Unity. Tyto doplňkové soubory nejsou samotnými výsledky diplomové práce, slouží pro ověření, že byla práce vytvořena samostatně a nebyly nahrány do IS jako její součást, protože by došlo k překročení maximální datové kapacity. Na DVD je navíc přiložena i verze aplikace pro Mac OS.

8 Seznam obrázků

9 Seznam tabulek