VYSOKÁ ŠKOLA

KREATIVNÍ KOMUNIKACE

Katedra vizuální tvorby

BAKALÁŘSKÁ PRÁCE

Unreal Engine a průlomy v real-time renderingu

2022 **Šimon Kryštof Honal**

VYSOKÁ ŠKOLA KREATIVNÍ KOMUNIKACE

Katedra vizuální tvorby

Vizuální a literární umění

Animace a vizuální efekty

Unreal Engine a průlomy v real-time renderingu

Teoretická část: Unreal Engine a průlomy v real-time renderingu

Praktická část: Animovaný 3D film

Autor: Šimon Kryštof Honal

Vedoucí práce: MgA. Martin Hovorka

2022

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze kterých jsem čerpal. Souhlasím s tím, aby práce byla zpřístupněna veřejnosti pro účely studia a výzkumu.

V Bdeněvsi dne…… Podpis autora:

PODĚKOVÁNÍ

Rád bych touto formou poděkoval vedoucímu bakalářské práce, MgA. Martinu Hovorkovi, který mi nejednou pomohl, a to nejen s bakalářskou prací. Dále chci poděkovat svým rodičům, kteří mi studium na VŠKK umožnili. Také chci poděkovat Janu Kastnerovi, se kterým jsme spolupracovali na praktické části bakalářské práce. Děkuji.

ABSTRAKT

Práce se zabývá o průlomy v oblasti real-time (tzn. online) renderingu, tj. renderingu v reálném čase, který se doposud užíval především, či skoro exkluzivně, v počítačových hrách. Tento trend nyní postupně upadá a díky novým technologiím v oblasti renderingu, jmenovitě programu Unreal Engine, se rendering v reálném čase užívá čím dál tím více i v mainstreamové vizuální produkci, např. v seriálu „*The Mandalorian*“. Práce také porovnává přístupy real-time (online) a offline renderingu, jejich využití a specifika.

Největší část práce je věnována programu Unreal Engine, původně videohernímu enginu, který nyní vede audiovizuální průmysl a klestí cestu novým technologiím. Unreal Engine mění paradigma real-time renderingu jako takového a má hluboký efekt napříč celým průmyslem audiovize.

Hlavním cílem práce je obhajoba real-time renderingu jakožto plnohodnotného způsobu vykreslování nejen pro videohry, ale i pro filmovou produkci.

Dále se práce věnuje budoucnosti audiovize a implikacím, které nové technologie přinášejí. Sekundárním cílem práce je předpovědět vývoj audiovizuálního průmyslu.

Poslední část práce popisuje kreativní proces praktické části bakalářské práce a její technické řešení při využití real-time renderingu v programu Unreal Engine.

KLÍČOVÁ SLOVA

Grafika, počítačová grafika, geometrie, engine, rendering, Unreal Engine, animace, vizuální efekty, virtuální produkce, filmový průmysl, videoherní průmysl, photogrammetry, ray-tracing, virtuální realita, augmentovaná realita, mixovaná realita.

ABSTRACT

This thesis describes breakthroughs in the area of real-time (online) rendering, which had, up until recently, been used mostly, or almost exclusively, in videogames. This trend is now gradually declining thanks to new rendering technologies, namely the Unreal Engine program. Real-time rendering is now used more and more in mainstream visual productions, eg. In ‚*The Mandalorian*‘ series. This thesis also compares different workflows, use-cases and specifics of real-time (online) and offline rendering.

The bulk of the thesis is dedicated to the Unreal Engine program, originally a videogame engine, which now leads the audiovisual industry and is paving the way for new technologies. Unreal Engine is changing the paradigm of real-time rendering as a whole and has a profound effect on the entirety of the audiovisual industry.

The primary goal of this thesis is the defense of real-time rendering as a fully-fledged method of rendering, not only in the case of videogames but also for filmmaking.

Furthermore, this thesis talks about the future of the audiovisual industry and the implications of the aforementioned new technologies. The secondary goal of this thesis is to predict the development of this industry.

The final part of the thesis describes the creative process of the practical part of the bachelor thesis and its technical realization using Unreal Engine’s real-time rendering.

KEYWORDS

Graphics, computer graphics, geometry, rendering, Unreal Engine, animation, visual effects, virtual production, film industry, videogame industry, photogrammetry, ray-tracing, virtual reality, augmented reality, mixed reality.

OBSAH

1. ÚVOD

Ať už jde o animovaný film, fyzikální simulaci, či počítačovou hru, všechna tato média musela projít dlouhým a náročným kreativním procesem. Onen proces si je ve spoustě ohledech podobný – ale přehlédněme nyní scénáristy a herce, skladatele a kameramany, režiséry a modeláře. I bez nich nám zbývá jeden společný termín, něco, čím si v dnešní době projde jak zmiňovaná počítačová hra, tak i velkometrážní celovečerní film. Ptáte se co?

Rendering.

Řekněme, že natáčíme film odehrávající se v daleké budoucnosti na odlehlé planetě, s herci procházejícími nereálnými scenériemi a bojujíc s efektními bouřemi. Samozřejmě, že natáčení v takové lokalitě je – alespoň prozatím – nemožné. Pokud chce režisér natáčet on-location (tj. natáčení filmu ve skutečné scenérii), může při nejlepším aproximovat svou vizi při výběru lokace.

Jenže je zde i jiná možnost. Ohlédněme se nyní za počítačovými hrami. Počítačové hry se odehrávají v plně virtuálním, fiktivním prostředí, které je cíleně vytvořeno týmem umělců. Ti onu vizi odlehlé planety někde daleko v budoucnosti změní ve skutečnost a hráč tak může prožít dobrodružství v prostředích, které jsou limitovány pouze představivostí scénáristů.

Fiktivní prostředí jsou pouze jedním příkladem užití renderingu jak ve hrách tak ve filmech, dále se můžeme bavit o vizuálních efektech, či využití fiktivních postav např. místo kaskadérů. Tak či onak, ani film, ani hra se bez renderingu – tj. vykreslování – neobejde.

1.1 Co je to rendering?

Co tedy je rendering? Rendering (česky renderování, či vykreslování) je proces, při kterém jsou pomocí počítačové grafiky vizualizována data ve formě obrazu. Vždy jde o poslední fázi produkce ať už animovaných, či jinak digitálně upravovaných filmů (např. filmů, které uživají digitální vizuální efekty). V případě videoher jde o finální proces zpracování grafických dat v reálném čase, jehož výstupem je obraz.

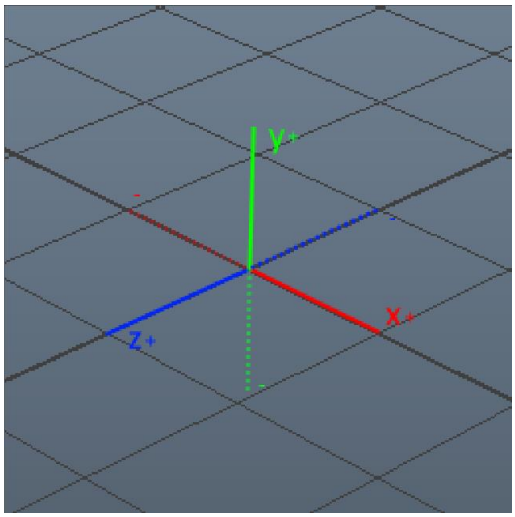
Základem renderingu je jedna jediná otázka: Jakou barvu má daný pixel na obrazovce? Tuto otázku si klade tzn. „renderer“, který na ni musí nalézt odpověď pro miliony pixelů, snímek po snímku, dvacetpětkrát za vteřinu.

Podobně jako je tomu u umělecké fotografie, rendering je jednodušše vyobrazení podoby scény, prostředí, či modelu. Kvalitu a celkový vzhled takové fotografie určují dva hlavní aspekty: Sensor a světlo. Každý vizuální vjem je výsledkem kontinuelních odrazů trilionů světelných paprsků, které nasvětlují a tím i barví scény kolem nás. Více k tomuto tématu v kapitole Ray-Tracing.

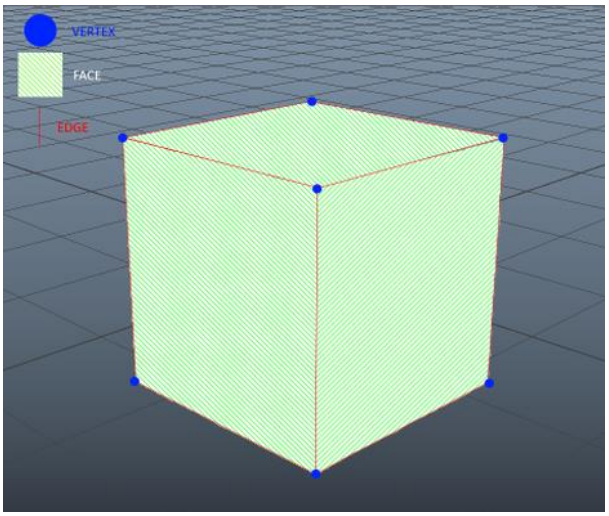
Přístup k renderování okem fotografa je ideální přístup k pochopení základů renderingu. Pro hlubší pochopení termínu rendering si musíme vysvětlit základy 3D grafiky.

1.1.1 Základy 3D grafiky

Trojrozměrná grafika je ve svém základě klasická geometrie. Pracujeme tedy s kartézskou soustavou souřadnic. Máme tři osy – X, Y, Z. Typicky osy X a Y znázorňují dvojrozměrný prostor a osa Z popisuje hloubku obsahu.



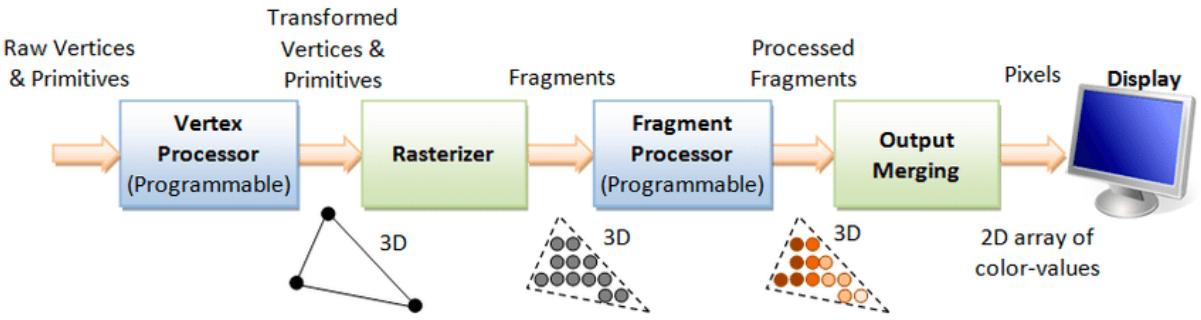
Obr. 1 – Osový kříž souřadnic



Obr. 2 –Vertexy, faces a edges krychle

Do souřadnic popisujeme modely. Tyto modely se skládají z polygonů – tzn. „mnohoúhelníků“ – nejčastěji jde o quady (čtyřhrany) či trojuhelníky. Tyto polygony jsou definovány tzv. vertexy. Vertexy jsou body určeny souřadnicemi na kartézské sousavě. Pokud jsou tři vertexy spojeny tzv. edgem – hranou (či linkou), definujeme nejjednodušší možný geometrický tvar, trojuhelník. Jakýkoliv dvojdimenzionální tvar takto definovaný známe pod pojmem face (rovina). Nadefinovali jsme si trojrozměrný model.

2. TECHNOLOGIE RENDERINGU



Obr. 3 – Flowchart rasterizace[1]

2.1 Proces renderingu a jeho metody

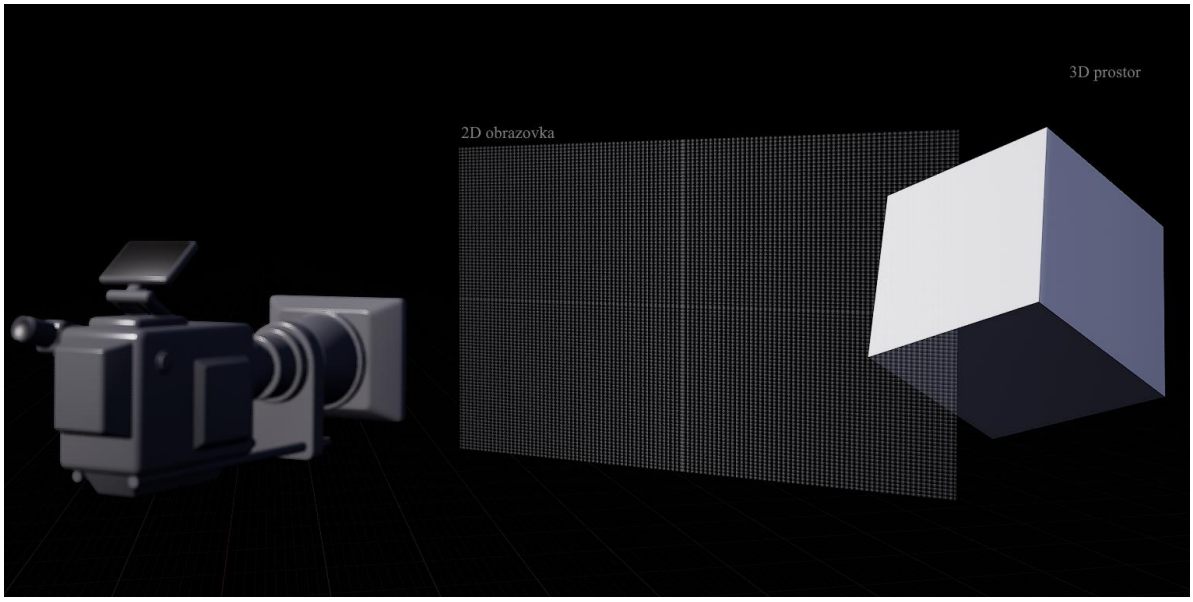
Proces renderingu můžeme zjednodušit na pět hlavních kroků. Prvním krokem je získat ta nejjednodušší surová data geometrie naší scény – vertexy (body). Vertexy přes proces rasterizace promítáme na dvojdimenzionální plochu pomyslné obrazovky. Tímto získáváme fragmenty. Z fragmentů vytváříme tvary, které se následně barví a výsledkem tohoto procesu jsou pixely na počítačové obrazovce.

Definováním našeho trojrozměrného objektu se dostáváme za fázi Vertex Processor. Nyní musíme napamovat model na dvojrozměrnou obrazovku. Tento proces se nazývá Rasterizace.[2]

[1],[2] Chaoyang *2019, s. 2*

2.1.1 Rasterizace

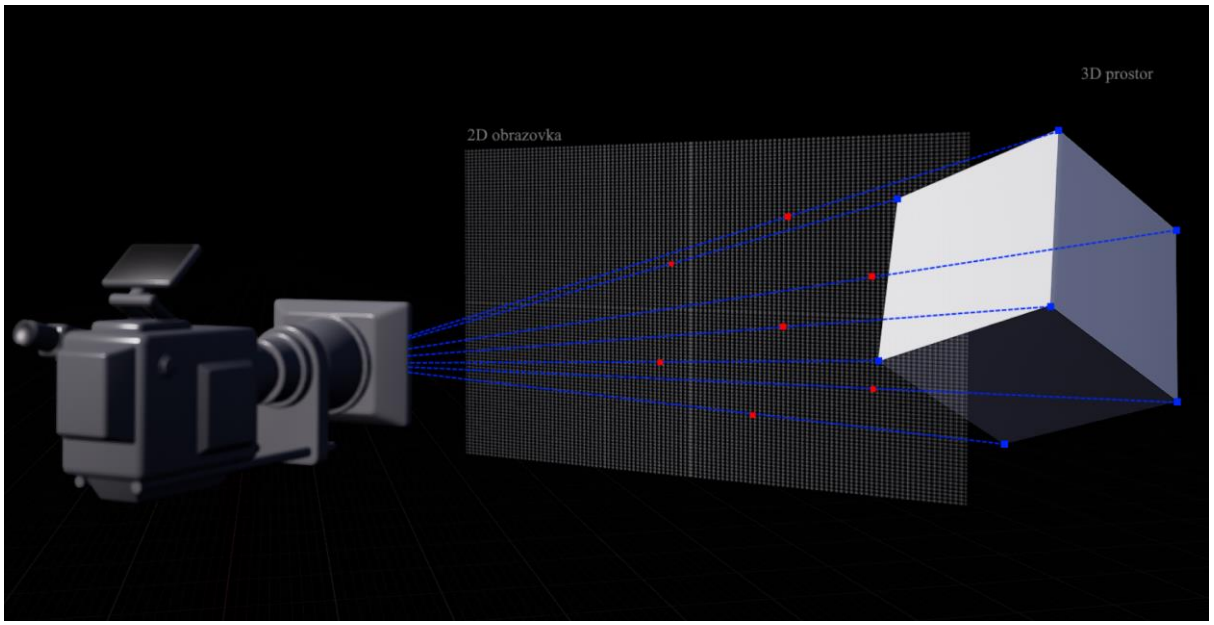
Rasterizace je metoda vyobrazení trojdimenzionálních předmětů na dvojdimenzionální plochu – přesněji řečeno, jde o mapování polygonů k pixelům na obrazovce.



Obr.4 – Vyobrazení 3D objektu před rasterizací.

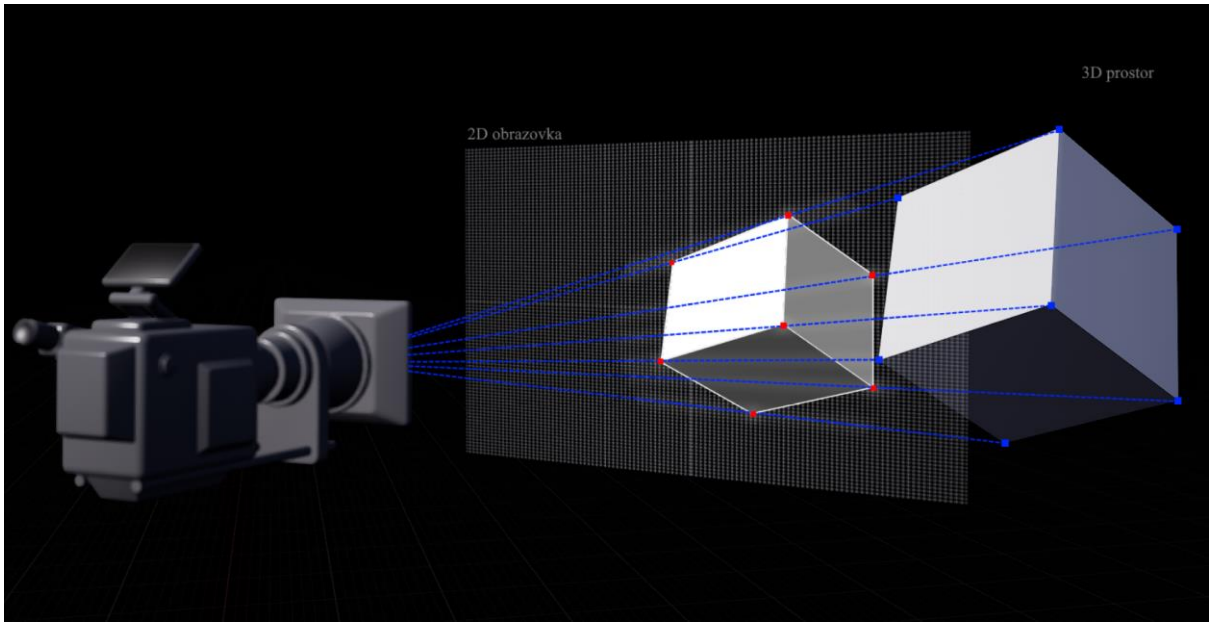
Představme si krychli (náš 3D objekt) v prostoru. Před něj vložme dvojdimenzionální plochu – ta nám poslouží jakožto pomyslný model obrazovky. Tato plocha se skládá z mnoha, mnoha čtverečků – pixelů naší obrazovky. Celou tuto scénu dále sleduje, pro zjednodušení, kamera – rasterizer (česky rasterizátor). Rasterizátor zná polohy vertexů naší krychle. Poté spočítá a promítne polohy všech těchto vertexů na obrazovku.[3]

[3] Lebrov, *2018. Parafráze*



Obr. 5 – Rasterizátor mapuje vertexy na fragmenty.

Vertexy namapované na obrazovku se nazývají fragmenty. Fragmenty nejsou to samé, co pixely. Pixely jsou finálním výstupem rasterizátoru, avšak fragmenty jsou pouze body namapované (přiřazené, či patřící k) finálním pixelům. Více fragmentů může být namapováno k jedinému pixelu! Tato situace nastává v momentě, kdy se více objektů přes sebe překrývá. [4]



Obr. 6 – Rasterizátor namapoval fragmenty k pixelům.

Nyní máme fragmenty a prostory mezi nimi. Ale stále nemůžeme objekt zobrazit, protože neznáme jeho vlastnosti. S tím nám pomůže tzv. pixel shader.

[4] Lebrov, *2018. Parafráze*

Pixel shader je počítačový program, který se provádí na GPU (grafické kartě) našich počítačů. Tento program nám určuje barvu každého jednoho fragmentu. Pokud se tedy snažíme zobrazit 7 fragmentů, vyžadujeme si 7 invokací pixel shaderu.

Pixel shader má přístup k datům, které určuje umělec – k texturám, funkcím a podobně. Dále mu jsou přístupná per-fragment data – data každého fragmentu, která jsou generována předchozí fází procesu rasterizace.

Data, která určuje umělec mimo jiné zahrnují tzv. textury. Textury jsou obrázky, často čtvercových (pravidelných) dimenzí, které určují barvu – popisují barevná data. Textur je mnoho druhů, každá se svým vlastním využitím. V moderním PBR (physically based rendering) workflow nejčastěji pracujeme s texturami popisujícími tzv. Albedo (barvu), Metallic (metalličnost – kovovost), Roughness (hrubost), Normal (normálová mapa, popisuje normálové vektory a jejich velikost) a AO (ambient occlussion – falešné předpočítané stínování).

Textury nejsou ani zdaleka jediným způsobem popisování vlastností našich modelů. Dále můžeme pracovat například s vektorovými parametry. V paradigmatu shaderů se vektorový parametr využívá především pro určení solidní, tzv. jednolité barvy. Vektor je ve trojrozměrném prostoru definován třemi komponenty - můžeme nad nimi přemýšlet jakožto nad komponenty X, Y, Z. Podobně, komponenty barvy na jakémkoliv monitoru jsou R, G, B. Mapujeme tedy tyto tři vektorové komponenty na komponenty barvy, čímž získáváme vektorový parametr, který nám určuje jednu solidní, difuzní barvu.

Dále stojí za zmíňku skalární parametry. Jak již jméno naznačuje, jde o číselnou hodnotu definujíc velikost. Zjednodušeně se bavíme o čísle, kterým můžeme vynásobit např. texturu pro ovládání její intenzity. Pokud texturu vynásobím skalárním parameterem 0, její data se vynulují. Stejně tak, pokud vynásobím texturu skalárem 2, všechna data dané textury se zdvojnásobí. Dle užití různých renderingových metod interpretujeme a pracujeme s těmito (a dalšími) instrukcemi pro stínování (shading) našich modelů.

Lokální per-fragment data zahrnují například tzv. vertex color. Vertex color je barevná hodnota RGB a její alpha kanál. Alpha kanál zajišťuje průhlednost či neprůhlednost barvy. Proces renderingu, který generuje samotné fragmenty interpoluje (lineární interpolace – zaplňování mezer mezi čísly, využívá se při hladkých přechodech mezi dvěma hodnotami či barvami, dále pro dopočítávání chybějících snímku v animaci)

barvy vertexů přes každý daný polygon a poté fragmentu přiřadí vhodnou interpolovanou barvu.

Takto jsme procesem rasterizingu schopni vyrenderovat jednoduchý 3D model. Rasterizace je nejjednodušší a nejužívanější formou renderingu. Při jakékoliv práci s 3D programem, který užíváme pro vytvoření samotných modelů vidíme rasterizovaný obraz. To také znamená, že samotný akt vytváření modelů, ještě před oním skutečným finálním vyrenderováním, je, již samo o sobě, rendering.

Rasterizace je extrémně rychlá, protože jde pouze o proces transformace geometrie naší scény do pixelů. [5]

Další, poněkud starší, metodou renderingu je tzv. Ray-casting.

[5] Lebrov, *2018. Parafráze*

2.1.2 Ray-casting

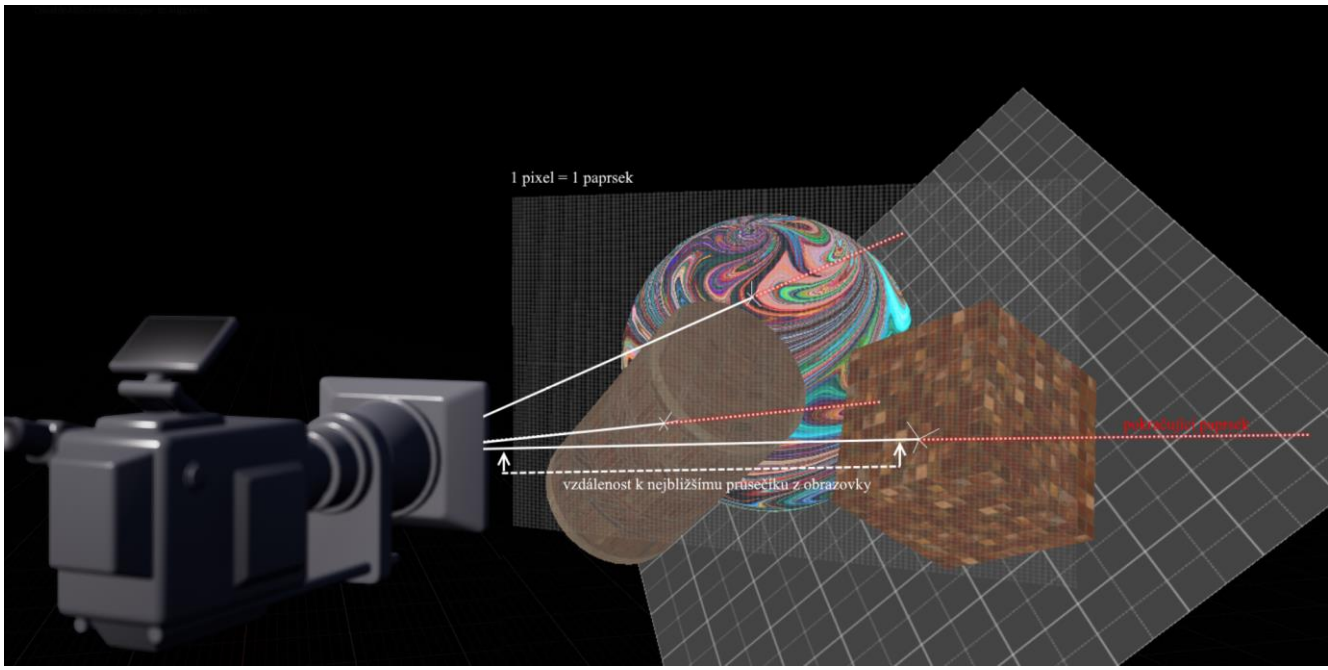
Ray-casting je další metodou renderingu. S termínem Ray-casting přišel Scott Roth, když pracoval v General Motors Research Labs v roce 1978. Jeho přístup značně zjednodušuje a také zlepšuje paradigma renderingu.

Na rozdíl od rasterizace, ray-casting již nepotřebuje znát polohy vertexů trojrozměrných objektů. Namísto toho ray-casting pracuje již s celými objekty. Rendering pomocí ray-castingu probíhá následovně.

Z kamery vyšleme paprsky – každým pixelem obrazovky jeden. To znamená, že pro rozlišení obrazovky 1920x1080px musíme vyslat 2 073 600 paprsků. V momentě, kdy paprsek pronikne jakýmkoliv objektem v scéně, počítač zaznamená nejbližší bod průniku vůči kameře a dále zjistí barevnou hodnotu povrchu protnutého paprskem.

Dále, protože paprsky vysíláme pouze jedním směrem – tzn. z obrazovky (z pixelů) do prostoru, tyto parpsky dále nijak neinteragují s prostředím. Tudíž se například neodrážejí, a tím pádem nemáme možnosti spekularity nebo stíny. Proto jsou barevná data povrchu určena pouze difuzní texturou. Takto zpracovaný rendering využívají první 3D a 2.5D hry – *Wolfenstein 3D, Doom* a další.

Obr. 7 – Screenshot z počítačové hry DOOM II



Obr. 8 – Znázornění principu ray-castingu

Koncem 70. Let minulého století přichází na scénu muž jménem J. Turner Whitted. Před jeho příchodem do sféry počítačové grafiky, Turner Whitted zkoumal chování fotonů a světelných paprsků v prostoru. Svou znalost světla uplatnil později, když navždy změnil metodu Ray-castingu a tím vytvořil nový, revoluční způsob renderingu; ray-tracing.[6]

2.1.3 Ray-tracing

J. Turner Whitted přišel s revoluční myšlenkou – Co, kdybychom paprsky, které posíláme do scény, emulovali světlo? Turner Whitted bere ray-casting a zavádí do něj tzv. ray-bouncing (odrazy světla, někdy také light-bounce). Místo toho, aby při průniku objektu paprskem počítač okamžitě zaznamenával vlastnosti povrchu tohoto objektu, při každém průniku paprsku objektem je vygenerován další paprsek. Každý tento paprsek se „odráží“ od zmiňovaného průsečíku a je veden k bodům, které emulují světlo – zdroje světla. Každému zdroji světla je přiřazen svůj vlastní paprsek. Jde tedy o opak chování světla, jak ho známe z reality. Proč je tomu tak?

Paprsky jsou vrhány z kamery především z důvodu toho, abychom ušetřili na výkonu počítače – každý jeden paprsek s sebou přináší značnou výpočetní cenu. Pokud

[6] Lebrov, *2018. Parafráze*

bychom posílali paprsky ze zdrojů světla, museli bychom počítat světelné interakce celé scény – to znamená i částí scény, které kamera nevidí.

Když posíláme paprsky „opačně“ – z kamery, renderujeme pouze to, co kamera vidí – pouze zorné pole kamery a tím ušetříme drahocenný čas. Této optimalizace se hojně využívá v online (real-time) renderingu například v počítačových hrách, kde je vše podřízeno výkonu počítače a kde potřebujeme, aby se obrázky dostávaly na obrazovku hráče co možná nejrychleji. Na rozdíl od filmu (offline rendering), který je produkován na frekvenci 24 (v televizní produkci 25) FPS (snímků za sekundu), počítačové hry mají za cíl minimální frekvenci 60FPS.

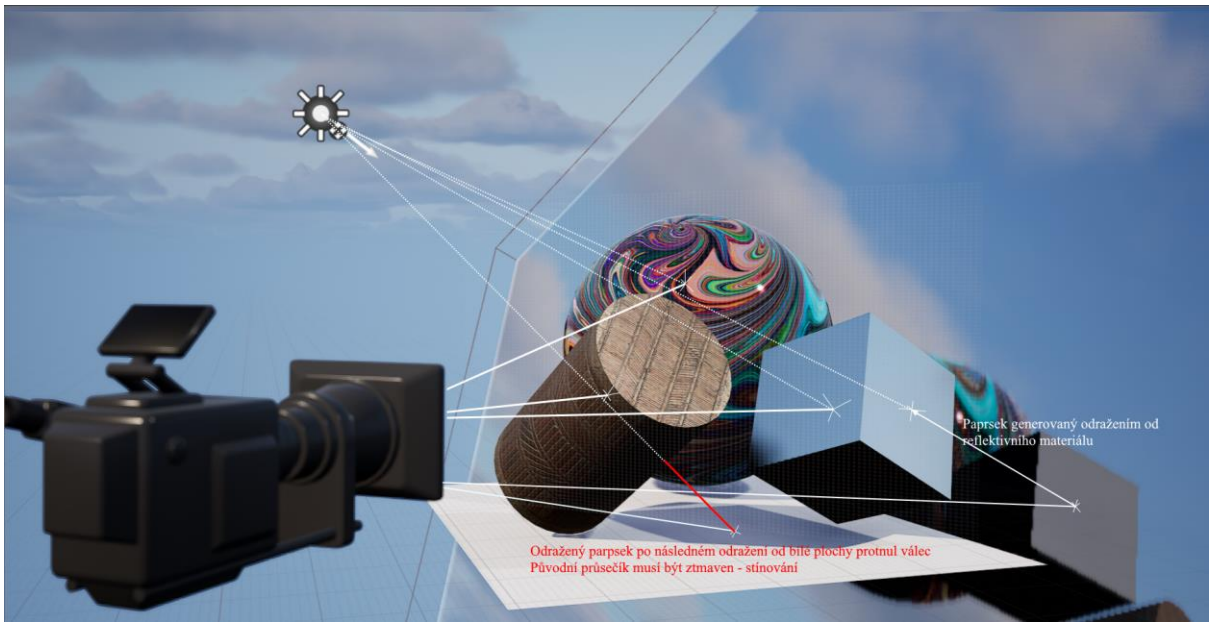
Tedy, pro rekapitulaci. Z každého pixelu na pomyslné obrazovce vysíláme paprsek. Pokud je naše výstupní rozlišení 1920x1080 pixelů, počet těchto paprsků je tedy 2 073 600. Tyto paprsky putují prostorem, dokud „nenarazí“ na objekt – model. Z bodu tohoto průsečíku se generují další paprsky. Počet těchto, odražených parpsků je závislý na počtu zdrojů světla – ke každému zdroji světla je vyslán jeden odražený paprsek.

Díky tomu je nyní možné stínování. Pokud odražený paprsek při své cestě ke zdroji světla narazil na další objekt, znamená to, že zdroj světla je něčím blokován – tudíž tento průsečík musí být temnější.

Dále, již nepracujeme pouze s jednou difúzní texturou – umělci nyní mohou užívat předem zmiňovaných textur, které ovládají např. spekularitu. Ray-tracing otevřel dveře pro tzv. materiály – Pixel-shader instrukce, které umělcům umožňují nastavovat vlastnosti povrchů modelů v jejich scénách. Pokud má, například, být objekt reflektivní, umělec tento fakt může nastavit pomocí reflektivního materiálu.

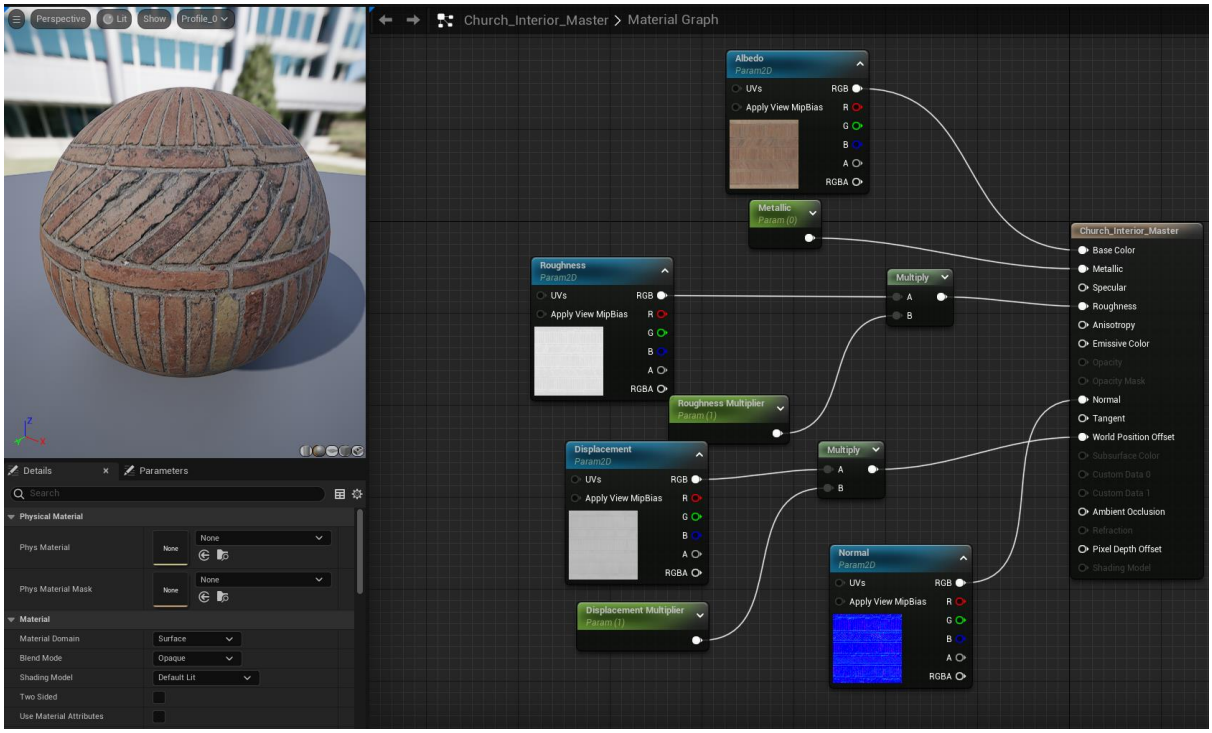
V momentě, kdy paprsek protne tento reflektivní materiál, počítač z tohoto průsečíku vyšle další paprsek pod určitým úhlem, dle nastavení materiálu. Tento odražený paprsek pod daným úhlem dále cestuje, dokud nezasáhne další objekt. Dle nastavení povrchu tohoto, nového, objektu jsou pak vystřeleny další paprsky, dokud nenacházejí zdroje světla. Tímto procesem materiál zjišťuje, které objekty se mají v reflektivním materiálu (povrchu původního průsečíku) odrážet.[7]

[7] *Lebrov, 2018. Parafráze*



Obr. 9 – Znázornění Ray-tracingu a odrážení paprsků

Užíváním materiálů se otevřel naprosto nový svět kontroly nad renderovanými scénami. Materiály určují chování světelných parpsků – zda se má paprsek odrazit či zda se má vstřebat, nebo obojí a jak moc. Ray-tracing dodnes užíváme v prakticky všech formách real-time renderingu. Rendering všech počítačových her a obecně 3D grafiky, kterou vidíme v reálném čase využívá ray-tracingu.



Obr. 10 – Materiálový editor programu Unreal Engine

Do roku 2018 se ray-traced rendering prováděl na grafických kartách softwarovým řešením. Ačkoliv se ray-tracing hodí pro real-time rendering, i tak jde o velice náročný proces, který i přes všechny své optimalizace umí počítač zaneprázdnit. Ono softwarové provedení bylo jen dalším krokem, který proces renderingu zpomaloval.

20. září roku 2018 přišla společnost nVidia, výrobce a inovátor grafických karet, s novou řadou grafických čipů, kterou nazvala nVidia RTX. Zkratka RTX napovídá hlavnímu využití těchto karet – ray-tracing. Nvidia přišli s novým hardwarovým řešením ray-tracingu a tím extrémně zrychlili proces real-time renderingu. O tom ale později.

I ray-tracing má své limitace. Největšími problémy ray-tracingu jsou tvrdé stíny a globální iluminace (GI). Ve zkratce, globální iluminace je jakási simulace nasvěcení scény pouze nepřímými – odraženými – paprsky světla. Dosažení obou těchto světelných fenoménů je pomocí ray-tracingu možné, avšak nejde o skutečnou simulaci, spíše o různé úpravy a podvádění lidských očí. Například globální iluminace se dosahuje pomocí post-processingu, při kterém detekujeme edges, které si jsou blízké a tak tvoří jakési „rohy“. Tyto rohy jsou pak uměle zteměnené. Real time indirect global illumination – nepřímá globální iluminace v realném čase - je považována za jakýsi svatý grál počítačové grafiky.

Pokud se chceme bavit o skutečném chování světla, jeho simulace a z té získání zmiňovaných měkkých stínů, či globální illuminace, musíme se bavit o další, a poslední, metodě renderingu. Jde o čistě offline renderovací metodu, jménem Path-tracing.

2.1.4 Path-tracing

Path-tracing je forma renderingu, která se pevně řadí do kategorie offline renderingu. To znamená, že výstup (obraz) se nezobrazuje na obrazovce ve skutečném čase a tudíž nemůže být interaktivní. Používá se výhradně pro filmovou produkci, či produkci, se kterou nebude nijak interaktivně zacházeno.

I path tracing se pořád drží pravidla vysílání paprsků z kamery. Avšak nyní nejde o jeden paprsek za každý jeden pixel, ale tisíce paprsků za každý jeden pixel. Dovedeme si tedy přestavit, o jak obrovských číslech se nyní bavíme.

Místo toho, aby se při dopadu paprsku generovaly paprsky další, path-tracing při průniku objektem dále odráží jeden a ten samý paprsek, co možná nejvícekrát. Tím se získávají informace o objektech, které jsou blízko u sebe a tím je možná i již zmiňovaná globální iluminace. Jedním z jejich největších vlastností je tzv. color bleeding. Jde o jev, kdy barva jednoho objektu se vlivem světla promítá na okolní povrchy.

Jednoduše řečeno, paprsky fungují naprosto stejně, jako světlo ve skutečném světě, když opomineme jejich vysílání z kamery.

Obr. 11 – Znázornění paprsků v path-tracingu

Dále jsou možné i skutečné měkké stíny. Skutečně simulované měkké stíny v ray-tracingu nejsou možné již z definice. Ray-tracing používá zdroje světla, které jsou ve skutečnosti pouze body v prostoru, které mají určité vlastnosti, data, dle kterých se poté chovají paprsky. Tím, že zdroje světla jsou pouze body zamezujeme jakémukoliv pravému změkčení stínů, protože paprsky tak putují pouze do jednoho bodu – stíny jsou tím pádem ostré.

Pro simulaci skutečných měkkých stínů potřebujeme zdroje světla, které mají plochu. Toho dosahuje path-tracing. Odražené paprsky totiž nyní trefují „světelnou plochu“ – zdroj světla, který je ve skutečnosti dvojdimenzionální rovinou. Čím větší je tato rovina, tím měkkčí bude náš stín. [8]

[8] *Lebrov, 2018. Parafráze*

Path-tracing je nejdetailnější možný způsob, jak renderovat obrazy co možná nejvěrnější realitě. Ovšem, tato extrémní přesnost také znamená extrémní náročnost. Path-tracing je velice náročný na hardware počítačů, na kterých pracuje. Proto je path-tracing odkázán jako forma renderingu určená čistě pro „předpočítaný“ (pre-calculated) rendering – jinými slovy offline rendering.

2.2 Offline vs Online (real-time) rendering

Jako takový, rendering dělíme do dvou hlavních skupin: Offline (pre-calculated) a online (real-time) rendering. Offline rendering se používá pro média, která nejsou interaktivní. Filmy, reklamy, televizní produkce a podobně. Offline rendering si tedy může dovolit dlouhou časovou prodlevu, během které je produkce odeslána do tzv. render farem, na kterých se následně renderuje třeba celé týdny. Render farmy jsou velké střediska počítačů určená výhradně renderingu nebo například propočítávání fyzikálních simulací.

Naproti tomu, online, tedy real-time rendering je forma vykreslování, která musí být co možná nejrychlejší a nejvíce optimalizovaná. Donedávna se užívala skoro exkluzivně ve sféře počítačových her.

Oba druhy renderingu také vyžadují vlastní přístup umělců i techniků. Pokud je offline rendering pomalý, ale zato detailní, můžeme si při práci s ním dovolit užívat velice detailní modely, ve kterých jsou všechny detaily přímo vymodelované v do sítě polygonů model tvořících. Assety tak není nutné optimalizovat.

Také se nemusíme starat o jejich strategické rozložení v kompozici, na rozdíl od rozložení assetů v počítačových hrách, kde potřebujeme, aby byly všechny objekty optimálně vykresleny s ohledem na výkon počítače. To samozřejmě neznamená, že v offline renderingu si můžeme dovolit plýtvat výkonem, nicméně přístup k němu je mnohem volnější a dostupnější umělcům, kteří chtějí pracovat s maximálním detailem.

Na druhé straně, real-time rendering potřebuje velice specifický workflow, přípravu assetů a naprosto strategické využití nám dostupných zdrojů (tzn. výkonu počítače). Výrobu assetů do počítačových her provází nejen vlastní vymodelování, texturování a nasvěcování, ale také zdlouhavá a často komplikovaná optimalizace, vytváření různých

permutací modelů, které užíváme pro různý Level of Detail (LOD) – různé úrovně detailu ve vzathu k vzdálenosti od kamery a podobně.

„První výhodou je čas strávený renderingem. Pokud si vezmeme za příklad statický obrázek, ten se může renderovat až 8 hodin na jednom počítači. Je jasné že použití Unity, real-timemového enginu, je neuvěřitelně rychlé.“ (oneirosvr.com, 2020)[9]

Efektivita času je obrovským faktorem například při práci s klienty. Představme si, že vlastníme designové studio zaměřené na architekturu. Pokud bychom užívali klasický offline rendering, je pravda, že vizuální výstupy by byly až neuvěřitelně přesné. Nicméně každý jeden výstup by stál obrovské peníze, a především čas. Článek blogu společnosti Oneiros, která se zaměřuje na architektonickou vizualizaci, dává za příklad 8 hodin offline renderingu pro jednu statickou fotografii. Při použití rendereru programu Unity tento čas padá na 0,1 sekund. Co víc, real-time render enginy jsou dnes tak propracované, že obyčejný člověk rozdíly mezi offline vyrenderovanou kuchyní a kuchyní renderovanou v online rendereru jednodušše nepozná.

Další výhodou je bezpochyb ona zmiňovaná interaktivita. O počítačových hrách se ani nebavme. Zůstaňme u příkladu architektonické vizualizace.

Díky vykreslování scény, jejího osvětlení a i možnostech například propočítávání dynamických světel (=světel, které se pohybují, či mění své vlastnosti, jako je například barva) v reálném čase, jinými slovy, prakticky okamžitě, můžeme našemu klientovi vyjít vstříc přímo na místě. Onen klient má možnost vidět změny, které si od nás vyžádá, naprosto okamžitě.

S neustálým vývojem výpočetní techniky bylo možné v real-time vykreslovat složitější a složitější scény, více polygonů, jemnější světla a stíny a ray-tracing se tak čím-dál-více přibližoval vizuální kvalitě path-tracingu. Jedním z největších kontributorů vývoje real-time renderingu jsou Epic Games, Kalifornská společnost zodpovědná za kultovní hry *Unreal* a *Unreal Tournament*.

Pro vývoj svého prvního titulu, Unreal, Epic Games přišli s vlastním herním enginem, který hru pohání a renderuje. Unreal Engine tak poprvé vidí světlo světa v roce 1998. A již tehdy šlo o revoluční technologii.

[9] Oneirosvr.com*, citát, vlastní překlad*

3. UNREAL ENGINE

Unreal Engine je program využívaný především pro výrobu a „pohon“ počítačových her. Obdobné programy nazýváme „herními enginy“. Jako takový je tedy především zaměřen na real-time rendering, avšak podporuje prakticky všechny druhy renderingu a to i offline-rendering v podobě path-tracingu.

Herní enginy neslouží počítačovým hrám (a jiným aplikacím) pouze jako renderingové programy. Dále obstarávají podporu programování, či skriptování.
V podstatě jsou jakousi kostrou jakékoliv interaktivní aplikace. Dnes jej využívají krom jiných architekti, filmoví producenti, či muzea, která chtějí svým návštěvníkům dopřát interaktivní zážitek.

3.1 Historie, vývoj a význam Unreal Engine

Je rok 1989. Někde ve státu Maryland si mladý mechanický inženýr hraje se svým novým IBM Personal Computer/AT, který dostal jako dárek od svého otce. Ačkoliv studuje mechanický inženýring, je fascinován počítači a ví, že s nimi chce v budoucnu pracovat.[10]

Tento mechanický inženýr se jmenuje Timothy Dean Sweeney. Tim chtěl svůj koníček proměnit v byznys a tak z pokojíku v domě svých rodičů zakládá svou první firmu – počítačové konzultační služby. Tim za poplatek radil lidem s jakýmikoliv počítačovými problémy, na které narazili. Tuto společnost nazývá Potomac Computer Systems, podle města, ve kterém žil. Bohužel, Potomac Computer Systems vydělával málo a tak Tim s firmou, přibližně po roce aktivity končí. [11]

I přes první neúspěch se Tim nadále zajímá o počítače a přichází s myšlenkou vytvoření své vlastní počítačové hry. V raných devadesátých letech minulého století nejsou

[10],[11] Edwards, Benj. *From the Past to The Future: Tim Sweeney Talks,* Game Developer, 2009. [online]. parafráze

dostupné manuály na vytvoření své vlastní počítačové hry, natož aby existovaly herní enginy se kterými by člověk mohl pracovat. Ale Tim si poradit uměl. [12]

Programuje textový editor v jazyce Turbo Pascal a samotnou hru vytváří z onoho textového editoru. Dvacetiletý Tim Sweeney vydává svou první hru, *ZZT*.

„*ZZT* jsem napsal v jazyce Turbo Pascal. Použil jsem styl programování, orientovaný na objekty. *ZZT* byl designován tak, aby poskytoval jednoduchý způsob kontroly herních objektů, bez komplexity ‚skutečného' programovacího jazyka."[13] (Thomsen, a další, 2010)

Hra *ZZT* byla úspěchem především, protože jde o historicky první případ programování hry na základě objektů – to znamená, že jiní vývojáři mohli hru jednoduše modifikovat a tak vytvářet vlastní hry pouze záměnou daných objektů. Dá se tedy mluvit o jedné z prvních emergencí tzv. moddingu – modifikování počítačové hry pro vytvoření hry nové, či upravené.

Selhaný Potomac Computer Systems Tim přejmenuje na Epic Megagames a rozhodne se, že se stane vývojářem počítačových her. Právě Epic Megagames je základem dnešní světoznámé společnosti Epic Games.

Kolem roku 1992 - 1995 se na herní scéně poprvé ukazují známky trojrozměrného vykreslování. Dvojapůldimenzionální (2.5D) *Wolfenstein 3D* či *DOOM* od průkopníků iD Software Tima Sweeneyho inspirují a on sám začíná pracovat na vlastním 3D herním enginu pro svou novou hru, *Unreal*. Určitě také pomohla známost s Markem Reinem, který byl zrovna z iD Software vyhozen…

Onen nový herní engine pojmenují Unreal Engine. Po vydání je hra *Unreal* jednoznačně chválena a setkává se s velkým úspěchem nejen u hráčů, ale i u konkurence. Poháněn novým enginem, *Unreal* měl plně 3D vykreslovaný svět, podporu dynamických světel, či skeletální animace – inovativní a průlomové technologie. Tim Unreal Engine již od začátku koncipoval jako nástroj pro vývoj počítačových her – jako herní engine – který by se neužíval pouze pro jejich vlastní hru, ale mohl by být licensován jiným firmám a

[12] Edwards, Benj. *From the Past to The Future: Tim Sweeney Talks*, Game Developer, 2009. [online]. parafráze

[13] Thomsen, Mike a Sweeney Tim. *History of the Unreal Engine; an interview with Tim Sweney*, *IGN*. 2010 [online] [Citace: 18. Duben 2022.]

vývojářům, kteří by ho dále za poplatek mohli používat. A tak začíná příběh o úspěchu jednoho muže, který navždy změnil audiovizuální průmysl.[14]

„Přes tři a půl roku jsem přepsal jádro svého renderingového algoritmu nejméně sedmkrát. Nakonec jsem skončil s dvojúrovňovým systémem, který umožňoval měkké stíny, dynamická světla, volumetrickou mlhu a efektivní způsob filtrování textur.“[15] (Thomsen, a další, 2010)

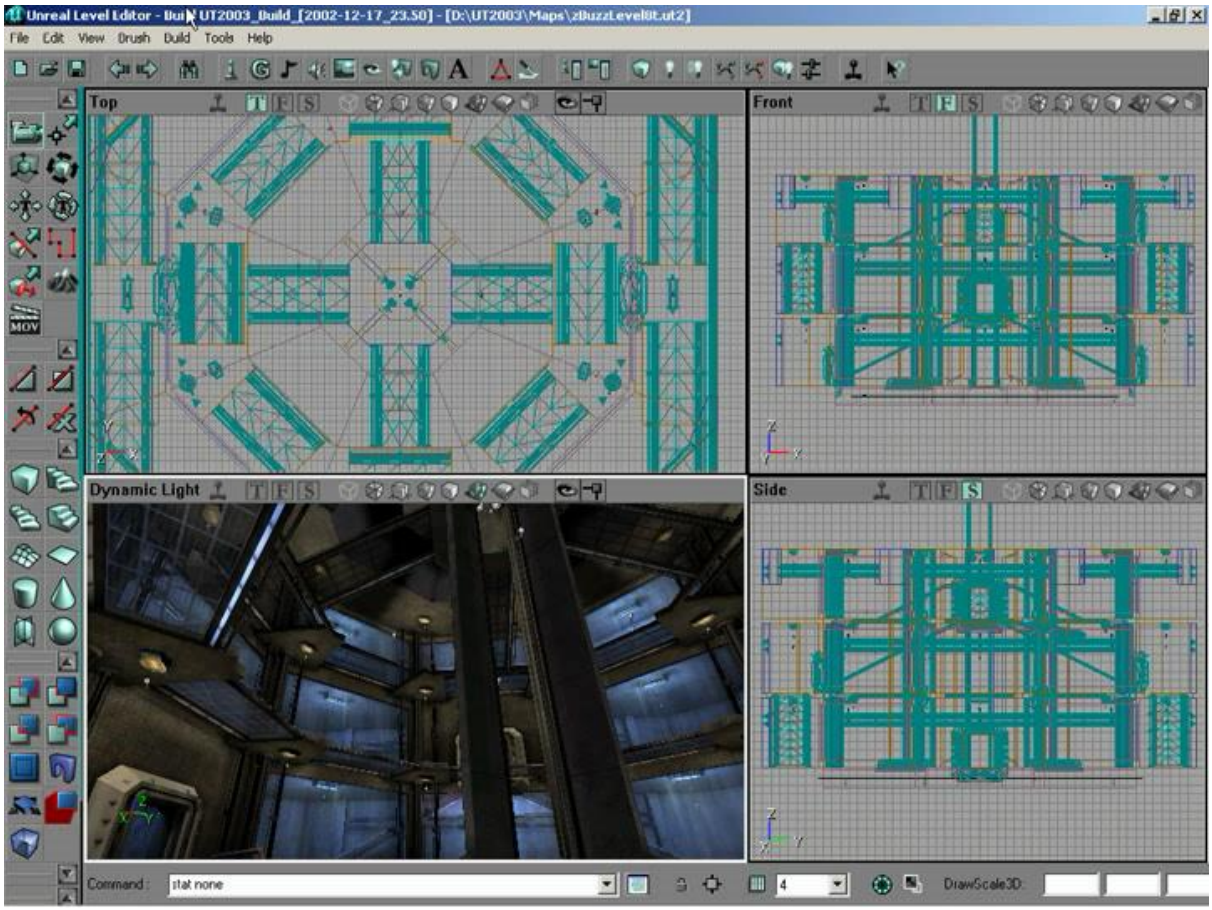
Roky šly a Unreal Engine procházel různými verzemi a revizemi. Unreal Engine 2.0 byl vydán v roce 2001. Byl užit v kultovních klasikách jako jsou *Unreal Tournament 2002* a *Unreal Tournament 2004*. Znovu se setkal s obrovským úspěchem. *Jak Unreal Tournament 2002,* tak i *Unreal Tournament 2004* jsou obě hry multiplayerové – tzn. hry více hráčů. Jde o arénové střílečky (FPS – first person shooter), ve kterých soutěží hráči z celého světa.

Mimo jiné, *Unreal Tournament* dal hráčům schopnost vytvářet vlastní arény – vlastní komunitní levely, ve kterých pak mezi sebou mohli soutěžit. Tento level editor se jmenuje Unreal Level Editor. Unreal Level Editor dovolil vytvoření projektů, které si vyráběli sami hráči. Byly vytvořeny celé komunity a internetová fóra, kde se z hráčů stávali level designéři, a své výtvory mezi sebou volně sdíleli.

Epic Games tímto dali příležitost obyčejným lidem vytvářet levely, jako by byli herními vývojáři. Mohl se tak projevovat talent u hráčů, kteří by se jinak k vývoji počítačových her nedostali a Epic Games je následně třeba i zaměstnali. Z některých těchto uživatelů se tak stali skuteční vývojáři a designéři, kteří v průmyslu pracují dodnes.

[14] Edwards, Benj. *From the Past to The Future: Tim Sweeney Talks,* Game Developer, 2009. [online]. parafráze

[15] Thomsen, Mike a Sweeney. Tim. *History of the Unreal Engine; an interview with Tim Sweney, IGN*. 2010 [online] [Citace: 18. Duben 2022.]



Obr. 12 – Náhled programu Unreal Level Editor

Novinky a vylepšení Unreal Enginu 2.0 zahrnovaly například zlepšení jeho nástrojů, podpora jednoduchých fyzikálních simulací, či zvýšení možných vyrenderovaných polygonů na obrazovce, kterých se nyní mohlo vykreslovat stokrát více, než v enginu minulé generace.[16] (Epic Games, Inc., 2005)

V roce 2007 byl Unreal Engine 2.0 nahrazen další verzí, Unreal Engine 3. Unreal Engine 3 vychází společeně se hrou *Unreal Tournament 3*, která svým uživatelům znovu dovoluje otevřeně a volně editovat, měnit, modifikovat, či vytvářet vlastní herní prostředí a tím se i učit procesy vývoje počítačových her, či specifika real-time renderingu.

Obrovským průlomem byl systém dynamických světel, kdy objekty vykresleny na obrazovce poprvé vypadaly velice věrohodně jejich skutečným předlohám. Tento systém byl zlepšen o nové tzv. light functions, světelné funkce, které se užívají dodnes. Engine zlepšovaly další vylepšení ve formě zvýšení možných vykreslených polygonů, propracované systémy optimalizace, či například podpora korektního provedení renderingu

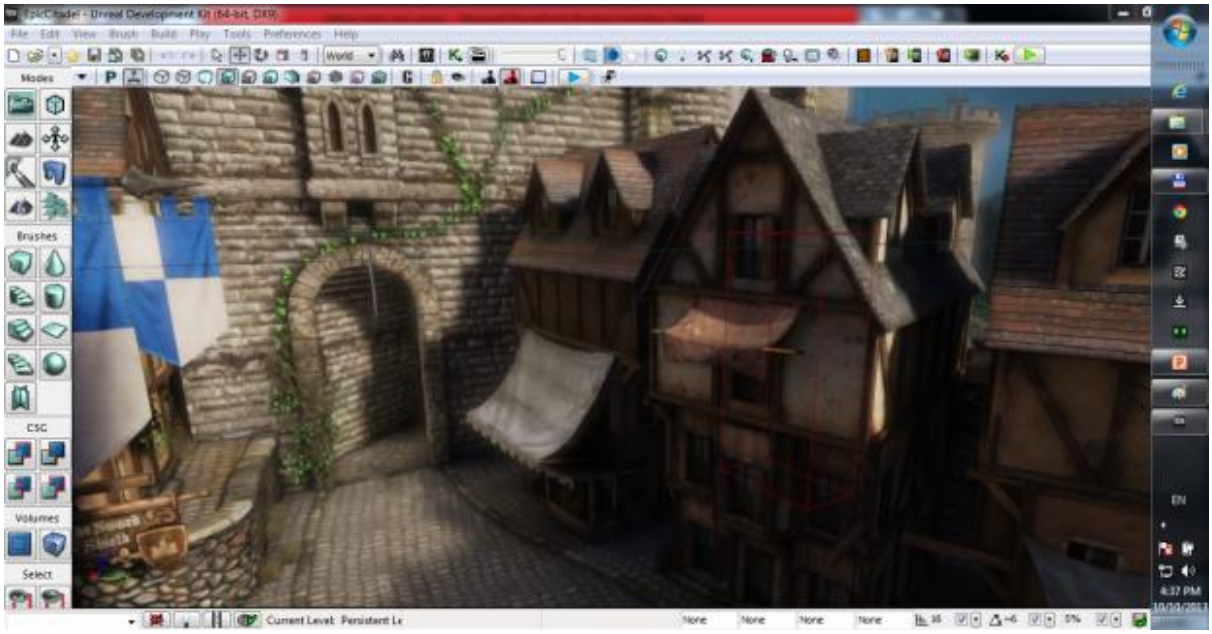
[16] Epic Games, Inc. UDN - Two - SiteMap. *Site Map for Unreal Engine 2 Documentation.* Epic Games, Inc., 2005 [Online]. [Citace: 18. Duben 2022.]

HDR – tzv. high dynamic range. Zjednodušeně jde o rendering barev, které jsou fyzikálně korektní – tím v některých případech mají větší hloubku, či saturaci.[17] (Epic Games, Inc., 2009)

Krom nových technologií, zlepšení kapabilit rendereru a dalších, technických novinek, Unreal Engine 3 byl revoluční především z jednoho hlavního důvodu. Devatenáctého listopadu roku 2007 Epic Games vypouštějí naprosto volně dostupnou verzi Unreal Engine 3 známou jako Unreal Development Kit.

Unreal Development Kit (zkráceně UDK) je dalším případem jakési demokratizace výroby obsahu ať už pro nezávislé vývojáře, či nadšence a koníčkáře. V UDK jsem osobně začínal i já. UDK přicházelo se všemi vlastnostmi Unreal Engine 3, několika „example mapami“ (tzn. scénami s příklady toho, co engine umí, na kterých se nováčci mohou učit) a především plnou, volně dostupnou dokumentací.

Další obrovskou výhodou byla evoluce programovacího jazyka UnrealScript, nyní pod jménem Kismet. Kismet byl revoluční, protože dovoloval umělcům programovat (či skriptovat) vizuální formou, pomocí tzv. nodes. Prototypování, či jednoduché programování tedy již nebylo výsadou pouze čistě programátorů. Kismet byl nadále zlepšen a přepracován v Unreal Engine 4 do podoby, kterou známe dnes – tzv. blueprints.



Obr. 13 – Náhled programu Unreal Development Kit

[17] Epic Games, Inc. UDN | WebHome. *Unreal Engine 3 Documentation*, 2009 [online]. [Citace: 18. Duben 2022]

Je rok 2012 a Epic Games konečně vypouštějí vysoce očekávaný Unreal Engine 4. Nejprve pouze pro komerční využití, avšak po dvou letech, 4. Září 2014, Epic vydává naprosto volně dostupný Unreal Engine 4. Ten znovu přichází se značnými zdokonaleními oproti předchozím verzím. Krom vylepšení, které od nové verze herního enginu již očekáváme, jako je vyšší číslo renderovaných polygonů, vyšší rozlišení vykreslovaných stínů, či podpora mnohem větších prostředí, byl do enginu zakomponovaný tzv. PBR materiálový workflow.

Obr. 14 – Příklad SDR vs PBR

PBR (physically-based rendering; renderování na základě fyzikálních vlastností) byl pro Unreal Engine novinkou. Předchozí verze Unrealu (a i ostatní herní enginy) do té doby užívali workflow názvem SBR – specular-based rendering. SBR renderuje materiály a jejich reflektivitu, či interakci se světlem, dle tzv. spekulární mapy – textury, která nese černobílá data, určující vlastnosti odrážení světla po povrchu. Je nutno podotknout, že tato data nemusí být určena pouze texturou, ale i jednoduchou číslenou hodnotou.

PBR přidává materiálům nové vlastnosti – tzv. metallic a roughness. Místo využívání spekulárních dat využívá data nová, která určují metaličnost a hrubost daného povrchu. Tento přístup, jak již jméno napovídá, je fyzikálně korektním způsobem vykreslování materiálů a tak dovoluje umělcům se mnohem lépe přiblížit podobě objektů skutečných.[18] (McDermott, 2018)

Z logiky ray-tracingu můžeme usoudit, že nový způsob vykreslování materiálů také znamená nový způsob renderingu světel a nasvěcování scén. A je tomu tak. Unreal Engine

[18] McDermott Wes. *The PBR Guide:A Handbook for Physically Based Rendering,* 2013. Parafráze.

nyní podporoval plně dynamické nasvěcování. Již nebylo nutností světlo předpočítávat (před-renderovávat) a poté užívat data takto získána na nasvěcování scén. S dalšími aktualizacemi přišla další zlepšení i tohoto systému, například různé optimalizace.

Unreal Engine 4 také zlepšil svůj skriptovací systém Kismet. Nová verze Kismetu, známá pod názvem Blueprint je natolik robustní, že může plně nahrazovat klasický C++ kód, s minimální ztrátou výkonu. Umožnil tak umělcům stát se plnohodnotnými vývojáři a zároveň sdílí logiku s jazykem C++, který se formou Blueprintu můžeme naučit.

Mimoto, vývojáři u Epic Games začali pořádat pravidelné livestreamy, ve kterých komunitu učili a vysvětlovali jim různé funkce jejich nového enginu. Koníčkář i nováček se tedy mohl učit přímo s vývojáři enginu a mít tak informace z první ruky. Zároveň měl možnost se přímo vývojářů ptát na konkrétní otázky.

Krom kompletní, volně dostupné dokumentace, s nástupem Unreal Engine 4 Epic Games přicházejí s online výukovými lekcemi pod názvem Unreal Engine Academy. Unreal Engine Academy je volně dostupné online výukové středisko, které je plné návodů, instruktážních videí, či celých kurzů, dedikovaných ať už programování, tak renderingu.

To jsou pouze některé příklady pokračující demokratizace produkce ať už videoher, tak animovaných filmů. Epic Games tuto tradici pokračují dodnes, například akvizicí firmy Quixel, či každoměsíčním vydáváním vybraných assetů z online tržiště Unreal Engine Marketplace zdarma. Právě akvizice Quixel firmou Epic Games se stala důležitým mylníkem pro další vývoj programu Unreal Engine.

Quixel byl založen v roce 2011 Teddy Bergsmanem a Waqar Azimem. Jejich vizí bylo zrychlení produkce virtuálních prostředí pomocí obrovské knihovny 3D naskenovaných objektů, které by umělci následně mohli využívat ve svých scénách. Tak vznikla, dnes již notoricky známá knihovna, Quixel Megascans.

Modely dostupné v této databázi jsou trojrozměrné objekty naskenované technologií zvanou „photogrammetry“ – fotogrammetrie. Jak již název napovídá, jde o kombinaci fotografie a geometrie. Proces fotogrammetrie je následující. Fotograf vyhlídne skutečný objekt a začne jej fotografovat ze všech možných stran a úhlů. Tyto fotografie se pak využívají jako vstupní data programů, které umí z těchto fotografií vyrobit trojrozměrnou geometrii. Ta se často musí dále ručně čistit a upravovat, aby byla kvalitní a

využitelná. Výsledkem je extrémně realistický a detailní model, který odpovídá skutečné předloze 1:1.

Obr. 15 – Příklad využití fotogrammetrie. Nalevo – Socha Sedící ženy Václava Bejčka, 1967. Uprostřed - naskenovaný 3D model. Napravo – Rekonstrukce sochy využitím 3D tisku.[19]

Detaily takto naskenovaných modelů jsou zachyceny samotnými polygony modelu – to znamená, že mohou obsahovat miliony těchto mnohouhelníků a tím pádem jsou velice náročené na rendering. Bez úprav či přepracování modelu do verze s nižším počtem polygonů se ještě do nedávna nehodili pro využití v real-time renderingu . Co se tedy změnilo?

3.2 Real-time rendering ve videohrách

Abychom si na tuto otázku mohli odpovědět, musíme nejdříve pochopit některé základy real-time renderingu. A jak lépe uchopit real-time rendering, než skrze pole, které jej využívá nejvíce – videohry.

Jak jsme si již vysvětlovali, online rendering musí být, na rozdíl od offline renderingu, velice optimalizovaný. Pro rekapitulaci; pokud vytváříme počítačovou hru, vytváříme interaktivní prostředí. Toto prostředí musí na interakci hráče reagovat v realném čase, umět se měnit a proměňovat a to co možná nejrychleji. Standardní frekvencí těchto

[19] Zuza, Mikolas. *3D skenování s použitím fotoaparátu či mobilu – Prusa Printers.* 2018 [online]. [Citace: 18. Duben 2022]

změn pro videohry je 60 snímků za sekundu, což odpovídá frekvenci většiny konzumních monitorů, které svůj obraz aktualizují 60 krát za vteřinu – mají frekvenci 60Hz.

Pokud tuto frekvenci nedosáhneme, hráč má pocit „oddělení“ od světa, do kterého by měl být ponořený. Pokud frekvenci dosáhneme, ale v některých momentech ji prudce zpomalíme, hra se „trhá“ a hráč je znovu od hry oddělen. Proto potřebujeme pro videohry modely a obecně assety vytvářet specifickým a optimálním způsobem.

Zaprvé, méně je více. Při real-time renderingu vždy pracujeme s určitým rozpočtem. Tento rozpočet rozdělujeme mezi kalkulace pro render modelů (v případě her renderujeme trojuhelníkové sítě), jejich materiálů, ale také herních systémů – to znamená skriptované sekvence, samotný kód hry a podobně.

Real-time rendering vždy provádíme na grafické kartě (GPU), nikoliv na procesoru (CPU) počítače. Offline rendering je tradičně činností především centrálního procesoru – CPU. Ten je o hodně silnější, než grafická karta, avšak pro rendering v realném čase se nehodí, protože offline rendering, jak jsme si již řekli, vyžaduje obrovskou časovou prodlevu.

Bez hlubšího zacházení do počítačové vědy, při renderingu videohry spolu procesor a grafická karta komunikují. Procesor grafické kartě rozkazuje který objekt a jak má vyrenderovat, zároveň také počítá všechny různé další instrukce, jako samotnou logiku hry, interakce hráče s ní a podobně. Grafická karta má na starosti vykreslování samotného obrazu.

Komunikaci mezi procesorem a grafickou kartou nazýváme tzv. draw calls. A právě zde přicházíme k prvnímu kameni úrazu real-time renderingu. Jakákoliv instrukce od centrálního procesoru, řečeno extrémně zjednodušeně, nás „stojí“ tento draw call. Příkladem takového draw callu je materiál. Každý materiál na každém objektu je jeden draw call. V praxi toto znamená, že pokud chceme renderovat krychli, která má na každé své straně jeden materiál, stojí nás to šest (plus jedna, tedy sedm) draw calls. Proč?

Pracujeme totiž na bázi tzv. „per object“ – „od objektu k objektu“. Místo jednotlivých trojuhelníků vnímáme celé objekty, ze kterých až poté získáváme data o materiálech, které potřebujeme vyrenderovat. Jinými slovy, procesor vidí skupiny

polygonů se stejnými vlatnostmi – stejnými materiály. Je důležité zmínit, že tyto vlastnosti se nesdílí mezi více objekty! Poté informuje grafickou kartu, které skupiny polygonů má jak vyrenderovat.

Vraťme se nyní k geometrii. Jak jsme si již řekli, na rozdíl od offline renderingu, v online renderingu vždy vykreslujeme triangulované modely – trojuhelníkové sítě. Quady a jiné mnohouhelníky se při importu do herního enginu automaticky triangulují – mění se v trojuhelníky.

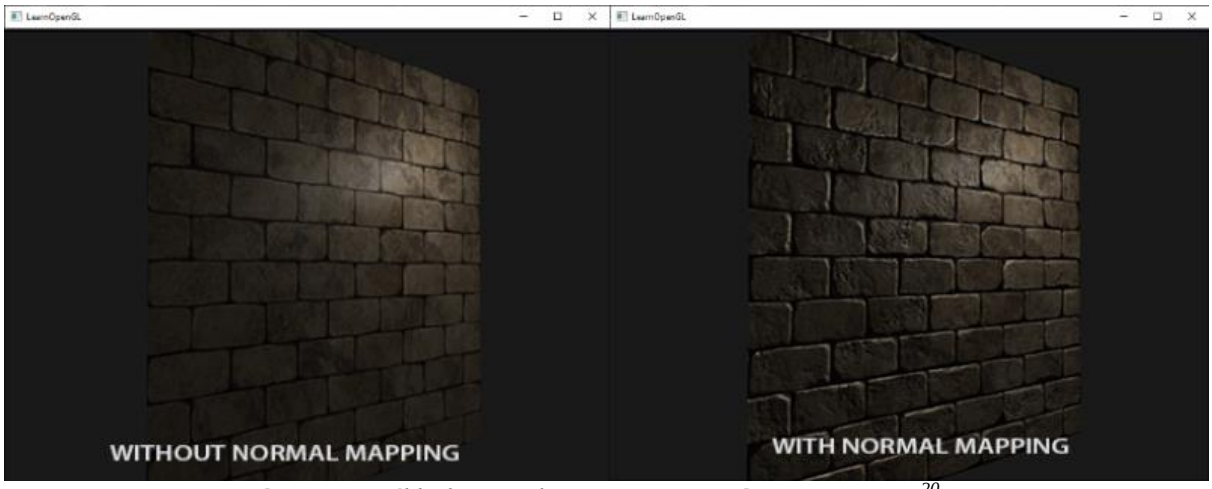
Pokud tedy zobrazujeme šestistrannou kostku, ve skutečnosti renderujeme 12 (+1 pro samotnou geometrii) = 13 trojuhelníků. I na počet zobrazených trojuhelníků se vztahuje náš zmiňovaný rozpočet. Ale jak tedy v real-time renderingu zobrazujeme detailní modely, či modely s velkým počtem polygonů?

Odpovědí je, že je nezobrazujeme. Pro real-time rendering totiž využíváme verze modelů, které mají menší počet trojuhelníků. Těmto modelům říkáme tzv. „low-poly modely“, zatímco modely s vysokým počtem polygonů nazýváme „high-poly modely“. Tradiční workflow výroby modelů pro videohry, či real-time rendering, je tedy následující.

Modelář vyrobí high-poly model, do kterého vymodeluje všechny detaily pomocí polygonů. Polygony tvoří geometrickou síť, která představuje samotný model. Podobu a rozložení této sítě nazýváme topologií. Po dokončení high-poly modelu se provede tzv. retopologie – Model se začne optimalizovat na low-poly verzi a při tom se musí měnit a optimalizovat topologie původního high-poly modelu. Při zhotovení low-poly modelu máme model, který můžeme využít v real-time renderingu.

Ptáte se, jak tedy zobrazíme detaily? Využijeme vlastnosti materiálů a vypůjčíme si původní high-poly model. High-poly a low-poly modely vložíme do stejné scény libovolného 3D modelářského programu, jako je např. Blender, Maya či 3DS Max. Následuje proces, který nazýváme baking.

Baking je proces, při kterém se data z high-poly modelu zapisují do textur – nejčastěji normal mapy, či displacement mapy. V případě normal mapy jde o datovou mapu, která obsahuje informace o normálech vektorů vůči vektorům, které určují geometrii modelu. Jinými slovy se bavíme o kolmicích k existujícím povrchům modelu. Metoda normal mappingu nám umožňuje předstírat nerovnosti a promáčkliny pomocí materiálu, který dle normal mapy tyto detaily vykresluje.



Obr. 16 – Příklad normal mappingu v rendereru OpenGL[20]

Tímto způsobem získáváme data ve formě textur – nejčastěji jde o normal mapy a height mapy (někdy zvané displacement mapy). V programech dedikovaných pro texturování objektů a vytváření materiálů takto získáváme: Albedo mapy, Specular mapy, Ambient Occlusion mapy, Displacement mapy, Normal mapy a nakonec také Roughness a Metallic mapy.

Albedo nám popisuje barvu povrchu – Tím není myšleno pouze, jestli bude povrch červený nebo zelený, ale přímo, jak bude vypadat. Pokud, například, vyrábím cihly, poté bude albedo mapa popisovat vzhled a barvu cihel.

Specular mapy se užívají při využití SBR (specular-based rendering) workflow. Určují nám vlastnosti intearkce světla s daným povrchem. To zahrnuje lesklost, matnost apod. Dnes jsou již nahrazeny Roughness a Metallic mapami.

Ambient Occlusion mapy, česky „ambientní okluze“, jsou textury, které nám určují předem „vypečené“ (baked) stíny, či zatemnění, které chceme na povrchu objektu mít za jakýhkoliv světelných podmínek.

Displacement (height) mapy jsou výškovými mapami daného povrchu. Určují nám výšku vertexů na povrchu objektu. Pokud displacement mapu užijeme, užíváme ji pro tzv. displacement za užití teselace.

Teselace (tessellation) je proces, při kterém uměle během běhu videohry „podrozdělujeme“ plochu (povrch) objektu, čímž uměle přidáváme více a více vertexů a tím i polygonů. Při užití displacementu se tyto nové vertexy umístí do prostoru dle dat,

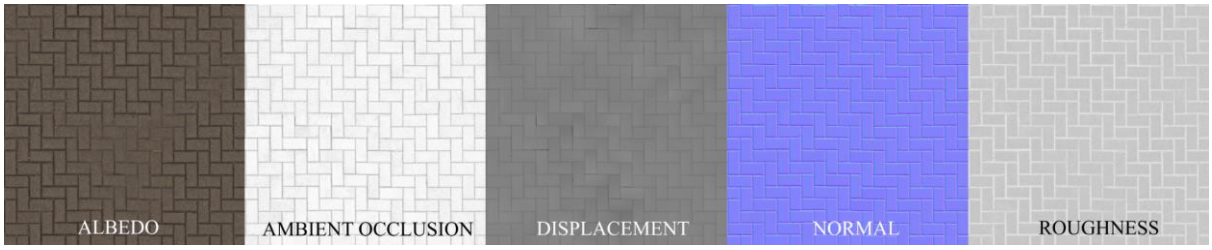
[20] De Vries, Joey. *Learn OpenGL – Normal Mapping* 2017 [online], [Citace: 18. Duben 2022]

které popisuje displacement mapa a tím získáváme skutečnou hloubku v 3D prostoru. Nutno podotknout, že tento proces je velice náročný na výkon počítače a pokud se ho užívá, tak jen ve specifických případech.

Normal mapy obsahují data o vektorech kolmých (normálech) vůči povrchům modelu. Užíváme jej pro předstírání nerovností a různých promáčklin.

Roughness mapy jsou textury specifické pro tzv. PBR (physically-based rendering) workflow. Popisují hrubost daného povrchu.

Metallic mapy jsou též textury specifické pro PBR workflow. Popisují metaličnost (kovovost) povrchu. Společně s Roughness mapami v PBR nahrazují Specular mapy.



Obr. 17 – Příklad map užitých pro PBR materiál non-metalického povrchu chodníku.

Nyní máme naše „baked“ textury a low-poly model po retopologii. Můžeme tedy začít renderovat model v reálném čase?

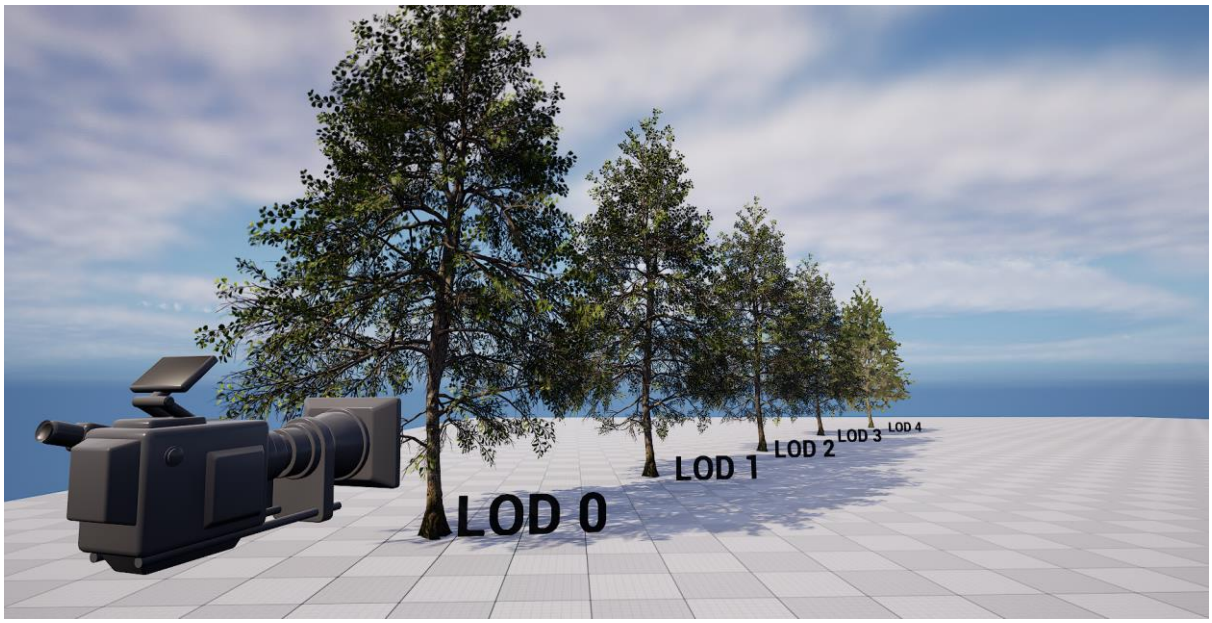
Ne tak rychle. Představme si například videohru, ve které má hráč absolutní volnost pohybu a může tak prozkoumávat velký, otevřený svět. Scény v takové hře jsou pochopitelně obrovské. Dejme tomu, že náš nově vymodelovaný a natexturovaný model je modelem stromu.

Takových stromů ve scéně najdeme třeba stovky. Možná i tisíce, pokud naše scéna zahrnuje, například, tropický prales. Pokud bychom v takovém tropickém lese měli vykreslovat tisíce stromů najednou, náš počítač by tuto aktivitu pravděpodobně nepřežil. Co s tím?

Podobně, jako tomu bylo u redukování počtu vykreslovaných trojuhelníků využitím low-poly modelu, i nadále budeme modely zjednodušovat. Tentokrát ale budeme simplifikovat již optimalizovaný low-poly model pro výrobu takzvaných LODs.

LOD – Level of Detail modely jsou ještě více zjednodušené modely, které užíváme ve scéně dynamicky v závislosti na tzv. screen size. Screen size je hodnota, která určuje

velikost vykreslovaného objektu na obrazovce. Čím více místa na obrazovce model zabírá, tím detailnější potřebujeme, aby byl. Každý model ve scéně má určitý počet LOD modelů. Obvykle se pohybujeme kolem pěti až šesti LOD modelů na každý model, ačkoliv toto číslo záleží na využití daného modelu.



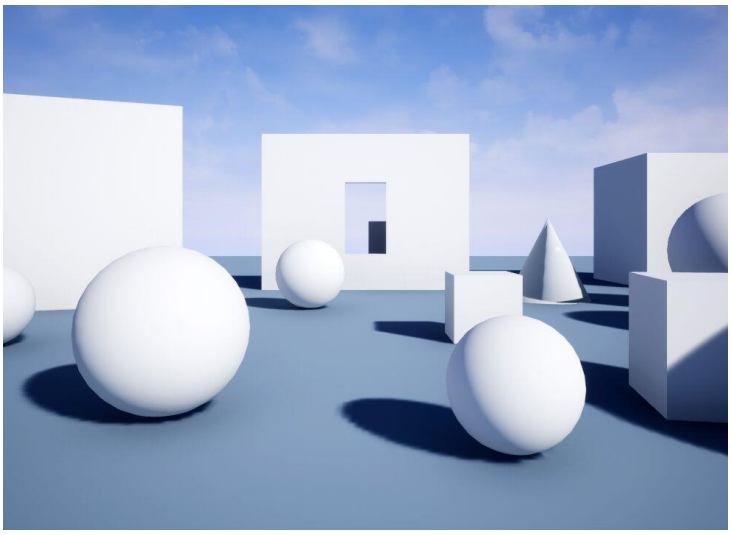
Obr. 18 – Příklad využití LOD modelů v Unreal Engine

V závislosti na hodnotě screen size tedy vybíráme LOD modely, které budeme na obrazovce vykreslovat. Pokud je tato hodnota vysoká, potřebujeme co nejdetailnější model – LOD 0. Model LOD 0 bývá sám původní low-poly model. Další LOD modely jsou poté méně detailnější modely odvozeny od tohoto modelu. Naštěstí se dnes již generují automaticky, není tedy nutné je ručně vyrábět. Poslední LOD model, v našem případě LOD 4, zpravidla bývá jednoduchou dvojdimenzionální rovinou, na které je vykreselená plochá reprezentace objektu, který optimalizujeme – jde tedy o obrázek. Tím pádem objekty v dálce nepředstavují prakticky žádnou geometrii a tak šetříme na našem render rozpočtu.

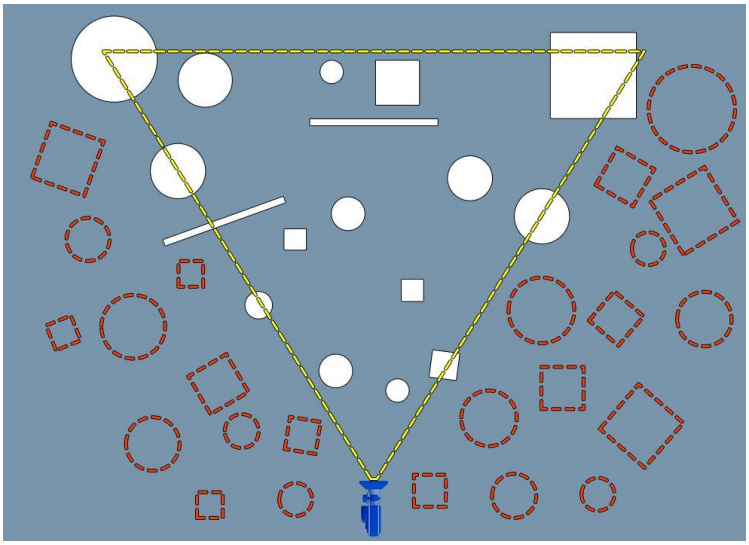
Další nutnou optimalizací, která je ovšem plně automatizovaná a 3D umělec ji tak nemusí řešit, je tzv. Visibility and Occlusion Culling. Pro nás je důležitá, protože jde o nezbytnou část real-time render pipelinu.

„Obecná idea metod visibility a occlusion culling je zmenšení počtu objektů, které jsou v jakémkoliv daném okamžiku viditelné na obrazovce, s cílem získání výkonu.“[21] (Epic Games, Inc., 2022)

„Pokud například začneme pouze tím, co kamera vidí ze své pozice, je viditelných pouze několik objektů. Víme však, že to tak úplně není, protože scénu tvoří spousta objektů, které z této pozice prostě vidět nejsou.“[22] (Epic Games, Inc., 2022) viz. Obr. 19.



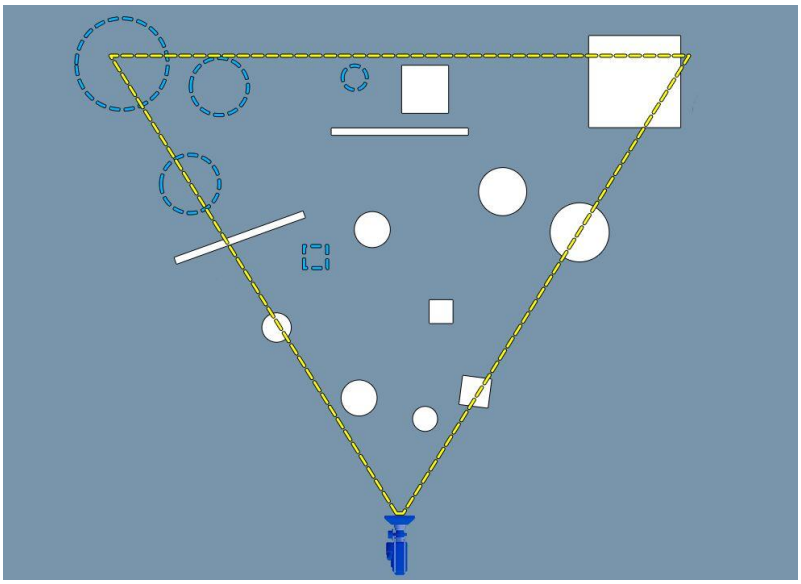
Obr. 19 - Kamerový pohled do scény[23]



Obr. 20 – Náhled scény shora[24]

[21], [22], [23], [24] Epic Games, Inc. *Visibility and Occlusion Culling | Unreal Engine Documentation.* 2022 [online] [Citace: 18. Duben 2022]

Červeně vyznačené objekty díky Visibility culling nerenderujeme. Nicméně, pořád si můžeme všimnout, že dle obrázku 20 stále vykreslujeme některé objekty, které z pohledu kamery nejsou vidět. Jde o objekty, které jsou „schované“ za jinými objekty. Pokud chceme renderovat efektivně, potřebujeme se zbavit i těchto modelů.

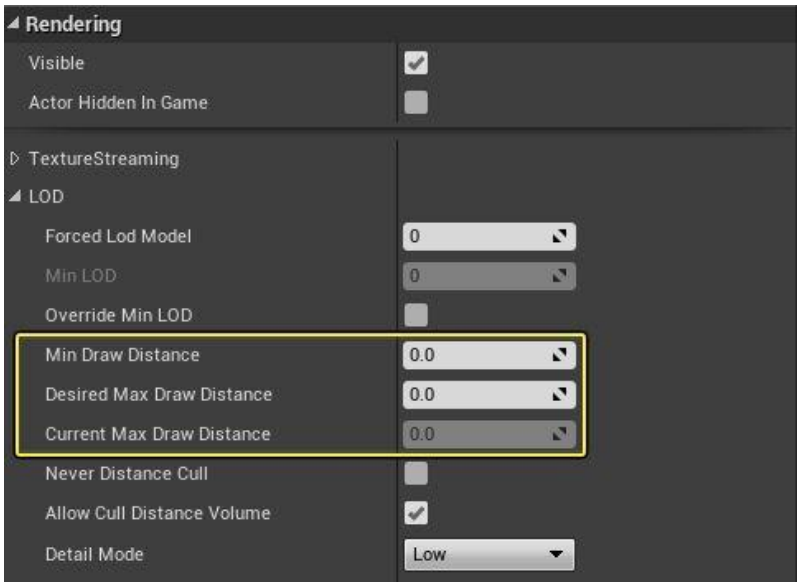


Obr. 21 – Znázornění objektů ve scéně, které jsou zakryté jinými objekty[25]

Díky occlusion culling zbytečně nerenderujeme objekty, které kamera neuvidí. Stojí za to zmíňit, že ale pořád renderujeme objekty, které vidíme byť jen z části. Jak jsme již zmiňovali, renderer pracuje na bázi objektů a tudíž pokud vidí byť jen část daného objektu, musí jej renderovat celý. Více o tomto později.

Optimalizujeme obsah také na základě vzdálenosti od kamery. Každý objekt ve scéně se řídí pravidly renderingu, které nastavuje umělec. Jedním z těchto pravidel je tzv. Draw Distance – vzdálenost vykreslování.

[25] Epic Games, Inc. *Visibility and Occlusion Culling | Unreal Engine Documentation.* 2022 [online] [Citace: 18. Duben 2022]



Obr. 22 – Tabulka nastavení renderingu objektu v Unreal Engine[26]

„Min Draw Distance: Minimální vzdálenost vykreslení, pří které by objekt měl být viditelný kamerou. Jde o nejbližší vzdálenost ke kameře, při které se ještě bude objekt renderovat. Jde o nejmenší možnou vzdálenost kamery od objektu, po jejím překročení se objekt renderovat přestane. (např. kamera prostoupí objektem).

Max Draw Distance: Maximální vzdálenost vykreslení, při které by objekt měl být viditelný kamerou. Jde o nejvyšší vzdálenost od kamery, při které se ještě bude objekt renderovat. To znamená, nejvyšší vzdálenost od kamery, před tím, než se renderovat přestane.“ [27] (Epic Games, Inc., 2022)

Tyto a další optimalizační systémy nám pomáhají renderovat dnes již fotorealistické scény v reálném čase. Vysoké vizuální kvality a především extrémně rychlého procesu iterací, tzn. změn a úprav scén, si časem všimla i Hollywoodská studia. Tím se dostáváme k využití technolgie real-time renderingu ve virtuální produkci.

[26],[27] Epic Games, Inc. *Visibility and Occlusion Culling | Unreal Engine Documentation.* 2022 [online] [Citace: 18. Duben 2022]

3.3 Real-time rendering ve virtuální produkci

Začněme definicí virtuální produkce dle příručky „*The Virtual Production Field Guide*“ od Noah Kadner.

„Virtuální produkce je široký pojem onzačující spektrum počítačem podpořených produkčních a vizualizačních filmařských metod. Pro začátek, zde je několik prominentních definic o virtuální produkci. Podle týmu Weta Digital, „Virtuální produkce je tam, kde se střetávají světy fyzické a digitální.“. Moving Picture Company (MPC) k této definici přidává technický detail, „Virtuální produkce kombinuje virtuální a augmentovanou realitu s CGI (computer generated imagery) a technologiemi herních enginů, čímž umožňuje produkčnímu štábu sledovat své scény tak, jak se komponují a zachycují přímo na place.“[28] (Kadner, a další, 2019)

Tradičně, produkce filmového CGI (computer generated imagery – počítačová grafika) patřila skoro exkluzivně offline renderingu. Při natáčení filmu, který vyžadoval například fiktivní prostředí, ve kterém se pohybují skuteční herci, byl použit tzv. green screen, či blue screen. (zelené / modré plátno).

Obr. 23 – Ewan McGregor před blue screen při natáčení Star Wars, Episode II.

[28] Kadner, Noah a Epic Games, Inc. *The Virtual Production Field Guilde, Volume 1*. 2019 [Dokument] [Citace: 18. Duben 2022]

Jde o zelené, či modré plátno, kterým je potažený prakticky celý plac. Při tvorbě setu se z různých ramp, plošin a krabic vytvořil přibližný tvar, prozatím neexistující scény. Vytvořením této aproximace scény, ve které se má film odehrávat, umožňujeme hercům pohybovat se v tomto fiktivním prostředí. Kvůli absenci skutečného prostředí musí být herci dobře seznámeni s vizí režiséra a zároveň potřebují velkou špetku představivosti.

Zelené či modré plátno slouží pro následovné „klíčování“ – color keying. Color keying je proces, ve kterém při postprodukci odstraníme ono zelené plátno a nahradíme ho fiktivním prostředím, které nám vyrenderovalo CGI studio. Distinktní modré a zelené barvy tedy slouží pro identifikaci pozadí scény počítačem, který následovně tuto barvu nahradí požadovaným prostředím.

Obr. 24 – Fotografie z natáčení Star Wars: Episode III. na setu s užitím green screen.

Green screen s sebou také přináší spoustu komplikací – prvním z nich je nasvěcování scén. Ty musely být nasvěcovány s obrovskou opatrností a precizností. Barva zeleného plátna se totiž může rozplývat po rekvizitách, či v nich odrážet. Využívání kovových, či lesklých rekvizit tak buď vůbec nepřišlo v úvahu, nebo jejich odlesky musely být zdlouhavě a namáhaně upravovány v postprodukci.

Proces offline renderingu fungoval následovně. Režisér si nechal od koncept umělců (concept artists) vytvořit návrhy prostředí, dle jeho zadání. Následovně vybral ty, které se mu líbily nejvíce a předal je týmu 3D umělců, kteří měli za úkol tuto vizi převést do reality.

Jenže workflow offline renderingu znamenal, že ačkoliv umělci užívali vysoce detailní modely, ze kterých mohli vytvářet ono prostředí, jakékoliv úpravy tvraly třeba celé týdny. I ta nejmenší chyba, či nejmenší úprava znamenala pře-renderování celých scén na render farmách, které jsou drahé a proces tak byl bolavý, drahý a zdlouhavý.

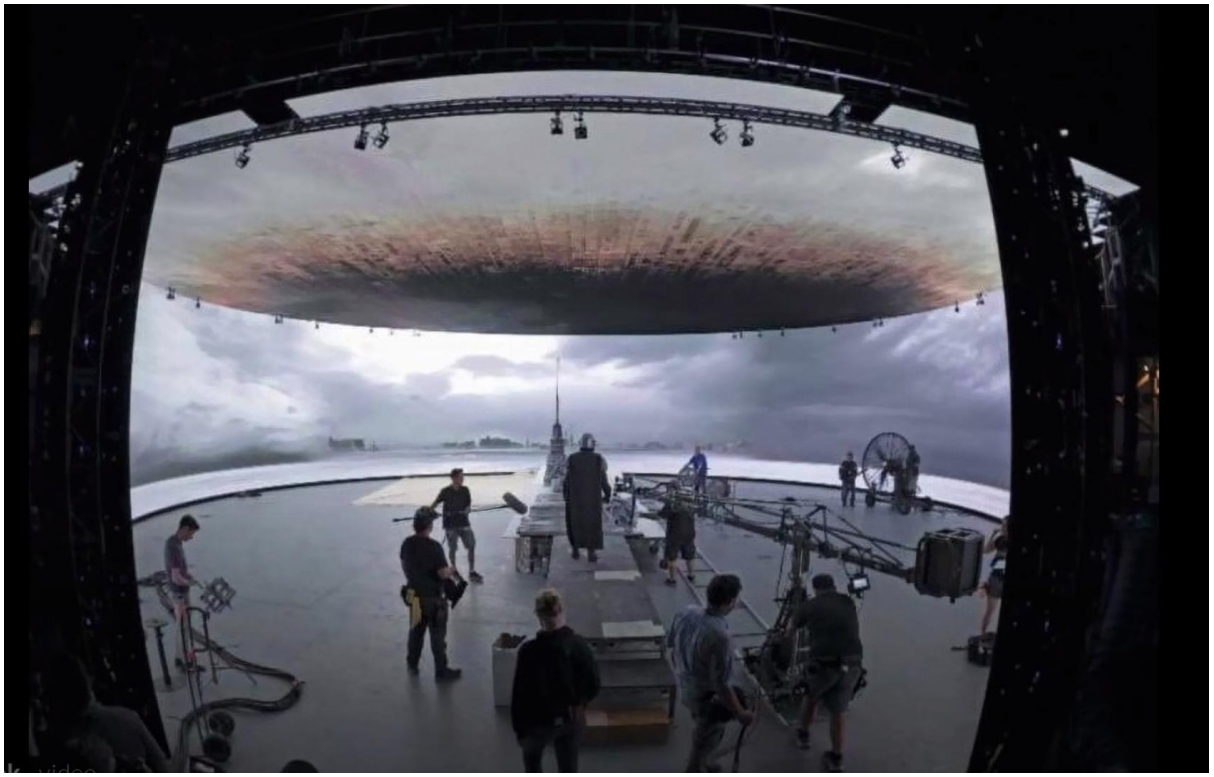
Obr. 25 – Fotografie z natáčení The Mandalorian za využití Unreal Engine a videostěny

Při natáčení seriálu The Mandalorian se LucasArts a jejich VFX studio Industrial Light & Magic spojili s firmou Epic Games. Následuje citát režiséra Jon Favreau.

„Částí toho, co bylo tak zábavné na spolupráci s LucasFilm a Disney na The Mandalorian, je, že jsme byli schopni uskutečnit několik technických inovací a i několik prvenství, které, myslím, budou mít velký dopad na způsob, jakým se televize a filmy budou pohybovat vpřed. Ve spolupráci s ILM a Epic jsme dali dohromady systém, s kterým můžeme mít real-time render herního enginu, spojený s technologií videostěny, aby vytvořili pozadí pro velký, krásný svět Star Wars.“[29] (Industrial Light & Magic, 2020)

Jon Favreau mluví o virtuální produkci za užití tzv. „virtuálních setů“. Více než polovina lokací, které jsme viděli v sérii The Mandalorian byla totiž renderovaná real-time v Unreal Enginu. Jak takový virtuální set vypadá?

[29] Industrial Light & Magic. *The Virtual Production of The Mandalorian Season One.* 2020 [YouTube video] [Citace: 19. Duben 2022]

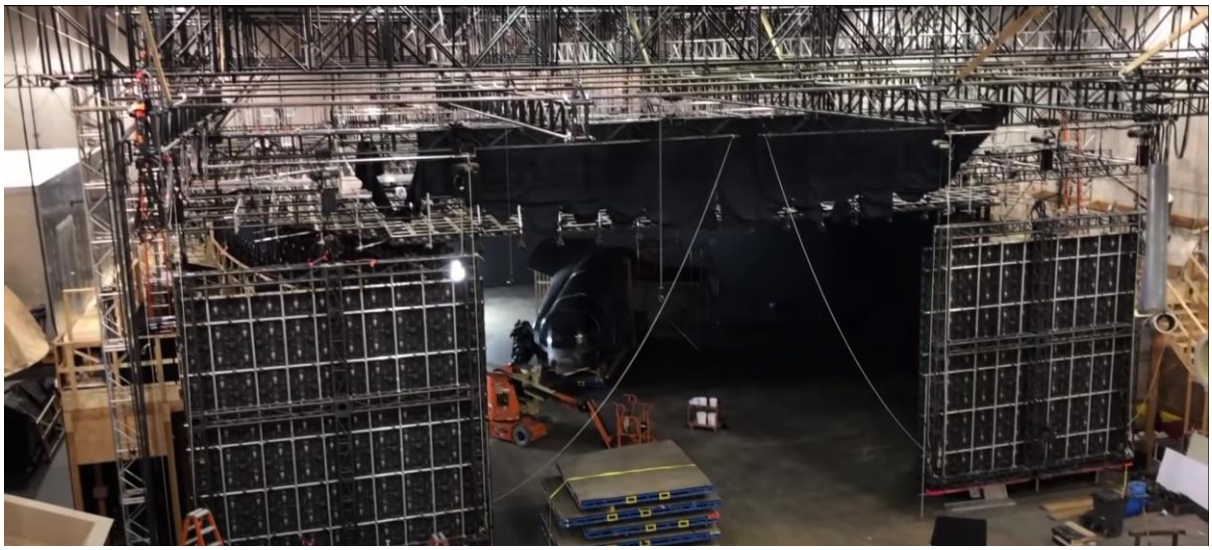


Obr. 26 – Fotografie z natáčení The Mandalorian za využití Unreal Engine a videostěny

„Set je 21 stop vysoký, má 75 stop v průměru, provozován je sedmi stroji (počítači), které pumpují vizuály na obrazovku. Vizuály se vytvářejí v předprodukci a mohou být na obrazovce do 24 hodin od jejich dokončení.“[30] (Industrial Light & Magic, 2020)

Tak komentuje virtuální set VFX Supervizor The Mandalorian, Richard Bluff.

[30] Industrial Light & Magic. *The Virtual Production of The Mandalorian Season One.* 2020 [YouTube video] [Citace: 19. Duben 2022]



Obr. 27 – Fotografie virtuálního setu z vnějšku.

Jde tedy o kulatý plac pro natáčení o průměru 75 stop, který je obklopen 21 stop vysokou videostěnou. Nad setem je další videostěna, která funguje jako obloha scény. Videostěny jsou, zjednodušeně, obrovské monitory, či chcete-li, obrazovky, na kterých se v reálném čase promítá scéna, ve které se herec nachází.

Obr. 28 – Fotografie z natáčení The Mandalorian za využití Unreal Engine a videostěny.

Kamera tím pádem natáčí již hotové pixely, zobrazeny na videostěně. Režisér tak může plně užívat všech schopností a vlastností herního enginu, které zahrnují například dynamický systém dne a noci. Díky tomu štáb již nemusí čekat na perfektní světelné podmínky při natáčení on-location a zároveň se nemusí bát, že jim ona „perfektní hodina“ natáčení unikne.

Herci si při svých výkonech již nemusejí představovat prostředí, ve kterém se nacházejí – vidí ho všude, kolem sebe. Dále, díky real-time renderingu jsou změny okamžité a chyby se dají rychle napravit. Efektivita času je tedy nesrovnatelná a iterace obsahu jsou neprodlené.

Tyto prostředí jsou také plně trojrozměrně propracované a nespoléhají se na kompoziting scén v postprodukci. Tudíž, pokud se režisérovi nelíbí např. úhel, pod kterým je prostředí prezentováno, prostředí se dá jednoduše v reálném čase rotovat a režisér tak získá úplně jiný pohled.

„Když tam poprvé vejdete, je to neuvěřitelně působivé, protože [videostěna] zcela obklopí vaše periferní vidění. A velmi rychle zapomenete, že jste někde vevnitř a že nejste venku na povrchu nějaké cizí planety. Máte pocit, že vás obklopuje skutečné trojdimenzionální prostředí, protože ono to skutečné trojdimenzionální prostředí je.“[31] (Industrial Light & Magic, 2020)

Virtuální sety také kombinují praktické a počítačové efekty. Podlaha ve virtuálním setu může být posypána hlínou či pískem, trávou a podobně. Po tomto „podiu“ se pohybují herci, což dává celé scéně hloubku. Nejde tedy již pouze o „popředí a pozadí“, ale natáčíme i „třetí dimenzi“, jinými slovy, osu Z. Jde tedy o skutečně revoluční přístup k virtuální produkci.

3.4 Quixel Megascans

Firmu Quixel a její databázi 3D skenovaných modelů jsme již zmiňovali. Fotogrammetrie je velice důležitou součástí virtuální produkce, protože skenované modely, například kamenů či skal, jsou extrémně důvěryhodné svým skutečným předlohám. Jsou tedy perfektním případem použití v situaci, kdy potřebujeme fotorealistické assety a Epic Games si toho byli vědomi.

Dvanáctého listopadu 2019 Epic Games kupuje firmu Quixel a tím i celou její knihovnu Megascans. Jde o další historický krok demokratizace techniky vytváření 3D

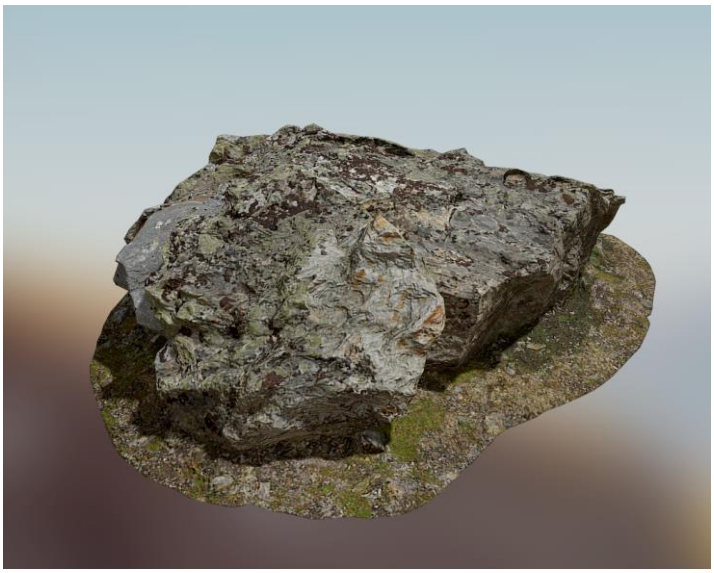
[31] Industrial Light & Magic. *The Virtual Production of The Mandalorian Season One.* 2020 [YouTube video] [Citace: 19. Duben 2022]

obsahu. Epic Games totiž ihned po dokončení akvizice celou knihovnu Megascans, která byla do té doby placená formou předplatného, uvolňuje zcela zdarma pro užití v Unreal Engine.

„To znamená, že pokud používáte Megascans s Unreal Engine 4, získáte bezplatný, neomezený a okamžitý přístup ke všem Megascans prostřednictvím programů Quixel Bridge a Mixer a k velkému množství balíčků Megascans na Unreal Engine Marketplace... ...pokud používáte Megascans pouze s UE4, vrátíme Vám všechny Vaše předplatné za rok 2019. Pokud máte aktivní předplatné, přihlaste se a zjistěte, jak získat své peníze zpět. Pokud jste v roce 2019 nakoupili Megascans, ale nemáte aktivní předplatné, kontaktujeme Vás ohledně vrácení peněz.”[32] (Quixel, 2019)

Proč jsou Megascans tak důležité? Vraťme se k přípravě modelů pro real-time rendering. Jak jsme již zmiňovali, 3D umělec potřebuje z high-poly modelu vyrobit low-poly model, který si real-time render engine může dovolit vykreslovat.

V případě fotogrammetrie je naskenovaný 3D model extrémně high-poly. Následuje tedy náročný a nepříjemný proces zjednodušování, čištění a optimalizace pro to, aby se mohl využít, například, v Unreal Engine. Co kdyby tomu tak ale nemuselo být?



Obr. 29 – Vysoce detailní model kamene zachycen technikou fotogrammetrie

[32] Quixel. *Quixel joins forces with Epic Games.* 2019 [online] [Citace: 19. Duben 2022]

4. TECHNICKÉ PRŮLOMY A INOVACE UNREAL ENGINE 5

Unreal Engine 5 nedočkavě vyhlíželi jak koníčkáři, tak profesionálové, ze všech možných sfér produkce 3D obsahu. Nová verze enginu byla poprvé demonstrována 13. Května 2020 a slibovala zdánlivě nemožné.

Tato první demonstrace běžela na herní konzoli PlayStation 5. Již tento fakt je z technického pohledu zajímavý. Takzvaná „tech-dema“, technické demonstrace nových technologií, jsou často demonstrována na počítačích vyšší třídy, aby byla zajištěna co možná nejlepší kvalita obrazu, frekvence snímků a tak dále. Typicky se herní konzole totiž nemohou rovnat výkonu klasických stolních počítačů.

Již první snímky pořízené v Unreal Engine 5 jsou, odpustíme-li si technický slovník, dechberoucí. Hráč je vržen do fotorealistické jeskyně vyrobené především ze skenovaných modelů již zmiňované knihovny Quixel Megascans.

Obr. 30 – Screenshot z real-time technické demonstrace Unreal Engine 5

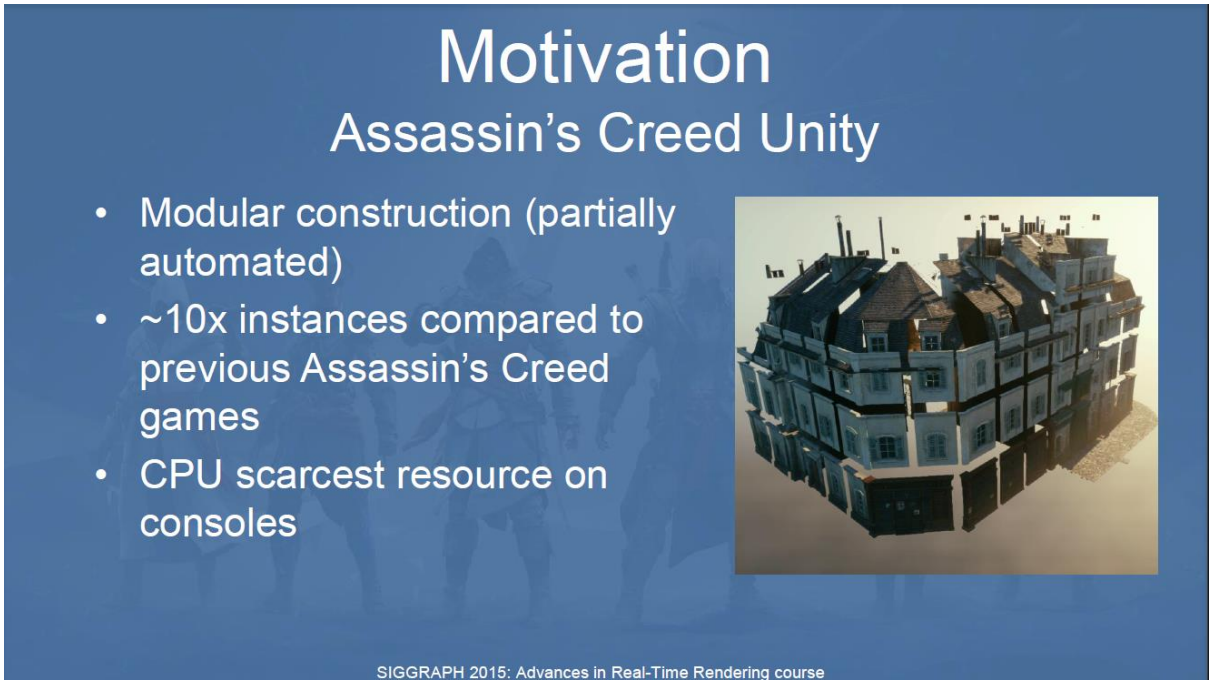
Demonstraci komentuje (tehdejší) Technical Director of Graphics firmy Epic Games, Brian Karis. Následuje citát.

„Mnoho z toho, co vidíte, bylo sestaveno z assetů knihovny Quixel Megascans. Ale místo herních [low-poly] verzí jsme použili filmové [high-poly] verze, které by se obvykle používaly pouze ve filmu. Každý z nich má kolem milionu trojúhelníků a díky virtuálnímu

texturování všechny používají také 8K textury. Technologie Nanite dokáže velmi rychle vykreslit šílené množství trojúhelníků. V každém snímku je více než miliarda trojúhelníků zdrojové geometrie, které Nanite plynule rozdrtí na přibližně 20 milionů vykreslených trojúhelníků.“[33] (Unreal Engine, 2020)

4.1 Technologie virtualizované geometrie - Nanite

Unreal Engine 5 umí vykreslovat miliardy trojuhelníků v reálném čase díky průlomové technologii jménem Nanite. Avšak Nanite není až tak nový, jak by si člověk mohl myslet. Má své kořeny u videoher jako Assassin’s Creed Unity.



Obr. 31 – Slide z prezentace SIGGRAPH 2015: Advances in Real-Time Rendering

Autoři Assassin’s Creed Unity měli vizi herního prostředí zasazeného do Paříže během Velké francouzské revoluce mezi lety 1789 – 1794. Výroba a rendering takového prostředí by ale byla konvenčními způsoby extrémně náročná a v praxi real-time renderingu vlastně i nemožná. Dali si tedy za úkol přijít s renderingovým řešením. [34]

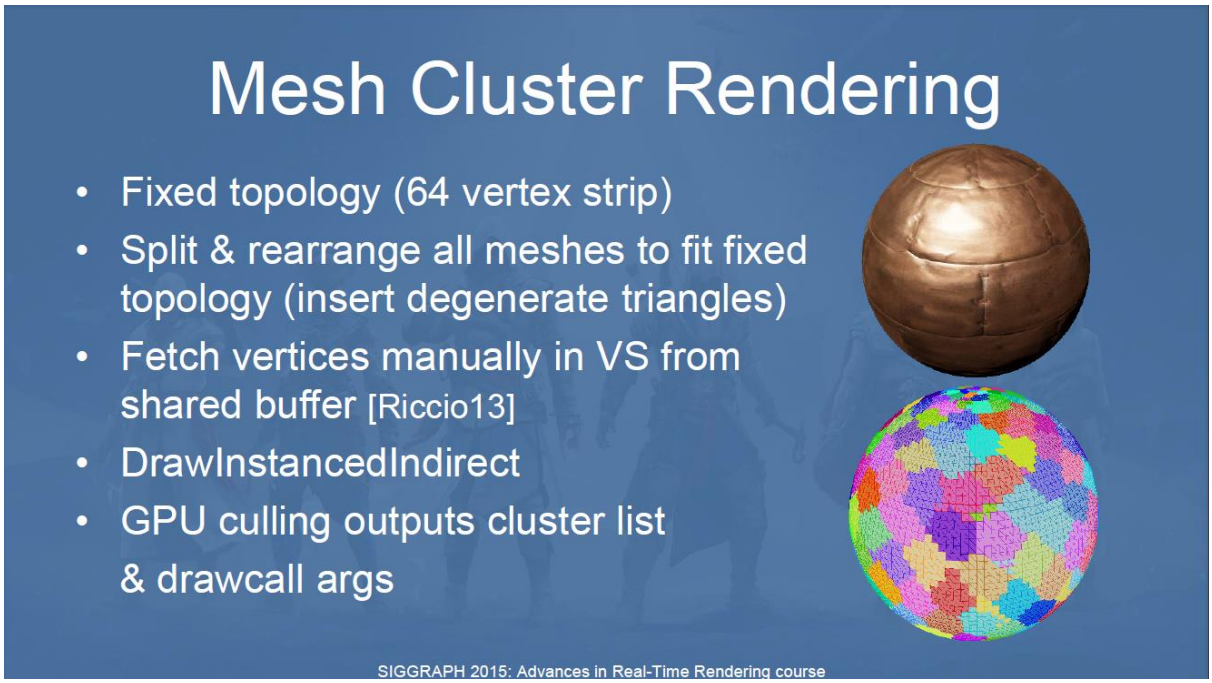
[33] Unreal Engine. *Unreal Engine 5 Revealed! | Next-Gen Real-Time Demo Running on PlayStation 5*. 2020. [YouTube video] [Citace: 19. Duben 2022]
[34] Haar, Ulrich a Aaltonen, Sebastian. *SIGGRAPH 2015: Advances in Real-Time Rendering in Games*. 2015 [Parafráze: 19. Duben 2022]

Ulice a budovy Paříže měly být vyrobeny z modulárních bloků. Typicky by tento přístup znamenal, krom neuvěřitelného množství trojuhelníků, především nepředstavitelné množství draw calls. Jak jsme si již říkali, zjednodušeně, každý objekt je vyrenderován tolikrát, kolik má materiálů. Další překážkou bylo zpracování samotných interiérů budov, které měly obsahovat obrovské množství rekvizit. (Haar, a další, 2015)

Mimoto, v prostředí velkého města musíme řešit již zmiňované visibility a occlussion culling. Ty sice real-time rendering extrémně zrychlují, ale dokud pracujeme s paradigmatem „per-object" – dokud pracujeme s celými objekty, nemůžeme si dovolit například nevykreslit polovinu domu, ačkoliv ji nevidíme. Toto je velký problém ve scéně, kde hledíme na mnoho velkých, někdy se částečně překrývajících budov, protože v takové situaci musíme vykreslovat i velkou část geometrie, kterou vůbec nevidíme.

Obr. 32 – Slide z prezentace SIGGRAPH 2015: Advances in Real-Time Rendering

Inženýři ze studia Ubisoft tedy přicházejí s nápadem. Co kdybychom se přestali pohybovat v paradigmatu celých objektů a začali pracovat na úrovni skupin trojuhelníků? Model tak můžeme rozdělit do tzv. clusters a dále pracovat s těmi. Právě na tomto principu funguje Nanite.



Obr. 33 – Slide z prezentace SIGGRAPH 2015: Advances in Real-Time Rendering

Než si začneme vysvětlovat principy renderingu s využitím technolgie Nanite, dovolím si lehkou předmluvu. *Tuto technologii práce popisuje velice zjednodušeně a z pohledu technického umělce, nikoliv optikou počítačového inženýra.*

Připomeňme si pipeline real-time renderingu. Víme, že modely vykreslujeme pomocí trojuhelníků – vykreslujeme tedy trojuhelníkové sítě. Každý model ve scéně vnímáme jako jeden objekt. Ten může mít nějaké číslo vlastností – materiálů. Tyto vlastnosti se nesdílí mezi objekty. To znamená, že pokud chceme vyrenderovat dvě identické krychle, každou se třemi identickými materiály, i přes to, že jsou krychle naprosto stejné, pořád musíme využít 3x2 draw calls.

Každý model v real-time renderingu má své level of detail modely – LODs. Ty vykreslujeme za pomoci hodnoty screen size. Tato hodnota popisuje, jak velkou část obrazovky onen objekt zabírá. Čím větší je tato hodnota, tím detailnější LOD model vybíráme pro rendering. A naopak, čím menší je tato hodnota, tím méně detailní LOD model vykreslíme.

Modely, které jsou mimo zorné pole kamery, nevykreslujeme. Avšak protože pracujeme s celými objekty, pokud vidíme i jen tu nejmenší část nějakého modelu, musíme ho vykreslit celý, a to bez ohledu na to, jak malou část jej vidíme.

A nakonec, můžeme si dovolit vyrenderovat pouze určité číslo trojuhelníků. Čím více je trojuhelníků na obrazovce, tím více dáváme zabrat počítači.

Mezi všechny tyto aspekty rozdělujeme určitý budget – rozpočet výkonu počítače s tím, že máme určitou cílovou frekvenci výstupu snímků. Tato frekvence bývá (v případě videoher) nejméně 60 snímků za sekundu. Takto tedy pracoval real-time rendering v Unreal Engine 4.

Technologie Nanite naprosto mění paradigma real-time renderingu právě tím, že již nezachází s jednotlivými objekty jako s celky. Místo toho, díky tzv. mesh cluster renderingu, rozděluje modely na clustery - skupiny trojuhelníků, se kterými dál pracuje. Tento přístup má za následek například kompletní deprekaci a konec LOD modelů, které již nejsou potřeba. Navíc, díky „podrozdělení“ modelů do clusterů nyní můžeme vykreslovat pouze některé části modelu, v závislosti na zorném poli kamery. Místo low-poly modelů nyní můžeme používat high-poly modely bez optimalizací. A třešničkou na dortu je sdílení vlastností clusterů mezi více clusterů na úrovni celé scény, což znamená extrémní redukci draw calls.[35] (Epic Games, Inc., 2022)

Pro hlubší pochopení se nyní soustředme na LOD modely. Cituji video autora Zero Conditional na platformě YouTube.

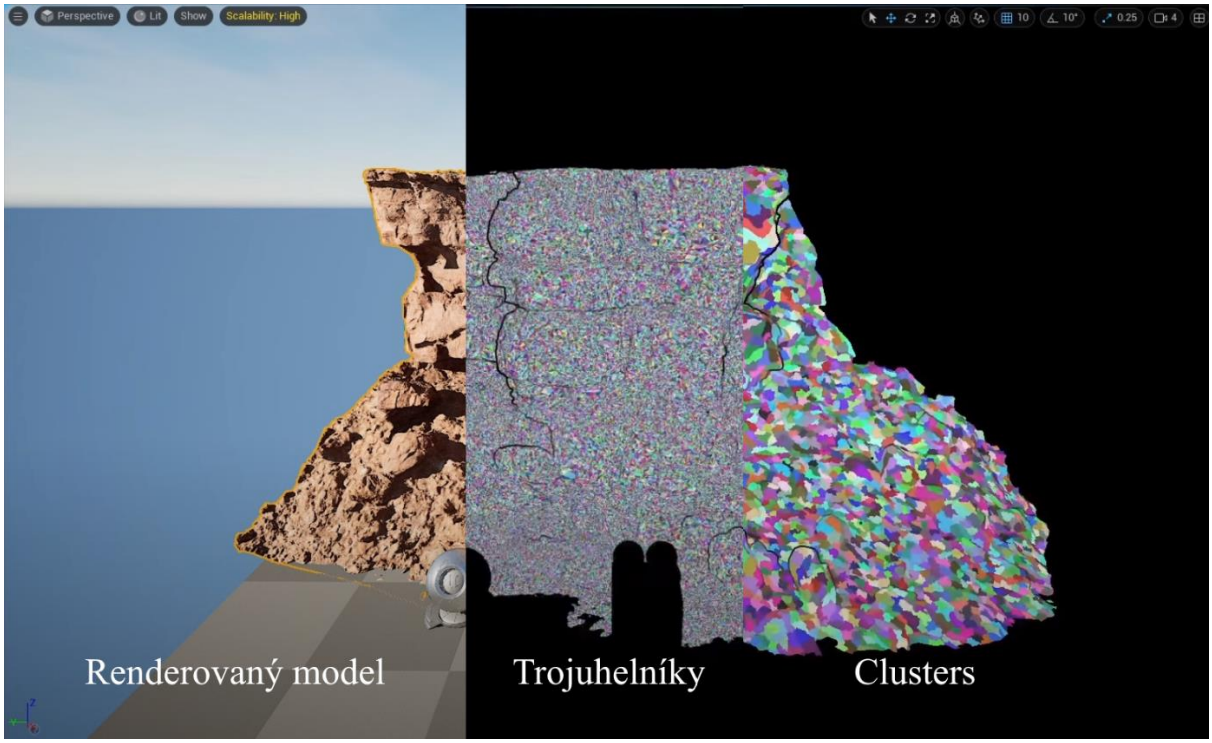
„Co [Nanite] tedy dělá, když víme, že se nemusíme starat o vytváření LOD modelů a retopologizaci modelů, co Nanite tedy dělá? Dělá to [LODs a retopologii] za nás? Nebo provádí nějakou zvláštní konverzi?“[36] (Conditional, 2021)

„Ve skutečnosti [Nanite] dělá to, že analyzuje trojuhelníkovou síť modelu, když je importována do Unreal Enginu. Potom tyto trojúhelníky rozdělí do skupin a tyto skupiny se nazývají clusters… … proč to tedy dělá, proč vytváří skupiny (clustery)? No, vytvořením skupin v podstatě síť rozbíjí. Rozděluje trojúhelníky do skupin, které může dále ovládat. A jakmile to udělá, je schopen tyto skupiny trojúhelníků dynamicky upravovat a vyměňovat za různé úrovně detailů, řekněme na základě pohledu kamery.”[37] (Conditional, 2021)

[35] Epic Games, Inc. *Nanite Virtualized Geomtery in Unreal Engine | Unreal Engine Documentation.* 2022 [online] [Parafráze 19. Duben 2022]
[36],[37] Zero Conditional. *Absolute Beginner’s Guide to Unreal Engine 5 Nanite.* 2021 [YouTube video] [Citace: 19. Duben 2022]

Nanite tedy vezme trojuhelníkovou síť modelu a tu rozdělí do více menších skupin trojuhelníků. Tyto skupiny, clustery, může dále dynamicky upravovat. Například „zhoustnutím“ sítě trojuhelníku v této skupině dociluje vyššího detailu dle zorného pole kamery. Toto je princip již zmiňovaného Mesh Cluster Renderingu (=renderingu clusterů sítě).

Proč již nepotřebujeme LOD modely? Pokud máme přístup k těmto clusterům, skupinám trojuhelníků, na individuální bázi a můžeme dynamicky měnit jejich hustotu (či rozlišení), poté není třeba vyrábět více celých modelů v různých statických rozlišeních. Dalo by se říci, že pracujeme s „cluster-based dynamic LOD systémem“ – dynamickým systémem level of detail, který je založený ne na celých modelech, ale na jejich clusterech.



Obr. 34 – Vizualizace modelu za užití technologie Nanite v Unreal Engine 5

Mimoto, nově můžeme renderovat pouze část modelu – Již nezacházíme s celými objekty, ale pouze s jejich clustery. Pokud polovina modelu není v zorném poli kamery, Unreal Engine jednodušše řekne rendereru, že clustery zapadajíc do poloviny, která není obsažena v kameře, nemá vykreslovat.

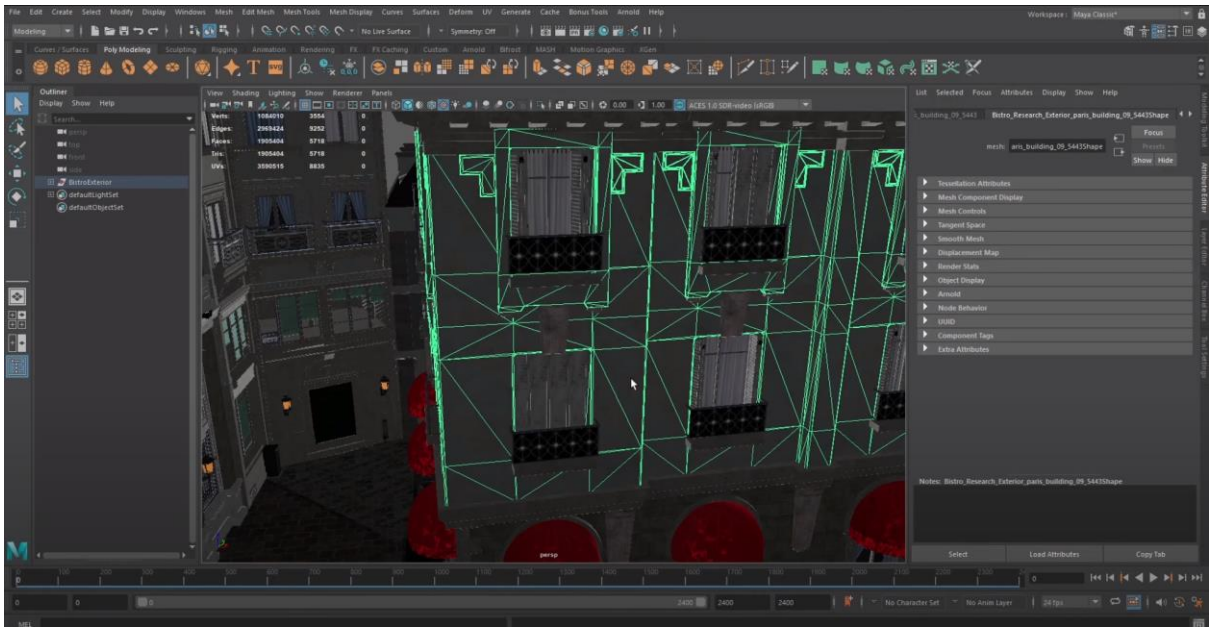
Navíc, vlastnosti těchto clusterů – to znamená jejich materiály – jsou sdílené přes celou scénu všem clusterům v ní obsažené. Tudíž, pokud bychom renderovali dvě indentické krychle s těmi samými třemi materiály, nově budeme potřebovat pouze 3 draw calls, a to nehledě na počet renderovaných krychlí.

Obr. 35 – Náhled vizualizačních módů technologie Nanite scény v Unreal Engine 5

Obr. 36 – Příklady geometrie vyrenderované v Unreal Engine 5 pomocí technologie Nanite

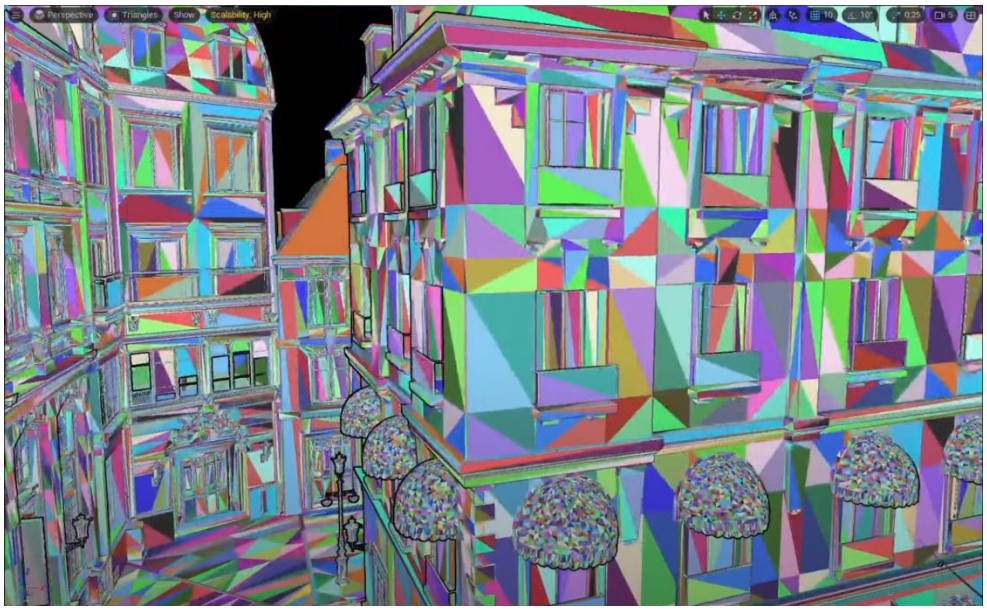
Technika mesh cluster renderingu, potažmo Nanite je tedy perfektním nástrojem pro rendering velkých měst s obrovskými budovami, které se navzájem překrývají a okludují. Na obrazovce jsou miliardy trojuhelníků, všechny renderované v real-time. Toto Epic Games ukázali ve svém demu *The Matrix Awakens.*

Nanite má ale i jeden velký háček. Pro to, aby Nanite mohl model rozdělit na skupiny trojuhelníků (clusters), model potřebuje mít hustou topologii. Při renderingu low-poly modelů tudíž výhody Nanite neužijeme, protože ty nejmenší trojuhelníky, zaznamenané ve vstupních datech modelu importovaného do enginu, jsou již při importu velké – optimalizované.



Obr. 37 – Příklad topologie low-poly optimalizovaného assetu, který může pro Nanite být problematickým.

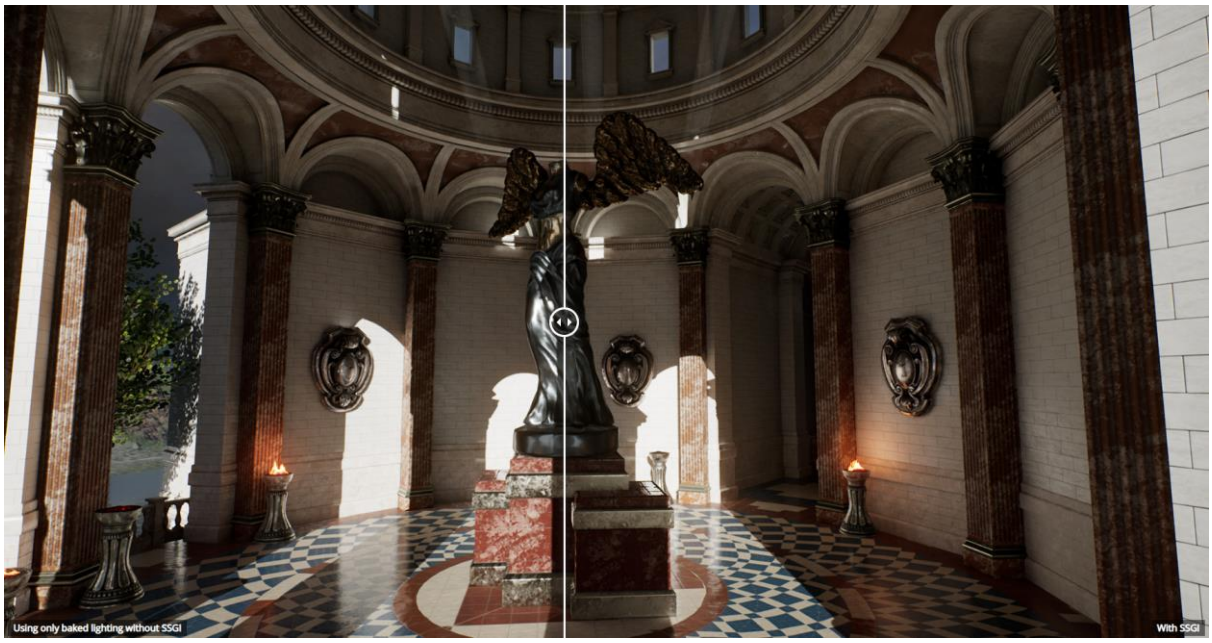
Toto může být i problém, protože ačkoliv onen model bez problému vyrenderujeme, málo trojuhelníků a tím i málo clusterů znamená to, že při oddalování, či přibližování kamery nebudeme na modelu pozorovat žádné změny – přestane se optimalizovat dle vzdálenosti od kamery, protože při přerozdělování trojuhelníků do skupin byly nejmenšími vstupními trojuhelníky již velké, optimalizované trojuhelníky.[38]



Obr. 38 – Trojuhelníky assetu na obr. 37 po importu do Unreal Engine 5 s použitím Nanite.

[38] Zero Conditional. *Absolute Beginner's Guide to Unreal Engine 5 Nanite.* 2021 [YouTube video] [Parafráze: 19. Duben 2022]

4.2 Technologie real-time global illumination – Lumen



Obr. 39 – Nalevo scéna nasvícená čistě offline, bez užití SSGI. Napravo ta samá scéna s užitím SSGI.

„Globální iluminace (někdy nazývaná indirect lighting, nepřímé osvětlení, či indirect illumination, nepřímá iluminace) simuluje interakce světla s geometrií a povrchy materiálů, za cílem doručit realistické nasvětlení Vašim scénám a projektům.“[39] (Epic Games, Inc., 2021)

Scény pro real-time rendering nasvěcujeme dvěma způsoby. Buď si předem offline vyrenderujeme již nasvěcenou scénu a pak užíváme tzv. lightmaps – texturové mapy, které popisují interakce světla s geometrií scény, nebo užíváme metody real-time nasvěcování za užití dynamických zdrojů světla.

Historicky byla nepřímá iluminace vždy renderována offline, kvůli obrovskému počtu kalkuací nutných pro simulaci nepřímého nasvěcování – to znamená, sekundární nasvěcování využitím již odražených paprsků světla. Jsou ovšem i další metody, kterými „podvádíme“ a vytváříme efekt globální iluminace. V Unreal Engine 4 bylo globální osvětlení zpracováváno především přímou iluminací. Všechny druhy indirektní iluminace

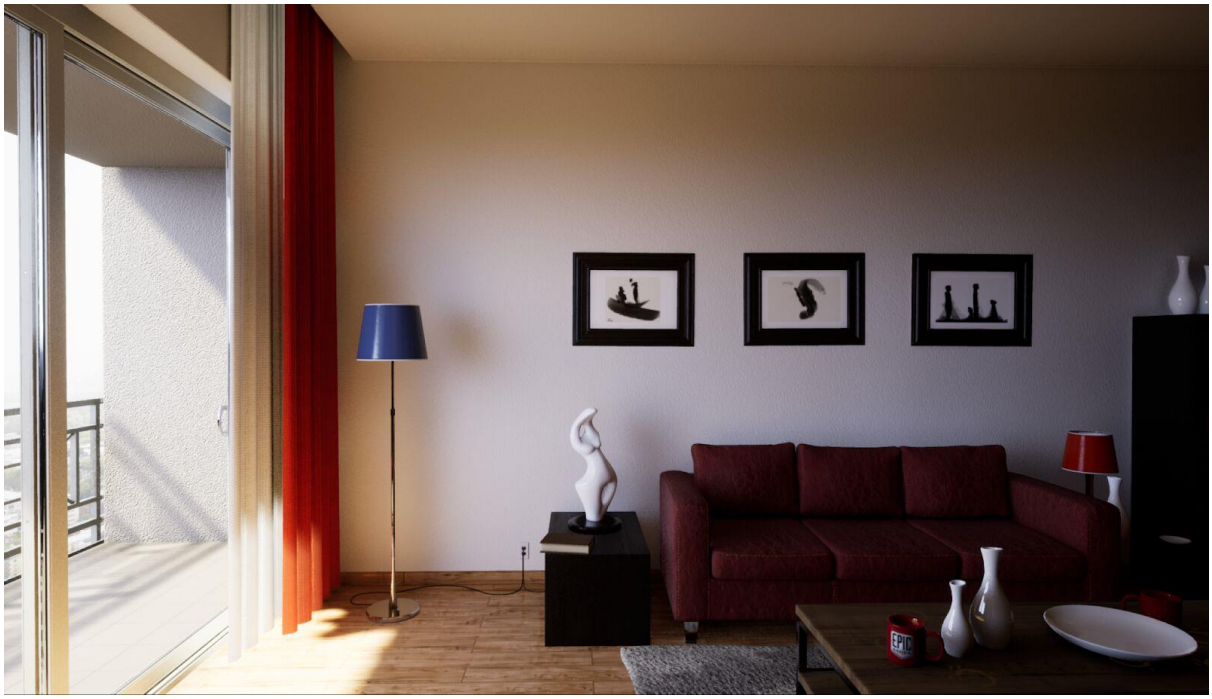
[39] Epic Games, Inc. *Global Illumination | Unreal Engine Documentation* 2021 [online] [Citace: 19. Duben 2022]

se spoléhaly na hrubé odhady či efekty, jako je například Screen Space Global Illumination, viz. Obr. 39.

„Screen Space Global Illumination (SSGI) využívá screen-space postprocessingový efekt pro generování dynamického nepřímého osvětlení. Tato metoda je omezena na objekty a osvětlení v pohledu kamery pro generování světelných dat. Případy, kdy jasná světla nejsou vidět nebo jsou blokována objekty ve scéně, mohou způsobit potíže.“[40] (Epic Games, Inc., 2021)

Skutečně dynamická globální iluminace v reálném čase byla až donedávna nemyslitelným úkolem. Tato skutečnost se změnila v roce 2018, kdy firma nVidia vydala první RTX (ray-tracingové) grafické karty. Avšak epidemie viru Covid-19 a další tržní podmínky znemožnily koupi těchto karet většině konzumentům.

Dynamická globální iluminace vyžaduje extrémní výkon výpočetní techniky a právě proto se považuje za „Svatý grál počítačové grafiky“. (Unreal Engine, 2021) Pokud opomeneme technologii RTX firmy nVidia, můžeme říct, že do vydání Unreal Engine 5 byla indirektní globální iluminace v reálném čase nemožná.



Obr. 40 – Příklad nasvícení scény za využití Lumen globální iluminace. (Epic Games, Inc., 2022)

[40] Epic Games, Inc. *Global Illumination | Unreal Engine Documentation* 2021 [online] [Citace: 19. Duben 2022]

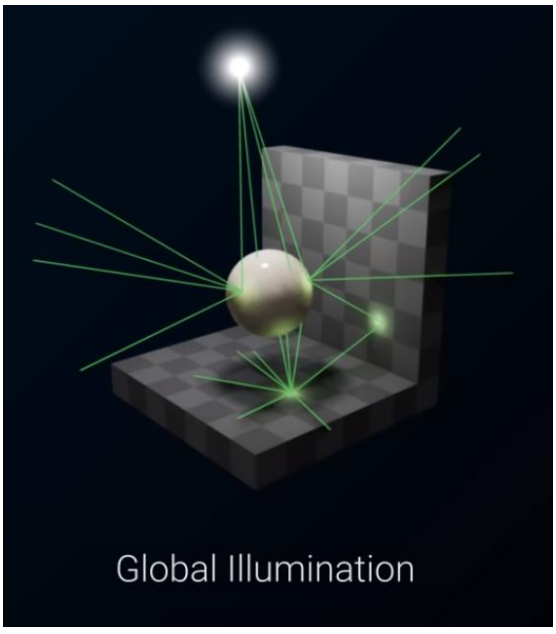
„Lumen je plně dynamický systém globálního osvětlení a odrazů v Unreal Engine 5, který je navržen pro konzole nové generace a je také výchozím systémem globálního osvětlení a odrazů. Lumen poskytuje rozptýlené vzájemné odrazy s nekonečnými odrazy a nepřímými zrcadlovými odrazy ve velkých, detailních prostředích v měřítku od milimetrů po kilometry.”[41] (Epic Games, Inc., 2022)

Unreal Engine 5 přichází s plně dynamickým řešením real-time globální iluminace v podobě systému Lumen. Lumen kombinuje přímé osvětlení, které známe z Unreal Engine 4 a přidává mu novou dimenzi – nepřímé osvětlení, které poskytuje fyzikálně korektní řešení nepřímo osvětlených míst v našich scénách.

Mezi nové funkce globální iluminace Lumen mimo jiné patří: rozptýlení barvy na okolní objekty, fyzikálně korektní měkké nepřímé stíny a také fyzikálně korektní reflekce, které byly nově poprvé unifikovány se světelným systémem. Lumen je tak tím nejbližším, co máme k naprosto fyzikálně přesnému chování světla v real-time renderingu.

Jak Lumen funguje?

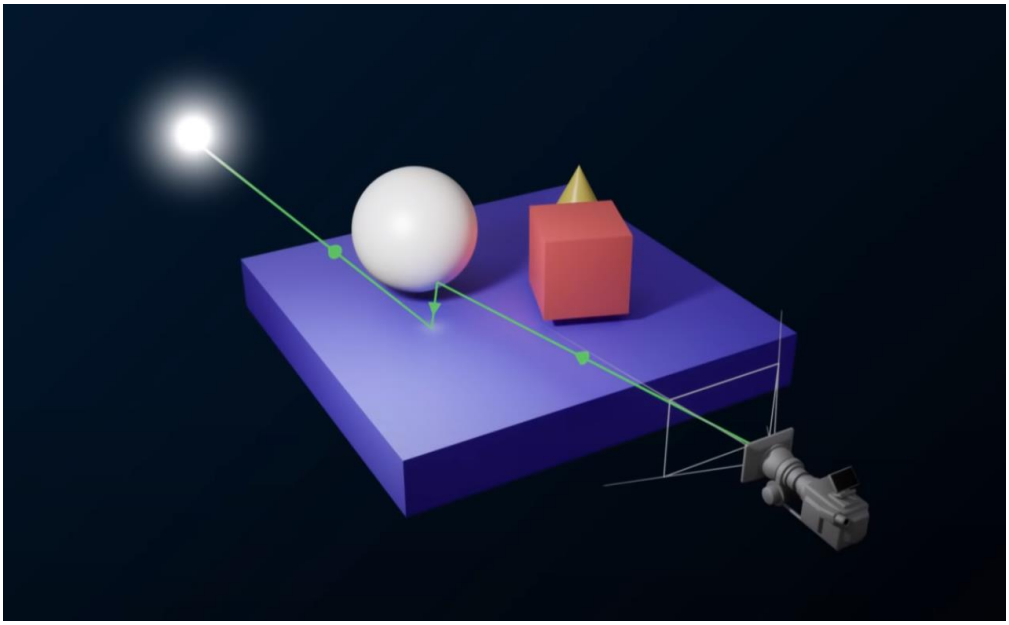
Již víme, že ray-tracing vytváří obraz tím, že sleduje světelné paprsky a zaznamenává jejich interakce s objekty ve scéně.[42] (Unreal Engine, 2021)



Obr. 41 – Princip globální iluminace za užití ray-tracingu

[41] Epic Games, Inc. *Lumen Global Illumination and Reflections.Unreal Engine Documentation,* 2022. [online] [Citace: 19. Duben 2022]
[42],[43] Unreal Engine. *Lumen in UE: Let there be light! | Unreal Engine.* 2021 [YouTube video] [Parafráze: 19. Duben 2022]

Pokud chceme docílit efektu globální iluminace za užitím ray-tracingu, potřebujeme obrovské množství sledovaných paprsků. Toto je extrémně náročné na výkon počítače. Proto Lumen užívá hybridní přístup k sledování paprsků.[43] (Unreal Engine, 2021)



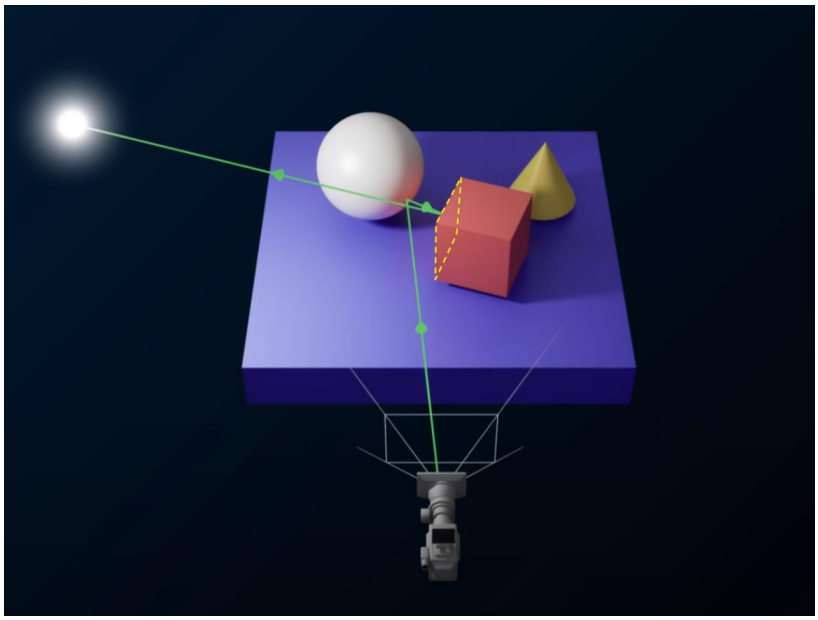
Obr. 42 – Sledování paprsku v ray-tracingu

Hybridní přístup Lumen k vykreslování je následující: Nejprve Lumen využívá klasického principu screen-space ray-tracing, kdy z pomyslné obrazovky posíláme a sledujeme paprsek, který při protnutí virtuálního objektu generuje další paprsky, které dále putují k zdrojům světla ve scéně.

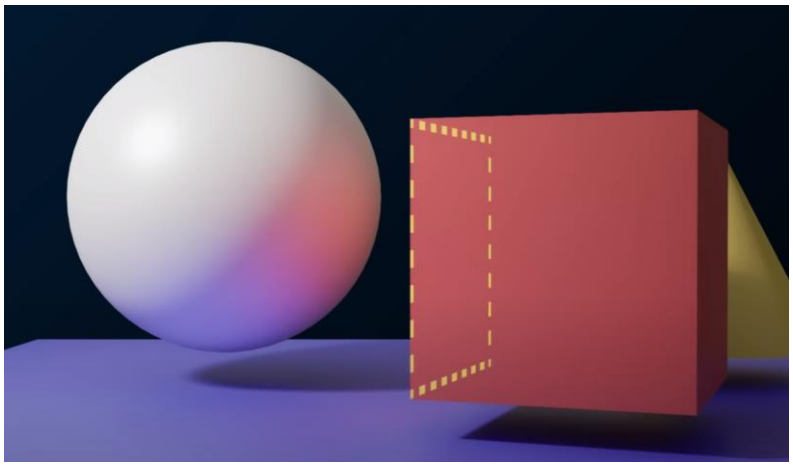
Jenže ray-tracing má jednu zásadní limitaci.

„Screen-space paprsky mohou interagovat pouze s objekty v zorném poli kamery. Ale ve většině případů existují části obrazovky,ke kterým se screen-space traces nemohou dostat. A pro správné osvětlení a odrazy musíme neustále brát v úvahu objekty i mimo obrazovku.”[44] (Unreal Engine, 2021)

[44] Unreal Engine. *Lumen in UE: Let there be light!* | *Unreal Engine.* 2021 [YouTube video] [Citace: 19. Duben 2022]



Obr. 43 – Příklad roviny (strany krychle), ke které se při užití screen-space ray-tracingu nemůže paprsek dostat.



Obr. 44 – Příklad roviny viz. Obr. 42

Unreal Engine mohl tento problém vyřešit jednoduše využitím RTX grafických karet fimy nVidia a jejich technologie Hardware Ray Tracing (HRT), avšak Epic Games potřebovali řešení, které by bylo kompatibilní jak s předchozími tak i s budoucími řadami grafických procesorů. Vytvořili proto vlastní softwarový způsob vykreslování, který tyto podmínky splňuje.

Tato metoda využívá tzv. signed distance fields. (SDFs) „Signed distance field je objemová reprezentace povrchu ve 3D. Jde o aproximaci původní trojúhelníkové sítě.“[45] (Unreal Engine, 2021)

Obr. 45 – Vizualizace mesh SDFs v Unreal Engine.

Každá síť trojuhelníků tedy má svůj vlastní mesh SDF, který je objemovou reprezentací dané sítě trojuhelníků. Jde o tedy zjednodušenou verzi daného modelu.

„Na trasování do SDFs je opravdu skvělé to, že je velmi levné, protože celá trasovací smyčka je pouze o tom, že parpsky procházejí prostorem, dokud nenarazí na povrch. Máme řešení, jak trasovat povrchy a vypočítat GI. To je skvělé. Abychom však mohli správně odrážet světlo, musíme ještě vzorkovat vlastnosti povrchů ve scéně. Vzorkování těchto vlastností přímo z trojúhelníků pro každý zásah by bylo strašně drahé. Vymysleli jsme tedy chytřejší přístup.”[46] (Unreal Engine, 2021)

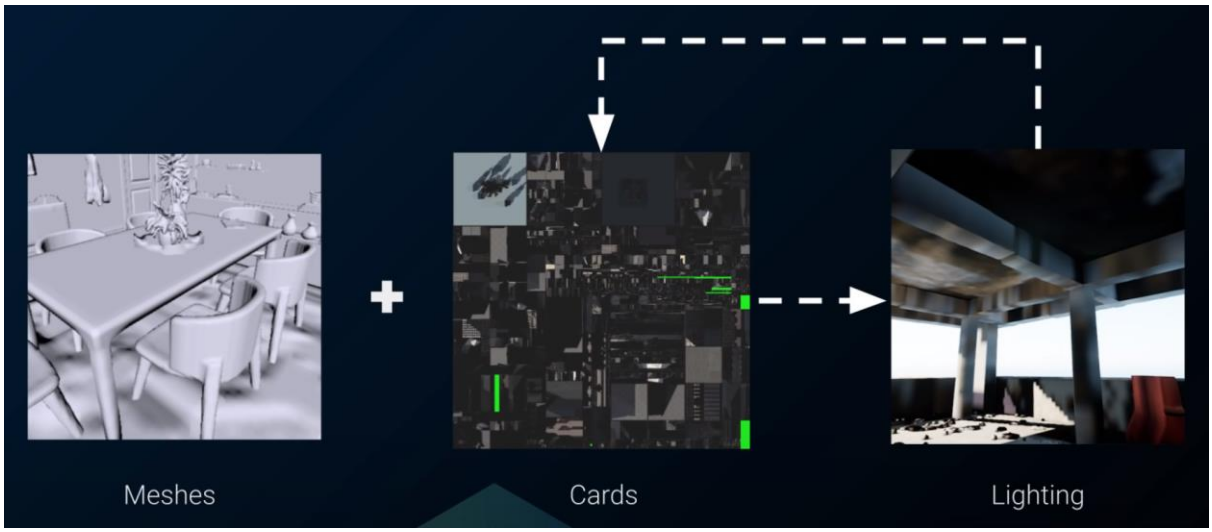
„Vygenerovali jsme parametrizaci povrchů blízkých objektů, kterou nazýváme „povrchová mezipaměť“. Tato mezipaměť se používá k vyhledání informací o povrchu a osvětlení, když paprsek zasáhne daný (tento) bod. Berte to jako způsob, jak uložit do mezipaměti informace o povrchu a osvětlení v GPU, abyste je mohli později použít.“ [47]

[45], [46], [47] Unreal Engine. *Lumen in UE: Let there be light! | Unreal Engine.* 2021 [YouTube video] [Citace: 19. Duben 2022]

Abychom získali vlastnosti povrchu blízkých objektů, využíváme tzv. cards. Cards bychom mohli přirovnat k „opačným cubemapám“. Zatímco tradiční cubemap nám popisuje offline-vygenerované odlesky okolního prostředí, cards obdobná data zachycují.

Tato data poté Lumen používá vyrenderování všech vlastností povrchů objektů ve scéně, z několika úhlů. Například, pokud chceme vykreslit krychli na chodníku, projekce cards vyrenderuje 5 stran naší krychle a tato data uloží do povrchové mezipaměti.

Z těchto dat nyní Lumen může spočítat jak přímé, tak nepřímé osvětlení. Výstup osvětlení se poté užívá ve smyčce, přičemž output osvětlení je zachycen cards, které tato data dále propagují v dalších snímcích.



Obr. 46 – Popis dat potřebných pro vyrenderování osvětlení pomocí Lumen.

Historicky poprvé se díky Lumen můžeme bavit i o fyzikálně korektních reflekcích v reálném čase, které se již nespoléhají na screen-space efekty. Osvětlení a reflekce musely vždy být naprosto oddělenými systémy, protože reflekce se nepočítaly na fyzikálním zakladě interakcí světla, ale buď předpočítanou cube-mapou, fotografií namapovanou na krychli, která sloužila jako obraz, který by byl poté promítnut v reflektivních materiálech, či již zmíněným způsobem přes prostor obrazovky – screen space reflections.

Obr. 47 – Příklad cubemapy.

„Po mnoho let ve videohrách musely být osvětlení a odrazy

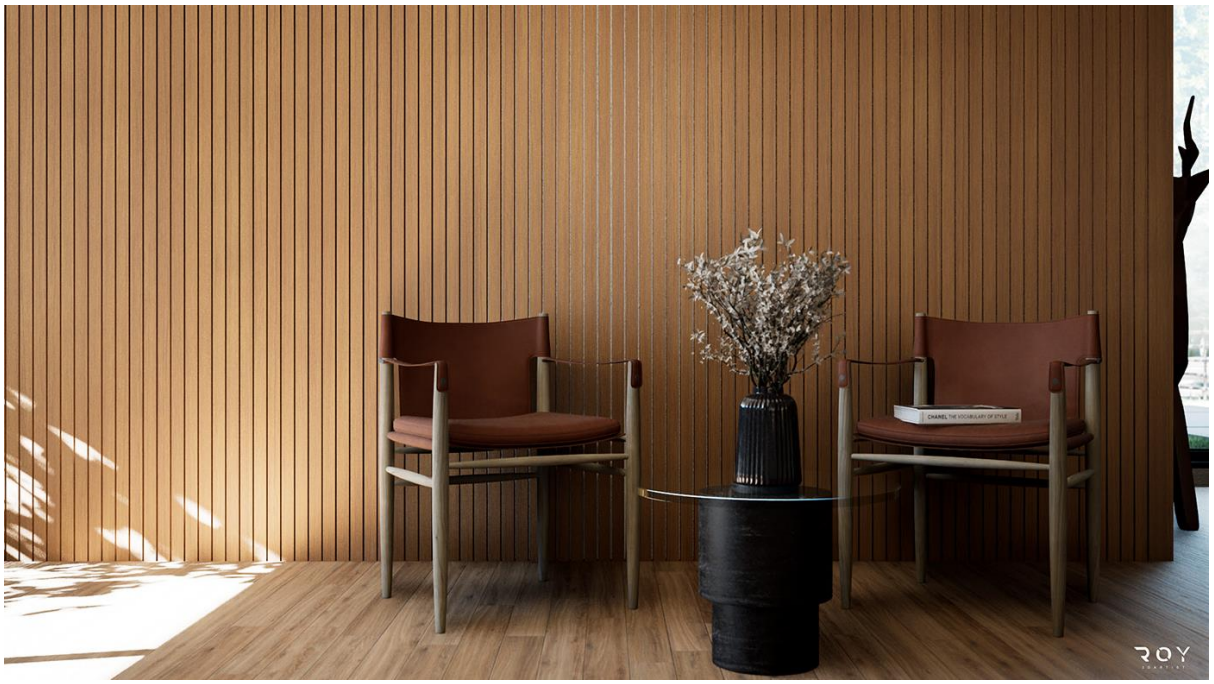
zcela oddělené systémy. Ale v reálném životě jde o jeden jediný fenomén. Lumen GI tyto systémy konečně sjednocuje. GI a odrazy v Unreal Engine 5 se počítají společně.. Pro Lumen je to další výhrou. Když už trasujeme řadu paprsků pro výpočet GI, proč bychom je nemohli znovu použít pro odrazy scény?"[47] (Unreal Engine, 2021)

4.2.1 Specifické případy užití Lumen GI

Obr. 48 – Příklad architektonické vizualizace v Unreal Engine 5

Možná nejpůsobivějším případem využívání globální iluminace Lumen jsou aplikace v architektonické vizualizaci. Díky nepřímé globální iluminaci nyní můžeme nasvítit interiéry pouze pomocí světla simulovaného Slunce za oknem. Interiéry vždy bývaly nasvíceny mnoha „falešnými" světly, protože pokud celý pokoj nasvicujeme pouze jedním oknem, setkáme se s problémem šumu (noise) kvůli opětovnému odrážení světelných paprsků. Z pohledu rendereru je celý prostor černý a odražené paprsky se snaží najít onen jeden zdroj světla.

[47] Unreal Engine. *Lumen in UE: Let there be light! | Unreal Engine.* 2021 [YouTube video] [Citace: 19. Duben 2022]



Obr. 49 - Příklad architektonické vizualizace v Unreal Engine 5 (Fredy, 2021)

Obr. 50 – Příklad architektonické vizualizace v Unreal Engine 5 (Scionti, 2021)

Obr. 51 – Příklad architektonické vizualizace v Unreal Engine 5 (Scionti, 2021)

Obr. 52 – Příklad architektonické vizualizace v Unreal Engine 5 (Dunlop, 2021)

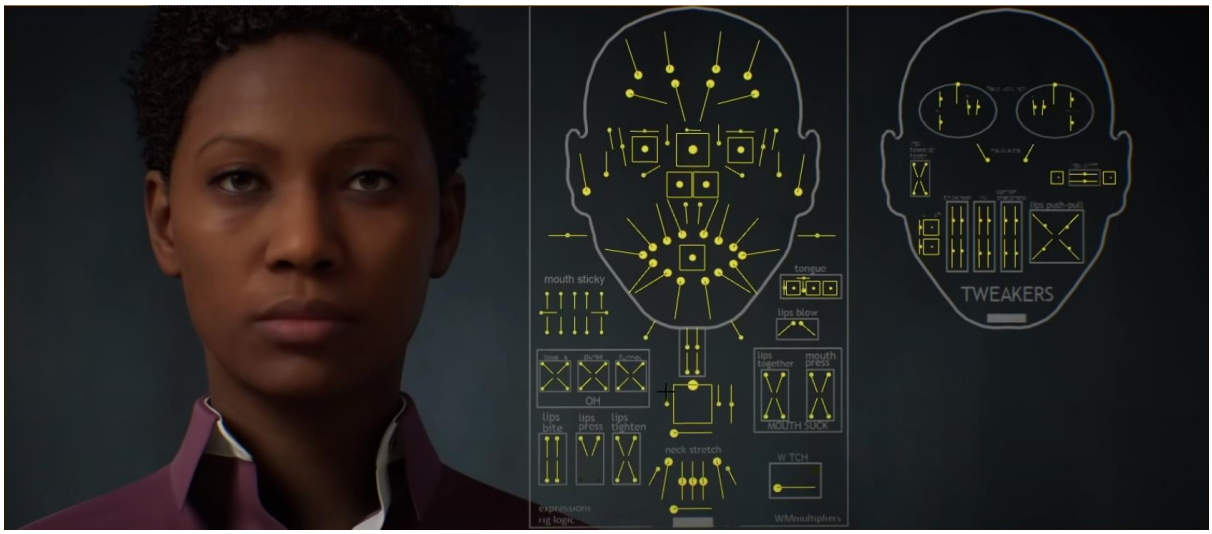
Obr. 53 - Příklad architektonické vizualizace v Unreal Engine 5 (Yilmaz, 2021)

4.3 MetaHuman

Obr. 54 – Obrázek z webuMetaHuman Creator Documentation

Program MetaHuman Creator je dalším programem od firmy Epic Games. Od dubna 2021 se nachází v otevřené beta verzi a je dostupný všem uživatelům Unreal Engine. Jde o cloudovou aplikaci, která uživatelům umožňuje vytvoření fotorealistické postavy, či chcete-li, loutky, během několika minut. Tuto postavu posléze můžeme vyexportovat buď do Unreal Engine, nebo například do programů jako je Autodesk Maya, kde se dá nadále animovat a podobně.

MetaHuman Creator je uživatelsky velice přátelský. Znovu se bavíme o demokratizaci a zvyšování dostupnosti nástrojů pro výrobu počítačových her, či animovaných filmů. Pro vytvoření vlastní fotorealistické postavy uživatel nemusí znát složité programy jako jsou Zbrush, či Blender. Program je součástí infrastruktury Epic Games a tudíž v Unreal Engine najdeme předem připravené podpůrné systémy, které nám dovolují s postavou dál pracovat. Mezi tyto nástroje patří například MetaHuman Facial Rig.



Obr. 55 – MetaHuman Facial Rig v Unreal Engine

5. BUDOUCNOST AUDIOVIZUÁLNÍHO PRŮMYSLU

„...a kamera se tak mohla dostat kamkoliv. Mohli jsme ukázat interiéry... No a potom jsme do toho [videostěny] začaly vestavovat sety. Měli jsme fyzickou polovinu vesmírné lodi s reflektivním povrchem, zatímco druhá polovina byla na videostěně. A tak to začalo být vzrušující, protože ke konci sezóny jsme si řekli „Začněme vyrábět sety kolem toho, co to umí dobře!". [48] (Industrial Light & Magic, 2020)

„...Je to opravdový přelom pro filmařinu.“ [49] (Industrial Light & Magic, 2020)

Obr. 56 - Fotografie z natáčení The Mandalorian za využití Unreal Engine a videostěny. (Industrial Light & Magic, 2020)

Kdybyste před pěti lety někomu tvrdili, že jedním z největším zvratů v produkci počítačově-asistovaných filmů, bude herní engine, jen těžko byste hledali někoho, kdo by se Vám nevysmál. A přesto tu jsme. Real-time rendering dnes již umí vše, co offline rendering. K tomu se s ním pracuje rychleji a jednodušeji a společně s technologií videostěn by během několika let mohl nahradit legendární green screen.

[48],[50] Industrial Light & Magic. *The Virtual Production of The Mandalorian Season One.* 2020 [YouTube video] [Citace: 20. Duben 2022]

Po straně televizní produkce, již dnes vidíme program Unreal Engine využíván k produkci setů televizních novin a zpráv. Tento workflow také užívá videostěny, avšak mnohem menší a primitivnější, než ty, o kterých jsme diskutovali.

Rozšířená realita se už dostala i za naše hranice. „Nehybné exponáty za skleněnými tabulemi byly často tím jediným, na co muzea návštěvníky lákala. Ve vytřeštěných skleněných očích vypreparovaných zvířat se však začaly postupem času odrážet obrazovky s videosekvencemi, později také tablety, díky kterým dostala výstava nový rozměr a návštěvníci ji mohli prožít interaktivním způsobem. Nyní však začínají lákat muzea i na další zábavu, o kterou se stará virtuální či rozšířená realita. Ta dokáže vtáhnout do děje expozice i ty, kteří by se jinak bezmyšlenkovitě šourali uličkami a jediné, co by je zajímalo, by byly jejich bolavé nohy…

… Studio Yord tímto způsobem oživilo již tři česká muzea. V Bruntále, kde se návštěvníkům přibližuje krása jesenické přírody, v muzeu značky Tatra v Kopřivnici a svými digitálními kouzli ozvláštnilo i výstavu Giganti doby ledové věnovanou mamutům a dalším obrovským tvorům minulosti, které je možné obdivovat v Černé v Pošumaví.“,[50] (Sedláček, 2022) píše o muzeích využívajíc rozšířenou realitu Vojtěch Sedláček, pro web CzechCrunch.

Je mým osobním názorem, že všechny momentální trendy ukazují na rostoucí popularitu Unreal Engine a obecně real-time renderingu. Jedním z nynějších nepopulárnějších témat je tzv. „metaverse“. Epic Games o metaversu a budoucnosti real-time grafiky píší:

„Nyní, když stojíme na okraji metaverse, jsme na křižovatce definující éru: kdo bude architektem tohoto nového světa? Ve společnosti Epic věříme, že bychom ji měli budovat my všichni, tvůrci nevyjímaje. Budujeme ekosystém kreativních nástrojů, které jsou otevřené a dostupné, aby umožnily každému mít podíl na metaversu.

Je příliš brzy na to, abychom přesně řekli, jak se metaverse vyvine, ale vidíme to jako sdílený sociální 3D svět s vytrvalostí, objevováním, umírněností a obchodem. Půjde o evoluci internetu, jak ho známe, a jeho základy budou postaveny na 3D technologii v reálném čase.

[50] Sedláček, Vojtěch. *Vycpané fretky střídají expozice plné života. Do českých muzeí si razí cestu virtuální a rozšířená realita.* 2022 [online] CzechCrunch s.r.o. [Citace: 19. Duben 2022]

V nedávném průzkumu, který pro nás provedla společnost Forrester, 85 % respondentů souhlasilo nebo rozhodně souhlasilo s tím, že technologie v reálném čase jsou pro budoucnost jejich společnosti velmi důležité, zatímco 82 % sdílí stejnou úroveň přesvědčení, že metaverse rozšíří způsob jejich interakce se zákazníky.

V tomto příspěvku vám poskytneme letmý pohled na některé statistiky z roku 2021, které odhalují boom v přijímání 3D technologie v reálném čase, a také to, co může rok 2022 připravit pro největší trendy v technologii..”[51] (Epic Games, Inc., 2022)

V lednu 2022 Epic Games evidují 500 milionů aktivních uživatelských účtů. Od konce roku 2020 do 2022 celkový počet stažení Unreal Engine vrostl o 40%.V programu MetaHuman Creator, který je dostupný od února roku 2021, bylo vytvořeno přes 1 milion postav. SketchFab, sociální síť pro 3D modeláře a umělce, též pod vlastnictvím Epic Games, eviduje 7 milionů uživatelů. 48% ohlášených, zatím nevydaných her příští generace, používá Unreal Engine. [52] (Epic Games, Inc., 2022)

[51],[52] Epic Games, Inc. *Real-time round-up 2022: the metaverse and emerging trends – Unreal Engine.* 2022 [online] [Citace: 19. Duben 2022]

5.1 Dostupnost a demokratizace výroby obsahu

Epic Games si udržují tradici zpřístupňování vývojářských a 3D-uměleckých nástrojů obyčejným lidem. Tato tradice započala již první hrou Tima Sweeneyho, *ZZT*, která se stala základem mnoha jiných her.

Již zmiňované tituly *Unreal Tournament* se svými level editory dělaly ze „střílečkových závisláků" level designéry, a následovné a opětovné vydávání dalších verzí Unreal Engine naprosto volně a zdarma přiblížilo svět vizuálních efektů, animace, filmu a vývoje počítačových her i mně, autorovi této práce.

Po akvizici společnosti Quixel, Epic dále koupil například SketchFab, socilní síť pro 3D umělce, kteří na této platformě mohou prodávat své modely. Velkou výhodou SketchFab oproti ostatním, podobným, platformám, je jeho schopnost renderingu 3D modelů přímo v internetovém prohlížeči. Zákazník si tedy mohl model zblízka a ze všech stran prohlédnout před tím, než jej koupil.

Známý web ArtStation, kde umělci prezentují svá portfolia a hledají kontakty, či práci, nebo kde prodávají své modely a grafiku, byl také koupen Epic Games.

„Drazí přátelé. Dnes s potěšením oznamujeme, že ArtStation se připojuje k rodině Epic Games. Společně urychlíme rozvoj a růst komunity autorů po celém světě. Toto partnerství podtrhuje poslání firem Epic i ArtStation - poskytnout tvůrcům nástroje a platformy, aby mohli prosperovat. Oznámení si můžete přečíst na Epic Games Newsroom."[53] (ArtStation Team, 2021)

Okamžitě po akvizici Epic snížil ceny poplatků na platformě ArtStation.

„Ode dneška masivně zvyšujeme výdělečný potenciál prodejců na ArtStation Marketplace. Snižujeme naše standardní poplatky z 30 % na pouhých 12 %. Pro členy „Pro" se poplatek snižuje z 20 % na 8 % a 5 % za vlastní prodej. Další podrobnosti o nových poplatcích ArtStation Marketplace najdete v naší kalkulačce výdělků…

[53] ArtStation Team. *ArtStation is joining the Epic Games Family.* 2021 [online] [Citace: 19. Duben 2022]

…také zpřístupňujeme ArtStation Learning všem členům zdarma po zbytek roku 2021. ArtStation Learning je naše streamovací video služba pro umělce. Nabízí neomezený přístup ke stále rostoucí knihovně obsahu od profesionálů z oboru. Umělci si mohou rozšířit své dovednosti v široké škále uměleckých témat a učit se vlastním tempem. Chcete-li začít, přejděte na stránku artstation.com/learning. Chceme poděkovat vám – naší komunitě a přátelům. Děkuji za důvěru a děkuji za vášeň. Byla to neuvěřitelná cesta.Těšíme se na tuto další kapitolu ArtStation. Budoucnost je Epic! Všechno nejlepší! Podepsán Leonard Teo, generální ředitel a tým ArtStation.“[54] (ArtStation Team, 2021)

I Unreal Engine má své vlastní, proprieterní „tržiště“ – Unreal Engine Marketplace. Členové komunity zde mohou prodávat a nakupovat digitální obsah. Krom vysokého podílu pro umělce – při koupi assetu jde 88% z ceny umělci a 12% jako poplatek Epic Games – Epic také často pořádá slevy a výprodeje.

K tomu, každý měsíc Epic Games vyberou několik assetů, které onen daný měsíc budou zdarma. Často se jedná o velké a drahé balíčky assetů, které mohou nezávislým vývojářům pomoci při výrobě jejich projektů a koníčkářům přinejmenším udělají radost.

Dovolím si tvrdit, že žádná jiná společnost neudělala pro nezávislé umělce tolik, co Epic Games. Samozřejmě, Epic si mohou dovolit takto operovat z důvodu jejich behemotní pozice na trhu. Na tuto pozici se dostala, krom jiného, díky své počítačové hře, Fortnite. Fortnite se stal okamžitým hitem a dnes jej hraje 350 milionů hráčů. Forbes zakladatele Tima Sweeneyho cenní na 7,6 miliard amerických dolarů, v přepočtu 172 026 000 000 Kč dle nynějšího kurzu (19. dubna 2022). Tim Sweeney do dnes zůstává hlavním akcionářem společnosti.

[54] ArtStation Team. *ArtStation is joining the Epic Games Family.* 2021 [online] [Citace: 19. Duben 2022]

Obr. 57 – Fotografie z livestreamu „A Year in Review w/ Tim Sweeney & Joe Kreiner“ (Unreal Engine, 2017)

Epic Games svým otevřeným chováním vůči umělcům a své komunitě vývojářů pěstuje kulturu, která vychovává a inspiruje nové, mladší generace budoucích 3D umělců, vývojářů, scénáristů, kameramanů, fotografů a podobně. Tato kultura je kulturou nadšení, radosti, otevřenosti a výpomoci.

Ačkoliv následující tvrzení nemám jak fakticky podložit, dovolím si osobní vsuvku. Jako mladý „modder“ jedné nejmenované počítačové hry jsem se často nacházel na otevřených forech dedikovaných oné hře a jejím modifikacím. S úžasem v očích jsem hleděl na výtvory ostatních, již zkušenějších, modderů. Avšak kultura na těchto fórech nebyla kulturou pozitivní. Modely byly v té době vystavovány z důvodu jakési ješitnosti, ne pro inspirování ostatních. Každý, kdo se něco naučil, si své schopnosti držel pevně u těla a tak neexistovala jakákoliv jednotná dokumentace, moddeři si navzájem nepomáhali. Často vznikaly projekty, které měly naprosto stené cíle, ale místo sjednocení a výpomoci se jejich autoři pouze nesmyslně přemeřovali.

Ano, velká část z toho, co jsem právě popsal, je jednoduše lidská povaha. Ale z vlastní zkušenosti mohu potvrdit, že kultura, kterou Epic Games podporuje není touto kulturou ješitnosti, naopak jde o kulturu plnou výpomoci, návodů a sdílení znalostí. Snad osobnosti vychované touto kulturou budou ještě více ochotni pomáhat a náš průmysl tak bude nadále rozkvétat.

6. ZÁVĚR TEORETICKÉ ČÁSTI

Práce se věnovala real-time renderingu, jeho porovnání s offline renderingem a zaměřila se především na herní engine Unreal Engine. Uvedený text jasně vysvětluje výhody a nevýhody těchto renderingových metod a poukazuje na nové, teprve vznikající postupy práce například s virtuální produkcí.

Z textu práce je evidentní, že real-time rendering je budoucností vykreslování nejen počítačových her. S novými průlomovými technologiemi, které práce zkoumala, nadchází nová doba počítačového renderingu. Práce dokládá tuto domněnku nejen statistikou, ale také pozorováním trendů v oblasti jak vývoje videoher, tak produkce filmové.

První videohra s grafickým rozhraním, *Tennis for Two*, vyšla před šedesátičtyřmi lety. A tak když se vydáváme na nové dobrodružství počítačové grafiky, nezapomínejme na nespočet inženýrů, vědců a umělců, kteří tento neuvěřitelný vývoj umožnili.

7. PRAKTICKÁ ČÁST

Praktická část bakalářské práce má formu trojminutového animovaného filmu renderovaného v Unreal Engine. Jde o příběh vikingů, otce a syna, kteří spolu zažívají dobrodružství v cizí, nehostinné zemi.

Obr. 58 – Drakkar na moři

Původní vizi, námět a scénář filmu vypracoval Jan Kastner, se kterým jsem na filmu spolupracoval.

Zatímco Jan Kastner měl na starost charakter design, já jsem pracoval především na prostředích a zastupoval jsem pozici technického umělce. Také jsem využil společenských kontaktů při shánění zvukového inženýra, hudebního skladatele a hlasového herce, Davida Ephgrave. David Ephgrave pro film vytvořil originální zvukovou stopu, zvukové efekty a propůjčil svůj hlas oboum hrdinům filmu.

Unreal Engine jsme si pro render vybrali ze dvou důvodů. Prvním důvodem byla rychlost práce v něm v porovnání s offline renderingem a zároveň porovnatelná kvalita obrazu. Druhým důvodem byla již má předem získaná zkušenost s tímto programem, tudíž jsem se pohyboval ve familiarním rozhraní.

Cílem filmu *On the Edge of Glory* bylo vytvořit krátký animovaný film, který by se mohl vyrovnat cinematické kvalitě velkých animačních studií. Samozřejmě šlo o velkou vizi, ale i přes nevyhnutelné selhání jsme se mnoho naučili.

Obr. 59 – Římský chrám

Film užívá real-time renderingu Unreal Engine a některých technologií, které práce zmiňuje. Například používáme novou verzi Unreal Engine, Unreal Engine 5, a tím I systém globální iluminace Lumen. Ten byl instrumentální při svícení některých scén.

Ve filmu pracujeme s Lumen GI, fluidními simulacemi, dynamickým cyklem dne a noci, virtuálními kamerami, volumetrickými efekty – světly a mlhou a simulacemi

destrukce. Narazili jsme i na některé limitace, Unreal Enginu 5. Největší z nich je kompletní zrušení teselace. Unreal Engine 5 (alespoň v nynější formě) nepodporuje teselaci proprieterních objektů Landscapes (prostředí) a tak není jak dosáhnout efektu displacementu.

Obr. 60 – Fjord

Obr. 61 – Fjord

Obr. 62 – Fjord a renderování fyzikálně korektní vody jako geometrie

Obr. 63 – Prostředí interiéru kostela

Obr. 64 – Prostředí exteriéru kostela

Obr. 65 – Drakkar v bouři

8. ZÁVĚR

Cílem bakalářské práce byla obhajoba real-time renderingu jakožto plnohodnotného způsobu renderingu nejen pro videohry. Pro uchopení samotného renderingu jsem vysvětloval základy jednotlivých způsobů renderingu, od těch nejjednodušších, po ty nejsložitější. Čas jsem věnoval i historickému kontextu, bez kterého se neobejdeme.

Porovnával jsem přístupy online a offline renderingu a jejich specifika. Pro pochopení nových postupů jsem zašel až k materiálům. Velká část práce byla věnována nejmodernějším technologiím real-time renderingu a jejich hlubšímu porozumění. S tím také souvisí druhý můj druhý cíl, kterým bylo poukázat na budoucnost audiovizuálního průmyslu. Pozastavil jsem se tak u produkce s užitím virtuálních setů a významu real-time renderingu pro filmovou produkci. Také jsem psal o přístupnosti pole vizuálních efektů pro nováčky či nezávislé umělce a jak tato dostupnost může ovlivnit budoucnost audiovizuálního průmyslu. Domnívám se, že jsem téma bakalářské práce zpracoval dle mých nejlepších schopností.

9. TERMINOLOGICKÝ SLOVNÍK

Rendering - vykreslování, renderování.

Renderer - program zodpovědný za renderování

Real-time rendering - vykreslování v reálném čase, okamžitě

Offline rendering – předpočítávání renderovaných snímků

Rasterizace - proces renderingu

Ray-casting - proces renderingu

Ray-tracing - proces renderingu

Path-tracing - proces renderingu

Herní engine - program pohánějíc počítačovou hru

Virtuální produkce - produkce na place za užití počítače, který produkci podporuje ve skutečném čase

Fotogrammetrie - způsob skenování 3D modelů

Vertex - vrchol polygonu, bod v kartézské soustavě souřadnic

Face - rovina (geometrie)

Edge - hrana (linka)

Polygon - mnohoúhelník

Rasterizer (rasterizátor) - počítačový program, který provádí rasterizaci

Fragmenty - výstup rasterizátoru, body namapované ke skupinám pixelů

Pixel - nejmenší součást obrazovky

Pixel shader - počítačový program, využívaný pro proces renderingu

Textura - data ve formě obrázku

PBR - rendering na základě fyzikálních vlastností, druh renderingu

SDR - rendering na základě spekularity, druh renderingu

Albedo - druh textury

Specular - druh textury

Metallic - druh textury

Roughness - druh textury

Normal - druh textury

AO , ambient occlusion - Metoda 3D grafiky, také stejnojmenný druh textury

Vektorový paramtr - soubor dat o třech parametrech

Skalární parametr - číselná data, určují velikost něčeho

Difúzní barva - jednolitá barva

Shading - stínování, metoda 3D grafiky

Per-fragment data - data určená každému fragmentu

Alpha kanál - určuje průhlednost

Global illumination - nepřímé nasvětlení 3D scény

Lineární interpolace - metoda prokládání křivek (hodnot)

Vertex color - barva (parametry RGB + alpha kanál) s hodnotami určenými každému vertexu geometrické sítě

Mesh, neboli síť - triangulovaný model, specifikum herních enginů

Triangulace - proces konverze polygonů na trojuhelníky

Ray-bouncing - odrazy paprsků

Materiál - Soubor vlastností povrchu objektu

Post-processing - úprava obrazu prováděna jako poslední fáze renderingu

Render farma - skupina počítačů využívaná pro offline rendering

Asset - "aktivum", jakákoliv věc vyrobena za účelem užití v produkci

Level of detail - úroveň detailu.

Dynamický zdroj světla - zdroj světla, který mění své vlastnosti v realném čase

Turbo Pascal - programovací jazyk

Modding - akt modifikace počítačové hry

Skeletální animace - animace za užití skeletálního rigu - kostry

FPS (frames per second) - snímky za sekundu, frekvence snímků na obrazovce

FPS (first person shooter) - žánr počítačových her - střílečka z pohledu první osoby

Level editor - program, umožňujíc editaci, či výrobu herních levelů

Light functions - specifické funkce pro zdroje světla

HDR - high dynamic range - způsob zobrazení barev

Example mapy - scény s příklady obsahu herního enginu

UnrealScript - skriptovací jazyk

Kismet - vizuální skriptovací jazyk

Blueprint - vizuální skriptovací jazyk

C++ - programovací jazyk

GPU – Graphics processing unit- grafický procesor

CPU – Central processing unit - centrální procesor

Draw call - forma komunikace mezi GPU a CPU

Baking - proces vytváření textur

Teselace - subdivize (podrozdělení) roviny aby vzniklo více geometrických tvarů

Occlusion culling - utrácení okluze - utracení, či zrušení schovaných (okludovaných) objektů

Visibility culling - utrácení objektů mimo zorné pole kamery

Draw distance - vzdálenost vykreslení

Tech demo - krátká demonstrace, při které se předvádí nové techniky

Clusters - skupiny trojuhelníků

Screen-space - postprocessingové kalkulace, pro jejichž vstup slouží data z prostoru obrazovky

Signed distance fields - zjednodušené modely 3D scén

10. BIBLIOGRAFIE

ArtStation is Joining the Epic Games Family, 2021. [Online] ArtStation [cit. 19-04-2022]. Dostupné z: https://magazine.artstation.com/2021/04/artstation-is-joining-the-epic-games-family/.

Bourke, Paul. *Converting to/from cubemaps*, 2003. [Online]. [cit. 19-04-2022]..Dostupné z: http://paulbourke.net/panorama/cubemaps/.

Conditional Zero. *Absolute Beginner's guide to Unreal Engine 5 Nanite*, 2021. [online]. [cit. 18-04-2022]. Dostupné z: https://youtu.be/9NgDle3J9lo

Game Development Engines for Mobile Platforms: Unreal Development Kit [Online] DataArt. 2014. [cit. 18-04-2022]. Dostupné z: https://www.dataart.com/en/blog/game-development-engines-for-mobile-platforms-unreal-development-kit.

Dunlop, Jake. *Trippy Archviz interior with lumen* 2021 [Online] ArtStation. [cit. 19-04-2022] Dostupné z: https://www.artstation.com/artwork/YekQ3Y.

Edwards, Benj. *From The Past To The Future: Tim Sweeney Talks* 2009 [Online] Game Developer. - Informa PLC UK Limited. [cit. 19-04-2022]. Dostupné z: https://www.gamedeveloper.com/design/from-the-past-to-the-future-tim-sweeney-talks.

Getting Started with MetaHumans in Unreal Engine | MetaHuman Creator Documentation, 2021 [Online] Epic Games, Inc. [cit. 19-04-2022]. Dostupné z: https://docs.metahuman.unrealengine.com/en-US/MetahumansUnrealEngine/GettingStarted/.

Global Illumination | Unreal Engine Documentation, 2021[Online] Epic Games, Inc. [cit. 19-04-2022]. Dostupné z: https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/GlobalIllumination.

Lumen Global Illumination and Reflections 2022 [Online] Epic Games, Inc. [cit. 19-04-2022]. Dostupné z: https://docs.unrealengine.com/5.0/en-US/lumen-global-illumination-and-reflections-in-unreal-engine/.

Nanite Virtualized Geometry in Unreal Engine | Unreal Engine Documentation 2022, [Online]. Epic Games, Inc. [cit. 19-04-2022] Dostupné z: https://docs.unrealengine.com/5.0/en-US/nanite-virtualized-geometry-in-unreal-engine/.

Real-time round-up 2022: the metaverse and emerging trends - Unreal Engine 2022, [Online]. Epic Games, Inc. [cit. 19-04-2022] Dostupné z: https://www.unrealengine.com/en-US/blog/real-time-round-up-the-metaverse-and-emerging-2022-trends.

Site Map for Unreal Engine 2 Documentation, 2005. [Online]. Epic Games, Inc. [cit. 19-04-2022] Dostupné z: https://docs.unrealengine.com/udk/Two/SiteMap.html.

*Unreal Engine 3 Documentation***,** 2009. [Online]. Epic Games, Inc. [cit. 19-04-2022] Dostupné z: https://docs.unrealengine.com/udk/Three/WebHome.html.

*Visibility and Occlusion Culling***,** 2021. [Online], Epic Games, Inc. [cit. 19-04-2022]. Dostupné z: https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/VisibilityCulling/.

Fredy, Roy. *Unreal Engine 5 Archviz Exterior on Behance,* 2021. [Online] Behance. [cit. 19-04-2022] Dostupné z: https://www.behance.net/gallery/121582649/Unreal-Engine-5-Archviz-Exterior.

Gebel, Meira a Bilbert, Ben. *The life and rise of Tim Sweeney, the billionaire CEO behind 'Fortnite' who's now taking on Apple in a lawsuit that could have huge implications for the whole industry,* 2020. [Online] Business Insider [cit. 19-04-2022] Dostupné z: https://www.businessinsider.com/fortnite-maker-epic-games-ceo-tim-sweeney-history-timeline-2019-10. ISSN - 2225-2592.

Haar, Ulrich a Aaltonen, Sebastian. *SIGGRAPH 2015: Advances in Real-Time Rendering in Games,* 2015. [Referát]. Los Angeles, California.

He, Chaoyang a Li, Ming. *OpenGL Rendering Pipeline,* 2019. [Online] ResearchGate GmbH [cit. 19-04-2022] Dostupné z: https://www.researchgate.net/figure/OpenGL-Rendering-Pipeline_fig2_334129575.

The Virtual Production of The Mandalorian Season One, 2020. [Online] Industrial Light & Magic [cit. 19-04-2022] Dostupné z: https://youtu.be/gUnxzVOs3rk.

Unreal Tournament 2003 Map Editor Lessons Videos: Free Download, Borrow, and Streaming : Internet Archive , 2019. [Online] Internet Archive [cit. 19-04-2022] Dostupné z: https://archive.org/details/07StaticMeshesOuter.

Kadner, Noah a Epic Games Inc. *The Virtual Production Field Guide, Volume 1,* 2019. [Dokument].

Lebrov, Andrey. *RAY TRACING and other RENDERING METHODS,* 2018. [Online] YouTube. [cit. 19-04-2022] Dostupné z: https://youtu.be/LAsnQoBUG4Q

The Making of Star Wars - Attack of the Clones, 2002. LucasFilm.

McDermott, Wes. *The PBR Guide: A Handbook for Physically Based Rendering,* 2018 [Kniha]. - [místo neznámé] : Allegorithmic. - 978-2490071005.

The main advantages of real time engines vs offline rendering in architecture, 2020. [Online] Oneiros. [cit. 19-04-2022]. Dostupné z: https://oneirosvr.com/real-time-rendering-vs-offline-rendering-in-architecture/.

Quixel joins forces with Epic Games, 2019. [Online] Quixel - Epic Games, Inc. [cit. 19-04-2022] Dostupné z: https://quixel.com/blog/2019/11/12/quixel-joins-forces-with-epic-games.

Scionti, Pasquale. *ArtStation - Archviz Actorcore Reallusion Unreal Engine 5 Lumen,* 2021. [Online] ArtStation. [cit. 19-04-2022] Dostupné z: https://www.artstation.com/artwork/kDYLr2.

Scionti, Pasquale. *My new archviz photoreal scene Unreal Engine 5 Lumen and Nanite,* 2021. [Online] ArtStation [cit. 19-04-2022] Dostupné z: https://www.artstation.com/artwork/xJE2a2.

Sedláček, Vojtěch. *Vycpané fretky střídají expozice plné života. Do českých muzeí si razí cestu virtuální a rozšířená realita,* 2022. [Online] CzechCrunch. [cit. 19-04-2022] Dostupné z: https://cc.cz/vycpane-fretky-stridaji-expozice-plne-zivota-do-ceskych-muzei-si-razi-cestu-virtualni-a-rozsirena-realita/.

Thomsen, Mike a Sweeney, Tim. *History of the Unreal Engine; an interview with Tim Sweney,* 2010. [Online] IGN. [cit. 19-04-2022] Dostupné z: https://www.ign.com/articles/2010/02/23/history-of-the-unreal-engine.

A Year in Review w/ TIm Sweeney & Joe Kreiner | News | Unreal Engine Livestream, 2017. [Online] Unreal Engine [cit. 19-04-2022] Dostupné z: https://youtu.be/0xjJ5Chio7o.

Lumen in UE5: Let there be light! | Unreal Engine, 2021. [Online] Unreal Engine [cit. 19-04-2022] Dostupné z: https://youtu.be/Dc1PPYl2uxA

Unreal Engine 5 Revealed! | Next-Gen Real-Time Demo Running on PlayStation 5, 2020. [Online]. [cit. 19-04-2022] Dostupné z: https://youtu.be/qC5KtatMcUw

de Vries, Joey. *LearnOpenGL - Normal Mapping*, 2017. [Online] Learn OpenGL. [cit. 19-04-2022] Dostupné z: https://learnopengl.com/Advanced-Lighting/Normal-Mapping.

Wilson, Joe. *"EarthQuake" PBR Texture Conversion | Marmoset*, 2014. [Online] Marmoset. [cit. 19-04-2022] Dostupné z: https://marmoset.co/posts/pbr-texture-conversion/.

Yilmaz, Orhan. *Unreal Engine 5 Archviz Lumen Test*, 2021. [Online] ArtStation. [cit. 19-04-2022] Dostupné z: https://www.artstation.com/artwork/9m90WL.

YT a Franczak, Michal. *Testing Unreal Engine 5 Archviz Lumen* , 2021. [Online] Evermotion. Dostupné z: https://evermotion.org/tutorials/show/12485/testing-unreal-engine-5-archviz-lumen.

Zuza, Mikolas. *Fotogrammetrie - 3D skenování s použitím fotoaparátu či mobilu - Prusa Printers*, 2018. [Online] PRUSA RESEARCH by JOSEF PRUSA. - Prusa Research [cit. 19-04-2022] Dostupné z: https://blog.prusa3d.com/cs/fotogrammetrie-3d-skenovani-s-pouzitim-fotoaparatu-ci-mobilu_7811/.

11. OBRÁZKOVÉ REFERENCE

Obr. 1 – Osový kříž souřadnic.[foto] Honal, Šimon Kryštof, 2022.

Obr. 2 – Vertexy, faces a edges krychle. [foto] Honal, Šimon Kryštof, 2022.

Obr. 3 – Flowchart rasterizace. [foto] Dostupné z: https://www.researchgate.net/figure/OpenGL-Rendering-Pipeline_fig2_334129575

Obr. 4 – Vyobrazení 3D objektu před rasterizací. [foto] Honal, Šimon Kryštof, 2022.

Obr. 5 – Rasterizátor mapuje vertexy na fragmenty. [foto] Honal, Šimon Kryštof, 2022.

Obr. 6 – Rasterizátor namapoval fragmenty k pixelům. [foto] Honal, Šimon Kryštof, 2022.

Obr. 7 – Screenshot z počítačové hry *DOOM II*. [foto] Dostupné z: https://www.nintendolife.com/games/switch-eshop/doom_ii/screenshots

Obr. 8 – Znázornění principu ray-castingu. [foto] Honal, Šimon Kryštof, 2022.

Obr. 9 – Znázornění ray-tracingu a odrážení paprsků. [foto] Honal, Šimon Kryštof, 2022.

Obr. 10 – Materiálový editor programu Unreal Engine. [foto] Honal, Šimon Kryštof, 2022.

Obr. 11 – Znázornění paprsků v path-tracingu. [foto] Honal, Šimon Kryštof, 2022.

Obr. 12 – Náhled programu Unreal Level Editor. [foto] Dostupné z: https://archive.org/details/07StaticMeshesOuter

Obr. 13 – Náhled programu Unreal Development Kit. [foto] Dostupné z: https://www.dataart.com/en/blog/game-development-engines-for-mobile-platforms-unreal-development-kit

Obr. 14 – Příklad SDR vs PBR. [foto] Dostupné z: https://marmoset.co/posts/pbr-texture-conversion/

Obr. 15 – Příklad využití fotogrammetrie. [foto] Dostupné z: https://blog.prusa3d.com/cs/fotogrammetrie-3d-skenovani-s-pouzitim-fotoaparatu-ci-mobilu_7811/