Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Rozšíření výukové aplikace virtuální reality o přehrávání animací trojúhelníkových sítí

Plzeň 2022 Jakub Hejman

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub HEJMAN**
Osobní číslo:	**A19B0049P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Rozšíření výukové aplikace virtuální reality o přehrávání animací trojúhelníkových sítí**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se se softwarem pro podporu výuky ve virtuální realitě vyvíjeným na KIV a s technologiemi použitými pro jeho vývoj.
2. Navrhněte způsob, jak software rozšířit o možnost automatického i uživatelem ovládaného přehrávání animací trojúhelníkových sítí. Předpokládejte, že sekvence reprezentující animace mají společnou konektivitu. Při návrhu se soustřeďte na efektivitu datové reprezentace přenášené mezi klienty a serverem.
3. Diskutujte vhodnost řešení z hlediska možnosti zaznamenávání interakce s animací během výuky ve virtuální realitě.
4. Navržené řešení implementujte do systému a ověřte jeho funkčnost. Změřte praktické výkonnostní metriky (množství přenášených dat, doba nutná pro přenos atp.) v běžně očekávatelných podmínkách.
5. Řešení důkladně otestujte a zdokumentujte.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Doc. Ing. Libor Váša, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 23. června 2022

Jakub Hejman

Abstract

The goal of this bachelor's thesis is to propose and implement additional functionality as a part of an existing educational application of Virtual Reality VRClassroom which enables playback of three-dimensional triangle mesh animations. These animations are defined as sequences of triangle meshes with shared connectivity and as such must be compressed, which is a large part of this work. First, some necessary principles are introduced as well as the context of this work. Next a solution is proposed and implemented and its results are examined from the perspective of performance. Finally the functionality is described from a user's point of view and some ideas for improvements are described.

Abstrakt

Cílem této bakalářské práce je navrhnout a implementovat systém, který bude součástí aplikace pro výuku ve Virtuální Realitě VRClassroom a umožní přehrávání trojrozměrných animací trojúhelníkových sítí při výuce. Animace jsou zadané posloupnostmi trojúhelníkových sítí se společnou konektivitou, tato podoba je relativně datově objemná a proto je důležitým tématem komprese těchto animací. Tato práce nejdříve uvádí základní principy a prostředí do kterého zapadá. Dále navrhne možné řešení, popíše jeho implementaci a také vyhodnotí jeho výsledky z pohledu výkonu. Nakonec je popsáno řešení z uživatelského pohledu a jsou navrženy možnosti vylepšení.

Poděkování

Rád bych poděkoval vedoucímu této práce Doc. Ing. Liboru Vášovi Ph.D. za zkušené vedení v průběhu této práce, příležitost pracovat na tomto projektu a za všechen čas věnovaný konzultacím. Děkuji též mé rodině za podporu v mém studiu.

Obsah

1 Úvod

Zadáním této práce je rozšíření již existující aplikace, která je vyvíjena za účelem výuky ve Virtuální Realitě na Fakultě aplikovaných věd ZČU. Projekt se nazývá VRClassroom a zaměřuje se na výuku konceptů trojrozměrné geometrie a základů polygonálních sítí v předmětu KIV/ZPOS, určeném pro studijní programy se specializacemi Medicínská informatika a Počítačová grafika.

Použití této trojrozměrné aplikace a zvláště ve virtuální realitě má za výhody lepší a intuitivnější vnímání trojrozměrných konceptů, když jsou tímto způsobem vizualizovány. Jedny ze základních funkcionalit aplikace je šíření a synchronizace zobrazení trojrozměrných modelů pro všechny klienty a kreslení křivek (a také rovin) v prostoru. Toto pomáhá studentům získat představu o tom, jak vyučovaná látka funguje v interaktivním prostředí a poskytuje vyučujícímu vhodné nástroje pro výuku.

Existují koncepty, pro jejichž vysvětlení by bylo vhodné zároveň reprezentovat nějakou změnu v čase, nebo přechod a proto je vhodné zobrazení animace, která toto umožňuje zachytit. Jedním příkladem je takzvaný UV unwrapping — proces „rozbalení“ polygonové sítě do texturových souřadnic, které poskytují informaci o tom, které body polygonové sítě náleží kterému bodu textury. V tomto případě lze vizualizovat rozbalení modelu např. krychle nebo koule do roviny, která reprezentuje texturu.

Cílem této práce je tedy zavést systém přehrávání animací trojúhelníkových sítí do aplikace VRClassroom a to s podobnými výhodami a vlastnostmi, jaké poskytuje již existující systém pro polygonální sítě. Jde o možnost rozeslání animace klientům přes počítačovou síť v průběhu výuky a uživatelem řízené přehrávání animace, které je synchronizované mezi klienty, aby bylo možné na základě animace vést výuku.

Následující kapitola se věnuje prostředí, do kterého je práce zasazena a jejím prerekvizitám. Dále jsou popsány kroky komprese a navržen její postup. V další části jsou experimentálně otestovány varianty komprese a poté je komprese celkově vyhodnocena. Následuje kapitola věnující se implementaci. Předposlední kapitola popisuje řešení z pohledu uživatele. Nakonec navrhuji způsoby, jak lze na tuto práci navázat a zlepšit její výsledky.

2 Výchozí projekt

2.1 VRClassroom

VRClassroom je aplikace postavená v herním enginu Unity, která poskytuje nástroje pro výuku ve virtuální místnosti a zaměřuje se na prezentaci ve Virtuální Realitě, aby uživatel měl lepší představu o trojrozměrném prostoru důležitou pro výuku konceptů trojrozměrné geometrie.

Aplikace sestává z klientů a serveru. Klienti jsou podporováni na více platformách, zejména operační systémy Windows pro osobní počítače a také Android zaměřený na VR Headset Meta Quest, dříve označovaný jako Oculus Quest, ale nově byla přidána podpora pro další platformy Virtuální Reality - Steam VR [9]. Server běží na stolním počítači, očekává se operační systém Windows, popř. Linux. Protože je aplikace vyvíjena v prostředí Unity, je možné ji zprovoznit na dalších platformách.

Klienti se mohou připojit v roli studenta a nebo učitele, očekává se pouze jeden učitel a pro přihlášení je od něj vyžadováno heslo. Podle toho, jestli jde o studenta nebo učitele, má klient dostupné jiné funkcionality, učitelé mohou navíc například nahrávat polygonové sítě a prezentace pro zobrazení všem klientům.

V tuto chvíli poskytuje řadu funkcionalit včetně zobrazování polygonálních sítí, kreslení v prostoru, ukazování ukazovátkem, signalizace studentů vyučujícímu a systém záznamů pro umožnění pozdějšího přehrání přednášky offline [9].

2.2 Unity

Unity je vývojové prostředí s hlavním zaměřením na vývoj počítačových her, ale zároveň je vhodné pro obecnější 3D real-time aplikace a vizualizace např. architektury. Podporuje sestavení na široké spektrum cílových platforem včetně dříve zmíněných Windows, Androidu a Linuxu.

Prostředí využívá komponentovou architekturu pro sestavování objektů a scén a podporuje vytváření skriptů jako komponent objektů v programovacím jazyce C# s bohatým API.

2.3 Trojrozměrné trojúhelníkové sítě

Nejčastěji jsou trojrozměrné modely či objekty v počítačové grafice reprezentovány polygonovými sítěmi. Jde o množiny vrcholů, hran a ploch. Obvykle se ale tato reprezentace zjednodušuje, zcela se vypouští hrany, protože samy o sobě nejsou zobrazitelné a je tedy úspornější rovnou pracovat s plochami, které by mohly být zcela libovolné polygony, ale pro zrychlení renderování se nakonec pracuje nejčastěji jenom s trojúhelníky. Polygonální sítě je možné převést na trojúhelníkové sítě rozdělením mnohoúhelníků na více trojúhelníků.

2.3.1 Reprezentace trojúhelníkových sítí

Trojúhelníkové sítě jsou v počítačové grafice standardně reprezentovány primárně polem vrcholů a polem indexů trojúhelníků, ale také mohou obsahovat informace o normálách, texturových souřadnicích, deformačních vahách (bone weights) apod. [14], tyto další vlastnosti nejsou pro účely této práce nutně potřebné. Buď nejsou vůbec potřeba pro zobrazení sítě, nebo je lze dopočítat jako v případě normál, protože je definováno, že vrcholy trojúhelníku jsou ukládány proti směru hodinových ručiček, což určuje orientaci trojúhelníku. Specificky pole vrcholů uchovává polohy vrcholů v podobě struktur trojrozměrných vektorů reálných čtyřbytových čísel `float`. Pole trojúhelníků sestává z celočíselných čtyřbytových hodnot `int`, kde tři po sobě jdoucí hodnoty reprezentují trojúhelník a udávají indexy vrcholů, ze kterých sestává. V Unity se vrcholy indexují od nuly [14], zatímco v souborech Wavefront `.obj` se počítají počínaje jedničkou [2].

2.3.2 Animace

Existuje více přístupů k trojrozměrným animacím. Výsledně zobrazované trojúhelníkové sítě mohou být dynamicky generované, typicky jako výsledek fyzikální simulace. Druhým základním přístupem je rozpohybování modelu podle armatury. Pohyb armatury lze animovat ručně, generovat proceduralně a nebo použít techniku Motion Capture. Armatura zastupuje kostru animovaného objektu, takže je toto velmi vhodný přístup k animování hlavně živých subjektů, protože blízce odpovídá jejich skutečné fyzikální podobě.

Prezentovaný systém bude podle zadání pracovat s animací reprezentovanou posloupností trojúhelníkových sítí, takže bude kompatibilní s libovolným způsobem tvorby animace, protože animace musí být před použitím exportována do této podoby. Posloupnosti trojúhelníkových sítí také musí mít vlastnosti popsané v podsekci 3.1.1.

2.3.3 Analýza hlavních komponent

Analýza hlavních komponent angl. Principal Component Analysis (PCA) je metoda, která se používá pro nalezení vhodnější báze k vyjádření dat [10]. Metoda se zakládá na rozkladu na vlastní čísla a vlastní vektory autokorelační matice[13]. Výsledná báze sestávající z vlastních vektorů je dekorelovaná a ortonormální. Navíc velikost vlastních čísel nám říká, jak významné jsou příslušné vlastní vektory pro vyjádření dat.

Metodu aplikujeme na data animace, která vyjádříme maticí o velikosti dimenze x počet vzorků, což je dáno vlastnostmi animace (počet vrcholů a snímků alespoň v našem případě). Ke kompresi využijeme toho, že z těchto vektorů vybereme pouze určité množství nejvýznamnějších a použijeme je jako bázi prostoru. Výsledkem je ztrátová komprese, která pro daný počet bázových vektorů použije ty, které jsou nejvýznamnější a nesou tedy nejvíce informace. Jde o formu redukce dimenzionality, tedy sníží počet přenášených hodnot.

Autokorelační matici získáme tak, že napřed poskládáme matici $\mathbf{S}'$ z dat animace tak, že všechny pozice vrcholů animace budou reprezentovány sloupci, ve kterých budou postupně skládány složky pozice tohoto vrcholu napříč snímky animace. V následujícím zápisu x_i^j značí složku x j-tého vrcholu ve snímku i.

$$\mathbf{S}' = \begin{pmatrix} x_1^1 & x_1^2 & \dots & x_1^m \\ y_1^1 & y_1^2 & & y_1^m \\ z_1^1 & z_1^2 & & z_1^m \\ x_2^1 & x_2^2 & & \vdots \\ y_2^1 & y_2^2 & & \\ z_2^1 & z_2^2 & & \\ \vdots & & \ddots & \\ x_n^1 & x_n^2 & & x_n^m \\ y_n^1 & y_n^2 & & y_n^m \\ z_n^1 & z_n^2 & \dots & z_n^m \end{pmatrix}$$

Následně od každé dimenze (řádku) matice $\mathbf{S}'$ odečteme její průměr ve statistickém smyslu střední hodnoty, tuto matici označíme $\mathbf{S}$. Autokorelační matice $\mathbf{A} = \mathbf{S} * \mathbf{S}^T$. Tato matice je tedy symetrická, což je důležité pro singulární rozklad.

Nyní můžeme provést singulární rozklad této matice, tedy $\mathbf{A} = \mathbf{U\Sigma V}$. Sloupce matice $\mathbf{U}$ jsou naše nové bázové vektory, matice $\mathbf{\Sigma}$ je diagonální s vlastními čísly na diagonále. Použitá knihovna řadí bázové vektory vzestupně podle velikosti jejich příslušných vlastních čísel a vlastní čísla jsou

přístupná jako vektor.

Velikost vlastních čísel určuje, jak významné jsou vlastní vektory, takže víme, že sloupce jsou zprava nejvýznamnější. Nyní můžeme z matice U vybrat daný počet bázových vektorů, které budou nejvýznamnější a snížit tím rozměr dat ($\mathbf{U}'$). Nyní můžeme animaci vyjádřit v nové bázi $\mathbf{T} = \mathbf{U}'^{T}\mathbf{S}$.

Pro dekompresi je potřeba uchovat matice $\mathbf{U}'$ a $\mathbf{T}$ spolu s průměry. Při dekompresi vyjádříme animaci znovu v kanonické bázi $\mathbf{S} = \mathbf{U}'\mathbf{T}$, k hodnotám řádků matice $\mathbf{S}$ přičteme průměry a získáme rekonstrukci původní matice $\mathbf{S}'$ (pokud jsme nezkrátili bázi a neprovedli kvantizaci, tak je tato matice rovná té původní). Nakonec z této matice $\mathbf{S}'$ obnovíme pozice vrcholů a proces je u konce.

Velkou výhodou tohoto přístupu je, že dekomprese je výrazně rychlejší, než komprese. Také tato metoda může dosahovat výborných kompresních poměrů pokud výrazně snížíme velikost báze. Nevýhodou je, že pro velké animace je krok singulárního rozkladu náročný, takže v tomto případě je komprese zdlouhavá.

Při sestavování matice $\mathbf{S}'$ jsme reprezentovali trajektorie vrcholů sloupci matice, tato varianta PCA se tedy odehrává v prostoru trajektorií a je to výhodnější varianta, než kdybychom definovali matici $\mathbf{S}$ jako $\mathbf{S^T}$, což by odpovídalo prostoru tvarů/geometrií. Velikost matice $\mathbf{A}$ je vázaná na počet snímků animace, takže místo počtu vrcholů animace uvažujeme počet snímků animace, což je typicky mnohem nižší číslo.

Tato metoda je součástí velkého množství state-of-the-art algoritmů pro kompresi animací trojúhelníkových sítí, typicky je dále kombinována s postupy, které maximálně využijí podobu a vlastnosti vstupních dat [18].

3 Návrh komprese

Desktopový klient VRClassroom bude umožňovat připravení komprimovaných animací do souborů, které bude po připojení k serveru s rolí učitele možné v reálném čase nahrát na server pro zobrazení všem uživatelům. Následně bude možné animaci přehrávat, běh animace bude možné ovládat a stav přehrávání animace bude synchronizován mezi klienty VRClassroom aby uživatelé mohli o animaci věcně hovořit.

3.1 Data a komprese

Protože animace mohou být relativně datově objemné (např. animace z testovací sady „Walk“ má v původní podobě 443MB, nebo cca. 80MB v úsporné binární podobě), je vhodné a časově výhodné co nejvíce komprimovat animaci před odesláním. Čas komprimace není kritický, zatímco čas dekomprese musí být dostatečně nízký, aby se komprese za běžných podmínek vyplatila. Je důležité si uvědomit, že animace je přijímána a dekomprimována klienty VRClassroom v průběhu výuky a měla by mít co nejmenší dopad na kvalitu a plynulost přednášky. Tématem komprimace animace se bude zabývat většina této práce, zvláště její rychlostí a efektivitou.

Za některých okolností je vhodnější jít směrem k lepšímu kompresnímu poměru i za cenu nevýhodné doby dekomprese, protože síťový přenos není jediná situace, kdy se vyplatí menší soubor.

Data animace projdou několika přípravnými kroky, jejichž cílem je minimalizovat redundnanci v datech a nakonec využít standardní kompresní algoritmus, jehož výstupem bude co nejmenší datový proud, ze kterého půjde zpětně sestavit daná animace. Přenášená data budou popisovat vrcholy - neboli geometrii a trojúhelníky - neboli konektivitu všech snímků animace. Nebudou se zahrnovat normály, ty mohou být dopočítány (je to součástí rozhraní trojúhelníkových sítí v Unity [14]) ani texturovací souřadnice a další, protože nejsou potřeba k zobrazení animace.

3.1.1 Vlastnosti

Zadaná posloupnost trojúhelníkových sítí, která reprezentuje animaci, má několik klíčových vlastností, které jsou využity pro kompresi a jsou z tohoto důvodu nutné pro funkcionalitu navrženého řešení.

- Počet vrcholů a trojúhelníků je mezi snímky animace neměnný.

- Konektivita je konstantní mezi snímky animace - odpovídající vrcholy jsou vždy součástmi trojúhelníků se stejnými ostatními vrcholy.
- Stejný vrchol je pro jeho identifikaci vždy uložen na stejné pozici, resp. vrcholy jsou uloženy ve stejném pořadí.

Jedním častým omezením na komprimované animace je, aby byly manifoldy, ale přístupy ke kompresi, které prezentuje tato práce tento požadavek nemají. Dvourozměrný manifold, neboli geometrická plocha, je reprezentován trojúhelníkovou sítí, pokud sestává pouze z tzv. manifoldních hran, které jsou součástí maximálně dvou trojúhelníků. Všechny trojúhelníky, jejichž součástí jej tentýž vrchol tvoří buď uzavřený, nebo otevřený vějíř [3].

3.1.2 Přenášená data

Pro úspěšné přenesení animace je zapotřebí uchovat konektivitu - ve smyslu samotných trojúhelníků, stačí ji v komprimované animaci zahrnout jednou, protože se nemění a bude považována za součást prvního snímku animace, se kterým se bude zacházet jinak, než s ostatními. Také je potřeba obsáhnout informaci o poloze každého vrcholu animace pro každý snímek.

Dále budou popsány operace, které ztrátově a bezeztrátově zjednoduší přenášenou informaci, potřebují dále uchovat určité parametry se kterými budou zkomprimovány pro jejich inverzní operace, které proběhnou při dekompresi. Pro datovou integritu je dále potřeba identifikovat počet vrcholů a trojúhelníků ve snímku a počet snímků.

Lze uvažovat o zahrnutí dalších užitečných informací, jako je snímková frekvence animace, nastavení či informace o kompresi jako např. verze kompresoru pro kontrolu kompatibility. Tyto informace nejsou klíčové, ale jsou možností pro budoucí rozšíření.

3.1.3 Kvantizace

Kvantizace je pojem z oboru zpracování signálů, jeho významem je převod kontinuálního analogového signálu na diskrétní hodnoty, typicky celočíselné. Velmi často se používá v kontextu vzorkování a zpracování audia. V procesu přípravy dat série trojúhelníkových sítí budou pozice vrcholů, které jsou v Unity původně reprezentovány datovými strukturami `Vector3`, které sestávají ze tří hodnot typu `float` převáděny na celá čísla v rozsahu $< 0, \text{počet úrovní})$. Tato celá čísla tedy odpovídají buňkám pomyslné kvantizační mřížky do kterých původní vrcholy spadají.

Tím dojde k snížení počtu různých hodnot na relativně malý rozsah, tedy snížení entropie systému a umožní to kompresnímu algoritmu dosáhnout menší velikosti zkomprimované animace. Tento proces není potřeba provádět s trojúhelníky, protože již jde o integery v relativně malém rozsahu $(1, \text{počet vrcholů snímku} >$.

Kvantizace pozic vrcholů je řízena velikostí buňky, nebo maximálním rozlišením, mezi kterými platí převod:

$$velikost\_buňky = \frac{max.\_šířka\_bounding\_boxu}{max.\_rozlišení}$$

Cílem kvantizace je převést reálné hodnoty pozic vrcholů na celá čísla v malém rozsahu. Tato celá čísla lze lépe komprimovat, protože mají nižší entropii. Vztah použitý pro převod je přímočarý:

$$kvant.\_hodnota = round\_to\_int(\frac{pozice\_vrcholu - min.\_bounding\_boxu}{velikost\_buňky})$$

A vztah pro odpovídající rekonstrukci:

$$rek.\_hodnota = (kvant.\_hodnota + 0.5) * velikost\_buňky + min.\_b.\_boxu$$

Kvantizace probíhá stejně pro obecný vrchol a je tedy identická mezi prvním snímkem animace a ostatními snímky.

Pro kvantizaci hodnot pro metodu komprese využívající PCA je použit jiný přístup, místo hledání maxima a minima jsem se rozhodl použít kvantizaci na pevný počet desetinných míst, který je daný následujícím vztahem:

$$kvantizovaná\_hodnota = round\_to\_int(hodnota * rozlišení)$$

Rekonstrukce této kvantizace:

$$rekonstr.\_hodnota = \frac{kvantizovaná\_hodnota}{rozlišení}$$

Např. pro $rozlišení = 100$ tato metoda kvantizace odpovídá fixed-point reprezentaci čísla na dvě desetinná místa. Bylo to zvoleno proto, že komprimované hodnoty v PCA metodě mají jiný charakter, než v první metodě.

3.1.4 Komprese prvního snímku

Jako součást prvního snímku jsou zahrnuty parametry kvantizace, které jsou potřebné pro rekonstrukci kvantizovaných hodnot. Vrcholy nyní můžeme kvantizovat, ale je vhodné jejich data nějakým způsobem redukovat za použití vhodné predikce, abychom snížili celkovou entropii dat a umožnili kompresnímu algoritmu dosáhnout lepšího kompresního poměru.

Často používaným způsobem je využít konektivity snímku a očekávat, že dvojice sousedních trojúhelníků bude často mít tvar rovnoběžníku a tedy bychom mohli z jednoho počátečního trojúhelníku postupně odhadovat další a další trojúhelníky, až ve finále pokryjeme celou síť. Tento postup by měl pravděpodobně velmi dobré výsledky, ale není příliš jednoduchý pro implementaci od začátku a pro sítě, které nemají vhodné vlastnosti (nejsou manifoldy) [5].

Výrazně jednodušší je predikovat pozici vrcholu pozicí předchozího vrcholu. V tomto případě je optimálním řešením sestavit nejkratší možnou cestu, který jde přes každý vrchol sítě. Poté bychom šli po této cestě a každý vrchol predikovali tím předchozím tak, že bychom je odečítali a v důsledku získali pouze vektory mezi nimi, místo absolutních pozic všech vrcholů.

Protože očekávaný dopad na kompresní poměr finálního algoritmu není až tak velký, převážně proto, že úspora bude pouze v prvním snímku, tento problém nebude řešen. Místo toho budou vrcholy predikovány v pořadí, ve kterém se nachází ve vstupních souborech. V případě, že bychom přesto chtěli dosáhnout lepšího výsledku pro odečítaní pozic vrcholů, tak stačí vrcholy předem seřadit. Zároveň pro typické `.obj` soubory dojde k redukci entropie, protože jsou při exportu z 3D aplikací často vrcholy uloženy v takovém pořadí, že po sobě jdoucí vrcholy jsou blízko (např. po řádkách).

Na konec dat prvního snímku zahrneme jeho konektivitu ve formě trojúhelníků, které již nebudeme dále zpracovávat.

3.1.5 Komprese ostatních snímků

V této části komprese dojde k největší úspoře oproti původním datům, protože neopakujeme konektivitu snímků, která typicky sestává z více hodnot než pozice vrcholů samotné. Zároveň můžeme experimentovat s několika jednoduchými způsoby predikce, které v důsledku sice nedosáhnou nejlepších kompresních poměrů, ale dosahují velmi rychlé dekomprese zvláště pro menší animace. Predikce používá k odhadu pozice vrcholu informace o předchozích pozicích vrcholu z předcházejících snímků.

3.1.6 Predikce

Komprese za využití predikce vychází z předpovědi o budoucích datech vycházející z dat dostupných. Pokud je predikce úspěšná, tak není potřeba přenášet další informaci, nebo alespoň přenášíme informaci výrazně menší. Tato přenášená odchylka predikce od skutečné hodnoty se nazývá korekce. V této práci budou využity velmi jednoduché a základní formy predikce,

vycházející z očekávání, že nějaká vlastnost (pozice vrcholů, změna pozice mezi snímky) je typicky neměnná. Často bude jedna hodnota predikována předchozí hodnotou.

Tuto techniku bude možné aplikovat jiným způsobem na vrcholy prvního snímku a na vrcholy navazujících snímků. V případně první trojúhelníkové sítě bude predikce vycházet z ostatních vrcholů v téže síti, zatímco u navazujících budou pro předpověď budoucích poloh vrcholů použito odpovídajících vrcholů předcházejících snímků. Pro případ predikce vrcholů navazujících snímků budou představeny tři varianty predikce.

Lze uvažovat použití komplexnějších predikčních funkcí, které berou v úvahu více snímků a nebo více vrcholů a systém bude postaven tak, aby umožnil použití alternativních metod predikce bez rozsáhlého zásahu do kódu řešení.

Začneme definicí tří testovaných predikcí jako P_A, P_B a P_C. Pozice vrcholu s indexem i ve snímku s indexem j bude značena v_i^j a její rekonstrukce bude značena $\overline{v_i^j}$. Predikci vrcholu s indexem i ve snímku s indexem j označím $P_X(i,j)$ a k ní příslušnou korekci $c_X(i,j)$. Platí tedy, že:

$$c_X(i,j) = v_i^j - P_X(i,j)$$

a

$$\overline{v_i^j} = r(q(c_X(i,j))) + P_X(i,j)$$

Prakticky bude korekce $c_X(i,j)$ vypočítávána při kompresi a přenášena, zatímco rekonstrukce pozice vrcholu $\overline{v_i^j}$ bude vypočtena při dekompresi a použita pro zobrazení animace. Funkce $q()$ reprezentuje kvantizaci hodnot a $r()$ reprezentuje funkci rekonstrukce. Predikce budou definovány následovně:

$$P_A(i,j) = \overline{v_i^{j-1}}$$
$$P_B(i,j) = \overline{v_i^{j-1}} + (\overline{v_i^{j-1}} - \overline{v_i^{j-2}}) = 2 * \overline{v_i^{j-1}} - \overline{v_i^{j-2}}$$
$$P_C(i,j) = \overline{v_i^{j-1}} + ((\overline{v_i^{j-1}} - \overline{v_i^{j-2}}) + ((\overline{v_i^{j-1}} - \overline{v_i^{j-2}}) - (\overline{v_i^{j-2}} - \overline{v_i^{j-3}})))$$

P_A předpovídá, že pozice vrcholu zůstane konstantní, to znamená, že korekce budou nulové pro nehybné vrcholy a menší, než původní hodnoty v případech, kde uražená vzdálenost mezi snímky je nižší, než absolutní velikost pozice vrcholu. Toto se za běžných podmínek a zajisté ve všech testovaných animacích vyplatí vždy, viz tabulku 4.5.

P_B lze vnímat jako vyšší stupeň predikce P_A, obdobně jako ve fyzice první derivací pozice HB (Hmotného bodu) je rychlost tohoto HB. Předpokládáme, že změna pozice, neboli pohyb mezi dvěma po sobě jdoucími

snímky animace bude konstantní. V důsledku bude korekce nulová pro vrcholy, které se pohybují z fyzikálního hlediska rovnoměrným přímočarým pohybem a vrcholy, které se nehýbají jsou speciálním případem tohoto. Častěji tedy zde bude korekce nulová, než za použití P_A, ale v některých případech bude větší (a jak lze vypozorovat z měření 4.2, tak toto není výhodné pro všechny kompresní algoritmy, i když entropie dat je nižší).

P_C logicky navazuje na P_B a to tak, že očekává, že změna změny pozice bude konstantní, což je analogické zrychlení z fyziky.

3.1.7 Kompresní algoritmy

V posledním kroku komprese budou data o animaci v minimalizované formě předána konvenčnímu kompresnímu algoritmu, který konečně zmenší datový objem animace. V úvahu budou brány dva algoritmy, které byly vybrány hlavně díky jejich dostupnosti, ale jsou samozřejmě vhodné pro tento účel.

Oba algoritmy jsou bezeztrátové, ztrátové kroky jsou pouze kvantizace a snížení dimenzionality metodou PCA.

Aritmetický kodér

Aritmetické kodéry jsou entropické kodéry, které rekurzivně mapují kódované znaky na části intervalu číselné osy a tím redukují původní data na jednotlivá čísla z původního intervalu. Jsou optimální pro nezávislé stejně rozdělené zdroje dat, ale jsou efektivní a skoro optimální i pro takové, která nejsou stejně distribuované. [12] Zaměřením aritmetického kodéru je tedy minimální velikost komprimovaných dat a alespoň použitá implementace, která byla poskytnuta vedoucím práce, má časy komprese a dekomprese vážené spíše k dekompresi (čas dekomprese je relativně k času komprese v porovnání s Gzipem větší).

Gzip

Gzip je formát souboru a SW původně vyvinutý jako součást GNU [1], který používá DEFLATE kompresi, což je kombinace Lempel-Ziv LZ77 slovníkové komprese a Huffmanova kódování, což je entropické kódování. V principu LZ77 nahrazuje výskyty znaků odkazy na jejich předchozí výskyty. Následně Huffmanovo kódování nahrazuje znaky kratšími symboly podle stromu sestaveného tak, aby výsledné symboly tvořily prefixový kód, který je optimální pro dané znaky s jejich pravděpodobnostmi výskytu.

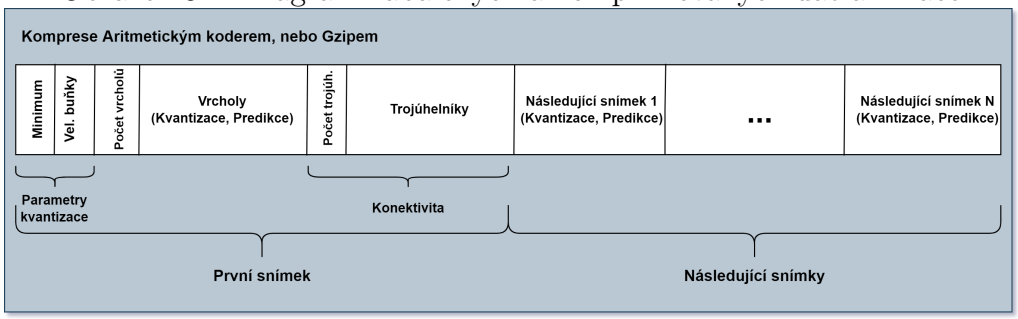
Jedním detailem, který se nějakým způsobem projeví v průběhu implementace mé komprese je, že aritmetický kodér si v komprimovaných datech musí ukládat délku komprimovaných dat, protože v důsledku jeho principu by mohl dekódovat z uložené hodnoty více či méně znaků, zatímco Gzip pracuje v použité implementaci do konce proudu dat. Aritmetickým kodérem je jednoduše a podle charakteristik dat i výhodně možné do jednoho proudu dat uložit více kódovaných bloků a velikost bloků není „navíc“ uložena mimo. Při dekódování jsou bloky přehledně dekódovány více voláními kodéru, což vede k trochu jiným detailům toho, jak jsou data strukturována.

3.1.8 Podoba komprimovaných dat

Je důležité stanovit, alespoň přehledově, strukturu komprimovaných dat. V mém návrhu jsou dvě hlavní části: první snímek animace a následující snímky. Všechna čísla jsou před finální komprimací čtyřbytová. Pro představu je podoba komprimovaných dat znázorněna na obrázku 3.1.

První snímek je samostatný a obsahuje napřed kvantizační parametry – minimum bounding boxu a šířku kvantizační buňky. Následuje počet vrcholů a vrcholy samotné, potom počet trojúhelníků a trojúhelníky samotné. Specificky minimum je trojrozměrná reálná hodnota a šířka kvantizační buňky je jednorozměrná reálná hodnota.Počty vrcholů i trojúhelníků jsou celočíselné, stejně tak vrcholy (díky kvantizaci) i trojúhelníky (jejich hodnoty jsou relativně rovnoměrně distribuované). Ostatní snímky jsou uloženy v druhé části, snímky v této části vycházejí z prvního snímku – používají jeho konektivitu, vycházejí z jeho počtu vrcholů pro oddělení snímků od sebe a také jeho vrcholy pro predikci. Počet snímků není potřeba ukládat, protože za vrcholy nejsou žádná další data a snímky lze dopočítat z počtu vrcholů. Jsou zde všechny hodnoty celočíselné a vrcholy prošly kvantizací a predikcí založenou na předchozích snímcích.

Obrázek 3.1: Diagram zabalených a komprimovaných dat animace.



K animaci je možné přidat také další data v případě rozšiřování systému, například informaci o verzi kompresoru, se kterou byla animace komprimována pro kontrolu kompatibility, specifikaci, jakým způsobem byla komprese provedena, nebo také snímkovou frekvenci animace, kterou by uživatel mohl specifikovat při přípravě animace. Tyto konkrétní údaje, v podstatě metadat, a bych nejspíše přidal na začátek struktury, jako hlavičku před první snímek animace, ale zatím se nezdá, že by to bylo zapotřebí.

Při kompresi animace dochází k převodu reálných souřadnic vrcholů do celých čísel relativních k bounding boxu prvního snímku a při dekompresi zase k opačné operaci, která za pomoci informací o bounding boxu zapsaných do komprimovaných dat animace transformuje tato čísla zpět do reálných trojrozměrných souřadnic vrcholů. Dochází tak k implicitní translaci a škálování těchto vrcholů. Tohoto můžeme využít a pouhým nahrazením informací o skutečném bounding boxu prvního snímku za informace o nějakém vhodnějším bounding boxu můžeme dosáhnout například translace animace do počátku souřadného systému a nebo škálování, které přizpůsobí rozměry animace tak, aby byla pohodlně sledovatelná ve virtuální realitě.

3.1.9 Využití PCA

Dále bude do komprese integrována metoda Analýzy hlavních komponent. Toto je velmi odlišný a komplikovanější přístup ke kompresi a proto se v tomto případě bude měnit jak struktura dat, tak přístup k nim, ale některé výše popsané principy mohou být rovněž použity s tímto přístupem.

Obrázek 3.2: Diagram zabalených a komprimovaných dat metodou PCA.

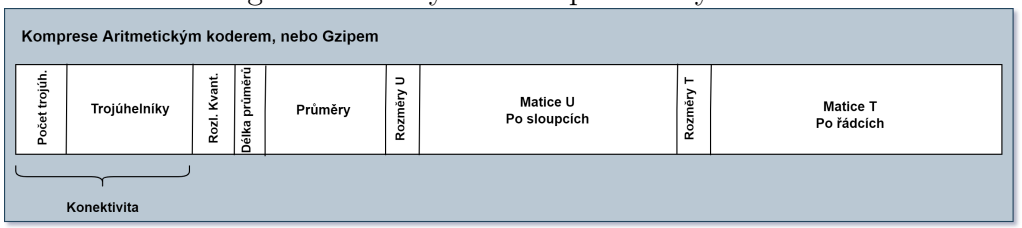


Diagram 3.2 znázorňuje, jak jsou při použití PCA komprese data strukturována. Napřed je uložena konektivita, rozlišení kvantizace, poté mediány a nakonec matice U a T. Přesný význam těchto matic je popsán v podsekci 2.3.3. Všechny hodnoty kromě konektivity jsou reálná čísla kvantizovaná se stejným rozlišením.

Hodnoty jsou v rámci bloků predikovány předchozí hodnotou, tedy hodnoty konektivity, průměrů a matice U jsou predikovány hodnotou, která jde

o tři místa před nimi, protože tato data jsou trojrozměrná. Tato predikce je obdobná predikci vrcholů prvního snímku u první varianty komprese. Matice T je predikována na stejném principu, ale vždy bezprostředně předcházející hodnotou, protože nejde o trojrozměrnou veličinu.

V kombinaci s PCA kompresí jsem se rozhodl vždy používat aritmetický kodér.

3.2 Normalizace

Normalizací v tomto kontextu myslím transformaci celé animace, popř. jejího prvního snímku tak, aby se vešla do nějakých standardních rozměrů, v mém případě krychle s protilehlými vrcholy (-1, -1, -1) a (1, 1, 1). V důsledku tedy skončí střed v počátku a rozměry zasahuje maximálně jednu jednotku do všech rozměrů. Účelem je přizpůsobit animaci k pohodlnému sledování ve Virtuální Realitě (ale i obecně), protože se většina z testovaných animací posouvá prostorem a mají různá měřítka.

Při použití přímé metody komprese jsou všechny vrcholy každého snímku při dekomprimaci transformovány následným způsobem:

$$\text{pozice} = \frac{(\text{kvantizovaná\_data} + 0.5) * \text{šířka}}{\text{rozlišení}} + \text{minimum}$$

Toto má za nevýhodu to, že to neumožňuje rotaci objektu, ale umožní to translaci a škálování. K docílení žádané transformace stačí přepočítat kvantizační parametry komprese a zapsat modifikované místo skutečných.

V případě PCA komprese nelze takto jednoduše dosáhnout výše zmíněné transformace, takže je potřeba před kompresí aplikovat žádanou transformaci na všechny snímky animace. Druhým způsobem je modifikovat lokální transformaci objektu jehož součástí je animace v Unity scéně.

V zásadě lze normalizovat animace buď podle prvního snímku, celé animace, nebo vůbec. Animaci bychom nechtěli normalizovat, pokud je přizpůsobená prostředí ve kterém bude zobrazována a normalizace by měla buď nevýznamný nebo dokonce nevítaný účinek. Normalizace podle prvního snímku má tu výhodu, že pokud animace sestává ideálně z jednoho objektu, jehož velikost se výrazně nemění v průběhu animace, tak docílíme vhodné velikosti objektu a neomezíme jeho pohyb. Normalizace podle celé animace zajistí, že úplně celá animace se vejde do krychle kolem počátku, takže tím eliminujeme případy, ve kterých by objekt „utekl“ z této krychle jako v předchozím případě. Pokud ale animace celkem zabírá výrazně větší prostor v porovnání s animovanými objekty, tak tyto objekty budou ve výsledku velmi malé.

K určení transformace vyhodnotíme původní bounding box, který chceme normalizovat a také cílový bounding box, které je vždy již dříve zmíněná krychle se začátkem/minimem (-1, -1, -1) a koncem/maximem (1, 1, 1). Původní bounding box bude buď určen bounding boxem prvního snímku, nebo v případě celé animace ho určíme následovně:

$$začátek = (\min_{i \leq n} začátek_i.x, \min_{i \leq n} začátek_i.y, \min_{i \leq n} začátek_i.z)$$

$$konec = (\max_{i \leq n} konec_i.x, \max_{i \leq n} konec_i.y, \max_{i \leq n} konec_i.z)$$

Zde $začátek_i$ značí začátek bounding boxu i-tého snímku a $konec_i$ značí konec bounding boxu i-tého snímku. Obdobně lze spočítat samotný bounding box snímku z pozic vrcholů snímku, ale to není potřeba, protože je to součástí rozhraní třídy Mesh v Unity [14].

Následně nalezneme vyžadovanou transformaci a aplikujeme ji. Protože půjde o translaci a škálování, tak napřed odečteme původní minimum, poté aplikujeme škálování faktorem $s' = \frac{1}{s}$, kde s je maximum ze složek rozměrů původního bounding boxu.

V kombinaci se systémem, který je již součástí VRClassroom a umožňuje uživatelům modifikovat transformaci zobrazovaného objektu v reálném čase by měl systém být dostatečně adaptivní pro účel sledování animace ve VR.

3.3 Samostatný snímek

Jelikož i první snímek animace je komprimovaný (i když méně efektivním způsobem oproti sofistikovanějším metodám, které se na toto zaměřují), tak je tato reprezentace velikostně stále výrazně menší, než reprezentace používaná existujícím systémem na zobrazování samostatných modelů, který posílá přes síť původní `.obj` soubor po Gzip kompresi. Bude tedy možné systém komprimace animací použít i na samostatné snímky a bude to výhodné, ale nikoliv v porovnání se systémem, který by cíleně komprimoval tento model samostatně, protože hlavní výhoda navrženého systému pochází z úspory mezi snímky.

Při použití PCA komprese se v tomto případě neočekává dobrý výsledek, protože pro samotný snímek je v důsledku potřeba přenést více hodnot.

4 Testované alternativy komprese

Až doposud bylo představeno několik částí kompresního procesu pro které bylo navrženo více alternativ:

1. různé kvality kvantizace podle rozlišení kvantizační mřížky
2. různé způsoby predikce vrcholů
3. dva kompresní algoritmy - Aritmetický coder vs. Gzip
4. využití PCA

Všechny z těchto možností jsou v této kapitole experimentálně porovnány, aby bylo zdůvodněno které možnosti jsou použity ve finální implementaci.

4.1 Vliv rozlišení kvantizace na kvalitu rekonstrukce

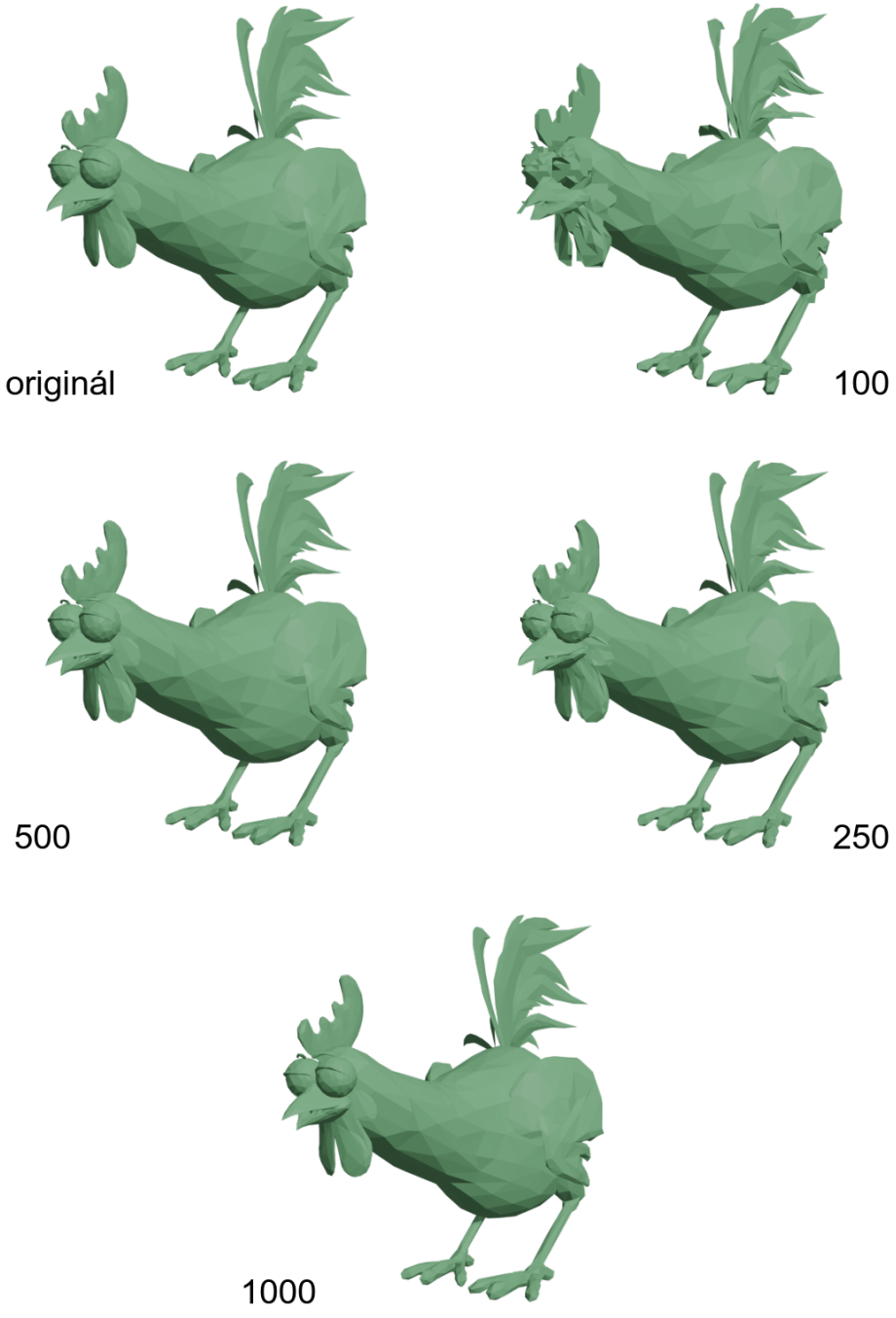
Pro tento experiment jsem využil variantu komprese, která nevyužívá normalizaci/transformaci animace a kvanizuje se zadanými *maximálními* rozlišeními a vynucuje krychlové buňky kvantizační mřížky. Normalizace není použita, aby výsledné snímky animace byly s originálními porovnatelné bez arbitrárních opravných transformací.

Na testování jsem ze sady testovacích animací poskytnutých vedoucím práce vybral první snímek animace „Chicken“, obdobné výsledky jsou očekávány od všech ostatních srovnatelných animací (popř. stačí zvýšit rozlišení, ale ne velmi významně) s níže popsanou výjimkou. Rozsah testovaných kvantizačních rozlišení a výsledky rekonstrukce si lze prohlédnout na obrázku 4.1.

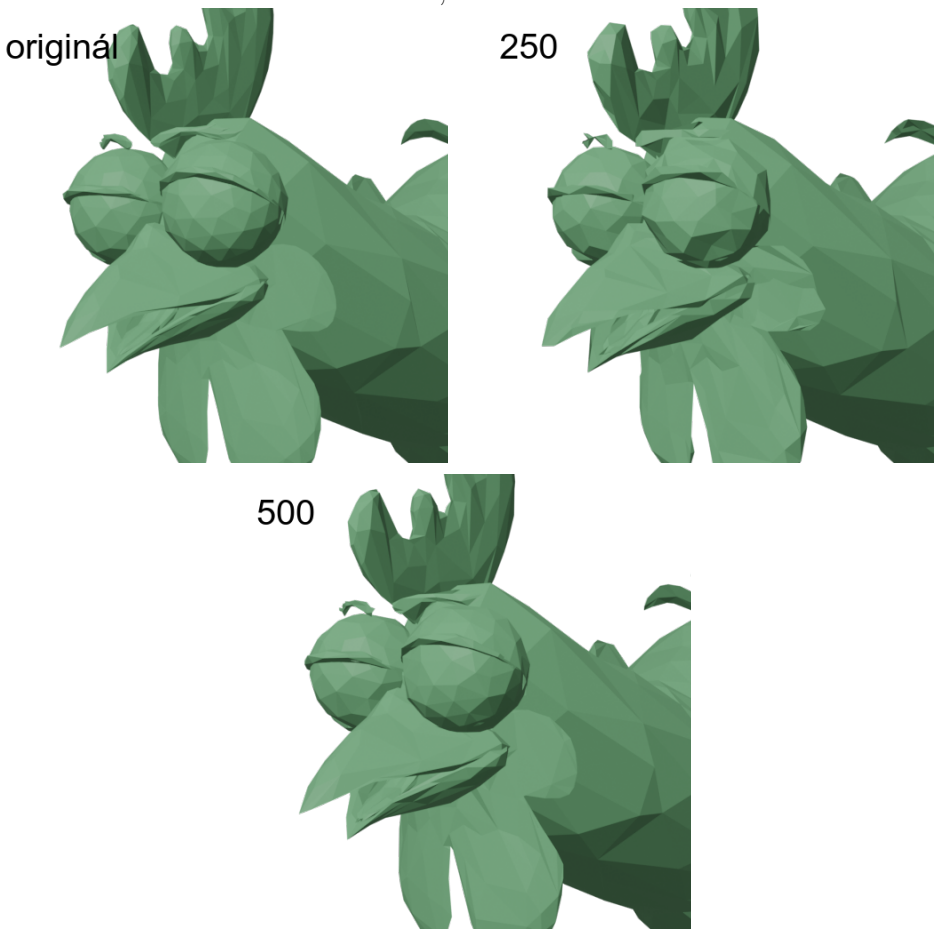
Z vizuálního hlediska je zřejmé, že kvantizace s navrženým maximálním rozlišením 100 je velmi hrubá a není velmi přesvědčivá, na obrázku s rozlišením 250 již nejsou rozdíly natolik zjevné, ale například v „obočí“ kuřete si můžeme povšimnout jasné ztráty kvality (detail viz obrázek 4.2). U rozlišení 500 je kvalita velice dobrá, i když lze stále najít rozdíly a nakonec u rozlišení 1000 jsou rozdíly prakticky neznatelné, pokud obrázky nějakým způsobem neprolneme nebo neměříme.

Takovéto výsledky můžeme očekávat za předpokladu, že animace neobsahuje jemný detail relativně k jejímu měřítku. Pro použití, ve kterém se na animaci díváme jako na objekt (narozdíl od prostředí) a má tedy velikost jednoho či dvou metrů (což je cílová velikost normalizace), lze očekávat srovnatelné výsledky. Z testovacích animací se zdá, že tomu tak je pro všechny animace, kromě animace „Cloth“, ve které můžeme sledovat fyzikální simulaci látky a v počátečním snímku jsou objekty vertikálně hodně rozprostřené, takže navyšují rozměry scény, a tedy když se podíváme na objekty samotné, hlavně na povrch látky který je tvořen velkým množstvím trojúhelníků a zveličuje tak chyby kvantizace, tak lze i s vyšším rozlišením pozorovat artefakty, viz obrázek 4.3.

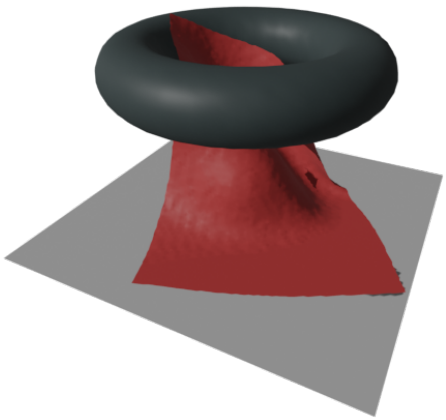
Obrázek 4.1: Přehled různých úrovní kvantizace - vlevo nahoře původní mesh, vpravo nahoře mesh kvantizovaná s rozlišením 100, vlevo uprostřed s rozlišením 500, vpravo uprostřed 250 a dole 1000.



Obrázek 4.2: Detail různých úrovní kvanizace - vlevo původní mesh, vpravo mesh kvantizovaná s rozlišením 250, dole 500.



Obrázek 4.3: Artefakty na simulaci látky s rozlišením 1000.



4.1.1 Porovnání různých rozlišení kvanizace

Z pohledu vlivu kvantizačního rozlišení na velikost komprimované animace se podíváme na stejný případ jako je na předchozích obrázcích Chicken. Tato animace má v zcela nekomprimovaném stavu (původní soubory `.obj`) velikost 67,9 MB, 400 snímků, 3030 vrcholů na snímek a 5664 trojúhelníků. Porovnání je v tabulce 4.1. Vidíme, že velikost kvantizační buňky má skutečně vliv na velikost komprimované animace, ale velikost nevzrůstá příliš rychle v závislosti na rozlišení a realisticky tedy můžeme pro tuto animaci používat vyšší rozlišení. Nadále budu pro animaci Chicken považovat za vhodné rozlišení kvantizace hodnotu 500 popř. 1000. (Komprese byla provedena s Gzipem a predikcí A)

Tabulka 4.1: Tabulka porovnávající kvantizační rozlišení a velikost komprimované animace Chicken. Pro rozlišení použitá v obrázcích 4.1 a 4.2.

Rozlišení	**Šířka buňky**	**Velikost komprimované animace**
100	0.02556	912 kB
250	0.01024	1,12 MB
500	0.00512	1,35 MB
1000	0.002556	1,65 MB

4.1.2 Vyhodnocení přesnosti rekonstrukce

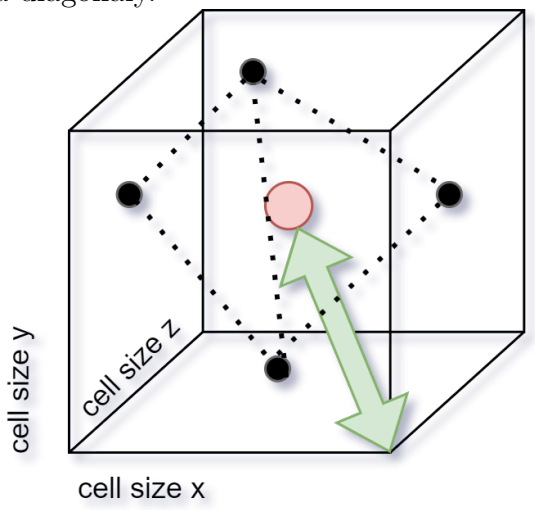
Existují různé metriky které porovnávají dvě polygonální sítě a vyhodnocují zkreslení. Tyto metriky do různé míry zachycují jak je zkreslení rekonstrukce vnímáno člověkem [16]. Tzv. perceptuální metriky se zaměřují na to, jak jsou rozdíly mezi modely vnímány člověkem. Toto by byl ten druh, který je pro tuto práci důležitější, protože jejím cílem je animace a potažmo modely zobrazovat a nejsou nadále zpracovány, protože v tom případě by mohly vzniknout nějaké požadavky na to, jak přesné rekonstrukce jsou.

Na druhou stranu jsou tyto perceptuální metriky komplikovanější a težší na vysvětlení i použití, takže se zaměřím spíše na základní metriku, která se jmenuje Hausdorffova vzdálenost. Tato metrika je definována mezi dvěma matematickými povrchy a ptá se, jaká je maximální vzdálenost libovolného bodu prvního povrchu od nejbližšího bodu toho druhého.

Program METRO, který vznikl jako výsledek výzkumu [4], právě tuto metriku umí vyhodnotit pro dva trojrozměrné modely. Do své práce jsem tedy přidal jednoduchý export rekonstruovaných snímků vygenerované animace a budu je porovnávat s originálními nekomprimovanými snímky.

Ještě než uvedu výsledky měření, tak popíši, jaké výsledky bychom mohli očekávat. Jelikož pracujeme s kvantizací, která má dané rozměry kvantizační buňky, tak je pro vyhodnocení očekávané chyby vhodné začít tam, viz obrázek 4.4

Obrázek 4.4: Kvantizační buňka, černě vrcholy trojúhelníkové sítě, které do ní spadají, červeně pozice v jejím středu, která ji reprezentuje v rekonstrukci a zeleně polovina diagonály.



Jelikož buňku ve snímku dekomprimované animace zastupujeme jejím středem, tak největší možná chyba kvantizace vychází z nejvzdálenějšího bodu od středu, což jsou rohy tohoto krychle. Maximální vzdálenost výsledného od původního bodu je tedy polovina diagonály. Výsledky viz tabluka 4.2. Předpokladem je, že buňka má krychlový tvar, což je splněno.

Tabulka 4.2: Vyhodnocení kvantizační chyby na dříve porovnávaném prvním snímku animace „Chicken“ s původními rozměry (2.56, 2.23, 1.07).

Rozlišení	**Šířka buňky**	**Diagonála / 2**	**Hausdorffova vz.**
100	0.02556	0.022136	0.018950
250	0.01024	0.008868	0.008012
500	0.00512	0.004434	0.003907
1000	0.002556	0.002214	0.002066

Výsledkem je, že skutečné vzdálenosti se v testovaných případech vejdou do odhadu nejhoršího možného zkreslení, tedy chyba kvantizace se chová podle očekávání. Naměřená Hausdorffova vzdálenost trojúhelníkových sítí je dokonce o zhruba 10% lepší, než bylo očekáváno, zdůvodněním je ale hlavně

to, že protože nejsou porovnávány pouze vzdálenosti dvojicí vrcholů, ale také hrany a trojúhelníky. Takže i když dojde k chybě blížící se maximální chybě, tak se ke zkoumanému vrcholu nalezne jiný bod sítě náležící zejména trojúhelníku jehož součástí je obraz zkoumaného vrcholu.

Nástroj METRO umí exportovat chybu konkrétních vrcholů do hodnoty vertex color modelu, ale z mého testování tento výstup není platný a nepovedlo se mi ho otevřít v žádné standardní aplikaci. Chtěl bych ale nějakým způsobem vizualizovat, jak je chyba distribuovaná na modelu, takže jsem se pokusil toho docílit renderováním obou modelů v prostoru přes sebe, viz obrázek 4.5. Cílem je ukázat, že chyba je náhodná až chaotická a nemá žádný trend, takže pravděpodobně nejde o základní numerickou chybu.

Obrázek 4.5: Vizualizace chyby: rekonstrukce modře – kvantizace s rozlišením 250 resp. 1000, červeně – originál.

4.2 Naměřené metriky výkonu predikce a kompresního algoritmu

Protože jsou ve skutečnosti výsledky těchto dvou faktorů do velké míry provázané, tak se následující dvě tabulky týkají obou dvou vlastností, a podívají se na velké spektrum případů, aby by závěr co nejkompletnější. První tabulka prezentuje výsledky Gzipu se všemi třemi možnostmi predikce a druhá tabulka prezentuje výsledky aritmetického kodéru.

Také by bylo vhodné představit animaci „Walk“. Animace má 187 snímků a v originální podobě `.obj` souborů má 433 MB, 35626 vrcholů a 67 571 trojúhelníků. Jde o zdaleka největší animaci, která byla poskytnuta pro testování a zároveň obsahuje asi nejkomplexnější pohyb. Animace vznikla technikou motion capture.

Do tabulek 4.3 a 4.4 jsem zahrnul i dříve zmíněnou animaci „Cloth“, která má 9 987 vrcholů, 19 494 trojúhelníků a 200 snímků. V nekomprimované podobě `.obj` souborů má 126 MB. Je tedy výrazně větší, něž animace „Chicken“ (ve všech vlastnostech kromě počtu snímků) a přesto je dokonce lépe komprimovatelná a to i ve velkém rozlišení, protože velké části této animace jsou dobře předvídatelné. V podstatě lze velkou část animace popsat rovnoměrnými přímočarými pohyby a proto má výborné výsledky pro predikci rychlosti (kde je ukládána informace o zrychlení) – v tomto případě jsou ukládané hodnoty nulové, což výrazně snižuje celkovou entropii dat a aritmetický kodér toho umí dobře využít.

Obrázek 4.6: Ukázka animace Walk na snímcích 0, 50, 120 a 186.

Tabulky porovnávají velikost komprimované animace a dobu dekomprese ve všech případech. Pro dobu dekomprese nebylo prováděno více pokusů, takže je pouze orientační a byla měřena na stolním počítači s procesorem Intel i7-6700K a 16GB RAM.

Tabulka 4.3: Tabulka komprimovaných velikostí a časů v případě kompresního algoritmu Gzip pro všechny varianty predikce.

Animace	**Predikce A**	**Predikce B**	**Predikce C**
Chicken 100	912 kB (0,16s)	1,19 MB (0,13s)	1,45 MB (1,31s)
Chicken 250	1,12 MB (0,16s)	1,40 MB (0,14s)	1,71 MB (1,34s)
Chicken 500	1,35 MB (0,16s)	1,55 MB (0,16s)	1,89 MB (1,34s)
Walk 100	4,23 MB (0,80s)	6,15 MB (0,73s)	7,31 MB (2,98s)
Walk 500	5,73 MB (0,78s)	7,78 MB (0,83s)	9,49 MB (2,96s)
Walk 1000	6,97 MB (0,82s)	8,54 MB (1,15s)	10,5 MB (3,17s)
Cloth 1000	1,28 MB (0,25s)	1,38 MB (0,23s)	1,72 MB (1,18s)

Tabulka 4.4: Tabulka komprimovaných velikostí a časů v případě aritmetického kodéru pro všechny varianty predikce.

Animace	**Predikce A**	**Predikce B**	**Predikce C**
Chicken 100	669 kB (1,37s)	721 kB (1,36s)	878 kB (2,41s)
Chicken 500	1,14 MB (2,16s)	0,98 MB (1,76s)	1,23 MB (2,66s)
Chicken 1000	1,45 MB (3,34s)	1,15 MB (1,98s)	1,38 MB (2,87s)
Walk 100	3,84 MB (7,33s)	4,09 MB (7,61s)	5,54 MB (9,46s)
Walk 500	5,72 MB (14,22s)	5,67 MB (8,54s)	7,05 MB (12,16s)
Walk 1000	7,26 MB (17,61s)	6,71 MB (10,72s)	8,12 MB (14,63s)
Cloth 1000	0,97 MB (2,34s)	878 kB (1,79s)	1,10 MB (2,79s)

4.3 Vliv predikce na velikost komprimované animace

Gzip nereaguje tak dobře na snížení entropie dat některým způsobem nebo pod nějakou hranici, protože kvantizace jednoznačně pomáhá, ale alternativy predikce již tolik ne, naopak jsou z nějakého důvodu na škodu. Toto je relativně zvláštní fakt, protože součástí této komprese je Huffmanovo kódování, což je podobně jako aritmetické kódování druh entropického kodéru, který má být z pohledu entropie blízko optimálnímu výsledku. Toto se ale v datech alespoň na základě mých testů neděje.

Gzip neprodukuje v kombinaci s vyššími stupni predikce (P_B, P_C) menší výstup, ale může v některých případech být o něco rychlejší (viz hodnoty naměřené pro Chicken), ale na takto malém počtu měření a s takto malými odchylkami nepovažuji zlepšení za významné.

Aritmetický kodér má nejlepší výsledky v kombinaci s Predikcí B, což podle zdůvodnění této metody dává smysl. Predikce C se již nevyplatí, intuitivně si můžeme představit, že se pohyb snažíme popisovat vyšším stupněm polynomu, což se často nevyplácí.

Celkem nejvýhodnější dvě možnosti jsou Gzip v kombinaci s Predikcí A, tato metoda je nejrychlejší, ale nedosahuje nejlepší komprese, zatímco Aritmetický kodér spolu s Predikcí B dosahuje doposud nejlepší komprese, za cenu zpomalení dekomprese.

Poté, co z experimentu vyplynulo, že Gzip dosáhne pro nižší stupně změny pozice lepší komprese, tak jsem se rozhodl ujistit, že je predikce skutečně výhodná oproti vynechání tohoto kroku, viz následující tabulka 4.5.

Tabulka 4.5: Porovnání jednoduché predikce s žádnou predikcí v případě Gzipu.

Animace	**Predikce A**	**Žádná predikce**
Chicken 100	912 kB (0,16s)	3,25 MB (0,23s)
Chicken 250	1,12 MB (0,16s)	4,31 MB (0,19s)
Chicken 500	1,35 MB (0,16s)	4,98 MB (0,19s)
Walk 100	4,23 MB (0,80s)	8,22 MB (0,76s)
Walk 500	5,73 MB (0,78s)	18,2 MB (0,96s)
Walk 1000	6,97 MB (0,82s)	22,7 MB (0,89s)
Cloth 1000	1,28 MB (0,25s)	7,41 MB (0,29s)

Z těchto výsledků lze usoudit, že predikce hraje v dobrém kompresním poměru velkou roli a výkon dekomprese je podobný.

4.4 Kompresní algoritmus

Výsledky testů v sekci 4.2 naznačují, že ArithCoder se skutečně nevyplatí v důsledku doby dekomprese i v případě s vhodnější predikcí a tedy jsem zvolil jako finální kompresní algoritmus použití komprese Gzip v kombinaci s touto metodou. Gzip sice může mít o něco horší kompresní poměr než ArithCoder ale prakticky vždy je znatelně rychlejší a je výrazně méně závislý na vstupních datech. Když jím projdou velká a těžko komprimovatelná data, tak to nezpůsobí problémy a dá se na jeho výkon více spoléhat. Jeho kompresní a dekompresní časy jsou vážené spíše pro rychlejší dekompresi, což je přesně, co je zde potřeba.

V průběhu implementace komprese animace mě napadlo, jestli by nemohlo být výhodné komprimovat animaci po částech, třeba i kombinací obou kodérů, pro co nejlepší výsledky. V jednu chvíli byl první snímek animace a zbytek animace komprimovány dvěma nezávislými průběhy kodérů, ale nevšiml jsem si žádných významných zisků, ale to může být proto, že nešlo o vhodné kombinace.

Jedním ze způsobů jak tohoto využít by bylo kódování části animace, která má nižší entropii, jako jsou např. data ostatních snímků s Predikcí B aritmetickým kodérem, a kódování zbylých částí, např. dat prvního snímku, Gzipem. Neočekávám ale nějaký významný zisk, protože největší rozdíl mezi Gzipem a AritCoderem mezi tabulkami výše je pod 60% , navíc každý kompresní algoritmus má svůj vlastní overhead a nebude schopný tolik benefitovat z objemu komprimovaných dat. Často jsou kompresní algoritmy tím efektivnější, čím více dat mají k dispozici, typicky pro slovníkové přístupy,

kde více výskytů umožní lepší využití slovníku a proto neočekávám, že rozdělení animace na více průchodů kompresními algoritmy by mělo významnou výhodu.

4.5 Kvalita PCA

Chyba rekonstrukce způsobená snížením počtu bázových vektorů má velmi rozdílný charakter od chyby rekonstrukce způsobené kvantizací. Oba dva způsoby zavedení chyby se objevují v mojí implementaci a vychází ze dvou parametrů komprese - rozlišení kvantizace a počet bázových vektorů.

Na následující sadě obrázků nastíním, jak se tato chyba projevuje. Obrázky také reprezentují fakt, že velikost této chyby se mění v průběhu animace od snímku ke snímku, na rozdíl od kvantizační chyby, která je ve všech snímcích velmi podobná.

Tabulka 4.6: Porovnání vybraných snímků animace Walk s různou kvalitou PCA komprese. (Kvantizace: 5 míst => rozlišení 10 000 a 2 místa => 100)

frame	originál	5 míst |báze| = 15	5 míst |báze| = 5	2 místa |báze| = 15
0				
50				
120				
186				

Chybu kvantizace si lze představit jako šum, který se přimíchá k pozicím vrcholů a pro každý vrchol v každém snímků má stejnou maximální velikost, viz poslední sloupec tabulky 4.6 (nebo dřívější porovnání u kvantizace přímé komprese viz obrázek 4.1). Takto se kvantizace projeví v důsledku kvantizace

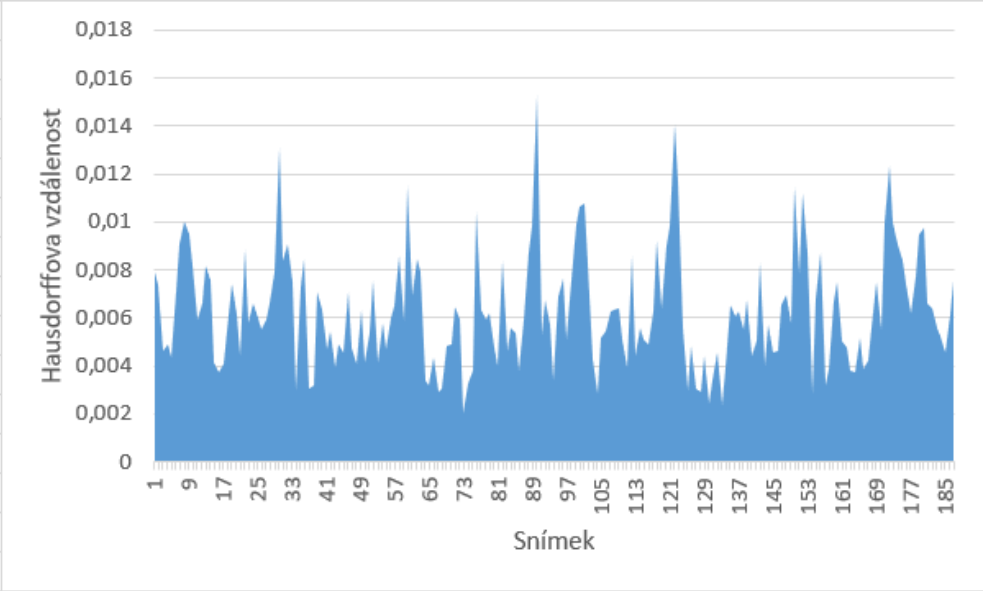
hodnot průměrů a podobně také v důsledku kvantizace koeficientů vrcholů, tedy matice T.

Oproti tomu chyba způsobená omezením velikosti báze má spíše charakter nepřesnosti pozic celých částí animace, který je zvláště bez znalosti originální animace těžší posoudit. Dobrým příkladem je pozice dlaní na prvním a posledním snímku, tedy v prvním a posledním řádku tabulky 4.6.

K druhému druhu chyby rekonstrukce také přispěje kvantizace bázových vektorů, tedy matice U. Různé hodnoty by tedy mohly být kvantizovány s různým rozlišením, aby se lépe využil rozsah hodnot a snížila velikost komprimované animace, ale z mého krátkého testování to nepovažuji to za velký nedostatek.

Také se lze pro představu podívat na chybu rekonstrukce pomocí Hausdorffovy vzdálenosti přes všechny snímky animace, viz graf 4.7.

Obrázek 4.7: Graf chyby rekonstrukce ve snímcích animace Walk za použití komprese PCA s rozlišením kvantizace 10 000 a velikostí báze 15.



5 Výsledky komprese

5.1 Porovnání s algoritmem CoDDyaC

CoDDyaC[18] je sofistikovaný algoritmus pro kompresi dynamických trojúhelníkových sítí, který byl state-of-the-art v době jeho publikace a je stále kompetitivní. Pro porovnání byla použita referenční implementace dostupná z `meshcompression.org` [15]. Tento algoritmus má na komprimované animace další požadavek oproti řešení prezentovaném v této práci a tím je, že animovaný model musí být manifoldem.

Dalším potenciálním problémem je, že CoDDyaC mění pořadí vrcholů, což by nepůsobilo potíže v aktuálním systému, ale komplikuje to využití mnohých metrik zkreslení rekonstruované animace. CoDDyaC poskytuje použitou permutaci vrcholů, takže je možné obnovit původní pořadí vrcholů, kdyby to bylo zapotřebí. Mnou prezentované algoritmy také mohou dosáhnout lepší komprese změnou pořadí vrcholů, ale tuto možnost jsem nechal v tuto chvíli na uživateli v kroku předzpracování a tedy její opominutí umožňuje rozšiřitelnost v tomto směru.

Algoritmus CoDDyaC se také zakládá na PCA kompresi, použitá implementace zároveň využívá princip komprese bází PCA[17]. Program také narozdíl od mých implementací pracuje ve více vláknech, takže přímé porovnání časů dekomprese není spravedlivé. Silnou nevýhodou využití této komprese pro účely této práce je, že není integrovaná do VRClassroom narozdíl od mého řešení, takže nelze výsledky rovnou předat enginu Unity pro zobrazení, protože komunikaci s programem by bylo potřeba započítat do času dekomprese.

CoDDyaC rovněž pro vstup i výstup používá souborový formát .OFF, nikoliv .OBJ, jak je zadáním této práce, což vyžaduje převod. Formát naštěstí není natolik odlišný od .OBJ, je textový, a má jisté podobnosti, ale jedním z velkým rozdílů je, že vrcholy se indexují od 0, takže při převodu je potřeba převést celou konektivitu, zatímco ostatní hodnoty lze povětšinou přeskládat. Pro účely porovnání s tímto omezením však lze pracovat.

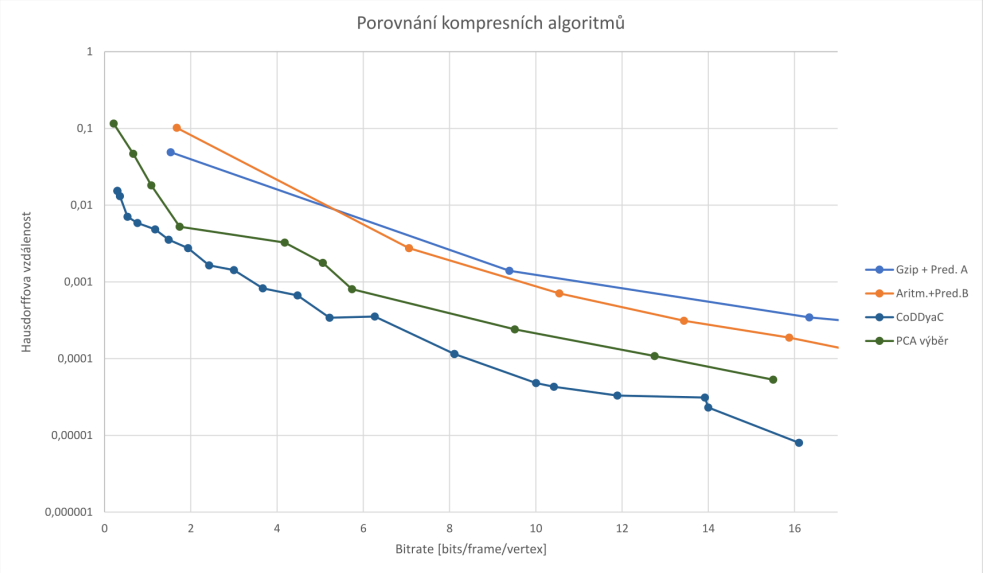
```
# Příklad triviálního OBJ souboru:
# Vrcholy
v 0.0 0.0 0.0
v 1.0 0.0 0.0
v 1.0 0.0 0.5
# Stěny
f 1 2 3

# Ekvivalentní minimální OFF soubor:
OFF
3 1 # Počet vrcholů a stěn
0.0 0.0 0.0 # Seznam vrcholů
1.0 0.0 0.0
1.0 0.0 0.5
3 0 1 2 # Seznam stěn (první číslo je počet vrcholů)
```

5.2 Výsledky porovnání

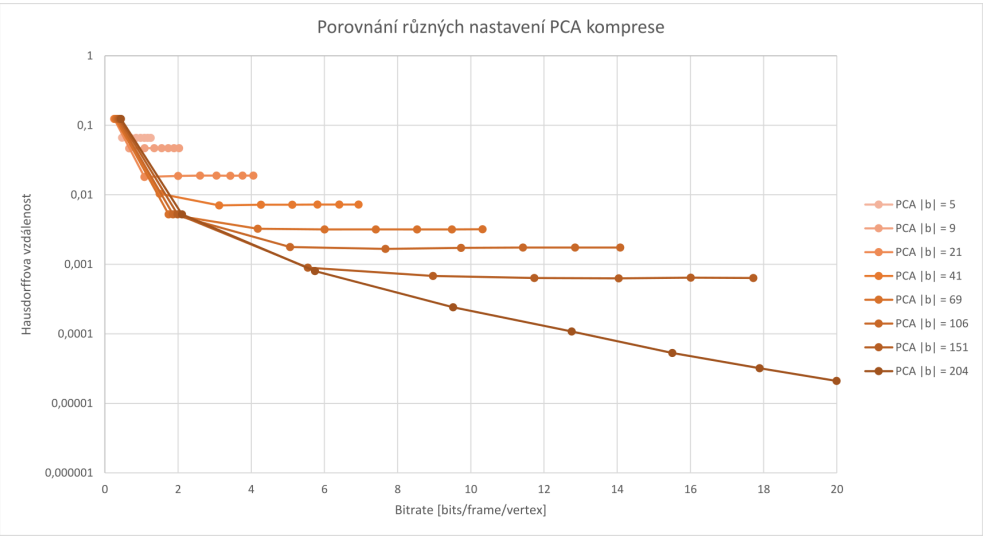
Pro detailnější vyhodnocení vlastností kompresního algoritmu se často používá tzv. RD digram, který vizualizuje vztah chyby (D - distortion) vzhledem k datovému toku (R - rate). Datový tok je vyjádřen jako $bity/snímek/vrchol$. Vzhledem k proměnlivosti chyby v průběhu animace při použití PCA komprese (viz sekce 4.5), tak by bylo vhodné využít metriku, která toto vezme v úvahu. Jedna často používaná metrika je KG Error[6], ale v tuto chvíli nemám dostupnou implementaci této metriky a také není kompatabilní s přeřazováním vrcholů, které vykoná CoDDyaC. Také existuje tzv. čtyřrozměrná Hausdorffova vzdálenost[19], ale výpočet této metriky je pomalý a rovněž nemám dostupnou implementaci. Jako kompromis bylo navrženo navzorkovat Hausdorffovu vzdálenost v průběhu celé animace, z dílčích vzdáleností se použije maximum. Výsledky porovnání si můžeme prohlédnout na grafu 5.1.

Obrázek 5.1: RD graf porovnávající kompresní algoritmy na animaci Cowheavy.



Křivka PCA komprese je ručně vybraná z více testovaných hodnot, hrubý přehled charakteristik komprese je znázorněn na obrázku 5.2. Všechny hodnoty vybrané pro obrázek 5.1 jsou podmnožinou těch zobrazených na obrázku 5.2.

Obrázek 5.2: RD graf různých nastavení kvality PCA komprese na animaci Cowheavy. Řady představují danou velikost báze s rostoucím rozlišením kvantizace od 10 po 100000.



Z porovnání pozorujeme, že moje implementace komprese má horší cha-

rakteristiky, což bylo možné očekávat z komplexity a sofistikovanosti algoritmů. Na druhou stranu moje varianta má několikero výhod (viz sekce 8.1 a 5.1), takže její použití je odůvodněné. Zároveň platí, že řekněme dvakrát, nebo třikrát horší komprese stále znamená, že naprostá většina úspory je zachována, díky tomu, jak velké jsou kompresní poměry. Navíc jsou moje časy dekomprese díky integraci přímo do aplikace v důsledku rychlejší. Zároveň je potřeba vzít v úvahu, že obrázky 5.1 i 5.2 pokrývají velmi široké spektrum kvality, pro cílové použití ve virtuální realitě budou pravděpodobně vhodné relativně nižší kvality v levé části grafů, kde se pro naměřené hodnoty PCA komprese přibližuje výsledkům CoDDyaCu.

5.3 Vyhodnocení řešení

Nejvhodnější možnosti ze všech navržených variant byly tedy vybrány pro porovnání. Cílem je vybrat finální kompresi pro použití v aplikaci VRClassroom. Hlavním cílem je redukování času přenosu, takže mimo velikosti komprimované animace je uvažována doba dekomprese, jak moc se komprese vyplatí z časového hlediska a za jakého připojení. Hlavní uvažované varianty jsou v tuto chvíli PCA komprese a Gzip komprese s Predikcí A. PCA zastupuje nejlepší kompresní poměr a časově je srovnatelná s kombinací aritmetického kodéru s Predikcí B, zatímco Gzip s Predikcí A je nejrychlejší, za cenu kompresního poměru.

Tabulka 5.1: Porovnání kompresí

Jméno animace	Walk			
Vlastnosti	35626 vrcholů, 187 snímků			
Nekompr. Velikost	187 * 35626 *12B = 79 944 744B = 76MB			
Komprese	Gzip + Predikce A		PCA	
Parametry	50	500	8, 1000	15, 10 000
Kompr. Velikost	3,54 MB	5,73 MB	620 kB	1,41 MB
Kompresní poměr	22:1	13:1	126:1	54:1
Bitrate [b/fv]	0,56	0,90	0,10	0,22
Chyba rekonstr.	0,011039	0,001292	0,034913	0,01528
Doba dekompr. PC	0,7s	0,8s	2,0s	4,0s
Max. rychl. přip.	104 MB/s	88 MB/s	38 MB/s	19 MB/s
Doba dekompr. VR	2,0s	2,7s	7,4s	13s
Max. rychl. přip.	36 MB/s	26 MB/s	10 MB/s	5,8 MB/s

Tabulka 5.2: Porovnání kompresí

Jméno animace	Chicken			
Vlastnosti	3030 vrcholů, 400 snímků			
Nekompr. Velikost	400 * 3030 *12B = 14 544 000B = 14MB			
Komprese	Gzip + Predikce A		PCA	
Parametry	250	500	10, 1000	15, 2000
Kompr. Velikost	1,12 MB	1,35 MB	139 kB	214 kB
Kompresní poměr	12:1	10:1	102:1	66:1
Bitrate [b/fv]	0,98	1,17	0,12	0,18
Chyba rekonstr.	0,008012	0,003907	0,915263	0,651364
Doba dekompr. PC	0,2s	0,2s	0,3s	0,5s
Max. rychl. přip.	64 MB/s	63 MB/s	46 MB/s	27 MB/s
Doba dekompr. VR	0,4s	0,4s	1,7s	2,4s
Max. rychl. přip.	32 MB/s	31 MB/s	8,1 MB/s	5,7 MB/s

Podle výsledků v tabulkách 5.1 a 5.2 lze konstatovat, že PCA komprese dosahuje typicky výrazně menší velikosti komprimované animace, tedy i nižšího datového toku a lepších kompresních poměrů. Na druhou stranu její dekomprese trvá déle a tím pádem se časově vyplatí pro nižší maximální rychlosti připojení.

Nastavení kvality byla vybrána jako vhodné hodnoty, které by bylo možné reálně použít s výjimkou prvního sloupečku komprese Gzip + Predikce A u animace Walk. Zde bylo vybráno velmi nízké rozlišení kvantizace, aby se velikost chyby přiblížila těm naměřeným u animace Walk. I v tomto případě je velikost komprimované animace výrazně vyšší, než za použití PCA komprese, zatímco vizuální kvalita animace je zase podstatně horší. Naměřené chyby rekonstrukce u PCA kompresí jsou velké, ale to je hlavně z toho důvodu, že Hausdorffova vzdálenost není nejvhodnější metrikou pro vyhodnocení chyby této metody, viz sekce 4.5.

Celkově se PCA komprese vyplatí v dobré kvalitě i s rychlostí připojení kolem 5MB/s. Dnes jsou zvláště pro podniky dostupné větší rychlosti připojení a od interních sítí se také očekává vyšší rychlost přenosu dat. Tato metoda komprese na druhou stranu dosahuje velmi dobrých kompresních poměrů, což přispěje ke snížení zátěže sítě, zvláště když aplikaci používá více uživatelů najednou a animace je tedy rozesílána opakovaně. Proto jsem vybral PCA kompresi jako tu, která bude použita ve VRClassroom.

6 Popis implementace

Bylo vytvořeno pět nových tříd v klientovi, navíc implementace systému celkem zasáhla do šesti tříd v klientské aplikaci a tří na straně serveru. Dále byly vytvořeny dvě nové síťové zprávy jak směrem od klienta k serveru, tak od serveru ke klientovi.

Z pěti nových tříd jsou dvě důležitější: `MeshAnimationCompressor` – tato třída se zabývá samotnou kompresí animací a `AnimationPlaybackState` – třída která uchovává aktuální stav přehrávání animace a jeho konzistenci. Méně důležité nové třídy jsou `AnimationMenuScript`, `PlaybackControlsScript` a `BatchCompression`. Z těchto tříd se první dvě týkají obsluhy GUI – menu komprese a panel ovládání přehrávání animace. Poslední třída `BatchCompression` umožňuje spuštění komprese z příkazové řádky, tato funkcionalita byla klíčová pro testování.

Třída `MeshAnimationCompressor` implementující samotnou kompresi umí načíst a komprimovat animaci ze složky a dekomprimovat animaci a exportovat ji do složky. PCA komprese se týkají metody `EncodePCA()` a `DecodePCA()`. Výstupem `DecodePCA()` není přímo použitelná animace, ale geometrie a konektivita animace, animaci je ještě potřeba dokončit funkcí `CompleteAnimation()`. Dekomprese je rozdělena do těchto dvou metod, aby byla kompatibilní se spuštěním v samostatném vlákně, některé části Unity API jsou dostupné pouze z hlavního vlákna, tyto části se nachází právě ve funkci `CompleteAnimation()`.

V aplikaci již existuje jako jedna z klíčových částí systém pro zobrazování polygonálních sítí, který je pro některé části (např. síťový přenos) použit jako vzor pro můj systém. Jelikož jsou animace uloženy v samostatných souborech, tak jsou obdobným způsobem načítány a rozesílány mezi klienty a serverem a naopak, ale protože jsou již komprimované, tak je vypuštěn krok komprimace, který původně probíhal až před odesláním. V důsledku je zpracování animace a samostatného modelu analogické a nezpůsobuje žádné problémy pro systém záznamů přednášek v aplikaci, který je založený na odchytávání paketů. Za tímto účelem byla tedy zavedena dvojice zpráv `MeshAnimationData`, jedna jako správa serveru a druhá na straně klienta. Obsahem těchto zpráv jsou pouze komprimované animace v binární podobě. Server přijme animaci, uloží si ji pro klienty, kteří se připojí později a rozešle ji ostatním klientům.

Druhá dvojice zpráv se týká přehrávání animace, také jsou symetrické mezi klientem a serverem a jejich obsahem jsou čtyři hodnoty, které definují stav `AnimationPlaybackState` – `boolean`: přehrávání je spuštěno(`true`) / zastaveno(`false`), `float`: snímková frekvence animace, `int`: číslo aktuálního snímku a `int`: délka animace.

Implementace síťové části vyžadovala změny ve třídách `MeshesPool`, `NetworkManager`, `NetworkSend` a `NetworRecieve` u klienta a `ClassroomManager`, `NetworkSend`, `NetworkRecieve` na straně serveru.

PCA komprese využívá některé maticové operace a singulární rozklad, pro tyto účely je použita knihovna. Knihovna vybraná pro mou implementaci je Math.NET Numerics[7], která požadované operace poskytuje a spadá pod MIT licenci, která je vhodná pro použití ve VRClassroom. Knihovna je cílená na Microsoft .NET prostředí, jehož součástí je programovací jazyk C#.

Vedoucím práce byla poskytnuta implementace aritmetického kodéru ArithCoder, který je napsaný v programovacím jazyce C# a vhodně do řešení zapadá. Tento kodér je ve finální implementaci využit jako součást PCA komprese.

6.1 Systém záznamů

Při implementaci bylo potřeba zohlednit systém přehrávání záznamů. Zpráva s obsahem animace netvoří pro tento systém problém. Synchronizace stavu přehrávání animace proběhne, a v případě, že se uživatel kontinuálně dívá na záznam bez pozastavování a přeskakování časem, tak bude synchronizace odpovídat tomu, jak vypadala původně při vytváření záznamu. V případě, kdy uživatel zastaví záznam ve chvíli, kdy je animace spuštěna, tak dále poběží a nebude zpětně synchronizována až záznam znovu spustí.

Tuto situaci bude ale možné řešit, pokud systém záznamů poskytuje např. informaci o čase relativním k začátku záznamu. Na druhou stranu je také potřeba detailněji specifikovat, jak by se přehrávání mělo v takovýchto situacích chovat, protože možných přístupů je více. Nepovažuji to tedy za problém, protože pravděpodobně existuje řešení, které lze postavit na vrch aktuálního systému přehrávání animace.

7 Funkcionalita řešení

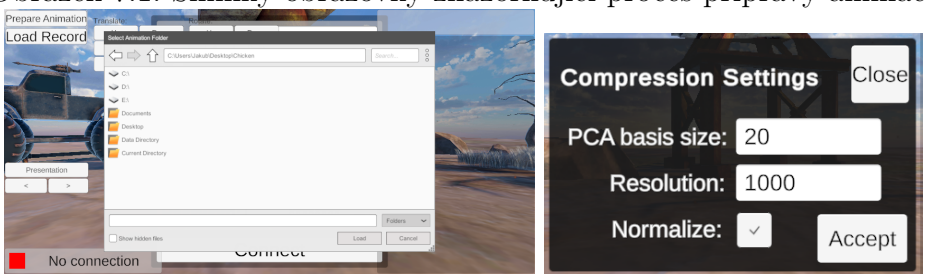
V této kapitole se nachází popis navrhované uživatelské zkušenosti s mým systémem animací ve VRClassroom.

7.1 Příprava animace

Menu přípravy animace je dostupné před připojením k serveru. Jeho účelem je nastavení parametrů komprese a připravení komprimované animace do souboru. Uživatel vybere složku se zdrojovými soubory, a kvalitu animace. Výstupní soubory mají koncovku `.ANIM`. Okno také poskytuje možnost normalizovat animaci. Výstřižek obrazovky si můžete prohlédnout na obrázku 7.1.

Tlačítko přípravy animace se nachází v levém horním rohu obrazovky pokud uživatel není připojen k serveru. Po stisknutí se zobrazí okno s výběrem složky obsahující animaci a následně okno s nastavením kvality a normalizace animace. Potvrzením dojde ke komprimaci animace.

Obrázek 7.1: Snímky obrazovky znázorňující proces přípravy animace.



7.2 Nahrání animace

Když je uživatel připojený k serveru VRClassroom jako učitel, je mu na stolní verzi aplikace nabídnuto v seznamu ovládacích tlačítek tlačítko „Load Mesh Animation“, které po stisknutí zobrazí okno file browseru, kterým se vybere soubor komprimované animace s koncovkou `.ANIM`. Po výběru je animace odeslána serveru a na všech klientech po přijetí dekomprimována a zobrazena.

Animace je dekomprimována v samostatném vlákně, aby nezmrazila běh aplikace, což by bylo velmi nevhodné zvláště ve VR. Vlákno nemůže dekompresi zcela dokončit, protože velkou část Unity API nelze využít z jiného,

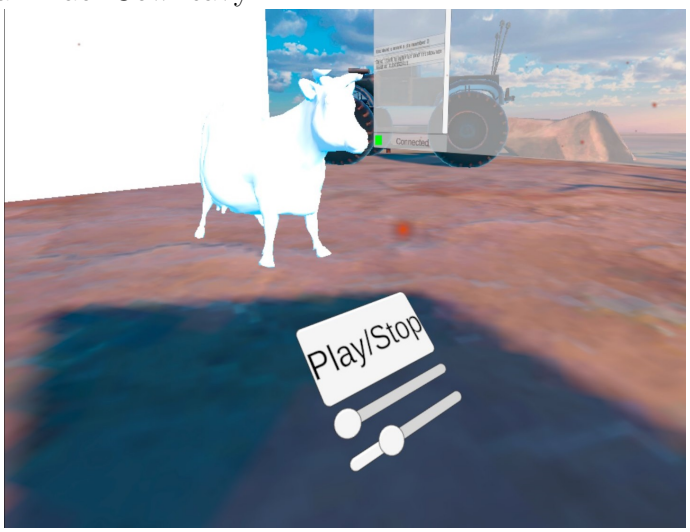
než hlavního Unity vlákna. Řešením je, že po dokončení všech oddělitelných částí jsou zbylé části – sestavení samotných `Mesh` objektů a logování provedeny při příštím průběhu hlavní smyčky. Celkově oddělení dekomprese do vlákna způsobí další zpomalení, ale je to nezbytné pro uživatelskou přívětivost systému.

7.3 Přehrávání animace

Po rozeslání animace klientům může libovolný uživatel jak ze stolního klienta, tak z VR klienta spustit nebo pozastavit přehrávání animace, nastavit rychlost přehrávání animace a přeskočit do libovolné části animace. Stav přehrávání animace je synchronizován mezi klienty.

Ovládací panel je stejný ve stolní aplikaci i ve VR, s tím rozdílem, že na stolním počítači je dvourozměrný a nachází se v levém spodním rohu obrazovky, zatímco ve VR je trojrozměrný a je postaven před počátek souřadného systému animací viz 7.2. Směrem shora dolů panel sestává z tlačítka, které přepíná mezi pozastavením a spuštěním animace, dále obsahuje posuvník průběhu animace a posuvník rychlosti přehrávání. Posuvník průběhu animace reprezentuje ve které části z časového hlediska se animace nachází a umožňuje do libovolné části přeskočit. Posuvník rychlosti přehrávání zobrazuje a nastavuje aktuální snímkovou frekvenci animace. Aktuální rozsah posuvníku je mezi 10 a 60fps.

Obrázek 7.2: Snímek obrazovky z VR s ovládacími prvky přehrávání animace a načtenou animací Cowheavy.



8 Návrhy na zlepšení

8.1 Technická zlepšení

Hlavní věc ke zlepšení by podle podílu na této práci byla samozřejmě komprese. Na obou přístupech, které byly prezentovány se dá dále stavět.

Hlavní přístup ke zlepšení první přímočaré metody komprese by bylo použití komplikovanější predikce, která využívá několika předchozích pozic vrcholů k co nejpřesnější predikci. Tento přístup by měl za cíl zlepšit kompresní poměr, ale za cenu delší doby dekomprese. Jsou i jiné směry zlepšení tohoto kompresního postupu, které přímo nenavazují na obsah této práce. Vzhledem k tomu, jak pozadu jsou aktuální výsledky tohoto postupu oproti tomu založenému na PCA, tak by se práce na nich pravděpodobně nevyplatila.

Co se týče metod založených na PCA, tak jeden způsob přímého navázaní na mou práci by bylo zlepšení metody kódování reálných čísel – kvantizace by mohla být efektivnější a predikce by se u některých polí určitě dala ještě vylepšit obdobným přístupem jako u první přímé komprese. Tato metoda má dále větší potenciál na významné zlepšení výsledků, existuje totiž spousta částí komprese, kterým nebyl v této práci věnován dostatek času – od efektivnějších kódování matic U a T, přes využití jednoho ze zavedených způsobů kódování konektivity trojúhelníkové sítě popsaných v publikaci [8]. PCA benefituje ze shlukování [11]. Nejvhodnější by bylo využít některý ze sofistikovaných přístupů např. [18], ale věnovat při implementaci pozornost době dekomprese, která je pro použití ke zrychlení přenosu po síti stejně významná, jako samotný kompresní poměr. Publikace o kompresi dynamických trojúhelníkových sítí se samozřejmě také zabývají dobou dekomprese, ale hlavně se zaměřují na kompromis mezi kvalitou komprese a kompresním poměrem.

Důvodem pro nepoužití optimální komprese je hlavně to, že pro jejich implementaci je potřeba porozumět ne jenom tomu, co jsem se naučil touto prací, ale i o mnoho více a proto by nebylo realistické docílit vhodné implementace založené na těchto sofistikovaných metodách.

Mnou implementované výsledky nejsou problematicky pozadu oproti těmto z pohledu komprese lepším metodám. Když se vezme v úvahu doba dekomprese a ostatní specifické parametry řešené úlohy, tak je moje řešení z dříve vysvětlovaných důvodů (komprese nemanifoldů, integrovaná implementace) nejlepším dostupným řešením.

8.2 Zlepšení aplikace z pohledu uživatele

Animace by mohla být integrována se systémem, který je již součástí VRClassroom a umožňuje uchopit trojrozměrné objekty a změnit jejich pozici pohybem ruky ve VR, toto by mělo kladný dopad na praktičnost zobrazení animace a na uživatelskou zkušenost s programem.

Dalším návrhem je přidání vizuálního indikátoru, který informuje, že probíhá dekomprese animace, např. ikona přesýpacích hodin. Tato informace je aktuálně dostupná pouze prostřednictvím Debug Logu.

Z důvodů praktičnosti by bylo vhodné přehodnotit možnosti, které jsou uživateli poskytnuty při přípravě animace, na jednu stranu by bylo možné přidat více možností, aby byl systém více flexibilní, a na druhou stranu by bylo možné zapracovat na uživatelské přívětivosti nastavení kvality. Hlavním nápadem zde je, že PCA komprese využívá dvou parametrů: velikost báze a rozlišení kvantizace, ale ani jeden z nich není velmi intuitivní. Alternativou by bylo je nahradit umělým parametrem „kvalita“, který by se vybíral z intervalu $< 0, 1 >$ a podle něj by se automaticky určily vhodné parametry komprese.

Ke komprimované animaci by mohlo být vhodné přibalit některá metadata, jako například snímkovou frekvenci a verzi kompresního algoritmu. Potíží se snímkovou frekvencí je, že ve vstupních datech není specifikována, takže by ji uživatel musel manuálně zadat, takže není významný zisk oproti změně rychlosti přehrávání za běhu. Verze kompresního algoritmu by mohla být použita pro kontrolu. V případě, že bude tento systém rozšiřován by mohlo dojít k potížím v případě, že uživatel připravil animaci s jinou verzí algoritmu, než se kterou se jí pokouší dekomprimovat. V tomto případě by bylo možné tuto chybu detekovat a podat uživateli smysluplnou neobecnou chybovou hlášku.

9 Závěr

Cílem této práce bylo vytvoření systému, který poskytuje funkcionalitu automatického a uživatelem ovládaného přehrávání animací trojúhelníkových sítí.

Bylo prezentováno a důkladně otestováno několik základních metod komprese animací trojúhelníkových animací. Také byly porovnány s alternativou CoDDyaC, která byla state-of-the-art v době její publikace a je stále kompetitivní. PCA metoda komprese, která byla vybrána jako ta nejvhodnější, poskytuje nejlepší kompresní poměry u všech dostupných animací a její chyba rekonstrukce je typicky vizuálně nevýrazná. Nevýhodou této komprese je ale doba dekomprese.

Dále byla implementována funkcionalita přehrávání těchto animací a synchronizace mezi klienty, implementace byla také popsána z hlediska programátora. Veškerá funkcionalita byla navržena tak, aby nebyla v rozporu se systémem záznamů a bylo diskutováno, že pro tento účel je vhodná.

Nakonec byly navrženy některé způsoby, kterými může být práce dále rozšířena a vylepšena v logické návaznosti na dokončené funkcionalitě.

Výsledkem práce bylo úspěšné rozšíření aplikace VRClassroom o tyto komponenty, celkem je systém funkční a plní zadané požadavky.

Přehled zkratek

VoIP – Voice over IP
KIV/ZPOS – Katedra Informatiky / Zpracování Polygonálních Sítí
QoL – Quality of Life

Literatura

[1] *GNU Gzip (oficiální stránka)* [online]. GNU, 2020. [cit. 2022/04/28]. Dostupné z: `https://www.gnu.org/software/gzip/`.

[2] *Object Files (.obj)* [online]. [cit. 2022/04/28]. OBJ file specification. Dostupné z: `http://paulbourke.net/dataformats/obj/`.

[3] Bonneau, D. – Difrancesco, P.-M. – Hutchinson, D. Surface Reconstruction for Three-Dimensional Rockfall Volumetric Analysis. *ISPRS International Journal of Geo-Information*. 11 2019, 8, s. 548. doi: 10.3390/ijgi8120548.

[4] Cignoni, P. – Rocchini, C. – Scopigno, R. METRO: Measuring error on simplified surfaces. *Computer Graphics Forum*. 06 1998, 17, s. 167 – 174. doi: 10.1111/1467-8659.00236.

[5] Isenburg, M. – Alliez, P. Compressing polygon mesh geometry with parallelogram prediction. In *IEEE Visualization, 2002. VIS 2002.*, s. 141–146, 2002. doi: 10.1109/VISUAL.2002.1183768.

[6] Karni, Z. – Gotsman, C. Compression of soft-body animation sequences. *Computers & Graphics*. 2004, 28, 1, s. 25–34. ISSN 0097-8493. doi: https://doi.org/10.1016/j.cag.2003.10.002. Dostupné z: `https://www.sciencedirect.com/science/article/pii/S0097849303002267`.

[7] *Math.NET Numerics* [online]. Math.NET, 2022. [cit. 2022/06/16]. Linear Algebra Library. Dostupné z: `https://numerics.mathdotnet.com/`.

[8] Peng, J. – Kim, C.-S. – Kuo, C.-C. J. Technologies for 3D mesh compression: A survey. *Journal of Visual Communication and Image Representation*. 12 2005, 16, s. 688–733. doi: 10.1016/j.jvcir.2005.03.001.

[9] Poór, V. Zaznamenávání akcí ve virtuální realitě pro účely výuky. Master's thesis, Fakulta Aplikovaných Věd Západočeské Univerzity v Plzni, Plzeň, 2022. Vedoucí diplomové práce Libor Váša, Doc. Ing. Ph.D.

[10] Pramoditha, R. *Image Compression Using Principal Component Analysis (PCA)* [online]. Towards Data Science, 2021. [cit. 2022/06/12]. Dostupné z: `https://towardsdatascience.com/image-compression-using-principal-component-analysis-pca-253f26740a9f`.

[11] Rus, J. – Váša, L. Analysing the Influence of Vertex Clustering on PCA-Based Dynamic Mesh Compression. s. 55–66, 01 2010.

[12] Said, A. *Introduction to Arithmetic Coding - Theory and Practice* [online]. HP, 2004. [cit. 2022/04/28]. Arithmetic coder introduction. Dostupné z: https://hpl.hp.com/techreports/2004/HPL-2004-76.pdf.

[13] Smith, L. I. A tutorial on principal components analysis. 2002.

[14] *Class Mesh* [online]. Unity Software, 2021. [cit. 2022/04/15]. Unity scripting API Documentation. Dostupné z: https://docs.unity3d.com/ScriptReference/Mesh.html.

[15] Váša, L. *Mesh Compression* [online]. Váša, L., 2022. [cit. 2022/06/22]. Homepage of Libor Vasa. Dostupné z: http://meshcompression.org/.

[16] Váša, L. – Dvořák, J. Error propagation control in Laplacian mesh compression. *Computer Graphics Forum*. 2018, 37, 5, s. 61–70. doi: https://doi.org/10.1111/cgf.13491. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13491.

[17] Váša, L. – Skala, V. COBRA: Compression of the Basis for PCA Represented Animations. *Computer Graphics Forum*. 2009, 28, 6, s. 1529–1540. doi: https://doi.org/10.1111/j.1467-8659.2008.01304.x. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2008.01304.x.

[18] Váša, L. – Skala, V. CODDYAC: Connectivity driven dynamic mesh compression. s. 1 – 4, 06 2007. doi: 10.1109/3DTV.2007.4379408. ISBN 978-1-4244-0722-4.

[19] Vasa, L. – Skala, V. A spatio-temporal metric for dynamic mesh comparison. In *International Conference on Articulated Motion and Deformable Objects*, s. 29–37. Springer, 2006.