FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Animace humanoidního 3D modelu
Student:	Oldřich Linhart
Vedoucí:	Ing. Jiří Kubišta
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

1. Vypracujte přehled současných řešení animací humanoidních 3D modelů.
2. Navrhněte postup přípravy 3D modelu pro následnou animaci, včetně vytvoření kostry.
3. S využitím Vámi zvoleného řešení implementujte sadu alespoň pěti základních animací končetin a hlavy humanoidního 3D modelu sbírkového předmětu, vybraného po dohodě s vedoucím práce.
4. Vámi vytvořené animace demonstrujte v jednoduché aplikaci vytvořené ve Vámi zvoleném herním enginu.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 29. ledna 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Animace humanoidního 3D modelu

Oldřich Linhart

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Kubišta

4. června 2020

Poděkování

Rád bych poděkoval mému vedoucímu práce Ing. Jiřímu Kubištovi za cenné rady a připomínky. Taktéž bych rád poděkoval své rodině za podporu při studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 4. června 2020 . . . . . .

Odkaz na tuto práci

Linhart, Oldřich. *Animace humanoidního 3D modelu*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato práce se zabývá problematikou animace humanoidních postav. Cílem práce je seznámit čtenáře se základními pojmy a principy, které jsou spojené s vytvářením humanoidních 3D postav a jejich animací. Je popsán proces přípravy modelů k animaci, který je prakticky předveden na modelu, přiděleného od vedoucího práce. Pro model je vytvořená ovládací kostra a pomocí metod dopředné a inverzní kinematiky je demonstrován postup vytváření kosterních animací. Výsledkem práce je upravený přidělený model, připravený k vytváření animací, sada základních animací a jednoduchá aplikace v herním engine `Unity`, kde lze animace demonstrovat.

Klíčová slova humanoidní animace, rigging, kloubová soustava, kosterní animace, procedurální animace, přímá kinematika, inverzní kinematika, demonstrační aplikace, Blender, Unity

Abstract

This thesis is focused on humanoid character animation. The goal is to introduce the basic terminology and principles associated with creating and animating 3D characters to the reader. The whole process of preparing a character for animation is explained and demonstrated on model assigned by thesis supervisor. Skeleton of the model is created with rigging and skinning techniques. Creation of skeletal animation using forward and inverse kinematics is demonstrated. The result of this project is a model with an advanced rig, appropriate for animation. A set of basic animations is created and demonstrated in simple `Unity` application.

Keywords humanoid animation, rigging, articulated structure, skeletal animation, procedural animation, forward kinematics, inverse kinematics, demonstrational application, Blender, Unity

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Humanoid je bytost, která má vzhled připomínající člověka. Chodí po dvou nohách, má dvě ruce, trup a hlavu. Běžně se s nimi lze setkat v seriálech, filmech i počítačových hrách, ať už ve formě robota, elfa, trola, zlobra, nebo jiné fiktivní bytosti.

Animace postav kombinuje uměleckou tvorbu s precizním studiem pohybu živočichů a jejich anatomie. Vyžaduje i znalost fyziky, zejména mechaniky. Vytvoření přesvědčivé animace není jednoduché. Každá scéna musí vyprávět svůj vlastní příběh, každá postava musí mít vlastní charakter. Jedno zdvižené obočí může kompletně změnit dojem, který má postava vyjadřovat. Diváci animovaných filmů ocení plynulé detailní animace u kterých lze odhadnout osobnost postavy čistě z toho, jak se pohybuje. Hráči zase chtějí, aby se postavy pohybovaly responzivně. Chtějí mít kontrolu nad tím, co a kdy jejich postava dělá. Pokud bojují s příšerou, chtějí z pohybu poznat, kdy bude útočit. Chtějí, aby jejich postava interaktivně manipulovala s herním světem. Animátoři musí spolupracovat s programátory, aby našli mezi ovládáním a animací kompromis.

Z důvodu stále většího rozvoje hardwarové technologie mají animátoři stále větší volnost ve vytváření technologií a algoritmů, které pomáhají postavy oživit ve virtuálním světě.

Tato práce se zaměřuje na celý proces, od přípravy postavy, k metodám samotné animace a využití těchto animací v počítačových hrách. Práce obsahuje kapitolu o reprezentaci postav v počítačové grafice, způsoby jejich tvorby, úpravy, manipulace a zásady nutné pro budoucí animaci. V další části jsou popsány obecné principy počítačové animace. Co to je vlastně animace, jakými způsoby se vytváří, jak se ovládá, jak se ukládá, atd. Další část se již zaměřuje na animaci humanoidních postav pomocí kosterní animace. Co je to kostra, jak se vytváří a k čemu slouží. Jak se pomocí kostry dá pohybovat s virtuální postavou, jaké jsou její možnosti a omezení. Důraz je kladen i na principy a algoritmy pro vytváření animací. Jak animátorovi ulehčit práci a zároveň mu nechat volnost pro kreativitu. V čem animátorovi může usnadnit práci

počítač. V poslední kapitole jsou zmíněna herní jádra a k čemu slouží. Využití některých těchto principů bude následně demonstrováno v praktické části.

Práce může posloužit studentům i nadšencům do počítačové grafiky jako úvod do problematiky humanoidních animací a seznámit je se základními pojmy a principy, které se v tomto oboru běžně využívají.

Kapitola 1

Cíl práce

Cíl práce je seznámit čtenáře se základními teoretickými pojmy používanými v počítačové grafice a animaci. Prozkoumat a vysvětlit základní zásady a techniky, které se využívají při vytváření animace a vzájemně je porovnat.

V praktické části budou některé tyto techniky použity a demonstrovány na modelu přiděleném od vedoucího práce. Výsledkem bude model se sadou vytvořených animací a jednoduchá aplikace vytvořená v herním engine, která umožní tyto animace demonstrovat. Proces začne od samotné úpravy přiděleného modelu, který je výsledkem fotogrammetrie. Oddělování dílčích částí modelu, vytváření chybějících částí modelu a následná úprava textur modelu. Následovat bude analýza tvorby kostry pro humanoidní modely a návrh kostry pro přidělený model. Bude proveden skinning modelu ke kostře. Dále bude vytvořeno rozhraní pro dopřednou a inverzní kinematiku pro manipulaci s modelem. Pomocí této kostry bude vytvořena sada základních animací pomocí klíčování, například chůze a běh. Model s animacemi budou exportovány do herního engine, kde bude vytvořena jednoduchá aplikace.

V aplikaci bude možné s modelem pohybovat a na základě uživatelského vstupu animace přehrávat. Dále bude procedurálně použitá inverzní kinematika pro interaktivní úpravu animací, aby postava v reálném čase reagovala na své měnící se prostředí. Aplikace bude obsahovat jednoduché menu a prostředí vhodné pro demonstraci jednotlivých animací.

Kapitola 2

Teoretická část

V první sekci této kapitoly jsou uvedeny způsoby reprezentace a vytváření 3D modelů. Druhá sekce se zabývá počítačovou animací a popisuje, jaké existují typy animace, jak se vytváří a jak se ukládají. Třetí sekce se zaměřuje na konkrétní typ animace – kosterní animace. V poslední sekci jsou stručně popsána herní jádra, jejich význam a využití při vytváření aplikací.

2.1 Reprezentace modelů ve 3D

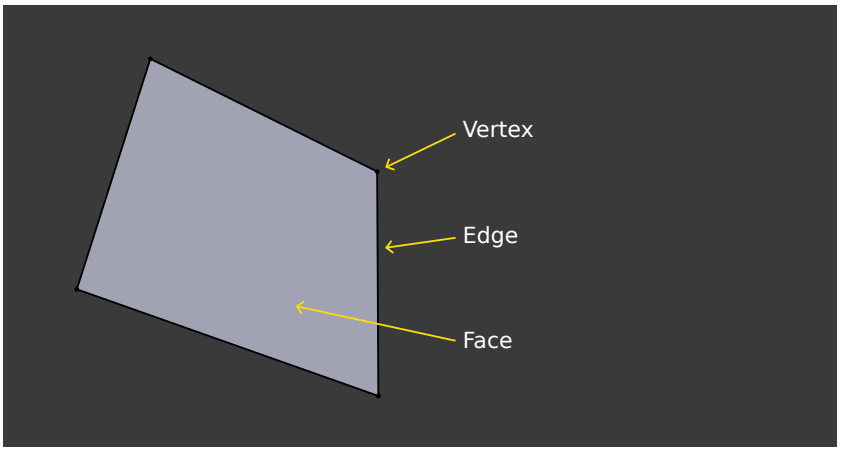
Ve 3D grafice se dají objekty reprezentovat několika způsoby. Tři nejčastější reprezentace jsou dle [1] polygonová síť, ve které jsou objekty reprezentovány jako množina bodů v trojrozměrném prostoru s určitými vlastnostmi. Další reprezentací jsou plochy tvořené množinou křivek, což může být vhodné zejména pro modelování hladkých povrchů. Třetím způsobem jsou dělené povrchy (*subdivision surfaces*), které jsou kombinací polygonové sítě a křivek. Každý způsob má své výhody i nevýhody a volí se na základě požadovaného využití objektu. Tato práce se zaměřuje na modely tvořené množinou bodů, které tvoří polygonovou sít.

2.1.1 Polygonová síť

V této reprezentaci jsou modely tvořeny polygonovou sítí, která reprezentuje povrch modelu. Polygon (mnohoúhelník) je dle [2] tvořen ze čtyř základních komponent:

Vrchol (bod, *vertex*), který je nejelementárnější prvek polygonu. Je to bod v prostoru, jehož poloha je určena souřadnicemi. Vrcholy lze spojovat a tím vytvářet složitější struktury.

Hrana (*edge*) je úsečka, která spojuje dva vrcholy.



Obrázek 2.1: Struktura polygonu, normála určuje orientaci plochy [3]

Plocha (*face*) vyplňuje prostor mezi množinou vrcholů, které jsou spojeny hranou. Nejjednodušší plocha je tedy tvořena množinou tří vrcholů, které dohromady tvoří v prostoru trojúhelník.

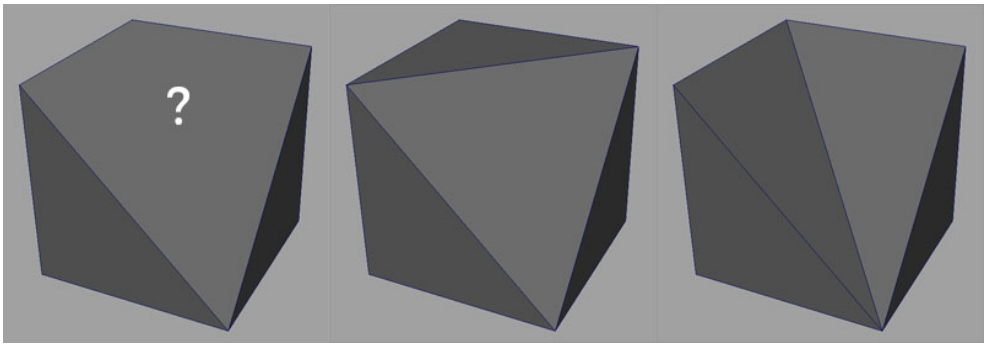
Normála (*normal*) je přímka, která je kolmá na plochu (nebo v rovině na hranu). Směr normály se nazývá normálový vektor a v grafice se používá například pro určení orientace plochy (to je důležité například proto, že některé softwary zadní stranu ploch vůbec nevykreslují).

Polygon obecně může mít libovolný počet hran. Množiny polygonů, které sdílejí vrcholy a hrany tvoří polygonovou síť. Polygonu se čtyřmi hranami se říká *quad (quadrilaterals)*. Polygonu, který má více než 4 hrany se říká *n-gon*. Polygon se čtyřmi a více hranami nemusí být nutně rovinný, čili všechny jeho vrcholy nemusí ležet v jedné rovině. Zároveň může být konvexní, ale i konkávní (jeho vnitřní úhly mohou být větší než 180 stupňů). [4]

Dle [6] je kvůli svým vlastnostem jako základní stavební prvek většinou využíván trojúhelník. Pro jakýkoliv trojúhelník totiž platí, že je konvexní a zároveň rovinný. To vede k tomu, že lze nad trojúhelníky provádět mnoho optimalizovaných výpočtů, například výpočet průsečíku paprsku s trojúhelníkem.

Dále je jejich vykreslování podporováno grafickými procesory (GPU), které jsou specializované na paralelní výpočty. GPU efektivně zvládá mnoho operací při vykreslování, jako počítání s čísly s pohyblivou řadovou čárkou, interpolace, maticové operace, sledování paprsku (*ray tracing*), atd.

Proto se dle [5] běžně v grafickém hardwaru všechny komplexnější tvary nejdříve převedou na trojúhelníky, než jsou poslány na rasterizaci. Jakýkoliv polygon se dá dekomponovat na množinu trojúhelníků, například přidáním úhlopříček. Tomuto procesu se říká triangulace. Způsobů provedení triangulace je více, s rozdílnými výsledky, viz obrázek 2.2. Tím pádem někdy může být vhodné provést triangulaci manuálně. [2]



Obrázek 2.2: Dvě možnosti triangulace nerovinného čtyřúhelníku [7]

Datovou strukturu, která popisuje 3D model, lze dle [6] rozdělit na geometrickou a topologickou část. Geometrická část obsahuje souřadnice jednotlivých vrcholů. Topologická část určuje jejich vztah, například které vrcholy tvoří trojúhelník, případně hrany, nebo které vrcholy jsou samostatné body. V ideálním případě je vhodné data uspořádat tak, aby každý vrchol byl zpracován právě jednou, například tím, že se trojúhelníky uspořádají do takzvaného pruhu trojúhelníků (*triangle strip*) nebo vějíře trojúhelníků (*triangle fan*).

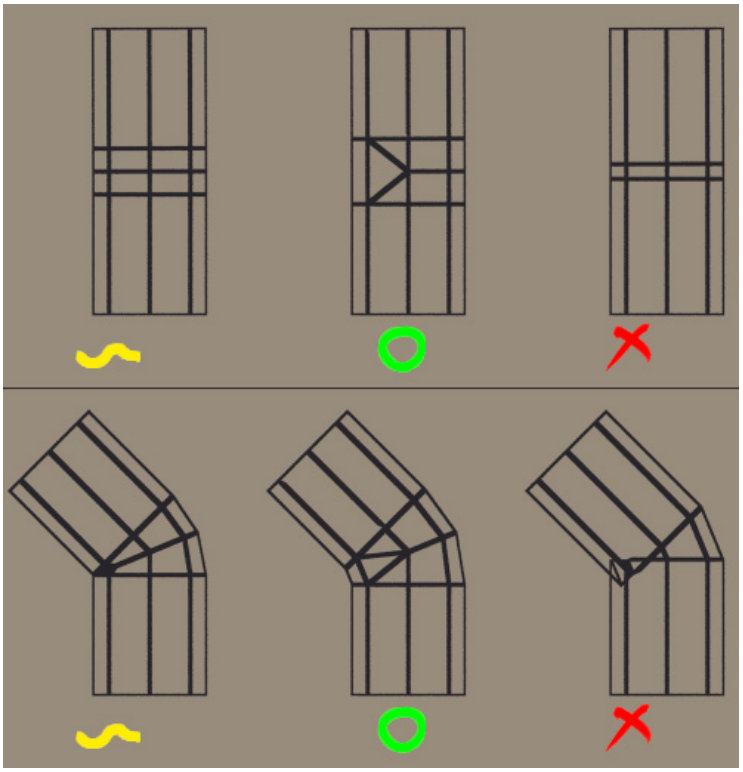
2.1.2 Topologie

Dalším důležitým pojmem je topologie. Obecně má tento pojem více významů, avšak ve 3D počítačové grafice popisuje rozložení polygonů, tedy jejich strukturu a distribuci na 3D modelu. Topologie ovlivňuje zejména to, jak snadno lze model upravovat a přidávat další geometrii, jak snadno se dají namapovat textury a jak se model deformuje při animaci. [8]

Cílem správné topologie je, aby měl model dostatečné detaily a přitom na dosažení těchto detailů bylo použito co nejméně polygonů. Zároveň v místech, kde se bude část modelu pohybovat a ohýbat, je vhodné mít více polygonů, aby se neprojevovaly deformace. Taková místa jsou typicky v kloubech a na obličeji modelu. [9]

Obecně je dle [2] a [10] v modelovací fázi vhodné, aby se model skládal ze čtyřúhelníků (*quadrilaterals*/*quads*), uspořádaných do podobně velkých a co nejvíce čtvercových ploch. Trojúhelníky lze využívat pro spojení tvarů v místech, které jsou skryté nebo kde se model při animaci příliš nedeformuje. Důvodem tohoto omezení je fakt, že trojúhelníky (a *n-gony*) se při animaci deformují nepředvídatelně a mohou vytvářet ostré hrany, boule a jiné artefakty. Na deformaci je třeba si dávat pozor i kvůli texturám, které se při deformaci mohou natahovat či smršťovat.

Dalším důvodem je, že většina modelovacích programů má nástroje, které podporují práci především se čtyřúhelníky, a proto je snazší a rychlejší upravo-

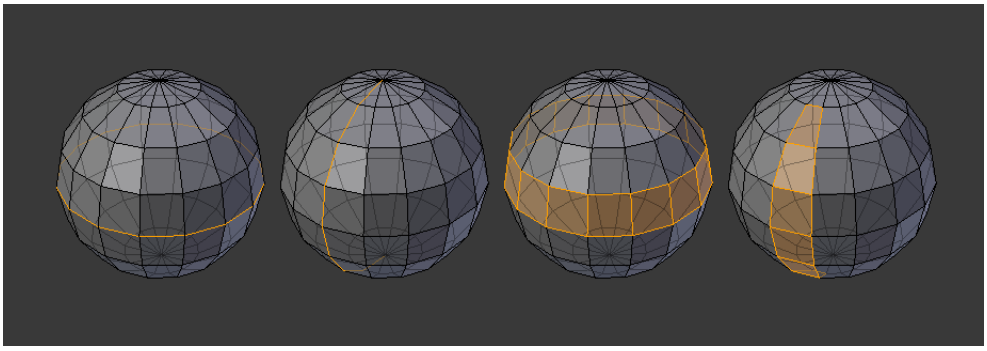


Obrázek 2.3: Deformace modelu v závislosti na topologii [11]

vat model se správnou topologií. Důležitý je zejména správný tok hranových smyček (*edge loop flow*). Hranová smyčka (*edge loop*) je souvislá linie hran, která na modelu tvoří smyčku a napojuje se zpět na sebe. Umístění těchto smyček je důležité, protože čím lépe jsou umístěny, tím méně vrcholů je potřeba pro vyjádření požadovaného tvaru. Zároveň určují, jak se model bude při animaci deformovat, viz obrázek 2.3. Z toho důvodu je u organických modelů důležitá jejich anatomie. Tok smyček by měl respektovat umístění a směr jednotlivých svalů. To zajistí, že při animaci se model deformuje přirozeně. Dobrý příklad je lidský úsměv. Při úsměvu se rohy úst vytáhnou, tváře vyboulí a oči zvrásní [1]. Na obrázku 2.4 lze vidět, že druhý a čtvrtý model neobsahují uzavřené smyčky, jelikož nejde implicitně určit, jakým směrem má smyčka pokračovat. Převod existujícího modelu do správné topologie se nazývá retopologizace.

2.1.3 Level of Detail

Při modelování je velice důležitý celkový počet polygonů. Čím více polygonů, tím mohou být na modelu zaznamenány jemnější detaily, ale zároveň pro vykreslování a související výpočty takového modelu bude zapotřebí vyšší výpočetní síly a model bude zabírat více paměti. Pokud jsou modely určeny



Obrázek 2.4: Smyčky hran a smyčky ploch [3]

pro využití například v počítačových hrách, bude existovat horní hranice celkového počtu polygonů vykreslovaných objektů ve scéně, aby bylo možné modely v reálném čase vykreslovat. [1]

Pro zjednodušování scény se používá pojem *Level of Detail (LoD)*, což lze přeložit jako úroveň či stupeň detailu. S nejvyšším *LoD* má model nejjemnější detaily. S nižším *LoD* se postupně detaily ztrácejí a s nejnižším *LoD* má model co nejjednodušší reprezentaci, která je přijatelná v kontextu využití modelu. [6] Detailní modely s hustou polygonovou sítí se nazývají *high poly (high polygon)*, naopak modely s řídkou polygonovou sítí se nazývají *low poly*.

Jako příklad uvedený v [12] je model kancelářské budovy s miliony polygonů, která má vymodelovaný interiér i exteriér. Interaktivní vykreslování ulice plné takových budov by při současné výkonnosti procesorů nebylo možné. Proto je při měnících se podmínkách vykreslované scény vhodné měnit i *LoD* modelů ve scéně. Tyto podmínky mohou být dle [6] například vzdálenost vykreslovaného modelu od kamery, počet pixelů, které vykreslený model na obrazovce zabírá, celkový počet polygonů vykreslených na obrazovce, nebo rychlost pohybu modelu.

Přístup implementace *LoD* se dle [6] rozděluje na spojitý *LoD (CLoD – Continuous LoD)* a diskrétní *LoD (DLoD – Discrete LoD)*.

U *CLoD* dochází ke změně modelu v malých krocích, například přidáváním a odebíráním jednotlivých vrcholů nebo trojúhelníků. Proces zjednodušování modelů se nazývá decimace. Tento přístup je vhodné provést v předzpracování, aby se náročné výpočty nemusely provádět až při vykreslování. Pro uložení modelu se využívají složitější a objemnější datové struktury, například stromové struktury trojúhelníků, aby bylo možné efektivně vyhledávat trojúhelníky potřebné pro zadaný *LoD*.

Jednodušší přístup je *DLoD*, při kterém se vytváří několik separátních modelů stejného objektu s různým množstvím detailů. Vytvořit tyto modely lze opět pomocí decimačních algoritmů modelu se všemi detaily, avšak někdy je lepší vytvořit jednotlivé modely ručně, čímž si sám modelář může určit, jaké detaily jsou více či méně důležité. Zároveň má kontrolu nad výslednou

topologií zjednodušeného modelu.

Zejména u *DLoD* se může projevit efekt zvaný *popping*, který je způsobený skokovou změnou geometrie modelu ve chvíli, kdy se objektu změní *LoD*. Tento efekt lze zmírnit vykreslením objektu dvakrát, jednou se současným *LoD* a jednou s novým *LoD* a mezi těmito snímky přes krátkou přechodnou dobu interpolovat přes alfa kanál. Druhou možností je takzvaný geomorfing (*geomorphing*), který provádí interpolaci mezi modely na úrovni jednotlivých vrcholů. [13]

Vzdálené objekty mohou být také aproximovány texturou, namapovanou na jednoduchý, plochý polygon. Těmto objektům se říká *billboard* a mohou být použity jako nejnižší *LoD* objektu. Jsou zejména vhodné na využití u rotačně symetrických objektů, jako například stébla trávy nebo skupiny listů na stromě. Obvykle se tyto polygony natáčí tak, aby vždy byly plochou kolmo ke kameře [12]. Další variantou je takzvaný *sprite*. Rozdíl je v tom, že *sprite* je tvořen z několika navzájem protínajících, různě otočených polygonů. Na každém polygonu je textura objektu z odpovídajícího pohledu. Zároveň se tyto objekty neotáčejí směrem ke kameře. [6]

2.1.4 Sculpting

Polygonové modelování dává absolutní kontrolu nad vytvářeným modelem, protože umožňuje manipulaci s každým samostatným vrcholem. Nevýhodou je, že takové modelování může být zdlouhavé a pracné, zvlášť, pokud je model veliký a složitý.

Sculpting umožňuje dle [14] simulovat sochařství z hlíny. Místo plné kontroly nad jednotlivými vrcholy nabízí rychlý a intuitivní způsob modelování. Umožňuje upravovat, přidávat a odebírat vrcholy, jako kdyby to byla keramická hlína. Modelář může využívat operace jako tahání, tlačení, vyhlazování hmoty, apod.

Tyto operace se aplikují na skupiny vrcholů v oblasti zájmu (*area of interest*). Ta má většinou sférický tvar a modelář si může nastavovat průměr. Důležitá vlastnost je tvrdost/síla nástroje, pomocí které se určuje, jak moc jsou vrcholy v oblasti zájmu ovlivněny. Čím nižší tvrdost, tím méně jsou ovlivněny vrcholy se zvětšující se vzdáleností od středu oblasti zájmu.

Sculpting se nejvíce využívá na modelování organických povrchů jako jsou postavy nebo oblečení.

Nevýhodou je dle [15] to, že modely vytvořené tímto způsobem nejsou vhodné pro animaci, kvůli vysokému počtu polygonů. Z toho důvodu se provádí takzvaná decimace, kterou se převádí komplexní model na jednodušší a snižuje počet polygonů. Příklad decimace modelu, spojenou s retopologizací lze vidět na obrázku 2.5.

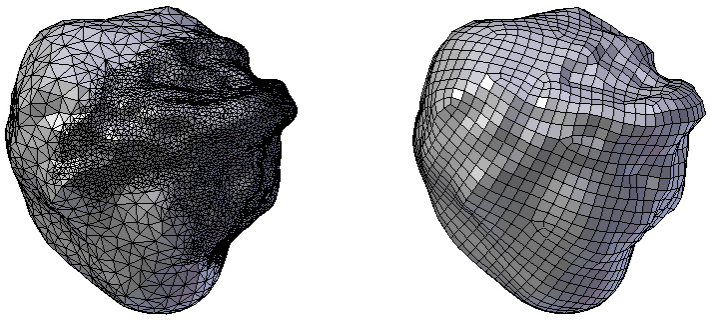
2.1.5 Fotogrammetrie

Fotogrammetrie je věda, která se zabývá extrakcí měření z dvojrozměrných obrázků, typicky fotografií. Výstup fotogrammetrie je typicky mapa, nákres, výsledek měření nebo 3D model nějakého reálného objektu či scény. [17]

Pomocí fotogrammetrických metod lze z fotografií budovy například změřit její výšku, délku nebo jiné přesné rozměry. Na to se využívá speciální fotogrammetrická metoda, stereofotogrammetrie. Ta je založená na principu stereoskopického vidění. Spočívá v odhadu 3D souřadnic bodů objektu pomocí měření provedených na dvou a více fotografiích objektu z různých pozic. Na stejném principu funguje i lidský zrak. Z levého a pravého oka získá mozek dva mírně odlišné obrazy ze dvou různých pozic a z těchto informací je schopen odhadnout vzdálenost mezi námi a ostatními objekty. [18]

V praxi se dle [19] fotogrammetrie často porovnává s metodou laserového skenování. Jedno skenování dokáže během krátkého času vyprodukovat velké množství dat, kterému se říká mračno bodů (*point clouds*), ze kterých lze vytvářet velmi přesné a detailní 3D modely. Na druhou stranu kvalitní skenery jsou obvykle velmi drahé a vyžadují speciální software na zpracování výsledných dat. Fotogrammetrie je typicky levnější a flexibilnější metodou, vzhledem k tomu, že k ní lze využít i řadové, levnější fotoaparáty. Kromě geometrických dat se dá zároveň získat i textury. Na vytvoření modelu je možné využít satelitní fotografie, fotografie pořízené drony nebo třeba podvodní fotografie. Navíc lze dle [18] fotogrammetrii použít i na rychle pohybující se objekty, což u laserového skenování nelze.

Na druhou stranu popisuje [20] i řadu omezení a problémů, kvůli kterým může být výsledný model necelý nebo zcela nepoužitelný. Jedním z problémů je okluze, kde jiný objekt zakrývá snímaný objekt. Komplexnější objekty mohou zároveň zakrývat samy sebe. To se dá řešit pořízením více fotografií s větším vzájemným překryvem v těchto místech. Další problém způsobují objekty s nevýraznou texturou, například prázdná rovná stěna, jelikož algoritmy pracují základě extrakce příznaků. Dále jsou problematické průhledné,



Obrázek 2.5: Decimace a retopologizace modelu [16]

lesklé a odrazivé objekty. Pro dobré výsledky je důležité i osvětlení. Blesk a silná směrová světla vytváří pro každou fotografii nežádoucí unikátní odrazy. Nejvhodnější světelné podmínky tvoří světlo ambientní, tedy všudypřítomné a všesměrové.

V závislosti na kvalitě a počtu fotografií je typicky výsledkem model s vysokým počtem polygonů. Proto stejně jako u sculptingu může být vhodné před dalším využitím modelu provést decimaci.

2.1.6 Zapékání textur

Dle [21] je zapékání textur *(texture baking)* proces ukládání specifických informací do textury. Ukládaná data mohou být například informace o barvě, osvětlení, odrazech, normálách, stínech, apod. To je významné, protože při vykreslování scény se zapečenými texturami nemusí tato data být počítány opakovaně na každém snímku.

Nevýhodou je, že zapečená textura je statická. Například zapečení stínu objektu vede k tomu, že se s tímto objektem nesmí ve scéně hýbat, jelikož pohyb objektu by nezpůsobil pohyb jeho stínu.

Zapékání nachází využití ve vytváření modelů s nižšími stupni detailů, viz sekce 2.1.3. Detailní informace lze na zjednodušený model přenést pomocí difuzních, normálových a dalších textur. Pomocí těchto metod lze mít ve scéně velmi detailně působící modely, přestože jsou složeny z relativně malého počtu polygonů.

2.2 Počítačová animace

Slovo animace se dá přeložit jako oživování/oživení. Prakticky se jedná o specifikaci (přímou či nepřímou), jak se objekty pohybují v čase a prostoru. Stejný význam má i v počítačové grafice. Základní myšlenka lpí v sekvenčním zobrazování po sobě jdoucích obrázků (*frame*) tak rychle, že je lidské oko vnímá jako plynulý pohyb.

Plynulost animace záleží částečně na tom, kolik snímků zařízení zobrazuje za jednotku času. To určuje takzvaná snímková frekvence (*frame rate*), která je obvykle udávaná v jednotkách snímků za vteřinu (*frames per second – fps*). Toto číslo se liší v závislosti na oblasti využití animací. Televizní formáty PAL například používají standardně 25 *fps*, NTCS formáty 29,97 *fps*. Filmy mají skoro výhradně 24 *fps* a počítačové hry mohou mít i stovky *fps*. [22]

Svět počítačové animace se dá rozdělit do dvou hlavních kategorií, a to do 2D animace (dvourozměrná) a 3D animace (trojrozměrná). Tato práce je zaměřena pouze na techniky 3D animace.

Vytvářet animace lze například pomocí klíčových snímků (*keyframe*), které vytváří animátor a software mezi nimi provádí interpolaci. Další způsob je snímání pohybu (*motion capture*), při kterém se nahrává pohyb skutečného

objektu a převádí na pohyb virtuálních 3D objektů. Animovat se dá i pomocí procedurálních metod, při kterých programátor stanoví řadu podmínek a parametrů, na základě kterých se objekt pohybuje a mění. [22]

3D počítačovou animaci [23] rozděluje na nízkoúrovňovou (*low level*) a vysokoúrovňovou (*high level*). Toto rozdělení poukazuje na výpočetní rozdíly mezi způsoby vytváření animace. V nízkoúrovňové animaci má animátor velmi precizní kontrolu nad vytvářeným pohybem. Do této kategorie spadá zejména pohyb objektů po určité dráze, jeho rychlost, směr, apod. Dále do ní patří již zmíněný pohyb na základě klíčových snímků. Vysokoúrovňová animace simuluje pohyb na základě množiny pravidel, omezení a počátečních podmínek, které určují co se má stát, namísto toho, jak se to má stát. Pomocí těchto metod se dá například nastavit, aby postava sledovala pohybující se cíl. Tyto operace lze dekomponovat na kolekce nízkoúrovňových operací.

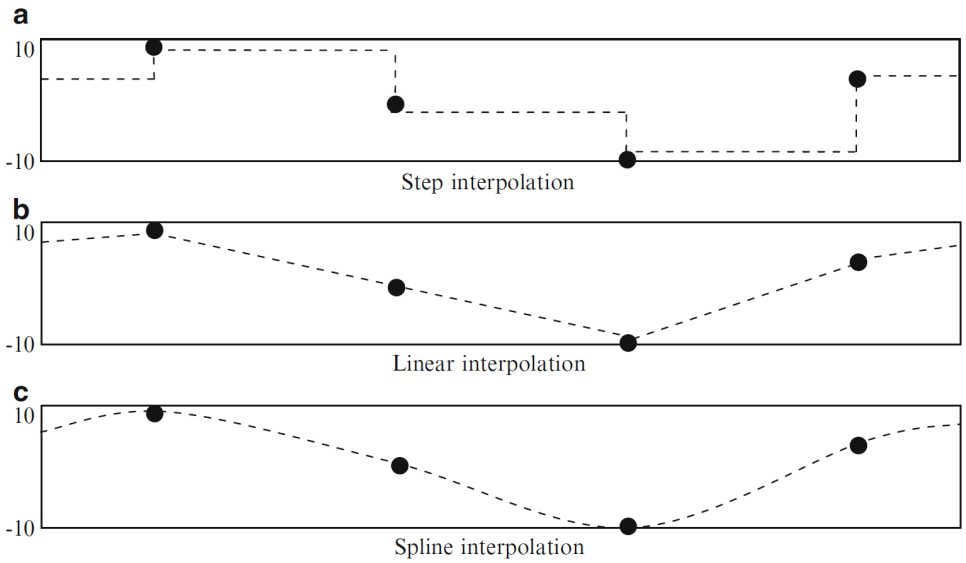
2.2.1 Keyframing

Jedním z nejvyužívanějších způsobů tvorby animací je klíčování (*keyframing*). Základním prvkem tohoto druhu animací je klíčový snímek (*keyframe*). Ten vyjadřuje konkrétní informace o objektu v konkrétním snímku. Do toho může spadat jak poloha a póza objektu, tak například jeho barva, průhlednost, textura, atd. Jak napovídá název, tyto snímky by měly obsahovat klíčové polohy v animaci, například počáteční a koncové pozice pohybu. Při animaci pálkaře odpalujícího míček by klíčové snímky byly například v nápřahu, při střetnutí pálky s míčkem a v koncové pozici po odpalu. Mezisnímky vytvoří software automaticky procesem zvaným *tweening* nebo *in-betweening*, který ve 3D grafice snímky dopočítá pomocí interpolace. [24]

V animaci není vhodné vždy využívat lineární interpolaci (obr. 2.6b), jelikož je nespojitá v první derivaci. To znamená, že v bodech napojení dvou segmentů si jejich tečné vektory nejsou rovny, což působí skokové změny v rychlosti. Vizuálně to vede k tomu, že se pohyb zdá trhaný nebo nerealistický, protože objekty v reálném světě se typicky nepohybují lineárně. Například padající objekt z výšky postupně zrychluje. Někdy může být žádoucí i skoková změna (obr. 2.6a), například pro probliknutí žárovky. Nejčastěji se využívají interpolační křivky (obr. 2.6c), které může animátor ovládat a měnit pomocí řídících bodů. Animační programy typicky mají pro tyto účely grafové editory. [23]

Možnost přidávat a volně upravovat klíčové snímky dává animátorovi velkou volnost k vyjádření cílového pohybu. Zároveň se tím zvyšují nároky na dovednosti animátora, aby tyto pohyby vyjádřil přirozeně. [24]

Nejvíce přímočarý způsob jak reprezentovat animaci objektů je specifikovat separátní model na každém klíčovém snímku a animovat na bázi samostatných vrcholů. Tomu se říká animace vrcholů (*vertex animation*). Pro reprezentaci objektů se složitější, pohyblivou strukturou a jejich animaci se častěji využívá kloubová soustava (*articulated structure*), popsaná v sekci 2.3.1.



Obrázek 2.6: a) kroková; b) lineární; c) křivková interpolace klíčových snímků [30]

2.2.1.1 Vertex animation

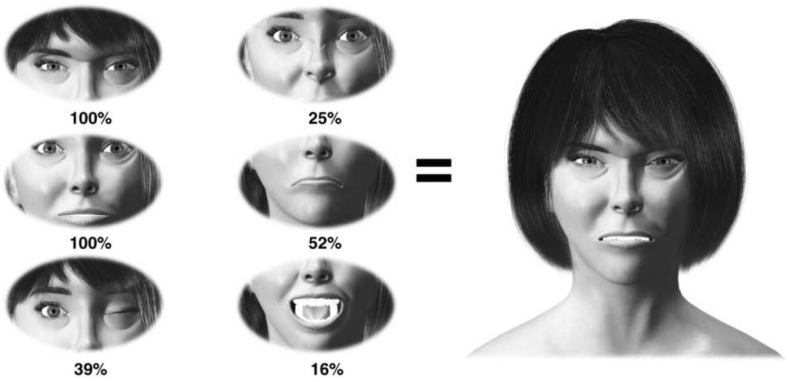
Tento způsob animace přináší nejvyšší stupeň volnosti pro animátora. Topologie modelů se typicky při běhu animace nemění, tedy se nemění počet vrcholů ani jejich vzájemné seskupení do hran a ploch. Animace spočívá v tom, že se ukládá informace o změně pozice každého jednotlivého vrcholu modelu v každém klíčovém snímku. To vede k tomu, že lze vytvořit jakákoli deformace modelu. [25]

Nevýhodou je dle [12] to, že takové modely jsou velmi složitě kontrolovatelné a vytváření klíčových snímků je pracné a časově náročné. Druhou nevýhodou je datová a výpočetní náročnost. Animační data musí obsahovat informace o poloze každého vrcholu v každém klíčovém snímku a musí se provádět lineární počet interpolačních operací v závislosti na počtu vrcholů modelu.

Jako další problém [12] uvádí, že neexistují omezení pro uchování objemu nebo povrchu části modelu, což může vést k ošklivým deformacím modelu, například při otočení objektu o 180° dojde v polovině interpolace ke zplacatění (všechny vrcholy jsou „na půl cesty“).

2.2.1.2 Morph target animation

Dalším druhem animací je tzv. *morph target animation* (někdy také nazývaná *morph targets*). Jedná se o variaci techniky animace vrcholů. Animátor vytvoří kolekci poloh a animace je tvořena mícháním dvou a více těchto poloh za běhu,



Obrázek 2.7: Morph targets [4]

viz obrázek 2.7. Pro výpočet poloh jednotlivých vrcholů na daném snímku se využívá lineární interpolace. [26]

Tato metoda nachází využití u měkkých či ohebných objektů, například kůže nebo tkanina. Typickým využitím je animace obličeje, protože umožňuje vytvářet detailní výrazy pro vyjádření emocí postavy. Polohy mohou reprezentovat například úsměv, zavřené oko, zdvižené obočí, apod.

2.2.2 Procedurální animace

Tradiční animace řízená klíčovými snímky umožňuje animátorovi vyjádřit libovolné pohyby. Avšak některé animace jsou příliš složité na ruční animování. Do toho spadají částicové systémy například na animaci mraků, vody, ohně, kouře nebo třeba vlasů či srsti. Také tam patří animace flexibilních objektů, jako je oblečení, nebo dynamiky tuhých těles, které berou v potaz parametry jako hmotnost, orientace, točivé momenty, lineární a úhlové rychlosti, apod. [27]

Takové animace může být velice obtížné vyjádřit pomocí klíčových snímků. Daleko lépe se dají sestavit algoritmy, které pohyb řídí pomocí matematických modelů. To umožňuje vytvářet například fyzikálně založené simulace, které využívají fyzikální zákony pro simulaci pohybu. Taková animace působí přirozeně, protože velmi dobře napodobuje chování objektů v reálném světě. Navíc si animátor může tyto zákony libovolně upravovat pro dosažení specifických výsledků. Nevýhodou je, že pro vytvoření takových algoritmů je zapotřebí hlubokého pochopení využitých modelů, typicky matematiky, fyziky, strojírenství, atd. [24]

Velice široké využití má procedurální animace i při animaci postav ve videohrách. Postava může reagovat na svoje okolí a akce v jejím okolí. Například může dynamicky balancovat na úzké plošině nebo lézt po stěně. Změnou vstupních parametrů se dají animace snadno modifikovat. Je nesmyslné, aby postava měla desítky animací chůze v závislosti na úhlu nahnutí podlahy.

Při chůzi se může postava procedurálně přizpůsobit nerovnému povrchu, při švihnutí mečem lze zaměřit cíl ve variabilní výšce. Listy na stromě se mohou třást v závislosti na nastavitelné rychlosti a směru větru.

Ačkoliv lze pomocí procedurálních metod vyvinout pohyb bez použití klíčových snímků, takto vyvinuté animace postrádají účinnou kontrolu nad stylem a osobností dané postavy. Proto se dle [28] často u herních postav využívají polo procedurální metody (*semi-procedural*), které využívají standardní animace s klíčovými snímky, rozšířené o procedurální metody. Například chůze může být řízena klíčovými snímky, ale v moment, kdy noha dopadá na podlahu, se její konečná poloha a rotace procedurálně upraví tak, aby seděla k nerovnosti povrchu.

2.3 Kosterní animace

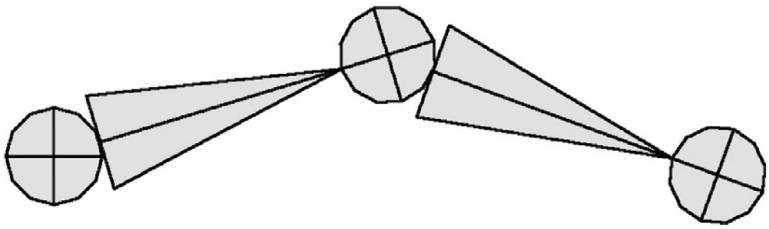
Velká část počítačové animace se zaobírá animací postav. Tyto postavy mohou být humanoidního typu, zvíře, robot nebo jiná virtuální stvoření. Takové postavy by bylo složité animovat pohybováním individuálních vrcholů. Zároveň ukládání animací pro takové modely je nepraktické, jak bylo uvedeno v sekci 2.2.1.1. Proto se části modelů kupí do skupin, které se pohybují jednotně, například lidská ruka může být složena z předloktí a nadloktí. Pro tyto modely se definuje takzvaná kostra.

2.3.1 Kostra

Kostra (někdy také nazývaná kloubová soustava) se v počítačové grafice skládá z jednotlivých kloubů/uzlů. Kloubům se někdy také říká kosti, ale z technického pohledu animátoři pracují s klouby a kosti reprezentují pouze prázdné místo mezi klouby. Obecně jsou dle [26] tyto termíny v počítačové grafice zaměnitelné. Běžnou vizualizaci lze vidět na obrázku 2.8. Animátor obecně nemusí být stejný člověk, jako ten, kdo vytváří kostru, v textu pro jednoduchost budou tyto role sjednoceny pod názvem animátor.

Tyto klouby jsou na sebe vázané a tvoří hierarchickou stromovou strukturu. To znamená, že pokud se pohne paže, s ní se zároveň pohne i předloktí a ruka, včetně prstů. Obecně změna transformace kloubu ovlivní všechny jeho podřazené klouby. Tato vlastnost se také dá nazvat vztahem rodič a potomek, například kloub v lokti je potomek kloubu v rameni a zároveň rodič kloubu v předloktí.

Pokud mají jednotlivé klouby konstantní vzdálenost a mění se pouze jejich rotace, nazývá se tato kloubová soustava rigidní (tuhá, nepružná). Pro humanoidní modely se dle [29] využívá rigidní kloubová soustava skoro výlučně. Jeden zvolený kloub se nazývá kořenový (*root*) a všechny ostatní klouby jsou jeho přímí či nepřímí potomci. Kořenový kloub nemá žádného rodiče. Pro humanoidní modely je typicky kořenový kloub v oblasti pánve a pomocí něj lze provádět transformace s celým modelem.



Obrázek 2.8: Typická vizualizace kostí a kloubů [1]

Samotná kostra nemá žádnou vlastní geometrii a nevykresluje se. Slouží pro usnadnění animátorovi s pózováním modelu, kterého je součástí. Výsledná póza je určena z úhlů mezi jednotlivými klouby. [26]

Polygonové síti, která tvoří povrch modelu, se říká kůže. Na klouby jsou napojené jednotlivé části kůže modelu, která se pohybuje relativně k pohybu kloubů. Místo toho, aby animátor pro vytváření animací ovládal jednotlivé vrcholy, pohybuje s kostrou modelu. V klíčových snímcích stačí ukládat pozice jednotlivých kloubů, čímž se efektivně snižuje paměťová i výpočetní náročnost.

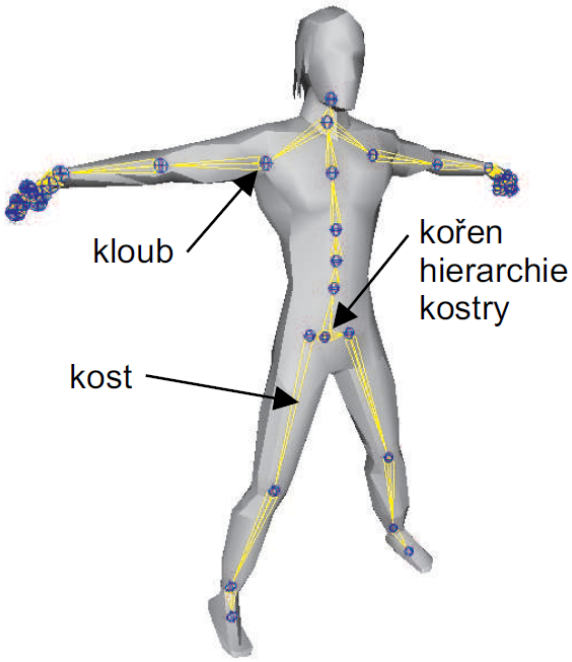
Animátor kostru pózuje tím, že jednotlivé klouby posouvá, rotuje, případně škáluje. V [26] je uvedeno, že každý kloub si ukládá informace o své poloze, rotaci, případně měřítku relativně k nějaké referenční póze (nejčastěji póze svého rodičovského kloubu). Tyto informace jsou typicky uloženy v matici afinní transformace o rozměru 4×4 nebo v SRT struktuře. SRT (*Scale, Rotation, Translation*) je datová struktura obsahující skalár nebo vektor pro škálu, kvaternion pro rotaci a vektor pro posunutí. Někdy se také nazývá SQT struktura (*Scale, Quaternion, Translation*). Výsledná póza celé kostry je určena polem těchto dat pro všechny klouby.

Základní póza kostry se nazývá *bind pose*. U humanoidních postav tato póza typicky připomíná písmeno T. Postava stojí vzpřímená s nohama u sebe a s rukama kolmo od sebe. Proto se někdy také nazývá *T-pose*. [26]

Další výhodou kosterní animace je, že pokud se animátor drží při tvorbě kostry určitých konvencí, lze animace z jedné kostry duplikovat do jiné kostry. Tato kostra může mít částečně jiné proporce, hierarchickou strukturu i počet kloubů. Tomuto procesu se říká *animation retargeting*.

2.3.2 Rigging

Vytváření hierarchické struktury kostí pro model se nazývá *rigging*. Při této fázi se animátor musí rozhodnout, jaké části modelu budou tvořit své individuální celky. Například lidské tělo má v dospělosti přibližně 206 kostí. Takový počet je pro animaci typicky příliš vysoký a tvoří příliš složitou strukturu. Animátor musí znát anatomii modelu a rozhodnout se, z kolika kloubů se bude výsledná kostra skládat v závislosti na tom, jak se bude postava pohybovat. Důležité otázky u humanoidních postav jsou například, z kolika kloubů



Obrázek 2.9: Příklad kostry pro humanoidní postavu [6]

bude tvořena páteř postavy nebo jestli postava bude moci hýbat s jednotlivými prsty u rukou a nohou. Příklad kostry člověka je uveden na obrázku 2.9.

Kostry se nevytváří jen v situacích, kde se vyskytují klouby v reálném světě. Kostru mohou mít i stromy a rostliny, například pro snazší animace působení větru. Dále se někdy využívají pro animaci obličeje nebo pro jiné sekundární animace, například zvířecí uši a ocas nebo části oděvu postav.

2.3.3 Degree of Freedom

Dle [6] lze pro každý objekt, čili i kostru nebo jednotlivé klouby, jednoznačně určit jejich polohu pomocí veličin zvaných stupně volnosti (*DOF – Degree of Freedom*). V trojrozměrném prostoru má každý nezávislý rigidní objekt 6 stupňů volnosti, tři pro pozici a tři pro rotaci v jednotlivých osách. Pokud jsou objekty seskupeny do segmentových struktur, jako je kostra, bude se jejich celkový počet stupňů volnosti snižovat. Například v rigidní kloubové soustavě je transformace kloubu určena jeho relativní transformací k rodiči a globální transformací rodiče. Zjednodušeně, pro jednoznačné určení polohy tohoto kloubu stačí tři stupně volnosti, které reprezentují relativní rotaci vzhledem k rodičovskému kloubu.

Někdy může být žádoucí stupně volnosti snížit manuálně a tím omezit pohyb kloubu. Lidské klouby mají přirozeně omezený pohyb. Například loketní

a kolenní klouby se mohou otáčet pouze ve směru jedné osy. Zároveň může být užitečné omezit v jednotlivých osách i rozsah hodnot, kterých může kloub nabývat. Taková omezení se nazývají *constraints*. Například loketní ani kolenní kloub nelze otočit o celých 360°. Nastavením těchto omezení lze animátorovi usnadnit práci a zamezit tomu, aby se modely dostaly do nepřirozené pózy. [29]

2.3.4 Skinning

V této kapitole bude pro snazší vizuální představu používán termín „kost" místo „kloub". Procesu přiřazení jednotlivých vrcholů kůže k jednotlivým kostem je proces zvaný *skinning* (někdy také nazývaný *binding*). Existuje řada způsobů jak toho docílit. Nejjednodušší případ je, když každý vrchol je ovlivněn pohybem pouze jedné kosti. To je přijatelné u rigidních modelů, které jsou tvořeny pevnými segmenty, jako jsou například roboti a stroje. Tomu se říká jednoduchý skinning. Každý vrchol se transformuje tak, jako kdyby byl pevně napojen na jednu kost. [6]

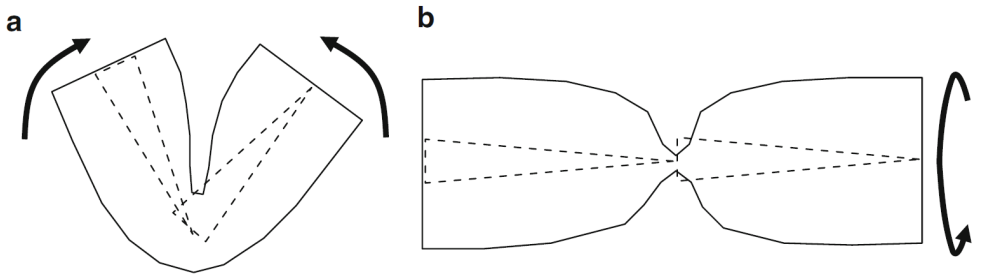
U organických modelů je toto řešení nedostačující. V místě, kde se dvě kosti setkávají se v takovém případě projeví nehezká deformace. Některé vrcholy jsou skřípnuté a kůže v oblasti ohybu není hladká. Lepších výsledků lze dosáhnout technikou míchání vrcholů (*vertex blending / smooth binding / linear blend skinning*). Základní myšlenkou této techniky je, že poloha jednoho vrcholu je ovlivněna vícero kostmi. Například ohnutí lidské ruky v lokti ovlivní nejen předloktí, ale i paži. [6]

Dle [22] má každá kost přidělenou hodnotu vlivu, jak moc daný vrchol ovlivňuje, takzvanou váhu. Váhu lze určit například v závislosti na vzdálenosti vrcholu od konkrétní kosti, nebo ji animátor může přiřadit ručně. Součet vah pro každý vrchol je typicky 1, neboli 100 % (ačkoliv to není nutná podmínka) a váha nikdy není záporná.

Pomocí této metody může být jeden vrchol ovlivněn libovolným počtem kostí. Avšak je nutné zmínit, že se zvyšujícím počtem ovlivňujících kostí se zároveň zvyšuje výpočetní náročnost a v praxi se běžně nedoporučuje, aby jeden vrchol ovlivňovaly více než 4 kosti. Tato metoda má ve většině případů dobré výsledky, ale pokud jsou rotační úhly oproti rodičovské kosti příliš velké, [30] popisuje dva typy nežádoucích deformací:

Collapsing elbow effect, který se objeví, když mezi sebou dvě kosti svírají malý úhel. Projevuje se tím, že vrcholy blízko vnitřní strany místa ohybu se propadnou ke středu kloubu a model tím vizuálně ztratí objem (obr. 2.10a).

Candy-wrapper effect, neboli efekt obalu od bonbónu je druhým typem. Tento efekt nastává, když se kost přetočí o 180°, například jako zápěstí při odemykání dveří. Projevuje se tím, že vrcholy na které oboje kosti působí podobnou váhou se přesunou blízko ke středu kloubu (obr. 2.10b).



Obrázek 2.10: a) Collapsing elbow efekt, b) Candy-wrapper efekt [30]

Jeden způsob zmírnění těchto efektů je rozdělit jednotlivé kostí na více částí. Například předloktí může být složeno ze tří kostí v jedné přímce. To umožní rozdělit hodnotu vlivu vrcholů mezi více kostí a zmírnit deformaci v extrémních rotacích jako na obrázku 2.10b.

Existují další a sofistikovanější třídy metod, které produkují lepší výsledky. Jedna z těchto tříd je stále lineární, ale lepších výsledků dosahuje tím, že zavádí větší počet vah pro každý pár vrcholu a kosti. Tyto metody se nazývají multi-lineární a mohou využívat i 12 vah pro každý pár vrcholu a kosti.

Druhou třídu tvoří nelineární metody, jako je *dual quaternion skinning*. Tato metoda produkuje lepší výsledky než metoda míchání vrcholů, ačkoliv je jejich výpočetní složitost porovnatelná. Paměťová náročnost je snížena ze dvanácti `float` reprezentující matici rigidní transformace, na osm `float` reprezentující dva kvaterniony. Tato metoda zabrání vzniku artefaktů, jako na obrázku 2.10. [31]

Pokud je vyžadovaný vysoký realismus, používá se i simulace svalů. Svaly jsou reprezentovány například přes *NURBS* (Non-Uniform Rational Basis Splines) křivky nebo povrchy, které jsou napojené na správných místech mezi klouby. Při pohybu kloubů mění svůj tvar a tím mění povrch kůže. Tímto způsobem lze vytvořit velice realistický muskuloskeletální systém. [1]

2.3.5 Kinematika

Studiu pohybu těles v prostoru a čase se zabývá obor fyziky zvaný mechanika. Podle vztahu k příčinám pohybu se mechanika dělí na kinematiku a dynamiku. Dynamika studuje vzájemné působení sil a objektů. Kinematika studuje pouze pohyb objektů, nezávisle na příčině pohybu, čili se zabývá pouze polohou, rychlostí a zrychlením v čase.

A právě kinematikou se zabývá část počítačové animace, ve které slouží k manipulaci s kloubovou soustavou. V souvislosti s kinematikou se dle [32] používá několik důležitých pojmů:

Spojení (*link*) je rigidní spojení mezi dvěma klouby. Řetězce spojů oddělené klouby tvoří kloubové soustavy. Tyto řetězce jsou zpravidla na jednom

konci pevně připevněny, například ruka je v rameni pevně připevněna k trupu.

Koncový efektor (*end effector*) je volný konec řetězce, v případě ruky se jedná o zápěstí nebo prsty.

Stavový vektor (*state vector*) je množina nezávislých parametrů, která definuje všechny možné stavy, do kterých se kloubová soustava může dostat. Jeho délka (dimenze) je stejná, jako počet stupňů volnosti celé kloubové soustavy (viz sekce 2.3.3). Stavový vektor se označuje

$$\Theta = (\theta_0, \theta_1, \ldots, \theta_n) \tag{2.1}$$

kde n je počet stupňů volnosti. Například nezávislý rigidní objekt v prostoru má 6 stupňů volnosti, tedy jeho stavový vektor je určen

$$\Theta = (x, y, z, \alpha, \beta, \gamma),$$

kde x, y, z jsou posuvy a α, β, γ rotace v jednotlivých osách.

V robotice existuje velký počet různých druhů kloubů, u kterých se dva *linky* pohybují relativně k sobě. V počítačové grafice se lze převážně setkat jen s několika druhy kloubů. První je posuvný kloub (*prismatic joint*), u kterého *link* mění svoji polohu relativně k druhému *linku*. Nejčastěji používaný je otočný kloub (*revolute joint*), u kterého jeden *link* rotuje po jedné určené ose. Tyto klouby mají jeden *DOF*. Existují i klouby s větším *DOF*, jako je *kulový kloub* nebo *Kardanův kloub*. Takové klouby lze vymodelovat tak, že se na stejné pozici uvažuje n kloubů s jedním *DOF*, které jsou spojeny $n-1$ *linkami* nulové délky. [23]

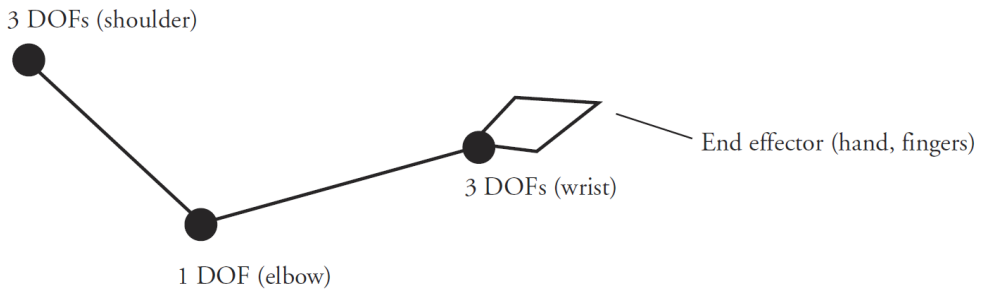
Stavový vektor jednoznačně určuje polohu koncového efektoru X. Existují dvě metody kinematiky, jak určit polohu koncového efektoru a jednotlivé hodnoty stavového vektoru. Dopředná kinematika (*forward kinematics*) a v dnešní době hojněji používaná inverzní kinematika (*inverse kinematics*). [6]

U dopředné i inverzní kinematiky se složitost výpočtu zvyšuje s počtem kloubů, jelikož každý kloub přidá do struktury minimálně jeden stupeň volnosti a tím pádem i minimálně jednu další dimenzi do stavového vektoru Θ. [32]

Na obrázku 2.11 lze vidět příklad kloubového řetězce reprezentující humanoidní ruku se sníženým počtem stupňů volnosti v lokti.

2.3.5.1 Dopředná kinematika

Dopředná kinematika (*Forward Kinematics – FK*) spočívá ve výpočtu polohy koncového efektoru postupným procházením řetězce od kořene až ke koncovému efektoru. V případě, že by animátor chtěl, aby postava uchopila objekt na stole, musel by nejdřív provést rotaci v rameni, pak v lokti, v předloktí a nakonec natočil jednotlivé prsty, aby držely objekt. V této metodě animátor



Obrázek 2.11: Řetězec kloubů reprezentující ruku humanoida [23]

pohybuje s více klouby a má větší volnost pro tvorbu expresivních animací. Zároveň je dopředná kinematika snazší na implementaci. Na druhou stranu tento postup může být zdlouhavý a v některých případech velmi obtížný. Pokud by postava měla v animaci udělat krok, na každém klíčovém snímku by animátor musel zkoušet a upravovat jednotlivé úhly v noze, aby ve všech snímcích chodidlo zůstalo na podlaze na stejném místě. Pokud bude koncová pozice na snímcích i trochu rozdílná, bude se chodidlo posouvat po podlaze, vznášet nebo propadat skrz podlahu. [1]

Výpočet polohy koncového efektoru X pomocí dopředné kinematiky lze napsat jako

$$X = f(\Theta). \tag{2.2}$$

Animátor postupně od kořene pózuje všechny klouby a poloha koncového efektoru je určena akumulací transformací jednotlivých kloubů. [32]

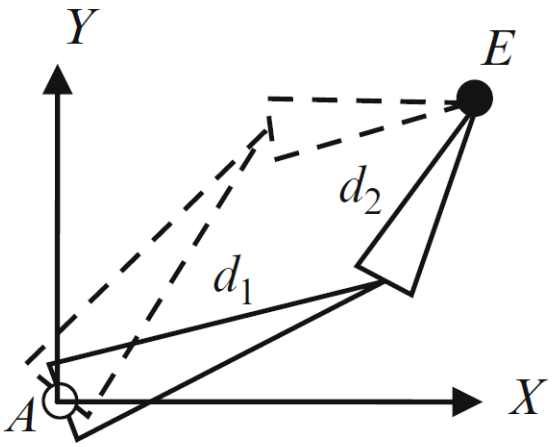
2.3.5.2 Inverzní kinematika

Inverzní kinematika (*Inverse kinematics – IK*) tento proces výpočtu otáčí. Jejím úkolem je se zadanou polohou koncového efektoru X nalézt jednotlivé hodnoty stavového vektoru Θ. Tento proces se dá popsat otázkou „Pokud postava bude mít ruku na tomto místě, jaké úhly budou svírat klouby v rameni a předloktí?“. Z tohoto důvodu se tato metoda nazývá také *cílem řízený pohyb (goal directed motion)* [32]. Formálně se tento vztah dá zapsat jako

$$\Theta = f^{-1}(X). \tag{2.3}$$

Nevýhodou *IK* je, že pro zadanou pozici koncového efektoru neexistuje vždy právě jedno řešení konfigurace stavového vektoru Θ. Řešení někdy nemusí existovat, nebo jich může být i dvě a více, jak lze vidět na obrázku 2.12. Omezit podprostor možných řešení lze částečně pomocí omezení (*constraints*), zmíněny v sekci 2.3.3. To například zajistí, že se klouby nemohou otáčet za určitou mez. Tímto se však podprostor řešení nezmenší vždy dostatečně.

Pokud je řetěz dostatečně krátký a jednoduchý, pak výpočet konfigurace stavového vektoru může být proveden analyticky. Avšak pokud je řetěz příliš



Obrázek 2.12: Více řešení při výpočtu inverzní kinematiky [30]

komplikovaný pro analytické výpočty, používají se iterativní metody. Iterativní metody využívají zjednodušené verze rovnic, které se opakovaně počítají a výsledky zpět dosazují, dokud se koncový efektor nedostane do cílové pozice. Na to se běžně používá takzvaná *inverze jacobiánu*, nebo jiné metody, podrobně popsané v [33]. *Jacobiho matice*, nazývaná také *jacobián*, je matice parciálních derivací, která vztahuje diferenciální změny vektoru Θ, zapsané jako $\Delta\Theta$, na diferenciální změny X, zapsané jako ΔX. Tento vztah lze dle [32] zapsat jako

$$\Delta X = \mathbf{J}(\Theta)\Delta\Theta, \tag{2.4}$$

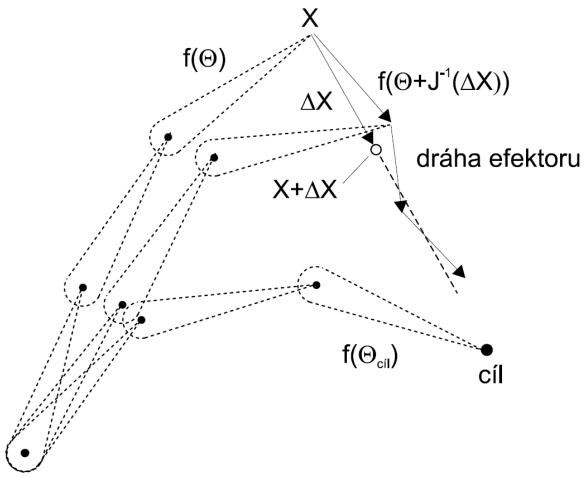
kde $\mathbf{J}$ je Jacobiho matice o rozměru $n \times m$, kde n je dimenze koncového efektoru X a m je dimenze stavového vektoru Θ. Tuto matici lze zapsat ve tvaru

$$\mathbf{J}_{m\times n} = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \cdots & \frac{\partial f_1}{\partial \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial \theta_1} & \cdots & \frac{\partial f_m}{\partial \theta_n} \end{bmatrix}. \tag{2.5}$$

Pokud existuje inverze jacobiánu, vzorec 2.4 lze upravit do tvaru

$$\Delta\Theta = \mathbf{J}^{-1}(\Theta)\Delta X, \tag{2.6}$$

ze kterého po dosazení změny souřadnic koncového efektoru ΔX lze vypočítat změnu stavového vektoru $\Delta\Theta$. Jelikož je jacobián závislý na aktuální konfiguraci stavového vektoru, jakmile se tato konfigurace změní, přestane vztah diferenciálních změn platit. To může vést k tomu, že pokud se zvolí příliš velký krok změny ΔX, koncový efektor se nemusí přibližovat ke svému cíli a je třeba volit menší změny. Tím pádem je potřeba více iterací, než se koncový efektor dostane do cíle, jak je znázorněno na obrázku 2.13. [23]



Obrázek 2.13: Iterační přibližování koncového efektoru ke svému cíli [6]

Zároveň typicky Jacobiho matice nemusí být čtvercová, čili neexistuje její inverze. V takových případech se dá vypočítat takzvaná *pseudo inverze (pseudo inverse)* $\mathbf{J}^+$. Formálně lze tento výpočet zapsat jako

$$\mathbf{J}^+ = (\mathbf{J}^T\mathbf{J})^{-1} = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}. \tag{2.7}$$

2.3.6 Motion capture

Další technikou vytváření animací je snímání pohybu (*motion capture – mocap*). Jak již název vypovídá, spočívá ve snímání a digitalizaci pohybu reálných objektů a následné převedení zaznamenaného pohybu na virtuální 3D model. Nejčastěji se používá na snímání pohybu lidských postav nebo zvířat, ale obecně lze snímat i jakékoliv jiné objekty. Využívá se jak v herním a filmovém průmyslu, tak i v medicíně, biomechanice, atd. Tato práce se zaměřuje pouze na snímání pohybu lidských herců.

Dle [23] a [22] je zaznamenávání nejčastěji založeno na dvou primárních metodách. První jsou elektromagnetické senzory, které jsou připevněny na klíčových místech, typicky kloubech. Senzory bezdrátově zasílají informace o své poloze a natočení centrálnímu procesoru, který informace zpracovává a ukládá. Výhodou je možnost snímat velikou plochu a jednoduchost přípravy. Nevýhodou je vysoká citlivost na zkreslení magnetického pole a šum v nahraných datech. Nahraná data nejsou tak přesná, což je zejména v medicíně a biomechanice důležitý faktor. Zároveň je složité nahrávat více než jednoho herce ve stejný čas.

Druhou metodou je systém značek, které má herec připevněné na oblečení. Tyto značky mohou být buďto pasivní – z reflexního materiálu, nebo aktivní – například svítící *LED*. Pohyb je zaznamenán pevně připevněnými kamerami z různých míst. V [22] je uvedeno, že pokud značku v daný moment

vidí alespoň tři kamery, pomocí triangulace lze vypočítat její polohu v prostoru. Výhodou této metody je, že lze nahrávat s vysokou snímkovou frekvencí i s levnějšími, řadovými kamerami. Vypočítané pozice značek jsou velmi přesné a nahrávat lze i více osob současně. Nevýhodou je dle [23] a [22], že je typicky potřeba využít více značek, než u elektromagnetických senzorů. Pokud kamery ztratí pohled na značky, nahraná data mohou obsahovat chyby a animátor musí později animace manuálně upravit. Zároveň je nahrávací plocha limitovaná svou velikostí, protože s větší vzdáleností se kamerám snižuje viditelnost značek.

Nahraná data jsou velice objemná. Typicky je vytvořen klíčový snímek pro každý nahraný snímek, závislý na *framerate*. Je vhodné, aby animátoři data vyčistili a zbavili se zbytečných klíčových snímků. Zároveň se sice postavy animované pomocí *motion capture* pohybují přesně a plynule, ale mohou působit mechanicky a ztuhle a animátor typicky musí nějaké pohyby manuálně zveličit a dodat jim hloubku. [2]

2.3.7 Míchání animací

Míchání animací *(animation blending)* spočívá v kombinování více animací současně, pro vyprodukování nové animace nebo pro vytvoření plynulého přechodu mezi jednotlivými animacemi. V prvním případě to může být kombinace animace chůze a animace běhu, pro vytvoření animace poklusu. Nebo mícháním dvou animací kopu v různých poměrech vytvořit desítky podobných, ale unikátních animací kopu. V druhém případě to může být rozeběhnutí postavy, tedy přechod z animace chůze do animace běhu. Animace, mezi kterými se vytváří přechod, je třeba dopředu plánovat už při vytváření animací, jinak může výsledný přechod vypadat nepřirozeně. [34]

Hojně využívaná je třída metod využívající lineární interpolaci *(LERP – linear interpolation)*. Výsledná pozice jednotlivých kostí je vypočítaná jako vážený průměr, kde váha určuje, jak moc mají jednotlivé animace přispívat do výsledku. Součet vah všech přispívajících animací by měl být jedna. Interpolace transformačních matic není praktické řešení, proto se interpolace provádí běžně zvlášť na každou komponentu SRT struktury, popsané v sekci 2.3.1. Vzhledem k tomu, že se interpolace provádí na jednotlivých kloubech, nezávisle na zbytku kostry, lze výpočet efektivně provádět paralelně pomocí GPU. Přechod mezi dvěma animacemi se může provádět překrytím dvou animací na časové ose a v místě překrytí provádět interpolaci. Tento způsob vyžaduje, aby míchané animace byly synchronizované, například aby pozice nohou v animaci běhu a v animaci chůze byly ve stejné fázi. Pro míchání animací, které nejsou synchronizované, má lepší výsledky metoda zmražení, při které se aktuální animace zmrazí a v ten moment se začne přehrávat cílová animace, do které se přes krátký časový úsek interpoluje ze zmražené pózy počáteční animace. [26]

Ve hrách se často využívá i míchání části kostry *(partial-skeleton blending)*. Spočívá v tom, že pro každý kloub lze určit jeho vlastní váhu míchání mezi animacemi. Tyto váhy lze uložit do takzvané míchací masky *(blend mask)*. Například pro animaci zamávání lze vytvořit masku, která přiděluje váhu 1 všem kloubům na ruce, kterou postava mává, a 0 na všech ostatních kloubech. Při LERP míchání animace chůze s touto animací vzniká animace, při které postava chodí a zároveň mává. Toto míchání může působit nepřirozeně, protože v reálném světě jsou tyto pohyby na sobě závislé. Další způsob je takzvané aditivní míchání *(additive blending)*, ve kterém se mezi dvěma animacemi vypočítá jejich rozdíl (tedy rozdíl pózy v jedné animaci od pózy v druhé animaci). Tento rozdíl se dá parametrizovat a procentuálně přidávat k existujícím animacím. [26]

2.4 Herní engine

Herní jádra (engine) usnadňují vývoj aplikací tím, že implementují základní softwarový framework, který obsahuje integrované nástroje pro vytváření aplikací a her. Herní jádra oddělují část implementace, specifickou pro konkrétní aplikaci, od části implementace obecných funkcionalit využívaných ve mnoho aplikacích. Každé herní jádro se liší tím, jaké funkcionality poskytuje, ale většina známějších obsahuje například jádro pro vykreslování grafiky (ať už 2D nebo 3D), jádro pro práci se zvukem, jádro implementující fyziku, jádro pro umělou inteligenci, apod. Dále může obsahovat funkcionality zařizující síťovou komunikaci, nástroje pro práci s animacemi, stará se o správu paměti, detekci kolizí a mnoho dalších funkcionalit. [35]

Dobrým příkladem je herní engine `Source`, pomocí kterého byly vyvinuty například hry jako `Team Fortress 2, Counter-Strike Global Offensive, Half-Life 2, Portal 2`, atd. Ačkoliv se jedná o naprosto odlišné hry, všechny využívají stejné základní funkcionality.

Na jednu stranu herní engine usnadňují vývojářům práci s tím, aby v každé hře nemuseli znovu implementovat stejné funkcionality, na druhou stranu odebírá flexibilitu v upravování těchto funkcionalit pro specifické potřeby v konkrétní aplikaci. Pokud uživatel v těchto funkcionalitách objeví chybu nebo nedostatek a engine není open source, nemůže chybu vlastnoručně opravit.

Na trhu je dostupné velké množství herních enginů. Některé jsou open source, jiné mají placené licence. Například `Unity` umožňuje zdarma využívat edici „*Personal*“, pokud příjem z prodejů nepřesáhne ročně 100 000 $. Některé společnosti využívají své vlastní herní engine, které nejsou veřejně dostupné.

Kapitola 3

Analýza a návrh

V této kapitole jsou cíle práce rozděleny do dílčích částí. Pro každou část je provedena analýza možných řešení a návrh postupu práce, včetně využitých technologií, nástrojů a metod.

3.1 Struktura projektu

Projekt lze rozdělit na tři hlavní části, na kterých lze pracovat z části nezávisle na ostatních. Tyto části vychází ze zadání práce:

Příprava modelu nezbytná pro animaci. Model bude nejdřív třeba analyzovat a určit, jak ho bude nutné upravit. Některé části modelu bude potřeba domodelovat, změnit bude třeba i jeho textury. Následně bude pro model vytvořená kostra a ovládací rig, pomocí kterého bude možné model animovat.

Vytvoření animací, které budou demonstrovány ve výsledné aplikaci. Tato část je závislá na vytvoření kostry a ovládacího rigu pro model. Následně bude třeba navrhnout, jaké animace budou vytvořeny a pomocí jakých metod budou vytvářeny.

Vývoj aplikace, ve které budou výsledky demonstrovány. Bude nutné určit, v čem bude výsledná aplikace vyvíjena. Dále jakou bude mít aplikace strukturu a uživatelské rozhraní, jak se bude ovládat a jakým způsobem budou animace demonstrovány.

3.2 Volba softwaru

3.2.1 Výběr modelovacího softwaru

Pro práci se 3D modely existuje řada softwarů. Mezi nejpopulárnější patří `Blender, ZBrush, Cinema 4D, Maya, 3ds Max, Mudbox`, aj.

	Blender	Zbrush	Cinema 4D	Maya	3ds Max	Modo
Company	Open-source (OSS)	Pixologic	MAXON	Autodesk	Autodesk	Foundry
Year founded	2003	1999	1990	1998	1990	2004
Key features	Skin modifier, keyboard shortcuts, addons	Customizable brushes, sculpting layers	Procedural and polygonal modeling, rigging, user-friendly interface	Motion-capture handling, nurbs modeling, layers	Scripting language, edit poly modifier, spline system	Programmability, organic and hard-surface modeling
Best for	Animation; independent projects	Hi-poly modeling; sculpting	Studio modelling and animation	Animation and visual effects	Modeling and rigging	Programming and modeling
Difficulty	Difficult	Average	Easy	Average	Average	Difficult
Cost	FREE	$795 per license	$700 per license (excluding variants)	$1,470 per year (FREE trial)	$1,470 per year (FREE trial)	$399 per year (FREE trial)

Obrázek 3.1: Porovnání několika softwarů pro 3D grafiku [36]

Obecně všechny tyto programy obsahují velmi podobné funkcionality. Liší se ve kvalitě jednotlivých nástrojů a jak snadno, rychle a flexibilně lze tyto nástroje používat a volba tedy závisí zejména na typu projektu a osobní preferenci. Například `Mudbox` a `ZBrush` mají velice kvalitní nástroje pro sculpting, `Maya` exceluje ve vytváření animací, `3ds Max` ve 3D modelování, atd. Dále se liší samozřejmě v ceně. Porovnání dle [36] lze vidět v tabulce na obrázku 3.1.

Z finančních důvodu, osobních zkušeností a preferencí v této práci bude využíván `Blender`. Obsahuje všechny důležité funkcionality pro tuto práci. Nástroje k modelování, sculptingu, texture painting, rigging, constraints, IK, keyframe animaci a export ve formátu FBX. K vývoji bude použitý `Blender` verze 2.81a. [37]

3.2.2 Výběr herního engine

Mezi aktuálně nejznámější veřejně dostupné herní engine patří například `Unity`, `Unreal Engine 4`, `Godot`, `CryEngine`, aj.

`Unity` umožňuje vývoj aplikací na mnoho platform, včetně mobilů i herních konzolí. V `Unity` byly vytvořeny známé hry, jako `Hearthstone`, `Cuphead`, `Deus Ex: The Fall`, `Rust`, atd. Umožňuje skriptování primárně v jazyku `C#` a obsahuje animační systém podporující *animation retargeting*. Unity obsahuje několik úrovní licencí, které se liší na základě příjmu z prodejů. Pokud roční příjmy přesáhnou částku 100 000 $, je uživatel povinen přejít z *Personal* verze na verzi *Plus*, která stojí 40 $ měsíčně. Pokud příjmy přesáhnou 200 000 $, je uživatel povinen přejít na verzi *Pro* za cenu 150 $ měsíčně. Dražší verze obsahují více funkcionalit. `Unity` je zejména známý svojí aktivní komunitou, která vytváří veliké množství návodů a dostupných zdrojů, jako jsou modely,

skripty a jiné funkcionality. [38]

Unreal Engine od firmy Epic Games je známý zejména pro fotorealistické vykreslování složitých a detailních scén. Umožňuje skriptování v jazyku C++, nebo využívat zjednodušený vizuální skriptovací systém *„Blueprints“*. V Unreal Engine byly vytvořené hry jako série Mass Effect, Bioshock nebo jedna z nejhranějších her roku 2019, Fortnite. License funguje na základě systému honorářů. V moment kdy výdělek z aplikace přesáhne 3 000 $ za čtvrtletí, je z této částky nutné zaplatit 5 %. V porovnání s Unity, C++ je nižší programovací jazyk než C# a kód lze lépe optimalizovat. Zároveň je ale snazší v něm dělat chyby. Zdrojový kód UE je veřejně dostupný, což může při tvorbě aplikací být užitečné. Dostupných zdrojů, návodů, modelů a dalších funkcionalit je v porovnání s Unity méně.

Godot je zdarma, open-source herní engine s MIT licencí, což znamená, že ho lze volně využívat, šířit a modifikovat, pokud se z něj neodstraní kopie licence. Godot je přijatelnou volbou pro menší projekty, specificky 2-D aplikace (ačkoliv umožňuje vyvíjet i 3-D aplikace). Nevýhodou je menší uživatelská komunita, stejně tak jako menší vývojářský tým, což zdržuje opravy chyb a vývoj nových funkcionalit. Zároveň obsahuje nepříliš kvalitní dokumentaci. Vyvíjet aplikace lze v několika jazycích, včetně C#, C++ nebo vlastním skriptovacím jazykem GDScript, což je dynamicky typovaný jazyk podobný jazyku Python. [39]

CryEngine od společnosti CryTek je herní engine který sloužil jako základ pro vytvoření her jako Far Cry, Kingdom Come: Deliverance nebo Crysis. Využívá podobný systém honorářů, jako Unreal Engine. Pokud příjmy z projektu přesáhnou částku 5 000 $ ročně, je z této části třeba zaplatit 5 %. CryEngine umožňuje tvorbu aplikací na širokém spektru platform, včetně virtuální reality. Je vhodný pro kvalitní vývoj takzvaných FPS her (first-person shooter) a umožňuje vytvářet velmi detailní a realistické prostředí. Veškeré zdrojové kódy jsou volně dostupné. Skriptovat lze v jazyku Lua. Nevýhody má podobné, jako Godot. Menší komunita a vývojářský tým vedou k menší uživatelské podpoře, pomalejší opravě chyb a nižší kvalitě dokumentace. [40]

Opět z důvodů osobních zkušeností a preference bude v této práci použit herní engine Unity.

Obrázek 3.2: Přidělený model robota

3.3 Úprava modelu

3.3.1 Analýza modelu

Přidělený model je sbírkový předmět humanoidního robota, viz obrázek 3.2. Model je v jednom celistvém kusu a ve výchozí poloze zaujímá T-pose. Byl vytvořený pomocí fotogrammetrie, což znamená, že má vysoký počet polygonů a složitou topologii. Model má následující parametry:

- 502 579 vrcholů,
- 1 005 142 trojúhelníků,
- texturu s 8K rozlišením a vygenerovanou UV mapu, která tuto texturu mapuje na robota.

3.3.2 Oddělení částí modelu

Jelikož se jedná o robota s mechanickými klouby, který je složený z rigidních částí, je nežádoucí, aby při animaci docházelo k deformaci polygonové sítě. Je zřejmé, že bude ideální použít jednoduchý skinning, popsaný v sekci 2.3.4.

To znamená, že bude třeba jednotlivé rigidní části od sebe oddělit do separátních modelů. Ze struktury robota lze určit, že objekt bude třeba rozdělit na 14 částí (každá končetina je složena ze tří částí, tělo je jeden veliký kus a poslední část je hlava).

Oddělení v programu `Blender` lze docílit několika metodami. Model se dá upravovat na úrovni jednotlivých vrcholů a hran, ale to není vzhledem ke složitosti modelu přijatelné řešení. Druhou možností je použití *boolean* modifikátoru, pomocí kterého lze například izolovat doplněk jednoho modelu v průniku s jiným modelem. Pomocí primitivních tvarů (jako krychle a koule) lze tedy vytvořit hrubou reprezentaci části kloubu, která patří k jiné rigidní části modelu a tuto část přes modifikátor odříznout.

Takové řešení je zdlouhavé a primitivní tvary nejsou kvůli specifickému tvaru kloubů vhodné. Lepší volbou je nástroj nůž (*knife tool*). Ten umožňuje objekt rozříznout podél uživatelem určených hran (ať už existujících v topologii modelu nebo nově vytvořených tímto nástrojem). Pokud se tímto nástrojem vytvoří smyčka hran, lze podél této smyčky objekt rozdělit na dva díly.

3.3.3 Domodelování chybějících částí

V místech kde bude model rozdělen na jednotlivé části tímto procesem vzniknou díry, kterými bude vidět dovnitř robota. To bude viditelné v moment, kdy se nějaká jeho část pohne z výchozí polohy v T-pose. Proto bude tyto části nutné ručně domodelovat.

Na tento úkol bude nejdříve využit program `Meshmixer` od společnosti `Autodesk` [41]. Důvodem je, že tento program má pro tuto práci užitečné nástroje. Prvním důležitým nástrojem je přemostění (*bridge*), které umožňuje mezi dvěma vybranými skupinami hran vytvořit „most“, který je složený z primitivních polygonů. Druhý důležitý nástroj je zaplnění (*fill*), který dokáže zaplnit díry v objektu polygonovou sítí. Tato síť zároveň respektuje tvar objektu, ve kterém je díra. Čili pokud je díra v modelu koule, vytvořená záplata je zaoblená, aby zachovala tvar koule. Tvar se dá interaktivně měnit pomocí parametrů.

Vedlejší výhodou je, že `Meshmixer` je schopen odhalit chyby v topologii. Dokáže nalézt díry v modelu, izolované vrcholy a jiné nemanifoldní objekty (objekty, které by nemohly existovat v reálném světě) a automaticky je opravit.

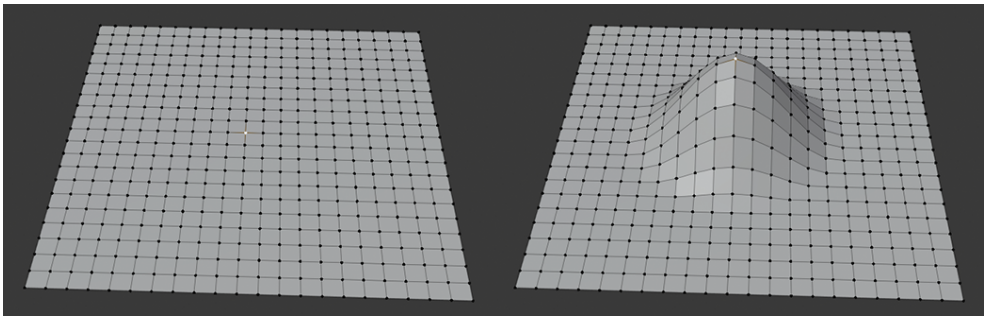
`Meshmixer` umožňuje domodelovat přibližně kulatý tvar kloubů. Stačí vybrat smyčku hran tvořící díru v objektu, použít nástroj *fill* a měnit parametry, dokud výsledek nevypadá přijatelně. Z osobních zkušeností vyplývá, že čím má díra v modelu pravidelnější tvar, tím lepší výsledky nástroj *fill* produkuje. Proto je lepší díru rozdělit na více pravidelných děr pomocí nástroje *bridge*. Na detailní zakulacení bude ideální použít sculpting nástroje, jelikož s nimi lze nafukovat části modelu, zahlazovat boule, apod. Přestože `Meshmixer` obsahuje i nástroje pro sculpting, těchto nástrojů je málo a obsahují menší množství

volitelných parametrů. Zároveň je výpočet změn značně pomalejší než v programu `Blender`. Proto pro sculpting úpravy bude lepší použít `Blender`.

Pro vytvoření co nejkulatějších kloubů lze ve středu kloubu vytvořit separátní primitivní objekt koule, která se zvětší na požadovanou velikost kloubu. Tomuto pomocnému objektu se nastaví drátěný zobrazovací mód *(wire)*, což způsobí, že se budou vykreslovat jen jeho hrany. Poté je možné pomocí sculpting nástrojů kloub upravovat tak, aby byly jen sotva viditelné hrany tohoto pomocného objektu. To zaručí, že výsledný kloub bude mít skoro perfektní sférický tvar. Toho lze docílit pomocí nástrojů *inflate/deflate*, které vyznačenou oblast zájmu „nafukují“, nebo „vyfukují“. Pro odebírání přebytečného materiálu je možné využít záporný mód nástroje *blob*. Pro vyhlazení povrchu lze použít nástroj *smooth*.

Upravit bude třeba i druhou část, tedy otvor, ve kterém se kulatý kloub otáčí. I po domodelování kloubů bude stále vidět dovnitř modelu v místech, kde se kloub na originálním modelu napojuje na tyto části a porušuje tím svůj sférický tvar. Jeden způsob řešení, je vytvořit jednoduchou polygonovou síť, která bude tento otvor vyplňovat. Na tuto nově vytvořenou síť lze aplikovat tmavý materiál, který pohlcuje světlo. V místech kde by bylo možné vidět dovnitř robota tedy bude tma. Tento přístup by znamenal, že by se musela tato nová polygonová síť vázat na existující okraje. Okraj má nepravidelný tvar a kvůli topologii robota nebude možné pro vytvoření této sítě používat většinu ulehčujících nástrojů a funkcí v programu `Blender`. Zároveň by se tímto způsobem přidala do modelu další geometrie. Druhý způsob je existující části posunout a ohnout tak, aby nebylo možné do robota vidět. Prakticky to znamená ohnout a „zapíchnout“ okraje do kulatého kloubu. Takto posouvat jednotlivé vrcholy by při hustotě polygonové sítě robota nebylo přijatelné.

`Blender` umožňuje aktivovat takzvané proporcionální editování *(proportional editing)*. To znamená, že transformace vybrané části polygonové sítě zároveň transformuje oblast v okolí této části. Uživatel může například vybrat jeden vrchol a s ním pohybovat. Všechny okolní vrcholy ve zvoleném rozsahu se budou pohybovat relativně k vybranému vrcholu. V základním nastavení platí, že čím blíže jsou tyto vrcholy ke zvolenému vrcholu, tím více jsou jeho pohybem ovlivňovány, viz obrázek 3.3. Další možnosti jsou například konstantní ovlivnění, lineární ovlivnění nebo náhodné ovlivnění. Nevýhodou této metody je, že je snazší dopustit se chyby a vynechat místo, ze kterého je v extrémním úhlu stále vidět dovnitř robota. Druhou nevýhodou je, že tímto dochází k natahování textur. Použitím proporcionálního editování se textura natahuje na více polygonech v menším měřítku, čímž je deformace méně viditelná. Zároveň z analýzy modelu vyplývá, že v místech kloubů nemá textura příliš specifické detaily. Výhodou je, že se tímto způsobem nepřidává do modelu žádná další geometrie. Pokud bude deformace textury příliš viditelná, dá se zamaskovat úpravou samotné textury.



Obrázek 3.3: Proporcionální editování v programu `Blender`

3.3.4 Úprava textur

Pro nově vytvořené části bude třeba vytvořit novou texturu. Zároveň aby přechod vypadal přesvědčivě, bude třeba upravit i originální texturu robota v místech, kde se napojuje na nově vytvořené části. Nová textura bude mít svoji vlastní UV mapu, na kterou se rozbalí nově vytvořené části. Textura musí mít dostatečné rozlišení, aby na ní bylo možné zaznamenat dostatečně drobné detaily.

Texturu lze upravovat nezávisle v programech pro úpravu rastrové grafiky, nebo pomocí nástrojů přímo v programu `Blender`. Ačkoliv programy jako Photoshop nebo Gimp obsahují více a lepších nástrojů pro takové úpravy, problém je v tom, že nelze interaktivně pozorovat změny na modelu. Textury se musí ukládat a následně v programu `Blender` obnovovat.

Pokud se úpravy provádí přímo v programu `Blender`, lze vidět v reálném čase, jak se změny projevují na modelu. `Blender` obsahuje prostředí pro úpravu textur zvané *Texture Paint*. Nejužitečnější nástroj pro tento úkol bude klonovací razítko *Clone*. Tímto nástrojem lze klonovat existující část textury z jedné části modelu na jinou část modelu. V případě modelu robota lze klonovat z originálních částí kloubu na dotvořené části. Během tohoto procesu dojde zároveň k úpravě originální textury robota v místech přechodu na dotvořené části kloubu.

V [42] je uvedeno, že grafické procesory rozdělují objekty před vykreslováním na základě materiálů na separátní části, které jsou vykreslovány zvlášť, což vede k tomu, že je potřeba více vykreslovacích volání *(tzv. draw calls)*, což typicky zhoršuje výkon. V `Unity` je doporučeno dle [43] pro zlepšení výkonu kombinovat více textur do jedné s větším rozlišením. V případě modelu robota to znamená zkombinovat originální texturu a dotvořenou texturu kloubů. `Blender` obsahuje pro tyto účely funkci pro zapékání textur *(texture baking)*, zmíněnou v sekci 2.1.6.

3.3.5 Level of Detail

Jelikož je model složen z vysokého počtu polygonů, mohl by způsobovat v aplikaci spuštěné na méně výkonném hardwaru pokles výkonu, což může mít za následek například sníženou snímkovou frekvenci. Z toho důvodu se pro modely vytváří více stupňů detailu, viz sekce 2.1.3

Proto bude pro model vytvořená druhá verze, složená z menšího počtu polygonů, která bude v aplikaci využitá místo originálního modelu s hustou polygonovou sítí. Zároveň by se na novém modelu mělo zachovat co nejvíce detailů. V ideálním případě by tato nová verze neměla být vizuálně rozeznatelná od originálu.

`Blender` umožňuje vytvořit nižší stupeň detailu buď manuálně pomocí modelovacích technik nebo použitím decimačních algoritmů. Vzhledem k tomu, že přidělený model se skládá z rigidních částí a nedochází u něj k deformaci kůže, je decimace dostačující řešení.

Decimace modelu způsobí deformaci textury, takže bude třeba jí opravit. Toho bude možné docílit zapečením nedeformovaných textur z originální verze modelu s hustou polygonovou sítí.

3.3.6 Kostra

Cílem této části je vytvořit kostru a ovladací prvky pro robota, které dovolí jednoduché a zároveň flexibilní ovládání. Dobrá kostra by měla být přehledná a ulehčit vytváření animací pomocí dalších ovládacích prvků. Pro přehlednost lze měnit i vzhled a barvy těchto ovladacích prvků.

V programu `Blender`, jsou ovládací prvky postavy primárně kosti. To znamená, že kosti nejsou jen uvnitř těla a neurčují pouze anatomickou strukturu postavy. Jako příklad může být kořenová kost *(root bone)*, která se někdy umisťuje na zem mezi nohy postavy a pomocí této kosti se hýbe s celou postavou. Další příklad je ovládací prvek, který určuje cíl, který postava pozoruje. To by mohlo být řešeno kostí umístěnou před postavou. Oči by následně měly nastavené omezení, které by jejich rotaci měnilo tak, aby se dívaly na tuto kost. Aplikace těchto omezení lze v programu `Blender` také animovat, takže by bylo možné přepínat mezi manuální manipulací očí a pozorováním jiných objektů.

Na ovládání veškerých jednotlivých částí kostry by stačilo 14 kostí, jelikož z tolika rigidních částí se skládá. Avšak kvůli nastavení v `Unity`, které bude zmíněno v kapitole 3.4 o vývoji aplikace, bude nutné vytvořit v torsu jednu kost navíc, dělící torso na pánev a páteř, která nebude ovládat žádnou část robota (nebude mít ve *skinningu* přiřazené žádné vrcholy). Tyto kosti se označují jako deformační kosti, protože u modelů, u kterých je potřeba zajišťovat ohyb kůže přímo způsobují deformaci polygonové sítě.

Kvůli přehlednosti a dělení funkcionality se nedoporučuje animovat pomocí deformačních kostí, takže bude vhodné vytvořit další dvě sady kostí

končetin, pro dopřednou a inverzní kinematiku, tedy dalších 24 kostí. Deformační kosti končetin mohou následně kopírovat polohu kostí vybrané kinematiky, což umožní přepínat mezi ovládáním pomocí dopředné a inverzní kinematiky.

Inverzní kinematika zároveň pro každou končetinu vyžaduje další dvě kosti jako ovladací prvky. Jedna slouží jako koncový efektor a typicky je tedy na konci končetiny. Druhá slouží pro specifikaci orientace, kudy se řetězec kostí ohýbá po cestě k cíli. V programu `Blender` se této kosti říká *pole target bone* a u humanoidních postav určuje směr zlomu v oblasti loktů a kolen.

Pro přepínání mezi kinematikami lze využít drivery na kostních omezeních *(bone constraints)*. U omezení lze definovat hodnotu vlivu těchto omezení na danou kost. Tato hodnota je v intervalu nula až jedna, kde nula znamená, že omezení kost neovlivňuje vůbec. Jednička naopak znamená, že kost dodržuje omezení na sto procent. Tuto hodnotu vlivu lze měnit pomocí zmíněných driverů pomocí definovaných vzorců. V tomto případě bude stačit, aby byly definované vlastní vlastnosti (v programu `Blender` takzvané *Custom Properties*) ve formě jednoduchých parametrů s hodnotou nula až jedna, které přímo ovládají hodnotu vlivu omezení.

V poslední řadě bude vhodné kosti od sebe barevně oddělit podle funkcionality. Pro přehlednost může být užitečné i vytvořit vlastní modely pro vizualizaci kostí. Volitelně lze i barevně oddělit kosti levých a pravých končetin, aby bylo na první pohled zřejmé, jakou končetinou animátor pohybuje i z bočního pohledu.

3.4 Animace a vývoj aplikace

Vývoj aplikace bude prováděn v `Unity` verze 2019.3.4f1.

3.4.1 Import modelu

Ačkoliv lze do `Unity` importovat celý soubor s příponou *blend* (nebo například přípony *max* z programu 3ds Max, atd.), tento postup není doporučován. `Unity` tento typ souboru převádí do formátu fbx jako součást importovacího procesu. Pro tento proces je zapotřebí mít na daném počítači nainstalovaný korespondující 3D modelovací software, nebo projekt nebude možné otevřít. Zároveň tímto způsobem nelze ovládat, jaké části budou importovány a soubor může obsahovat zbytečná data.

Animované postavy se nejčastěji do `Unity` importují v souborech formátu fbx. `Blender` umožňuje exportovat postavy i jejich animace v tomto formátu. Zároveň lze exportovat model a jednotlivé animace v oddělených souborech.

Animované postavy mohou v Unity využívat tři animační systémy. Systém „*Legacy*“, který byl vyvinutý v dřívějších verzích `Unity` a nedoporučuje se používat. Výjimkou je využití kvůli zpětné kompatibilitě ve starších projektech. Druhý systém je „*Generic*“, který slouží pro animaci všech postav,

které nemají humanoidní strukturu. Pro tyto modely je třeba určit kořenový kloub *(root bone)*. Pro humanoidní typy existuje systém „*Humanoid*“. Unity pro reprezentaci humanoidních postav využívá generickou humanoidní kostru. Importované postavy je třeba namapovat na tuto generickou kostru. Toto umožňuje pomocí automatického *animation retargeting* používat jednu sadu animací pro jakoukoliv humanoidní postavu. Toto mapovací rozhraní se v `Unity` nazývá *Avatar*. Aby bylo možné využívat humanoidní systém, musí kloubová soustava splňovat následující strukturu:

- kyčel → páteř → hruď → horní hruď → krk → hlava
- hlava → oko
- hlava → čelist
- kyčel → horní noha → spodní noha → chodidlo → prsty
- hruď → ramena → paže → předloktí → ruka
- ruka → proximální → střední → distální

Hruď, horní hruď, ramena, prsty, krk, oči a čelist jsou nepovinné. Zároveň lze mít více kostí než jsou zde definované, například pokud by páteř měla být složena z více kostí. Tyto kosti ale nebudou animovány, budou držet pozici relativní ke své rodičovské kosti. Zároveň lze využít i další kosti, například pro animaci doplňků nebo částí oděvu. Na to je potřeba využít animační masky, ve kterých je třeba tyto kosti u jednotlivých animací aktivovat.

Výhody humanoidního systému jsou následující:

- Pomocí *animation retargeting* lze využít humanoidní animace na libovolnou humanoidní postavu. Z toho plyne, že si vývojář může zakoupit nebo stáhnout sady animací a aplikovat je na svoji vlastní herní postavu nebo naopak aplikovat své vlastní animace na staženou postavu.
- Implementovaná inverzní kinematika pro ruce a nohy. Pomocí skriptů stačí zapnout výpočet inverzní kinematiky na končetině a nastavit cílovou polohu a rotaci.
- Možnost zrcadlení animací (například chůzi doprava zrcadlit a tím vytvořit chůzi doleva).
- Možnost pozorování místa či objektu. Přes skriptování lze nastavit, jak moc se při pozorování zapojuje tělo, hlava a oči. Zároveň lze omezit pozorovací rozsah.

3.4.2 Animace

Animace, které budou využívány v aplikaci, se dají vytvářet dvěma způsoby. V prvním se postava pohybuje pomocí skriptů v ovladačích a její stav určuje, jaká animace se přehrává. Stav může být například poloha nebo rychlost. Taková animace se vytváří na místě a kořenová kost se nepohybuje, z čehož plyne její název *in-place*. Například při animaci chůze nohy kloužou po podlaze a postava se nepohybuje dopředu – pohyb bude určen až pomocí kódu.

Druhý způsob je takzvaný kořenový pohyb *(root motion)*. V této animaci je proces opačný. Animace se vytváří s pohybem kořenové kosti a hýbe se s celou postavou. V aplikaci je následně pohyb postavy určen pohybem kořenové kosti v animaci.

Oba přístupy mají svá pozitiva i negativa. *Root motion* obecně působí vizuálně lépe, protože nedochází například k takzvanému klouzání chodidel *(foot sliding)*, kdy nastavená rychlost postavy není synchronizovaná s rychlostí animace a lze vidět, jak nohy kloužou při animaci po podlaze.

Na druhou stranu je složitější a méně intuitivní pohyb upravovat, protože není řízen čistě přes kód. Postavy mohou působit méně responzivně a hráčům může připadat, že na ovládaní reagují hůř nebo opožděně. Tyto dva přístupy lze i vzájemně kombinovat tak, že některé animace využívají první způsob a některé druhý způsob.

Z osobní preference a možnosti lépe pohyb ovládat pomocí kódu v této práci bude využíván způsob *in-place*. Tyto dva přístupy jsou na sebe zároveň převoditelné, ačkoliv převod z *in-place* animací na *root motion* je o něco složitější, jelikož animátor musí přidat informaci o poloze kořenové kosti, na rozdíl od opačného převodu, kde tuto informaci stačí vymazat.

`Blender` obsahuje prostředí pro vytváření animací *Animation*. Jednotlivé animace jsou zaznamenávány jako takzvané akce *(Action)*. Klíčovat se dají transformace kostí a jejich podčásti. Jelikož se typicky nemění měřítko kostí, do klíčových snímků se ukládá pozice a rotace, vyjádřena jako 6 čísel. Tři pro pozici a tři pro rotaci v Eulerových úhlech.

Kromě pózování a ukládání snímků lze jednotlivé hodnoty i mazat či upravovat (například odstranit posunutí v ose x na třetím klíčovém snímku). Jednotlivé hodnoty i celé klíčové snímky lze i posouvat po časové ose, kopírovat a pro osově souměrné modely zrcadlit.

V následující části je uveden popis jednotlivých animací, vybraných pro implementaci a využití ve výsledné aplikaci. Zároveň je popsán stručný návrh jejich tvorby.

3.4.2.1 T-pose

Ačkoliv se nejedná přímo o pohyb, bude vhodné vytvořit animaci s jedním klíčovým snímkem s robotem ve své základní T-pose. V této póze bude model importován a značí, že postava neprovádí žádnou animaci. Do `Unity`

je doporučováno humanoidní postavy importovat v T-pose kvůli internímu *animation retargeting* procesu. `Unity` umožňuje převod importované pózy do T-pose automaticky, ale kvůli chybám, které mohou vzniknout při vytváření kostry se může stát, že se tímto procesem část těla přetočí ve špatné ose.

3.4.2.2 Nečinnost

Takzvaná *idle* animace, při které bude robot stát v nečinnosti na místě. Tuto animaci lze obohatit jemným kolísáním pánve, aby model nebyl kompletně stacionární. Ve hrách mají často postavy více nečinných animací, které obsahují sekundární animace. Postava se pak může po chvíli nečinnosti rozhlédnout, podrbat, apod.

3.4.2.3 Chůze a běh

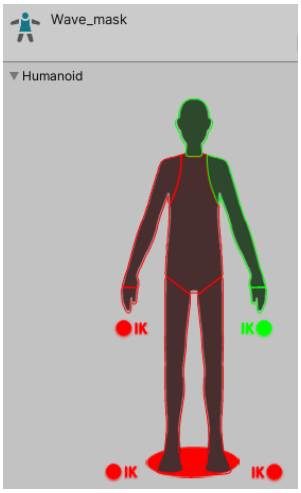
Obě tyto animace budou cyklické, čili poslední klíčový snímek musí navazovat na první. Postava v této animaci tedy udělá dva kroky. Na to se perfektně hodí možnost kopírovat klíčové snímky. Pro vytvoření jednoho kroku lze počáteční snímek zkopírovat a zrcadlit do posledního snímku, poté lze upravit přechod postupným přidáváním snímků mezi počátečním a koncovým. Jeden vytvořený krok se dá poté opět celý zrcadlit a poslední klíčový snímek, který je stejný jako úplně počáteční, se odřízne. Tím se zaručí plynulé opakování animace.

3.4.2.4 Zamávání

V této animaci robot jednou rukou zamává. Zároveň tuto animaci bude možné kombinovat s ostatními animacemi pro pohyb. Tento proces je vysvětlený v sekci 2.3.7. `Unity` nabízí primárně dva přístupy, které fungují na principu vytvoření masky vybraných kostí, viz obrázek 3.4. Pomocí této masky lze přehrávat na části těla, kterou maska pokrývá, jinou animaci než na zbytku postavy. Touto animací lze buď přepsat aktuální animaci na zbytku těla, nebo jí aplikovat aditivně k aktuální animaci. Toho se hojně využívá například v takzvaných *střílečkách*, kde jsou k animacím chůze a běhu přidány animace držení jednotlivých zbraní, animace střelby, nabití, apod.

Masek lze využít současně více. Každá maska má zároveň svoji váhu v rozmezí 0 až 1, která určuje, jak moc ovlivňuje celkovou animaci. Plynulý přechod této váhy zaručí plynulý přechod mezi hlavní animací a animací aplikovanou na masku.

Tímto způsobem bude možné spouštět animaci zamávání jak v nečinnosti, tak za chůze či běhu. Zároveň se při vytváření této animace stačí soustředit pouze na části těla robota, které budou pokryty maskou, tedy jedna ruka, případně hlava a trup.



Obrázek 3.4: Zjednodušené vytváření masky pro humanoidní postavy v `Unity` [38]

3.4.2.5 Skok

Pokud je skok využíván ve hře, kde postava musí interaktivně reagovat na své prostředí a na uživatelský vstup, je animace skoku postavy závislá na několika faktorech. Například rychlost postavy při zahájení skoku, tedy závislost na tom, jestli postava při skoku stojí, chodí či běží. Dále jak se má postava ze skoku zotavit, což záleží například na tom, z jaké výšky postava skočila.

Celý skok lze obecně rozdělit na tři stavy:

- příprava na skok a odraz ze země,
- let a volný pád,
- dopad na zem a zotavení.

Pro tyto stavy je vhodné vytvořit oddělené animace které se v průběhu skoku mezi sebou míchají. Tím způsobem lze interaktivně reagovat na stav postavy. Například pokud postava z objektu spadne, přeskočí se stav přípravy na skok. Pokud postava padá z veliký výšky, animace volného pádu se může přehrát i několikrát, dokud postava nedopadne na zem.

Pro tyto stavy lze vytvořit i více animací, například příprava na skok ve stoje a příprava na skok v běhu. Animace v letu lze dělit na základě toho, jestli postava skočila nebo spadla, nebo na základě výšky, ze které postava padá, apod. Moment dopadu lze například detekovat pomocí kolizí postavy s podlahou nebo pomocí vysílání paprsků směrem k zemi a dopočítávání místa a času dopadu na základě vzdálenosti, rychlosti a směru.

3.4.3 Design prostředí

Prostředí bude obsahovat objekty, pomocí kterých bude možné demonstrovat implementované procedury. Pro demonstraci adaptace pohybu na nerovnost povrchu lze vytvořit nahnuté plošiny, schody, boule, apod. Z objektů, na které schody a plošiny vedou lze demonstrovat skok a volný pád.

Tyto objekty lze buď vytvořit vlastnoručně nebo zadarmo stáhnout jednu z mnoha vytvořených sad objektů z *Asset store*.

Aplikace zároveň bude obsahovat menu, pomocí kterého bude možné hru pozastavit a zobrazit ovládání.

3.4.4 Pohyb a ovládání

Vzhledem k vytvořeným animacím bude vytvořeno ovládání, které umožní jednotlivé animace demonstrovat. Proto bude vytvořená komponenta ve formě skriptu, která bude připojena k postavě.

Tato komponenta bude spolupracovat s animační komponentou „*Animator component*“, která slouží pro správu a přehrávání animací postavy. Animační komponenta funguje na základě stavového stroje, kde stavy tvoří jednotlivé animace. Přechody mezi stavy mohou být závislé na hodnotách proměnných, které lze měnit například ze skriptů. Zároveň animační komponenta zaručuje míchání a přechod animací. V přechodech mezi stavy lze animace překrývat, což způsobí plynulý přechod animací. Dalším užitečným nástrojem jsou míchací stromy *(blend trees)*, které na základě jednoho či více parametrů míchají animace dohromady. Míchací strom může například přehrávat animaci nečinnosti při hodnotě parametru 0, chůzi s hodnotou 1 a běh s hodnotou 2. Tento parametr lze ze skriptů plynule měnit, což způsobí plynulý přechod mezi těmito stavy. Nastavením tohoto parametru například na hodnotu 1,5 vznikne animace poklusu. Ačkoliv animace vytvořené tímto procesem nemusí vypadat dostatečně realisticky.

Komponenta ovládacího skriptu bude mít následující funkce:

- Reagovat na uživatelský vstup a na základě toho pohybovat s postavou.
- Zajišťovat gravitaci a kontrolovat, zda je postava na zemi či ve vzduchu.
- Měnit proměnné, na kterých jsou závislé stavy animační komponenty. Toto zaručí změnu animací postavy.
- Aplikovat inverzní kinematiku pro rovnání nohou na nerovném povrchu.
- Poskytovat vývojářské rozhraní pro interaktivní změnu nastavení výše zmíněných funkcionalit.

Standardně ve hrách s postavami lze pohybovat pomocí šipek a kláves „WASD“. Směr lze určit z kombinace těchto kláves. Pokud lze zároveň pohybovat s kamerou kolem postavy, ovládání čistě na základě kláves je neintui-

tivní. Z tohoto důvodu bude směr pohybu určen kombinací relativní pozice kamery k postavě a kláves.

Dále bude pomocí klávesy možné aktivovat běh. Jelikož se jedná o demonstrační aplikaci, ve které není primární cíl rychlý pohyb, bude postava běhat jen v moment, kdy hráč bude držet klávesu pro běh. Alternativou je pomocí klávesy přepínat chůzi a běh.

Zbývá ovládání pro skok a zamávání. U zamávání je třeba zaručit, aby se animace nedala spustit znovu, pokud ještě hraje. Podobně u skoku je nutné zamezit tomu, aby postava skočila ve vzduchu.

3.4.5 Inverzní kinematika

Pro humanoidní modely lze využít interní inverzní kinematiku přímo v `Unity`. Vývojář musí nastavit polohu a rotaci koncového efektoru dané končetiny a váhu určující, jak moc se má IK aplikovat. Váha může mít hodnotu nula až jedna. Jednička značí značí cílovou polohu/rotaci IK, zatímco nula značí originální polohu/rotaci.

Polohu lze získat například pomocí sledování paprsku. Na předpokládanou lokaci dopadu chodidla lze vyslat paprsek. Místo, kde protne paprsek povrch lze použít jako cílovou lokaci. Pro rotaci lze použít normálu povrchu v místě dopadu paprsku. Délka vyslaného paprsku by měla být omezená, aby byl cíl v realisticky dosažitelné vzdálenosti postavy. Tento výpočet lze provádět na každém snímku určeným snímkovou frekvencí aplikace nebo snímkovou frekvencí fyzikálního systému `Unity`.

Váha působení nemůže zůstat celou dobu s hodnotou 1, jelikož by IK přepisovala například zvedání nohy ze země při chůzi. Proto lze pro jednotlivé animace definovat křivky, které mohou váhu měnit v závislosti na průběhu animace. Váhu je vhodné měnit plynule, jinak na animaci bude viditelná skoková změna. Při pokládání nohy na zem se váha plynule zvýší na hodnotu 1 a při zvedání nohy ze země se opět plynule sníží na 0.

Pro realističtější výsledky může být vhodné zároveň upravit polohu pánve postavy v závislosti na vertikální změnu výšky nohou.

3.4.6 Kamerový systém

Dle [44] existují tři primární způsoby pozicování kamery z pohledu třetí osoby, které se využívají ve 3D aplikacích:

- Automatizované, které neumožňují hráči s kamerou manipulovat, což může být pro hráče omezující.
- Volně ovládané hráčem. Typicky se nastavují omezení, aby se kamera nemohla dostat mimo hrací pole, do nežádoucích úhlů, dovnitř objektů, apod.
- Hybridní systém, který kombinuje první dva způsoby. Primární manipulace je automatizovaná, ale hráči je umožněno do jisté míry kameru ovládat.

Hybridní systém obsahuje výhody obou dvou systémů. Primární pohyb kamery je automatizován, čímž se zaručí, že hráč vidí jen to, co má vidět. V tomto prostoru je hráči umožněno kameru do jisté míry ovládat, například kameru přibližovat, oddalovat nebo otáčet kolem postavy.

Výběr kamerového systému je individuální pro každou aplikaci/hru. Vzhledem k tomu, že tato aplikace má za účel demonstrovat animace, je důležité umožnit hráči si animace prohlédnout z různých úhlů a vzdáleností. Z toho důvodu bude implementován hybridní kamerový systém. Kamera se bude pohybovat s postavou. Hráči bude umožněno s kamerou točit kolem postavy, přibližovat či oddalovat pohled a měnit střed zájmu (například se zaměřit na nohy postavy). Omezení zamezí, aby se kamera dostala mimo hrací plochu, pod zem, dovnitř objektů, apod. Zároveň bude omezená vzdálenost maximálního přiblížení a oddálení kamery.

Kapitola 4

Implementace

V této kapitole je popsán proces úpravy modelu, tvorba kostry, animovaní a vytváření demonstrační aplikace.

4.1 Model a textury

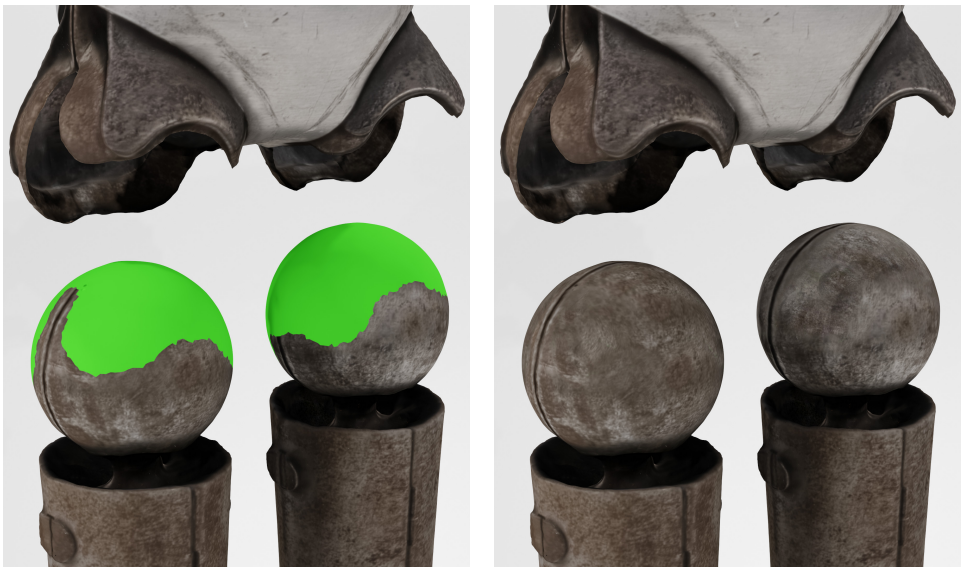
Model robota je rozdělený na 14 oddělených částí. Trup, na který je napojená hlava, ruce a nohy. Každá končetina se skládá ze tří částí, viz obrázek 4.1. Jednotlivé klouby jsou domodelované a upravené tak, aby nebylo možné vidět dovnitř robota.

Domodelované části robota obsahují 34 101 vrcholů. Jelikož je většina z dotvořených částí kloubů schovaná v důlku na kloub, bylo by možné toto číslo dále redukovat decimačními algoritmy bez zásadního rozdílu ve kvalitě.

Pro model je dotvořená textura v místech, kde jsou domodelované klouby. Na obrázku 4.2a je část robota bez dotvořené textury. Na obrázku 4.2b je stejná část robota s dotvořenou texturou.

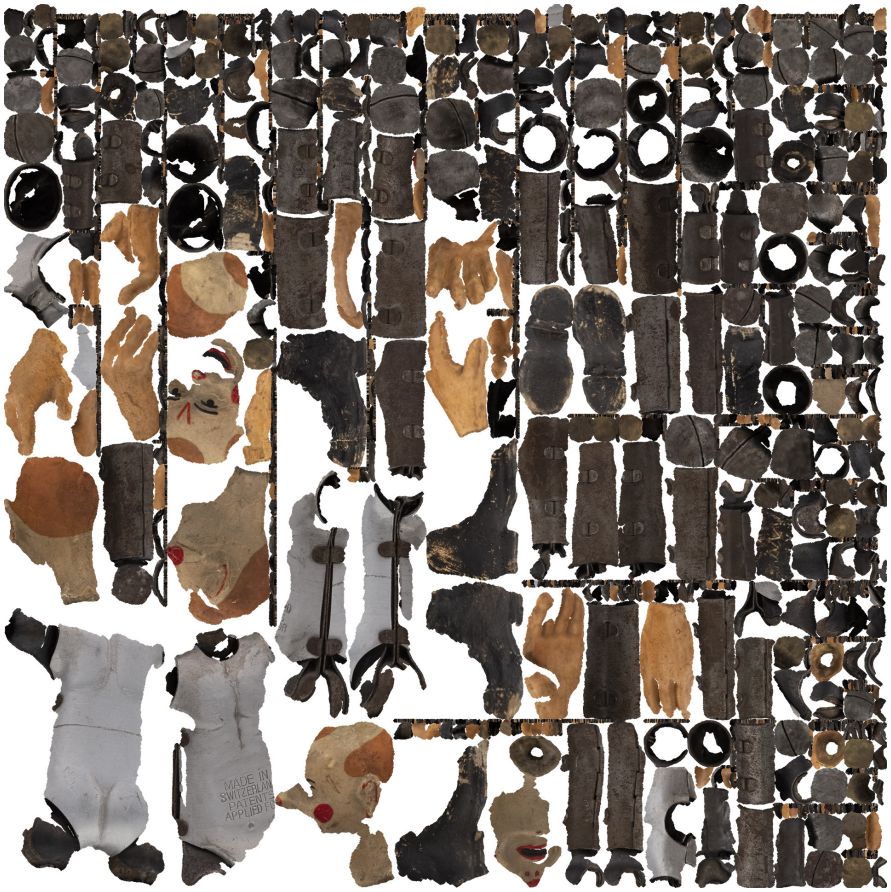
Ve výsledku má model jeden materiál s jednou difuzní texturou, která má rozlišení 8192×8192. Textura je kombinací dvou textur – upravené originální textury vygenerované z fotogrammetrie a dokreslené textury pokrývající domodelované klouby. Byla vytvořená pomocí techniky zapékání textur, zmíněné v kapitole 2.1.6. Výslednou texturu v malém rozlišení lze vidět na obrázku 4.3

Obrázek 4.1: Robot s oddělenými částmi



(a) Bez dotvořené textury (b) S dotvořenou texturou

Obrázek 4.2: Model s domodelovanými klouby



Obrázek 4.3: Zapečená textura robota

4.2 Level of Detail

Pro model je vytvořená druhá verze s nižším stupněm detailu, která je využitá v demonstrační aplikaci. Tato verze je složená z přibližně 24 700 vrcholů a 48 000 trojúhelníků. Jelikož decimace způsobuje deformaci textury, je pro model zároveň vytvořená nová textura se stejným rozlišením. Na obrázku 4.4 lze vidět, že jsou modely vizuálně prakticky nerozeznatelné, přestože zjednodušený model obsahuje méně než 5 % z celkového počtu vrcholů originálního modelu.

4.3 Kostra

Model má vytvořenou kostru, pomocí které lze pózovat a animovat. Kostra obsahuje kontrolní kosti, deformační kosti, kosti pro dopřednou kinematiku, kosti pro inverzní kinematiku a cílové kosti pro výpočet inverzní kinematiky.

Obrázek 4.4: Originální model (vlevo) a model se sníženým stupněm detailu (vpravo)

Tyto skupiny kostí mají v programu `Blender` přiřazené rozdílné barevné značení pro zvýšení přehlednosti. Barevné značení je následující:

kontrolní – žluté

deformační – bledě zelené

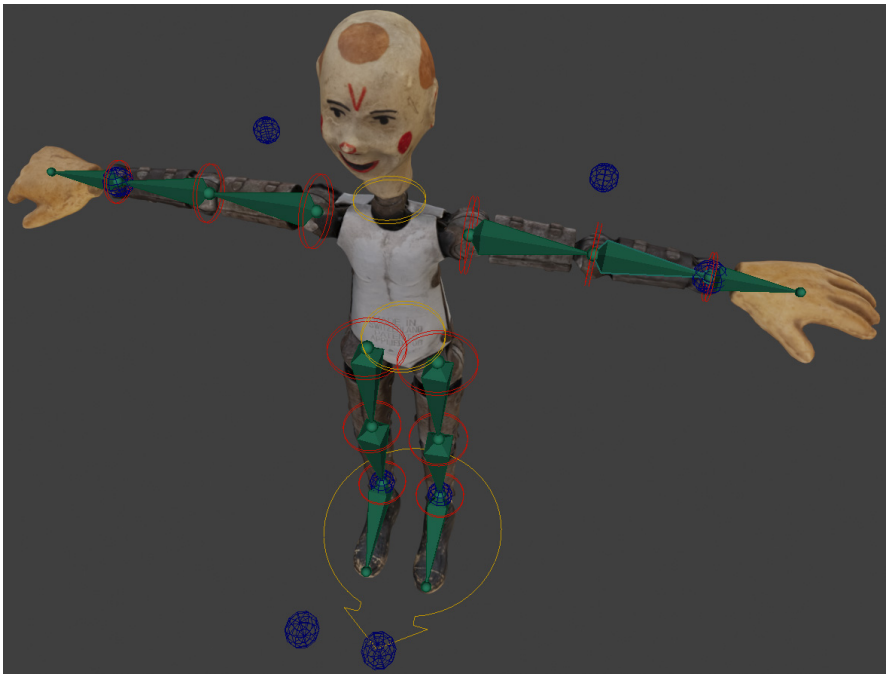
dopředná kinematika – červené

inverzní kinematika – fialové

ovladače inverzní kinematiky – modré

Pro některé kosti jsou zároveň vytvořené vlastní modely pro lepší odlišení a vizualizaci. Kořenová kost je reprezentovaná kruhem s šipkou, která určuje orientaci modelu, ostatní kontrolní kosti a kosti pro dopřednou kinematiku jsou reprezentovány dvojicí kružnic a ovladače inverzní kinematiky jsou reprezentovány drátěným modelem koule. Kostru lze vidět na obrázku 4.5. Vytvořenou kostrou lze identicky pózovat obě verze modelu.

Inverzní kinematika je implementovaná standardním `Blender` způsobem, tedy přes kosterní omezení *(IK Bone Constraint)*. Každá končetina má dva ovládací prvky – jeden pro pozici koncového efektoru a druhý pro orientaci kolen a loktů.



Obrázek 4.5: Kostra robota

Deformační kosti končetin kopírují polohu kostí inverzní a dopředné kinematiky. Mezi nimi je možno přepínat pomocí implementovaného ovladače.

4.4 Animace

Animace jsou vytvořeny v animačním systému, který je implementován v programu `Blender`.

Některé animace jsou vytvořeny pomocí ovladačů dopředné kinematiky a některé pomocí inverzní kinematiky. Zároveň jsou některé animace cyklické, což znamená, že se plynule opakují. To znamená, že poslední snímek musí navazovat na první snímek, nikoliv, že jsou tyto snímky stejné. Tyto detaily jsou popsány v tabulce 4.1.

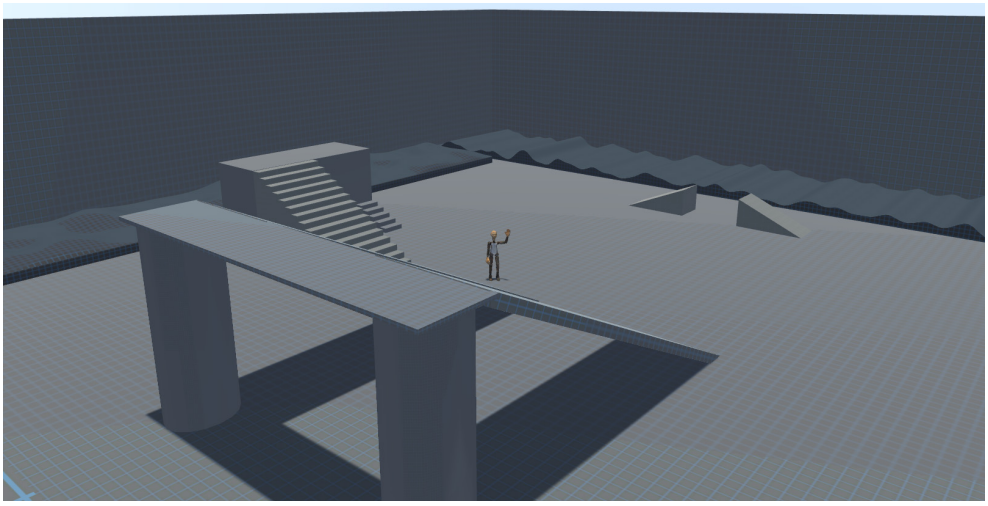
Animace jsou vytvořeny způsobem *in-place*, tedy na místě, protože se pohyb postavy implementuje v `Unity` pomocí skriptů. Tedy postava chodí i běhá na místě a při skoku nenabírá výšku.

4.5 Aplikace

Aplikace obsahuje základní hlavní menu, ze kterého je možné hru spustit, zobrazit informace o projektu nebo aplikaci ukončit. Po spuštění hry může hráč ovládat robota ve vytvořené jednoduché scéně.

Tabulka 4.1: Detaily vytvořených animací

Animace	Počet snímků	Cyklická	Kinematika
T-pose	1	—	—
Idle	49	ano	IK
Walk	24	ano	IK
Run	12	ano	IK
Wave	40	ne	FK
Jump start	14	ne	IK
Jump mid	18	ano	IK
Jump land	17	ne	IK



Obrázek 4.6: Scéna aplikace

Scéna obsahuje řadu objektů, které slouží pro demonstraci animací. Mezi objekty například patří rampy, schody a hrbolatý povrch pro demonstraci inverzní kinematiky. Scéna nemá žádný úkol ani cíl. Scénu lze vidět na obrázku 4.6.

Robot se ovládá pomocí vstupů popsaných v tabulce 4.2. Animaci zamávání lze spustit v kombinaci se všemi ostatními animacemi, viz obrázek 4.7. Pokud robot spadne či seskočí z příliš vysokého objektu, přehraje se animace dopadu. Pomocí klávesy *Escape* se hra pozastaví a zobrazí se menu, ve kterém si hráč může zobrazit ovládání nebo se navrátit do hlavního menu.

Kameru lze myší volně pohybovat dokola kolem robota. Pomocí kolečka lze kameru přibližovat či oddalovat. Podržením pravého tlačítka v kombinaci s vertikálním posunutím myši se mění střed zájmu kamery. Kamerou nelze projít skrze zdi a ostatní objekty ve scéně.

Inverzní kinematika srovnává robotovy nohy a trup v závislosti na nerov-

Tabulka 4.2: Ovládání aplikace

Chůze	šipky / „WASD“
Běh	levý shift
Skok	mezerník
Ovládání kamery	rotace – pohyb myší přiblížení – kolečko vertikální posunutí – RMB + pohyb myší
Zamávání	H
Zapnutí/Vypnutí IK	I
Menu + pauza	escape

Obrázek 4.7: Míchání animací nohou s animacemi rukou

nosti povrchu. Toho je docíleno vysíláním dvou paprsků z pozice každé nohy, jeden paprsek směrem kolmo do země a druhý ve směru normály plochy, nad kterou je noha. Z normály a míst dopadu paprsků se vypočítá cílová pozice a rotace koncového efektoru. Využití inverzní kinematiky v aplikaci lze vidět na obrázcích 4.8, 4.9 a 4.10.

Obrázek 4.8: Aplikace inverzní kinematiky (příklad 1)

Obrázek 4.9: Aplikace inverzní kinematiky (příklad 2)

Obrázek 4.10: Aplikace inverzní kinematiky (příklad 3)

Závěr

Cílem práce bylo upravit 3D model robota přiděleného od vedoucího práce, aby byl vhodný pro animaci. Následně tomuto modelu vytvořit kostru, pomocí které bude možné model animovat. S touto kostrou vytvořit sadu animací a nakonec vytvořit jednoduchou aplikaci, kde tyto animace bude možné demonstrovat.

Model byl v programu *Blender* upraven tak, aby se dalo manipulovat s jeho individuálními částmi a pózovat ho. Pro usnadnění pózování byla vytvořena kostra, která má jak ovládací a deformační kosti, tak kosti pro dopřednou a inverzní kinematiku. Mezi těmito systémy je možné se v programu *Blender* volně přepínat pomocí vytvořeného *driveru*. Zároveň byla vytvořená jednodušší verze modelu, s nižším stupněm detailů. Pro model byly vytvořeny animace základní pózy, chůze, běhu, skoku a zamávání. Tyto animace byly zaznamenány pomocí klíčových snímků. Animace s modelem byly exportovány do herního engine *Unity*, kde byla vytvořena aplikace.

Aplikace obsahuje hlavní menu, kde lze spustit hru, zobrazit informace o aplikaci nebo ji ukončit. Ve hře může hráč pomocí kláves chodit, běhat a skákat s postavou ve vytvořeném prostředí, nebo spouštět animaci zamávání. V aplikaci je implementovaný pohyb kamerou, která následuje postavu a lze ovládat myší. Pomocí procedurální inverzní kinematiky se nohy postavy přizpůsobují nerovnosti podlahy. Hru lze pomocí druhého menu pozastavit, zobrazit ovládání nebo se vrátit zpět do hlavního menu.

V budoucnu by bylo možné práci rozšířit v několika směrech. Pro model lze vytvořit více LoD. Následně je v aplikaci možné implementovat jejich přepínání v závislosti na vzdálenosti od kamery. Pro kostru je možné vyvinout `Blender` skript, který přesune kosti IK do polohy kostí FK a naopak. To umožní snazší kombinaci těchto přístupů při animování. Pro model lze vytvořit další animace, které lze následně v aplikaci demonstrovat. Příkladem může být bojový systém nebo překonávání překážek. Systém inverzní kinematiky je možné vylepšit pro lepší chování v extrémních úhlech, například vysíláním více paprsků.

Literatura

[1] CHOPINE, Ami. *3D art essentials: the fundamentals of 3D modeling, texturing, and animation.* Boston: Elsevier/Focal Press, 2011. ISBN 978-0240814711.

[2] ZEMAN, Nicholas Bernhardt. *Essential Skills for 3D Modeling, Rendering, and Animation* [online]. Boca Raton: CRC Press, Taylor & Francis, 2015 [cit. 2020-03-26]. ISBN 978-1-4822-2414-6.

[3] Structure – Blender Manual. *Blender Documentation – blender.org* [online]. [cit. 28.03.2020]. Dostupné z: `https://docs.blender.org/manual/en/latest/modeling/meshes/structure.html`

[4] RATNER, Peter. *Mastering 3D animation.* 2nd ed. New York, NY: Allworth Press, 2004. ISBN 1-58115-345-7.

[5] HORMANN, Kai a Marco TARINI. *A quadrilateral rendering primitive.* In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware - HWWS '04 [online]. New York, New York, USA: ACM Press, 2004, 2004, s. 7-14 [cit. 2020-03-26]. DOI: 10.1145/1058129.1058131. ISBN 3905673150. ISSN 17273471.

[6] ŽÁRA, Jiří, aj. *Moderní počítačová grafika.* 2., přeprac. a rozš. vyd. Brno: Computer Press, 2005. ISBN 80-251-0454-0.

[7] Triangulating before baking – Substance Bakers. [online]. [cit. 28.03.2020]. Dostupné z: `https://docs.substance3d.com/bake/triangulating-before-baking-159451841.html`

[8] Topology – 3D Modeling Resources. *TurboSquid* [online]. Copyright © TurboSquid 2017 [cit. 28.03.2020]. Dostupné z: `https://www.turbosquid.com/3d-modeling/checkmate/checkmate-product-presentation/topology/`

[9] How to Identify Topology in 3D Animation. *Lifewire* [online]. [cit. 2020-03-28]. Dostupné z: https://www.lifewire.com/topology-in-3d-animation-2181

[10] Quad-Based Toplogy in Your 3D Model – 3D Modeling Resources. *TurboSquid* [online]. Copyright © TurboSquid 2017 [cit. 28.03.2020]. Dostupné z: https://www.turbosquid.com/3d-modeling/training/modeling/quad-based-topology/

[11] Wunkolo [Do you have any good guides for topology when it...] In: Tumblr [online]. 9. června 2015 [cit. 28.03.2020]. Dostupné z: https://wunkolo.tumblr.com/post/121136952712

[12] HUGHES, John F. *Computer graphics: principles and practice.* 3rd ed. Upper Saddle River: Addison-Wesley, 2014. ISBN 978-0321399526.

[13] ZACH, Christopher. *Integration of geomorphing into level of detail management for realtime rendering.* In: Proceedings of the 18th spring conference on Computer graphics - SCCG '02 [online]. New York, USA: ACM Press, 2002, s. 115-122 [cit. 2020-03-26]. DOI: 10.1145/584458.584478. ISBN 1581136080.

[14] KRS, Vojtěch. *Sculpting in Virtual Reality* [online]. Czech Technical University in Prague, 2014 [cit. 2020-03-26]. Dostupné z: https://dcgi.fel.cvut.cz/theses/2014/krsvojte. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology.

[15] MODELING vs SCULPTING: how do you know which to use?. *METHOD: J* [online]. [cit. 2020-03-29]. Dostupné z: https://www.methodj.com/modeling-vs-sculpting/

[16] Sketchfab Community Blog - How to retopologize 3D scans into low poly game assets. *Sketchfab* [online]. [cit. 28.03.2020]. Dostupné z: https://sketchfab.com/blogs/community/retopologise-3d-scans-low-poly-game-assets/

[17] What is Photogrammetry? *Photogrammetry* [online]. [cit. 2020-03-29]. Dostupné z: http://www.photogrammetry.com/

[18] LINDER, Wilfried. *Digital Photogrammetry: A Practical Course* [online]. 4. vydání. Berlin: Springer, 2016 [cit. 2020-03-26]. ISBN 978-3-662-50463-5. Dostupné také z: 10.1007/978-3-662-50463-5

[19] BARSANTI, Sara Gonizzi, Fabio REMONDINO a Domenico VISINTINI. *Photogrammetry and Laser Scanning for Archaeological Site 3D Modeling - Some Critical Issues* [online]. University of Trieste, 2012 [cit. 2020-03-29]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.416.9595.

[20] FOSTER, Shaun a David HALBSTEIN. *Integrating 3D Modeling, Photogrammetry and Design* [online]. London: Springer, 2014 [cit. 2020-03-26]. ISBN 978-1-4471-6329-9. Dostupné také z: 10.1007/978-1-4471-6329-9

[21] TRAMMELL, Kent. Big Idea: „Baking". In: *Cgcookie.com* [online]. 3. května 2016 [cit. 2020-05-04]. Dostupné z: `https://cgcookie.com/articles/big-idea-baking`

[22] BEANE, Andy. *3D Animation Essentials.* Wiley, 2012. ISBN 978-1-118-14748-1.

[23] PARENT, Rick. *Computer animation: algorithms and techniques.* San Francisco: Morgan Kaufmann Publishers, c2002. ISBN 978-1558605794.

[24] HODGINS, Jessica K., James F. O'BRIEN a Robert E. BODENHEIMER. *Computer Animation* [online]. Wiley, 1999 [cit. 2020-03-26]. Dostupné z: `http://www.vuse.vanderbilt.edu/~bobbyb/pubs/ency99.html`

[25] JEPPSSON, Daniel. *Realtime Character Animation Blending Using Weighted Skeleton Hierarchies* [online]. Lund University, 2000 [cit. 2020-03-26]. Dostupné z: `http://fileadmin.cs.lth.se/graphics/theses/reports/jeppsson_masterthesis.pdf`. Master Thesis. Lund University, Faculty of Engineering.

[26] GREGORY, Jason. *Game engine architecture.* Third edition. Boca Raton: Taylor & Francis, CRC Press, 2018. ISBN 9781138035454.

[27] Procedural Animation. *The University of Texas at Dallas* [online]. [cit. 2020-03-29]. Dostupné z: `https://www.utdallas.edu/atec/midori/Handouts/procedural_animation.htm`

[28] JOHANSEN, Rune Skovbo. *Automated Semi-Procedural Animation for Character Locomotion* [online]. Aarhus University, 2009 [cit. 2020-03-26]. Dostupné z: `http://runevision.com/thesis/`. Master's Thesis. Aarhus University, Department of Information and Media Studies.

[29] PEČENKA, Michal. *3D animace postavy v počítačové grafice* [online]. Vysoké učení technické v Brně, 2008 [cit. 2020-03-26]. Dostupné z: `https://dspace.vutbr.cz/handle/11012/53224`. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

[30] MUKUNDAN, Ramakrishnan. *Advanced Methods in Computer Graphics: With examples in OpenGL* [online]. London: Springer London, 2012 [cit. 2020-03-25]. ISBN 978-1-4471-2340-8. Dostupné také z: 10.1007/978-1-4471-2340-8

[31] KAVAN, Ladislav, Steven COLLINS, Jiří ŽÁRA a Carol O'SULLIVAN. *Skinning with Dual Quaternions* [online]. 2007 [cit. 2020-04-14]. Dostupné z: 10.1145/1230100.1230107

[32] WATT, Alan H. a Mark WATT. *Advanced animation and rendering techniques: theory and practice.* Reading, Mass.: Addison-Wesley Pub., c1992. ISBN 978-0201544121.

[33] NILSSON, Rickard. *Inverse kinematics* [online]. 2009 [cit. 2020-03-26]. Dostupné z: `http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1018821&dswid=5431`. Master's thesis. Luleå University of Technology.

[34] KOVAR, Lucas a Michael GLEICHER. *Flexible automatic motion blending with registration curves* [online]. San Diego, California: Eurographics Association, 2003 [cit. 2020-04-15]. ISBN 1581136595. Dostupné také z: 10.5555/846276.846307

[35] HALPERN, Jared. The What and Why of Game Engines. In: *Medium* [online]. 11. prosince 2018 [cit. 2020-04-15]. Dostupné z: `https://medium.com/@jaredehalpern/the-what-and-why-of-game-engines-f2b89a46d01f`

[36] Comparison: the Top 6 Most Popular 3D Modeling Software. *Medium* [online]. 17. října 2018 [cit. 30.03.2020]. Dostupné z: `https://medium.com/imeshup/comparison-the-top-6-most-popular-3d-modeling-soft-1c6a9ed204a4`

[37] THE BLENDER FOUNDATION. *Blender 2.81* [software]. 13. října 2002. [přístup 2020-04-28]. Dostupné z: `https://www.blender.org/download/`

[38] UNITY TECHNOLOGIES. *Unity 2019.3.4f1* [software]. [přístup 2020-04-29]. Dostupné z: `https://unity3d.com/get-unity/download/`

[39] Godot Review. *Slant* [online]. [cit. 2020-05-18]. Dostupné z: `https://www.slant.co/options/1068/~godot-review`

[40] DEALESSANDRI, Marie. What is the best game engine: is CryEngine right for you? In: *gamesindustry.biz* [online]. 16. ledna 2020 [cit. 2020-05-18]. Dostupné z: `https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-cryengine-the-right-game-engine-for-you`

[41] AUTODESK. *Meshmixer 3.5.474* [software]. 2017. [přístup 2020-04-28]. Dostupné z: `http://www.meshmixer.com/download.html`

[42] PROVOST, Guillaume, 2003. Beautiful, Yet Friendly Part 1: Stop Hitting the Bottleneck. In: *Ericchadwick.com* [online]. [cit. 2020-05-04]. Dostupné z: `http://www.ericchadwick.com/examples/provost/byf1.html`

[43] Optimizing graphics performance, *Unity Manual* [online]. [cit. 2020-05-04]. Dostupné z: `https://docs.unity3d.com/Manual/OptimizingGraphicsPerformance.html`

[44] HAIGH-HUTCHINSON, Mark. *Real-time cameras: a guide for game designers and developers*. Oxford: Elsevier Science [distributor], 2009. ISBN 978-0-12-311634-5.

Příloha **A**

Seznam použitých zkratek

3D; 2D Three-dimensional; Two-dimensional

CLoD Continuous LoD

DOF Degree of Freedom

DLoD Discrete LoD

FK Forward Kinematics

FPS Frames Per Second

GPU Graphics Processing Unit

IK Inverse Kinematics

LED Light-Emitting Diode

LERP Linear Interpolation

LoD Level of Detail

NTSC National Television System Committee

NURBS Non-Uniform Rational Basis Splines

PAL Phase Alternating Line

Quad Quadrilateral

RMB Right Mouse Button

SRT Scale, Rotation, Translation

SQT Scale, Quaternion, Translation

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou aplikací
- `src`
  - `impl` ..... zdrojové soubory implementace
    - `blender` ..... `Blender` soubory
    - `unity` ..... `Unity` soubory
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF