České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Radek Smetana**

Studijní program: Otevřená informatika
Obor: Počítačová grafika a interakce

Název tématu: **Skripty pro tvorbu animační kostry**

Pokyny pro vypracování:

1) Popište a analyzujte postup aplikace zaznamenaného pohybu reálné postavy (RAW data z motion capture) na model postavy.
2) Vyberte kroky při tvorbě animační kostry s kontrolery, které by se daly vhodně automatizovat pomocí skriptovacího jazyka a vytvořte základní knihovnu skriptů.
3) Skripty použijte na třídách reálných dat vybraných po dohodě s vedoucím a zhodnoťte je z pohledu univerzálnosti a následné aplikovatelnosti na další data a modely.
4) Bod 1) pojměte jako tutoriál, použitelný jako učební text na VŠ.

Seznam odborné literatury:

How to Cheat in Maya 2012 - Tools and Techniques for Character Animation. Eric Luhta, Kenny Roy Focal Press (2011)

Art of Rigging Volume I. Kiaran Ritchie, Jake Callery, Karim Biri, CG Toolkit (2006)

Maya Python for Games and Film: A Complete Reference for Maya Python and the Maya Python API. Adam Mechtley, Ryan Trowbridge, Elsevier (2012)

Vedoucí: Ing. David Sedláček, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

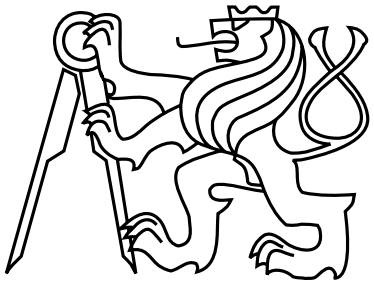
L.S.

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 4. 11. 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce



Diplomová práce

Skripty pro tvorbu animační kostry

MgA. Radek Smetana

Vedoucí práce: Ing. David Sedláček, Ph.D.

Studijní program: Otevřená informatika

Obor: Počítačová grafika a interakce

5. ledna 2015

Poděkování

Děkuji svým rodičům, bratorovi a Lucie Vrbské za podporu při studi, děkuji Luboru Zelinkovi za podnětnou konzultaci. Dále dlužím velké díky Gerardu Verronovi za otestování skriptů a v neposlední řadě děkuji Davidu Sedláčkovi za vstřícný přístup a cenné rady.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 5. 1. 2015 .....

Abstract

The diploma thesis deals with the preparation of computer model for animation. This process, also known as rigging, is present during creation of most of computer effects. The aim of the thesis is to explain, how to create a system that can simplify the process of rigging. Furthermore, it describes functioning of Autodesk Maya application and covers specific issues and solutions related to this topic. In conclusion all the results are tested, assessed and special attention is given to obstacles which may occur during the implementation.

Second part of the thesis contains a detailed guide, which describes a process of obtaining and application of motion capture system data. This is demonstrated on specific configurations and programs. Thanks to the guide it is easy even for an inexperienced user to orientate in the process.

Abstrakt

Tato práce se do hloubky zabývá přípravou počítačového modelu pro animaci. Tento proces, který je přítomný při vzniku téměř všech počítačových efektů, je ve světě známý jako rigging. Předmětem této práce je seznámit čtenáře s tím, jak vytvořit systém, který rigging dokáže usnadnit. Dále popisuje fungování aplikace Autodesk Maya v návaznosti na tuto problematiku a předkládá konkrétní úlohy a řešení, která jsou do hloubky rozebírána. V závěru všechny výsledky testuje, hodnotí a předkládá názor na zmíněné překážky, které se při implementaci vyskytly.

Druhou částí práce je podrobný návod, který popisuje na konkrétních konfiguracích a programech proces získání a aplikace dat ze systému motion capture. Díky němu je možné, aby se i nezkušený uživatel v programech zorientoval a celým procesem od počátku do konce úspěšně prošel.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Obor animace zažil s nástupem počítačové grafiky do kategorie audiovizuální tvorby velký převrat. Od doby, kdy byl uveden první animovaný film vygenerovaný počítačem, pohádka Toy Story, jsou ručně kreslené filmy na ústupu. Podobný trend je viditelný i u hraných filmů, přesněji trikových záběrů, kdy se dnes již téměř nepoužívají klasické triky využívající například optické klamy, kresby na sklo či loutky. Všechny efekty se tak provádí postprodukčně v počítači především díky technickému postupu v oblasti výpočetní a paměťové kapacity. Světlocitlivé snímače postupně vytlačily z pozice záznamového média filmový pás, který se v současné době již téměř nepoužívá, vše je zaznamenáváno digitálně. Díky technickému rozvoji přestávají být simulace a náročné programy výsadou trikových studií a jsou stále dostupnější i pro běžného uživatele.

Trikové záběry i animované filmy jsou stále složitější. Obsahují kvalitní fyzikální simulace, věrné fyzikální modely pro vykreslování a nemají problém s velkými scénami o vysokém počtu objektů. Tím rostou požadavky na um a specializaci všech osob, které jsou do výrobního procesu zařazeny. Tito pracovníci používající animační software se obvykle zaměřují na modelování, texturování, rigging, animaci, programování, dynamiku, osvětlení nebo renderování. Tato práce se zabývá právě jednou z těchto specializací, konkrétně riggingem.

Rigging se zabývá přípravou počítačového modelu pro animaci. Nastaví, jakých pohybů a výrazů bude model schopný a určí, jak při nich bude vypadat. Pro jeho ovládání vytvoří sadu kontrolerů, přes které bude animátor model ovládat tak, aby byl co nejvíce odstíněn od samotného fungování modelu a jeho rigu. Tato disciplína vyžaduje míru technických znalostí a zároveň i umělecké cítění. Rigger sice nastaví a naprogramuje rig tak, aby všechny jeho vrcholy, hrany a plošky, z kterých se takový polygonální model skládá, byly vždy v té správné pozici, zároveň v ní však musí vypadat přirozeně a výtvarně správně. Rigger tedy určí, jak objekt bude vypadat ve všech ostatních pozicích, než v kterých byl vytvořen. Toto odvětví tak vyžaduje dobrou znalost animačního programu, požadavků a standardů animátorů a zkušenosti s filosofií riggingu.

Rig modelu se dá rozdělit na dva typy. Prvním typem je systém, který obsahuje kotrolery přizpůsobené pro ruční animaci. Animátor často do podoby rigu zasahuje a vznáší na něho požadavky, aby se mu s rigem dobře pracovalo a aby ho rig při animaci co nejméně omezoval. Druhý typ rigu je připravený pro příjem dat ze systému motion capture. U tohoto typu je animátor zaměněn za herce, jehož pohyby jsou digitálně zpracovány a předány právě

rigovému systému. V první fázi je proces rigování shodný pro oba typy. V pozdější fázi se však zásadně rozcházejí.

Cílem této práce je zmapování oblasti riggingu bez předpokladu předchozí znalosti této profese, popsat problémy s riggingem spojené a představit jejich řešení. Tyto řešení by měly problematiku prozkoumávat do hloubky a používat ten nejvhodnější způsob. Tato práce prozkoumá a popíše řadu rigovacích technik a s využitím skriptovacího jazyka animačního programu tyto techniky zautomatizuje. Tak vznikne knihovna skriptů, která vznik rigu co nejvíce zjednoduší. Skripty budou odzkoušeny na konkrétních 3D modelech. Práce popíše vznik obou typů rigu, jak ty připravené pro ruční animaci, tak vznik rigu pro data ze systému motion capture. Část práce o motion capture navíc bude obsahovat menší odklon od tématu profese rigování, kdy bude detailně popsán proces pořízení dat tímto systémem až po aplikaci dat na vytvořený rig modelu. Tato část se nachází v příloze A.

Kapitola 2

Seznámení s problematikou

2.1 Prostorový počítačový model

V roce 1967 byl vytvořen základ počítačové grafiky, která je schopna pracovat s prostorovými modely a jejich zobrazením v perspektivě. [LG02] Od té doby vznikla řada způsobů, jak prostorový model popsat, každý z nich má své přednosti a tak i úlohy, pro které je jeho použití nejvhodnější. Jak vysvětluje [JS10], používá se několik typů prostorových křivek, které slouží jako základ pro popis plochy. Plošková reprezentace objekt zase dokáže popsat použitím mnoha malých rovných ploch, které na sebe navazují. Tyto přístupy se zabývají pouze plochou objektu, tato reprezentace se proto nazývá hraniční. Údaje o vnitřní struktuře objektů popisuje objemová reprezentace pomocí vzorků nesoucích informaci jak o své pozici v prostoru, tak o nějaké fyzikální veličině. Vzorky jsou často uspořádány v prostorové mřížce, mohou být však i rozptýlené.

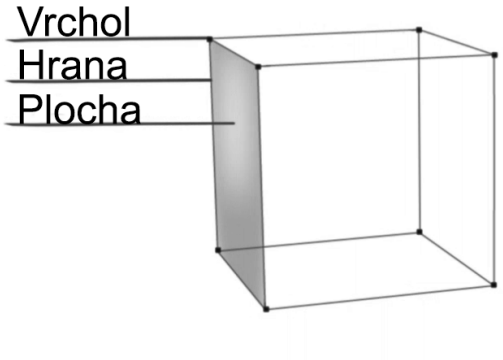
2.1.1 Struktura polygonálního modelu

Typ 3D modelů, kterými se tato práce bude zabývat, využívá ploškovou reprezentaci. 3D model tohoto typu se skládá ze tří druhů komponent: z vrcholů, hran a ploch. Vrcholy jsou umístěny v prostoru, jsou spojeny hranami. Skupiny hran, které uzavírají cyklus, mohou obsahovat plochu, která vytvoří část této hraniční reprezentace. Lze tedy říci, že vrcholy poskytují informaci o geometrii objektu, protože jsou na ně navázaná data o jejich pozici v prostoru. Hrany a plochy popisují topologii, neboli to, jak jsou vrcholy spojeny a jakým způsobem tvoří výsledný tvar. (Obrázek 2.1)

Plochy

Plochy označované také jako polygony neboli mnohoúhelníky by měly popisovat rovinný útvar. Pokud se jedná ovšem o komponent o více jak třech vrcholech, mohou vznikat nerovinné polygony. Proto některé aplikace dovolují práci pouze s trojúhelníky, jiné zase před vykreslením tyto polygony na trojúhelníky převádějí na pozadí tím, že do polygonu přidají vhodným způsobem další hrany.

Protože polygony jsou rovinná tělesa, je vždy možné na ně nahlížet vždy ze dvou stran. U polygonu pomocí daného předpisu je možné určit přivrácenou a odvrácenou stranu. Tu



Obrázek 2.1: Polygonální tvar tvořící krychli
Zdroj: vlastní

definuje normála plochy, vektor kolmý na daný polygon. Polygon by měl být v objektu vždy použit tak, aby byl vidět pouze z přivrácené strany. Toto pravidlo definuje přístup k polygonálnímu modelování, navíc mnoho programů pro urychlení vykreslování odvrácené plochy zanedbává.

Hrany

Při aproximaci tvaru polygonálním objektem je třeba rozlišovat, zda je hrana polygonu pouze hranou pomocnou nebo hranou ostrou. Pomocná hrana má simulovat zakřivení povrchu a neměla by být vidět, naopak hrana ostrá se má na objektu opravdu zobrazit. Klasifikace hrany může být ponechána na zobrazovací fázi, ve které se rozhoduje obvykle podle úhlu mezi sousedními polygony, který na hraně vzniká. Alternativou tohoto přístupu je nechat přímo na uživateli při vytváření modelu, aby rozhodl, která hrana má být ostrá a která nikoli.

Body UV

Aby byl popis komponent polygonálního modelu úplný, je třeba ještě zmínit body UV. Ty slouží k nanášení dvourozměrné textury na prostorový objekt. Jeden vrchol objektu může mít přiřazený jeden či více UV bodů, kdy každý z nich má definovanou polohu v dvourozměrném prostoru. UV body jsou spojeny stejně jako vrcholy objektu. Některé hrany však tvoří texturovací hranici a v dvourozměrném prostoru se vyskytují dvakrát. Je možné si tyto hrany představovat jako řez objektem, který pomáhá prostorový objekt rozvinout do plochy, v které se ideálně žádné polygony nebudou překrývat.

2.1.2 Model vhodný pro animaci

Je více technik, které vedou ke vzniku počítačového modelu. Podrobné informace o tvaru může poskytnout 3D scanner, který dokáže reálný objekt převést do digitální formy. Vysoké

detaily lze získat také pomocí programu pro 3D sochání, kdy výtvarník ve virtuálním prostoru přidává a ubírá materiál tak, jako by pracoval s opravdovou hmotou. U takových modelů však může nastat problém s jejich topologií a vysokou podrobností. Obzvlášť, pokud bude model využit pro animaci, je třeba jeho strukturu zjednodušit a vytvořit takovou topologii, která bude k animaci vhodná. Aby při zjednodušení modelu byla data zachována, ukládají se do výškové textury. Ta je využita při finálním vykreslování modelu, a tak jsou detaily modelu opět navráceny.

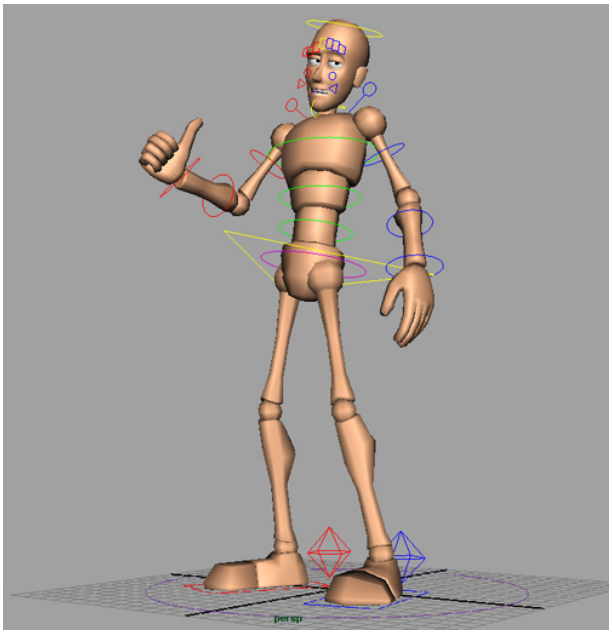
Zjednodušení modelu může program provést automaticky. Ten však topologii vhodnou pro animaci často nevytvoří, proto je třeba model ještě upravit nebo celý znovu vytvořit pomocí starší techniky. Tou může být polygonální modelování či tvorba modelu pomocí křivek, u kterých má výtvarník možnost zasáhnout do pozice a nastavení každého komponentu modelu. Topologie se prací s jednotlivými komponenty dá i snadno upravovat tak, že se tvar téměř nezmění. Jen se vytvoří alternativní propojení vrcholů s hranami. Tak vznikne objekt, který je možné při animaci správně ohýbat, protahovat či vlnit.

Jak popisuje [Tin12], 3D model vhodný k animaci, by se měl skládat z ploch o čtyřech vrcholech. Ty by měly tvořit prstencovité struktury v takových směrech, v kterých se dají logicky natahovat či vrásnit. Hrany těchto struktur se nazývají hranové smyčky (edge loops).Pozice každého vrcholu modelu musí mít logické opodstatnění plynoucí ze struktury a objemu modelu, popřípadě z výrazů, kterých má model docílit. Modelování například kvalitního obličeje tak podléhá jasným zásadám, které definují přesný počet hranových smyček jednotlivých částí obličeje nutných k vyjádření dané škály výrazů.

2.2 Rigging

Práce s 3D počítačovými modely obsahuje mnoho dílčích článků, které vedou k vytvoření výsledného záběru. Ať už se jedná o scénu, která kombinuje počítačovou grafiku se záběrem pořízeným v reálném prostředí, nebo jde o sekvenci čistě animovaného filmu, často je potřeba počítačové objekty animovat. A právě rigging je ten článek procesu, který z počítačových modelů vytvoří plně funkční systém, který je připravený pro animaci. Animátor má tak k dispozici model s kontrolery (Obrázek 2.2), který rozhýbe jen pomocí změny pozice, rotace, měřítka a speciálně připravených atribut těchto prvků. Právě rig určí to, jaké funkce jsou animátorovi k dispozici a jak se změny atribut projeví na modelu.

Pro vykonávání práce riggera je třeba znát velmi dobře funkce animačního softwaru. Až poté je rigger schopen úspěšně kombinovat funkce tak, aby vznikaly kontrolery, které pracují společně v jednom rigovém systému a které dělají přesně to, co se od nich očekává. Jak bylo zmínění na [Car13], k dosažení takového cíle není podmínkou, aby rigger byl programátor a rozuměl všem procesům animačního softwaru v úrovni kódu. Přístupů k riggingu je totiž celé spektrum, kde na jedné straně je pracovník zaměřený právě na kódování, na straně druhé je rigger prosazující více uměleckou stránku. Velké množství funkcí lze zpřístupnit bez znalosti nějakého programovacího jazyka, při mírné znalosti programování lze základní automatizace dosáhnout. Rozsáhlé skripty pro usnadnění rigovacích procesů sice mnoho času ušetří, ale poměrně často si každý rig žádá speciální úpravy, které program nikdy nezvládne provést s dostatečnou přesností, nebo se v budoucnu již nevyužijí.



Obrázek 2.2: Model s kontrolery
Zdroj:<http://seanburgoon.com/wordpress/wp-content/uploads/2011/03/goonRig_Splash.jpg>

2.2.1 Animační kostra

Hlavním animačním nástrojem pro 3D modely je virtuální kostra. Jde o virtuální prvek, který není zahrnut při vykreslování. Skládá se z řetězce kloubů, z jointů, které jsou navzájem v hierarchickém stavu potomek-rodič. Tento vztah je znázorněn grafickým propojením takto spojených kloubů, kterému se říká kost. Posuny, otáčení a v některých případech i změny měřítka kloubu jsou distribuovány pomocí hierarchických vztahů na všechny potomky upravovaného kloubu. Tak vzniká struktura, která svou funkcí připomíná kostru.

Skinning je proces, díky kterému je možné propojit pohyb kostry s polygonálním objektem. Typů propojení je více, všechny však řeší ten samý problém. Jak co nejvhodněji distribuovat posun, rotaci a změnu měřítka kostí na jednotlivé vrcholy přiřazeného objektu. Neohýbající se místa objektů lze ovládat transformacemi jedné kosti. Ty části objektu kolem kloubů, které se ohýbají a natahují, jsou ovlivněny kostí více, proto je třeba transformace těchto vrcholů ovládat pomocí nějakého nejlépe uživatelsky upravitelného předpisu. Protože pozice kloubu definuje střed otáčení kosti a tak i všech závislých vrcholů objektu, je přesné umístění kloubu v modelu velmi důležité a nepřesnosti jsou dobře viditelné. Proto je dobré při tvorbě kostry používat referenční materiály a strukturu si dobře promyslet.

Při skinningu může v extrémnějších ohybech docházet k úbytku objemu jednotlivých svalů. Jak ukazuje [Tin12], nápomocné může být využití modelů reálných kostí, které se umístí do objektu. Rigger má pak lepší přehled o tom, co se v objektu odehrává a zda k úbytku objemu dochází. Pokud se takový problém vyskytne, jednoduché přerozdělení vah kostí, které působí na jednotlivé vrcholy modelu, nemusí být pro odstranění takovéhoto problému dostatečné. Pozice vrcholů lze dále upravit v konkrétních ohybech dvěma způsoby.

Využití systému svalstva je prvním z nich. Pod hraniční reprezentací modelu je vytvořena síť důležitých svalů, kolem kterých se pak vrcholy (kůže) natahují. Druhou možností je dle [Car13] vytvořením tzv. correctives. To je soubor pomocných verzí modelu, v které se původní model transformuje právě v těch případech, které korekci potřebují. To znamená, že rigger upraví v hraničních pozicích umístění vrcholů ručně.

2.2.2 Dopředná a inverzní kinematika

Jednotlivé klouby je třeba nějakým způsobem ovládat. Technika zvaná dopředná kinematika, forward kinematic (FK), definuje otočení jednotlivých kloubů s dodržením jejich hierarchie. Rotace kloubu z vyšší hierarchie tak ovlivní všechny jeho potomky. Hodí se pro animaci například mávající ruky, pohyb hlavy či dlouhých uší. Pro animaci, kdy je třeba zapojit více kloubů tak, aby poslední kloub směřoval do předem zvoleného bodu, se však tato technika příliš nehodí. Jak vysvětluje [JS10], k tomuto účelu je vhodné použít inverzní kinematiku.

Inverzní kinematika (IK) je nejčastěji používaná u řetězců kloubů, které ovládají končetiny. Na konci takového řetězce je definován kontroler IK řetězce. Jeho pozice pomocí řady výpočtů definuje natočení všech kloubů řetězce tak, aby měl poslední kloub vždy shodnou pozici s kontrolerem. Pomocí animace IK kontroleru se dá celá končetina animovat tak, že má animátor kontrolu nad pozicí posledního článku řetězce. Výpočty mohou měnit rotace kloubů řetězce ve všech třech osách, přičemž délky kostí zůstávají konstantní. Úloha vytvořena pozicí kontroleru nemůže dát žádné řešení v případě, když se kontroler nachází ve větší vzdálenosti od počátku řetězce, než je součet délek kostí. Při jiné pozici kontroleru může být zase více možných řešení. Proto je třeba nějakým způsobem definovat rotaci celého řetězce. Na základě použité metody pro výpočet je ovlivněno to, jak je výsledná animace hladká, jak rychle je počítaná a další parametry.

2.2.3 Deformátory

Další nástroje, které lze pro animaci polygonálního objektu využít, je možné nazvat společným označením deformátory. Ty mohou upravovat pozici vrcholů objektu pomocí nastavitelného předpisu či funkce. Lze tak posunout se speciálním kontrolerem i vrcholy v jeho okolí. Jiné funkce mohou celý objekt ohnout, rozvlnit čí obtočit. Funkce blendshape zase pracuje se dvěma podobnými objekty. Porovnává u nich rozdílný posun vrcholů, který byl uživatelsky vytvořen. Posun vrcholu umí funkce plynule aplikovat na cílový objekt, aby se podobal objektu vzorovému.

2.3 Animace

Audiovizuální materiál navozuje dojem pohybu pomocí zobrazení statických obrázků rychle za sebou. Jak vysvětluje [RM00], pokud lidský mozek dostane do jedné patnáctiny sekundy nový obraz, který se mírně liší od předchozího, bude vjem interpretovat jako plynulý pohyb. Lidské oko není schopno rozeznat, zda světlo bliká či svítí nepřetržitě, pokud je frekvence rozsvěcení 50 Hz, tedy 50 krát do sekundy. Nicméně podle [Bra12] je lidské oko schopno zaznamenat objekt, který je vidět pouze 1/220 sekundy. Proto se používá více standardů rychlosti přehrávání, u počítačových her dokonce frekvence zobrazování kolísá. V Evropě

je standart přehrávání 25 snímků za sekundu, anglicky frame per second (fps). Moderní televizory jsou však schopny rychlejšího zobrazování a snímky pro plynulejší zobrazení dopočítávají. Dalším trendem je natáčení materiálu o rychlejší frekvenci záznamu, než udává standard.

Při generování snímků počítačem není tak těžké vytvořit dojem pohybu, stačí jen v každém novém snímku plynule měnit některé z atribut popisující generovaný objekt. Ač je řada animačních technik shodná pro různé druhy techniky, 2D i 3D animaci, z důvodu zaměření této práce se text od počátku vymezí na animaci polygonálních objektů.

2.3.1 Klíčové snímky

Pokud máme ve scéně definovanou proměnnou popisující pozici v čase, je možné dle ní řídit změnu atribut objektu, jako je například pozice či rotace objektu. Obecně řečeno jde o propojení pozice v čase s hodnotou dané atributy. Při vzniku propojení lze říci, že byl vytvořen klíčový snímek. Měnící se atribut může být jakákoliv hodnota, nejčastěji se však jedná o datový typ float, tedy aproximace oboru reálných čísel. Mezi klíčovými snímky se hodnoty v čase nějakým způsobem interpolují, v programu Maya se jedná o interpolaci pomocí křivky. Maya průběh křivky definuje pomocí směru, popřípadě i váhy spojitých i nespojitých tečen, které je možné libovolně editovat. Při přehrávání jsou pak pro každý snímek hodnoty atribut jednoznačně určeny.

2.3.2 Ruční animace pomocí rigu

Animace počítačového modelu nevyžaduje pouze umělecké cítění, kvalitní animátor musí mít především velké zkušenosti v oboru a sadu nástrojů, která ho při animování nebude omezovat. Počítačový model, který je připravený pro animaci, obsahuje sadu kontrolerů, s kterými animátor může pracovat. Systém kontrolerů, animační kostry a dalších nastavení by měl animátora odstínit od fungování rigu i celého programu a nabídnout mu jen ty atributy a nástroje, které bude opravdu potřebovat. Animátor pro ně pak vytváří klíčové snímky a upravuje animační křivku, aby rozhýbal celý model. Často disponuje řadou referenčních materiálů ve formě kreseb či videonahrávek, které mu slouží při animaci jako vzor.

Stejně jako samotný model mohou být pohyby postavy zcela realistické, nebo mohou být nadsazeny jako v kreslených filmech. Ty sice také vychází z reálných pohybů, ale zdůrazňují u postavy náladu, její charakter a pohyb obecně, postava se pak nerealisticky prohýbá a deformuje. Animace tak dopomáhá vdechnout pohybům život a jiskru, dělá postavy zábavnější a lépe čitelné. Tyto techniky vznikly a byly zdokonalovány s rozvojem animovaného filmu od 30. let 20. století ve společnosti Walt Disney a později byly popsány v [LR12]. V kapitole 3.2.1 jsou podrobněji popsány a analyzovány, jak se dají využít pro animaci 3D modelu a zda z nich plynou nějaké požadavky pro animační rig.

2.3.3 Motion capture

Existuje řada případů, kdy ruční animace nemusí být jediné řešení pro vytvoření počítačových pohybů. Nevýhodou ruční animace je fakt, že se jedná o zdlouhavý proces, který se dá zrychlit jen do jisté míry. Nezbytnost velmi zkušených animátorů a čas potřebný k

vytvoření pohybů mohou být i při efektivním rozvržení práce příliš vysoké. Pokud je třeba vytvořit realistickou animaci, může být vhodnější použít systém pro záznam pohybu, který se označuje termínem motion capture. Výhodným se stává především v situacích, kdy je třeba velkého množství animací pro stejnou kostru, nebo pokud chce tvůrce přiřadit 3D postavě herecké výkony konkrétního herce. Proto své místo najde jak u vzniku počítačových her, tak u trikového filmu.

Motion capture ve většině případů zaznamenává pohyb na určité ploše, která se nazývá scéna. Herec, který ve scéně předvádí pohyby, tedy Actor, musí mít pro fungování většiny systémů na sobě připevněny markery. Ty se dle použité technologie rozdělují na pasivní a aktivní. Aktivní markery musí vyvíjet nějakou činnost, aby záznam pohybů fungoval. Z toho plyne jejich nevýhoda a to závislost na zdroji energie. Některé technologie používají aktivní markery přímo pro záznam své pozice v prostoru. Pasivní markery zdroj energie nepotřebují, pro identifikaci a uložení jejich polohy je použit vnější systém. Jak z tohoto obecného popisu plyne, je více technologií, které se používají pro záznam pohybu.

Většinu systémů je třeba při nové instalaci zkalibrovat. Zaznamenaný pohyb Actora vyžaduje ještě zásah pro identifikaci markerů a vyčištění záznamu. Osoba, která tyto činnosti provádí, bude v této práci označována jako operátor. Výsledek tohoto procesu je pohyb markerů ve scéně. Ty se dají namapovat na animační kostru, v které jsou pak prezentovány sekvencí klíčových snímků ovládajících rotaci každé kosti animační kostry. Proces by tak měl poskytnout klíčový snímek pro každý obraz vytvářeného videa, tedy 25 klíčových snímků v jedné sekundě.

Fakt, zda systém ke sledování pohybu používá markery, nebo pohyb dokáže sledovat bez nich, je základním parametrem k rozdělení typů motion capture. Podle technologie zjištění pozice markerů ve scéně ho lze dále rozdělit na magnetický, optoelektrický, akustický, mechanický a optický. Protože je ze zadání práce jasné, že bude použit optický systém s pasivními markery, ostatní metody pro trackování pohybu budou popsány velmi stručně.

Mechanický systém

První motion capture byly mechanického typu. Actor byl omezen těžkopádným oblekem ve formě exoskeletu s množstvím drátů, které mu omezovaly volnost pohybu a tak i celé činnosti, které se pomocí systému daly zaznamenat. Ze současných systémů se do těch mechanických řadí systémy využívající potenciometry a slidery. Jejich výhodou je, že je neruší okolní prostředí a nevyžaduje zdlouhavý rekalibrační proces.

Magnetický systém

Magnetický motion capture vytváří pomocí vysílačů nízkofrekvenční elektromagnetické pole. Aktivní markery jsou prezentovány třemi na sebe kolmými cívkami, které jsou pomocí indukce schopny určit svoji pozici a rotaci ve scéně. Zařízení není příliš flexibilní a může být rušeno kovovými předměty, proto lze magnetický systém provozovat jen v některých prostorách. Nevýhodou je také množství kabelů na Actorovi, které se však s rozvojem technologie daří eliminovat. [YS00]

Akustický systém

Jak popisuje [Nog11], akustický systém tvoří zvukové vysílače v podobě markerů, na záznamové straně se nachází tři snímače. Markery jsou postupně aktivovány, při tom vytváří soubor charakteristických frekvencí, které snímače systému zaznamenávají. Díky porovnání časů, v kterých byl zachycen signál ve třech snímačích, lze pozici markeru v prostoru získat. Nevýhodou tohoto přístupu je to, že díky sekvenčnímu snímání nejsou data všech markerů zaznamenána v jednom okamžiku. Dalšími negativními vlastnostmi je omezený počet markerů v systému a snížení pohyblivosti Actora způsobené dráty markerů. Ač u systému nedochází k zakrývání a rušení kovovými objekty, je tento systém velmi náchylný na časté zachycení odrazů a vnějšího ruchu, díky čemuž je volen jako poslední možnost pro motion capture.

Ostatní systémy

Optoelektrický systém měří světlost optického vlákna, které je pohybem deformováno. Lze využít v kombinaci s jiným systémem pro sledování ohybu prstů ve formě speciálních rukavic. Pro některé systémy může být totiž pohyb prstů přiliž detailním pohybem.

Inerciální systémy využívající akcelerometry a gyroskopy se dají také efektivně využít na tvorbu rukavic pro sledování pohybu prstů. Zároveň je možné technologii zkombinovat například s mechanickým motion capture pro zpřesnění měření. [Mot12]

Optický systém s pasivními markery

Nejčastěji využívaným systémem motion capture pro tvorbu animace v zábavním průmyslu je v současnosti optický systém s markery. Jeho výhodami je přenositelnost systému a variabilita, díky které lze snímat nejen postavu, ale i obličej, objekty a zvířata. Obvykle je systém tvořen kamerami s vlastním zdrojem infračerveného světla. Kamery v infračerveném spektru snímají scénu, kolem které jsou rozmístěny. Actor má na sobě umístěny pasivní markery, které osvětlení efektivně odráží, takže jsou pak nejsvětlejším bodem ve scéně a ostatní objekty je možné odfiltrovat. (Obrázek 2.3) V některých případech je třeba pro přesnější záznam scénu potemnit, aby kamery nezaznamenávaly i jiné objekty mimo markerů. Kamery svůj vysokorychlostní záznam, který je v řádu až stovek snímků za sekundu, posílají do počítače, kde probíhá zpracování dat a záznam pozice markerů.

Optický motion capture nabízí alternatinu k základním markerům nazývanou rigid body. Rigid body je takový předmět, který obsahuje několik spojených markerů. Toto pevné spojení by mělo být unikátní pro každou scénu. Pokud se tedy používá více objektů rigid body, je třeba vždy zvolit objekt s jiným rozmístěním markerů. Výhodou tohoto řešení oproti běžným markerům je nezaměnitelnost jednotlivých rigid bodies a možnost zjistit nejen pozici tohoto objektu, ale také rotaci.

Před záznamem pohybu je třeba nejdříve systém nastavit a zkalibrovat. Kamery musí snímat scénu tak, aby každé místo, na kterém se bude Actor pohybovat, zabíraly alespoň dvě kamery. Dále se vymaskují rušivé signály, které systém není schopen odlišit od markerů. To znamená, že se označí místo obrazu kamery, které systém při výpočtech nemá brát v potaz. Může se jednat o lesklé či blízké předměty nebo zdroje světla ostatních kamer.

K samotné kalibraci se používá nástroj Wand, který se skládá z držadla a několika pevně umístěných markerů. Proporce tohoto objektu jsou pro systém známé, takže když systém

Obrázek 2.3: Optický systém motion capture s actorem
Zdroj: vlastní

sleduje pohyb tohoto nástroje v prostoru scény, je po získání dostatečného množství vzorků schopen vypočítat umístění kamer vůči sobě a provést případnou korekci zkreslení jejich objektivů. K nastavení počátku scény a pozice podlahy, konkrétněji všech tří os scény, slouží druhý nástroj ve tvaru písmena L. Ten je systém schopen opět pomocí markerů rozeznat. Proto po jeho položení na podlahu scény a nastavení jeho ramen do směrů os x a z je systém schopen kamery ve virtuálním prostoru umístit, a tak dokončit celý kalibrační proces.

Actor při zaznamenávání pohybu může díky vysokorychlostnímu snímání scény provádět i rychlé pohyby. Největším nedostatkem optického motion capture je nutnost toho, aby alespoň dvě kamery každý marker ve scéně nepřetržitě viděly. Herec totiž při svém pohybu může marker zakrýt tak, že ho systém není schopný nalézt. Trackování markeru je v takové situaci ukončeno. Když se pak marker systému znovu odkryje, systém ho ve většině případů nedokáže spojit se zmizelým bodem, a tak je toto přiřazení nutné udělat ručně po nahrání pohybů. Zároveň je třeba některým typem aproximace trajektorii bodu v zakrytém čase dopočítat.

Ostatní optické systémy

Pro rozlehlé scény může být vhodnější použít aktivní markery, které samy vydávají světlo pomocí LED osvětlení a napájení z baterie. Díky tomu jsou mnohem lépe viditelné kamerami, které je tak dokáží rozeznat i na větší vzdálenost. Velký rozvoj zažívá optický motion capture bez markerů. Actor bez speciálního vybavení je z více úhlů snímán kamerami. Jejich signál pak analyzuje systém pro identifikaci lidské postavy. Když je postava rozpoznána, je rozpoznána i póza, kterou zaujímá. Z ní jsou pak odvozeny a zaznamenány rotace kloubů, které jsou pro vyjádření takovéto pózy zapotřebí. Dobrým příkladem takového systému je zařízení Kinect od firmy Microsoft, který za nízkou cenu nabídl ovládání počítačových her pomocí pohybů.

2.3.4 Další animační techniky

Je mnoho pomocných technik, které se dají k animaci využít. Snadno lze použít prostorovou křivku pro určení trasy pohybu objektu. Je možné vytvořit zapínatelné propojení mezi objekty například pro animaci chycení, přenos a puštění objektu. Animační program dále může nabízet vytvoření animační křivky s klíčovými snímky, která však není závislá na změně času, ale na jiné uživatelsky animované hodnotě. Jako poslední animační techniku, která bude uvedena v tomto stručném výčtu, by bylo vhodné zmínit animaci pomocí skriptu. Lze tak vytvořit matematickou funkci, která bude objekt ovládat. Za speciální případ takovéto animace lze označit fyzikální simulaci, kterou animační programy nazývají dynamika.

2.3.5 Dynamika

V této práci a v některých animačních programech je slovem dynamika označována simulace, která napodobuje fyzikální procesy. Dynamika řeší konkrétní úlohu pomocí zjednodušení procesů. Řešení je možné využít ve více aplikacích díky atributům, které simulaci ovlivňují.

Simulace pevných těles

Díky detekci toho, zda těleso přišlo do kontaktu s jiným tělesem, je možné simulovat nárazy a odrazy těles. Na tělesa působí jednoduché síly ve formě silového pole nebo těleso může dostat při zahájení simulace silový puls, který ho rozhýbá. Změny sil také způsobují nárazy těles mezi sebou. Tato základní simulace se vyskytuje už několik let i v počítačových hrách, které ji dokáží vyhodnotit mnohokrát do sekundy. Simulace pevných těles tedy není příliš výpočetně náročná.

Simulace látky

Zvláštní přístup si žádná simulace látky. Ta by měla vytvořit na polygonálním modelu záhyby a zvlnění tak, aby geometrie neprostupovala skrz vlastní plochu a aby byla schopna reagovat na ostatní animované objekty. Jak zmiňuje [Ber13], je mnoho přístupů a modelů, které tento problém řeší. K napodobení chování například hedvábné sukně nebo kožené tašky je možné využít jeden model pomocí změny atribut objektů. Použitý model ovlivňují atributy, které slouží k ovládání simulace a specifikace materiálů. Uměním je tedy naučit se používat vždy konkrétní model k docílení co nejreálnější simulace.

Simulace tekutin, kouře a ohně

Pro prostorovou animaci tekutin, kouře a ohně představuje částicový systém nebo fluidní dynamika jediná řešení. Vyšší počet částic systému ovlivňuje pozitivně kvalitu výsledku a samozřejmě negativně výpočetní nároky. Složitější simulace fluidní dynamiky vytváří vektorová pole a počítá s různými veličinami, nejčastěji s hustotou, a tak je pohyb částic a vlastně celý výsledný efekt mnohem uvěřitelnější. Systém je možné ovlivňovat dalšími vlivy, jako jsou silová pole, nebo objekty, do kterých mohou částice narážet. Průběh simulace ovlivňuje i místo, kde částice případně vznikají a vývoj částic v čase. Jako u simulace látky, jedná se o složitý systém s řadou aplikací. Tvorba kvalitní simulace si žádá zkušeného pracovníka a často i značný výpočetní výkon.

2.4 Autorig

Nástroj nazývaný autorig má co nejvíce zefektivnit rigovací proces a vytvořit celý rig modelu s co nejmenším uživatelským zásahem. Každý rigger se snaží do jisté míry svou práci zautomatizovat, protože tvorba rigu zahrnuje takové úkony, které se stále opakují. Velké firmy vytvářejí vlastní komplexní systém pro rigování, který se stále zdokonaluje a rozrůstá o nové funkcionality s každým novým projektem. Malé firmy či freelanceři tvoří menší moduly pouze tak složité, jak jim to situace umožňuje. Na internetu jsou jak volně stažitelné autorigy, tak ty zpoplatněné pro uživatele, kteří nejčastěji z finančních důvodů nemají možnost s riggerem spolupracovat a přímo tvorbou rigu se nechtějí příliš zabývat. Samotní riggeři cizí autorigy nepoužívají, protože i ty jednodušší autorigy jsou poměrně robustní. Tím jsou méně transparentní a vyžadují nějaký čas, aby se s nimi uživatel naučil pracovat. Každý rig je unikátní, a tak snadno může dojít k situaci, kdy bude třeba nástroj překonfigurovat nebo dokonce opravit. Ve vlastním kódu se uživatel vždy mnohem lépe orientuje a mnohem rychleji problém vyřeší. Proto je pro riggera ve výsledku výhodnější si vlastní autorig vytvořit od začátku.

Snad všechny autorigy mají společný první krok, vytvořit skeleton proxy. Tak se označuje soubor objektů, díky kterým autorig pozná, na jakém místě je třeba vytvořit důležité klouby modelu. V této fázi je tedy na uživateli, aby nastavil atributy pro skeleton proxy jako počet prstů či kloubů v páteři, nechal skeleton proxy vygenerovat a umístil všechny jeho objekty ve svém modelu na správné místo. Poté je třeba nastavit funkcionality, které má vytvářený rig nabízet a pak se už jedním tlačítkem celý rig automaticky vygeneruje. Každý autorig nabízí jiný soubor funkcionalit, kontrolery rigu jsou různě vizualizované a vlastně i provedené, takže ovládání vzniklého rigu je do jisté míry také unikátní pro každý autorig. Některé autorigy umožňují dále pracovat se vzniklým rigem a nabízejí uživateli tvorbu ovladačů obličeje, asistenci při skinningu a při vytváření korekcí skinningu, vytvoření částí animovaných dynamikou a podobně.

Součástí rigu mohou být i nástroje pro usnadnění animace. Jak popisuje [KR05] a [KR06], vedle připravení dynamiky lze pomoci animátorovi i s tvorbou klíčových snímků. V knihách byla popsána tvorba skriptu pro zrcadlení animace na druhou půlku rigu, tvorba uživatelské knihovny pro ukládání výrazů obličeje a nástroj pro ukládání a načítání celých animací do externích souborů.

Kapitola 3

Analýza

3.1 Program Autodesk Maya

Expanze klasických stolních počítačů vedla k vydání prvního animačního systému jménem TOPAS, který byl představen v roce 1986. O čtyři roky později vydala firma AutoDesk první verzi programu 3D Studio, který je dodnes aktualizován a používán. V roce 1998 byla společností Alias|Wavefront vydána první verze programu Maya pro systém UNIX, o rok později i na platformu Windows. [LG02] Od té doby vyšla každý rok nová verze Mayi, z počátku i vícekrát do roka. [tox08] To platilo i po změně vydavatele v roce 2005, kdy se firma Alias stala částí firmy Autodesk. [Wik15]

Již od své první verze je program Autodesk Maya (dále jen Maya) vyzdvihována díky otevřenému programovatelnému rozhraní, promyšlené architektuře a extrémní využitelnosti. V současnosti je stále hojně používána ve velkých firmách pro tvorbu počítačových triků, animovaných filmů s 3D objekty a pro tvorbu grafiky do počítačových her. Díky rostoucímu výkonu stolních počítačů a klesající ceně programu je Maya používaná i freelancery a v menších firmách. Program nabízí dobrou dokumentaci, existuje kolem něho velká komunita, proto není problém na internetu dohledat řešení řady problémů i mnoho výukových videí. Další výhodou tohoto programu je to, že osoby zabývající se riggingem právě Mayu využívají nejčastěji pro vysvětlení problematiky.

3.1.1 Typy objektů

Maya nabízí několik typů objektů, s kterými lze pracovat pro získání určitého tvaru. Každý z nich má své výhody a omezení, díky kterým je třeba tyto typy kombinovat. Všechny typy objektů mají nějaký druh vrcholů, jejichž pozice řídí výsledný tvar objektu. Funkce pro deformaci objektů pracují s těmito řídícími vrcholy a jejich posunem dokáží tvar objektu měnit nezávisle na jejich typu.

Polygonální objekty

Pro tvorbu polygonálních objektů je v Maye bohatá řada nástrojů, které dovolují tvořit geometrii jak na úrovni jednotlivých komponentů, tak operacemi s celými objekty. Vzhledem

k tomu, že polygonální objekt je možné použít i na vytvoření hladkých tvarů, Maya nabízí funkci pro zahlazení tvaru. Výsledek této funkce je možné od verze Mayi 2008 snadno zobrazit ve Smooth Mesh Preview v prozatímním režimu bez změny geometrie. [Mog07]

NURBS objekty

Maya používá křivky typu NURBS pro řízení různých funkcí. Některé z těchto funkcí vedou ke vzniku polygonálního objektu či ploch typu NURBS. Vzhledem k tomu, že se křivky samyi o sobě nevykreslují a přitom mohou tvořit nejrůznější tvary, používají se při riggingu jako kontrolery, které ovládají celý rig.

Plochy typu NURBS se skládají stejně jako křivky z řídících bodů. Na rozdíl od polygonálních objektů přesně popisují hladké povrchy. Jejich další výhodou je uniformní definice texturovacích souřadnic, které vycházejíí ze struktury objektu. Texturovací souřadnice mají tedy konstantní měřítko v obou osách a jsou spojité. Práce s NURBS objekty je však v mnoha případech složitější ohledně jejich tvorby, editace i řízení. Objekty NURBS je možné převést do polygonálního objektu. Ostatně jak při náhledu scény, tak před vykreslením scény jsou tyto plochy převedeny na polygonální dle řiditelného procesu zvaného teselace.

Objekty subdivision

Objekty subdivision nabízí kompromis mezi polygonálními objekty a objekty NURBS. Umožňují většinu funkcí pro modelování, jako při práci s polygonálními objekty. Jejich báze je však podobná objektu NURBS, je tedy přesná ohledně hladkých tvarů. Rozlišení je navíc lokálně řiditelné, a tak je možné mít geometrii podrobnou jen v určitých místech. Tyto objekty nejsou však často využívány pro jejich složitost a robustnost. S příchodem polygonální funkce Smooth Mesh Preview byly jejich výhody poměrně potlačeny.

3.1.2 Graf scény

Pokud jsme schopni popsat prostorový model, je třeba tento objekt umístit v prostoru a kombinovat s ostatními modely v jedné společné scéně. Software pro tvorbu 3D grafiky, Autodesk Maya, využívá pro organizaci scény stejně jako ostatní 3D programy takzvaný graf scény. *„Graf scény (scene graph) je n-árn strom, tj. takový graf, v němž lze pro každý uzel nalézt právě jednoho předchůdce. Výjimku tvoří kořen stromu, který stojí na nejvyšší úrovni. Graf scény může obsahovat i několik stromů, tzv. les. Graf scény není ve všech systémech definován stejným způsobem, odlišnosti jsou v typech uzlů, v pravidlech pro stavbu stromu i pro interpretaci dat ve stromu uložených (. . .).*

Důležitou vlastností stromu je schopnost vyjádřit vztahy mezi uzly. Jedním z těchto vztahů je dědičnost. Umožňuje v jednom místě stromu definovat vlastnost, která bude platná pro řadu dalších uzlů. Rozsah platnosti je dán vzájemnou polohou uzlů v rámci stromu. Lze například stanovit, že vlastnost definována v uzlu je platná pro všechny následníky tohoto uzlu.“ [JS10] Konkrétněji pak strukturu v Maye popisuje [Aut07], který hovoří o grafu scény jako o směrovém acyklickém grafu. Jeho zkratka DAG (Directed Acyclic Graph) je používána jak v dokumentaci, tak přímo v programu. Tento graf může obsahovat dva typy uzlů: Transform Node a Shape Node.

Transformační uzel

Transformace je operace, která může prostorový objekt posunout, otočit či mu změnit měřítko (tedy smrštit či roztáhnout) libovolně ve třech osách. V programu je tato funkcionalita implementována pomocí transformační matice 4x4. Tu uživatel na pozadí vytváří používáním nástrojů k posunu, otočení a změně měřítka. Transformace je uložena v transformačním uzlu (Transform Node) a aplikována na všechny její potomky, tedy pro všechny uzly, které se nachází v hierarchii pod ní. Vytvářením vazeb potomek-rodič transformačních uzlů může uživatel vytvořit logické uskupení objektů.

3.1.3 Souřadné systémy

Pro popsání umístění objektu ve scéně lze použít více metod podle toho, vůči čemu je pozice popisována, formálněji jde o zvolení souřadného systému. Základním prostorem je světový prostor, world space. Díky transformačnímu uzlu Maya definuje i lokální a objektový prostor, tedy local space a object space. Maya dovoluje uživateli přepínat mezi těmito prostory při používání nástrojů pro provádění transformací objektů i jednotlivých komponentů. Pro manipulaci komponentů lze ještě využít prostor, který je definován směrem jejich normál. Z něho plyne i tečný prostor, tangent space.

Světový prostor

Světový souřadný systém (World Space) popisuje pozici objektu či vrcholu absolutními hodnotami, které jsou společné pro celou scénu, a tak i pro všechny objekty. Definují počátek scény, od kterého se pozice odvíjí. Tento počátek je konstantní, co se týče jak jeho pozice, tak i směru os a velikosti měřítka. Lze ho použít pro sjednocení souřadných systémů při práci s objekty v různých hierarchiích.

Objektový prostor

Pro vyjádření souřadného systému objektu (Object Space) je třeba na tento systém aplikovat všechny transformační uzle, které samotný objekt ovlivňují. Například vymodelovaná jehla, která se zužuje do špičky ve směru osy x, při umístění ve scéně pomocí (libovolného počtu) transformačních uzlů směřuje svou špičkou v objektovém prostoru stále ve směru osy x. Člověk v tomto systému porovnává dva objekty, kdy mu nezáleží na tom, zda je jeden z objektů položen na zemi nebo opřen o stěnu. Objektový prostor je tedy možné použít pro relativní porovnání objektů a částí objektu.

Lokální prostor

Stejně jako je tomu u objektového prostoru, i lokální souřadný systém (Local Space) ovlivňují všechny transformační uzly v hierarchii objektu. Je však ignorován ten poslední, transformační uzel samotného objektu. Proto je v některých verzích Mayi lokální prostor přehledněji nazývaný rodičovský prostor (Parent Space). [Aut15c]

Počátek lokálního souřadného systému Transform Nodu se nachází při vzniku modelu ve středu scény. Poté jsou na něj použity veškeré transformace hierarchicky spojených nodů s

objektem. Ač se jedná o local space, pozice počátku je ovlivněna i transformací samotného Transform Nodu. V Maye není počátek local space příliš využíván, je použit například pro popis umístění pivotu objektu.

Pojem local space je podle [Kel10] spojován i s relativní pozicí vrcholů polygonálního objektu. (Obecněji se jedná také o kontrolní body ostatních objektových typů.) Při vzniku objektu mají všechny vrcholy koordináty v local space nastaveny na nulu. Nenulových hodnot nabývá pouze při použití transformačních nástrojů přímo na daný komponent. K vynulování těchto hodnot dochází vždy při úpravě struktury objektu pomocí kterékoliv funkce, která není typu deformer.

Tečný prostor

Tečný prostor (Tangent Space) je definován povrchem objektu. [Aut15d] popisuje tečný prostor situací, kdy je magnet přichycený ke kovové konvici. Pokud posouváme magnet po povrchu konvice, nezáleží na tom, kde je magnet umístěn. Při jistém pohledu můžeme dát magnet nahoru, tedy dál od konvice, i když se vlastně nachází na boku tohoto kovového objektu. Tento prostor je využíván například pro popis výškových map, které díky tomu nemusí zohledňovat, jaká část textury se nachází na které části objektu.

Práce se souřadnými systémy

Při provádění transformací objektů a jejich komponent se v Maye používají manipulátory nabízející uživatelské rozhraní nebo samotné příkazy programovacích jazyků. V obou případech však výsledek transformace ovlivní to, v jakém souřadném prostoru se provádějí. Někdy se hodí například posunutí provádět vzhledem k natočení objektu (příkladem může být použití objektového prostoru při pohybu lodě ve směru přídě), někdy je třeba posunout objekt ve světových souřadnicích (použití světového prostoru například při přesunutí flotily lodí blíže k ostrovu).

Provádění transformací objektu je zaznamenáno v atributech transformačního uzlu, které jsou popsány v lokálním prostoru. Při změně hierarchie objektů zůstávají objekty na svých místech, jsou proto vhodně změněny údaje v transformačních uzlech. Pokud má transformační uzel rozdílnou změnu měřítka v osách, budou se transformace jeho potomků provádět nerovnoměrně. Nejvíce je tento jev vidět při rotaci, kdy dochází k dynamickému roztahování a smršťování objektu podle toho, jak je přeškolovaný i lokální prostor. Abychom se vyhnuli tomuto jevu je tedy vhodné při vytváření hierarchie používat transformační uzly s vyresetovanou změnou měřítka. Výhodou může být zachování nenulové rotace rodičovského uzlu tak, aby bylo možné přesouvat objekt v lokálním prostoru vhodným směrem pomocí posunu pouze v jedné ose transformačního uzlu. To samé platí i pro posun rodičovského uzlu, který přesune počátek lokálního prostoru potomka.

Funkce Freeze Transformation vyresetuje základní atributy transformačního uzlu při zachování pozic všech jeho potomků a pivotů. Ovlivněny jsou tedy přímo jejich komponenty a pivoty v lokálních souřadnicích. Tento úkon je realizován přesunutím počátku lokálního prostoru transformačního uzlu do počátku scény nebo na místo, kam ho z počátku přesunují rodičovské transformační uzly.

Pokud se jedná o transformaci typu rotace či změny měřítka, výsledek operace záleží především na umístění pivotu objektu. Pivot pro rotaci definuje střed otáčení, pivot pro změnu měřítka určuje střed expanze či komprese, tedy bod, který při této operaci zůstane jako jediný na původním místě. Pozice pivotu pro rotaci i pro změnu meřítka je definována v transformačním uzlu, kdy lze jeho umístění uživatelsky libovolně měnit. Všechny prováděné transformace se zaznamenávají do transformačního uzle. Je však dobré si uvědomit, že při přesunutí pivotu mimo počátek lokálního prostoru se ovlivní i fungování všech transformací. Aby mohlo být vyhověno transformaci pro pivot mimo počátek lokálního prostoru, Maya při rotaci či změně měřítka musí provádět i posunutí, které odpovídá vzdálenosti od zmíněného počátku. Tento posun však není zachycen v transformačním uzlu, je tak ovlivněna pozice ve světových souřadnicích všech vrcholů objektu, které jsou potomky upravovaného transformačního uzlu. Důsledek se dá snadno ukázat na příkladu, kdy je vytvořen objekt uprostřed scény. Pokud je jeho pivot posunut a je provedena libovolná rotace a následně je pivot přesunut zpět do počátku lokálního prostoru, objekt se po vynulování rotace nachází na jiném místě. V transformačním uzlu jsou atributy popisující transformaci shodné jako při jeho vytvoření, objekt je však na místě jiném. Záznam o této operaci není nikde číselně vyjádřen, oproti počátečnímu stavu je změněna pouze pozice komponentů a pivotu ve světových souřadnicích.

Při transformaci komponentů lze využít prostory stejné, jako nabízí daný objekt. Pivot komponentů však není na objekt navázán, ve výchozím nastavení je umístěn do průměru jejich pozice, je možné ho však volně přesouvat bez jakéhokoliv ovlivnění transformačního uzlu.

Graf závislostí

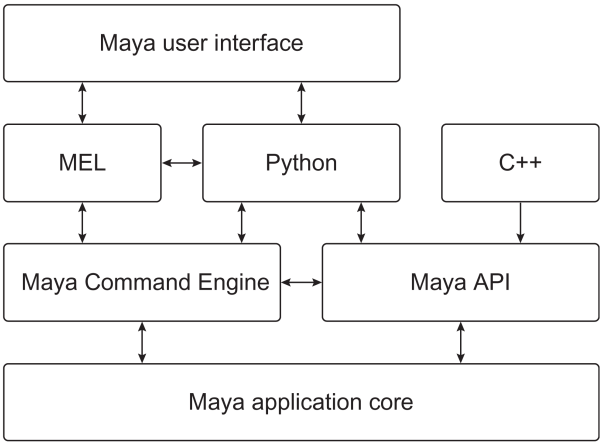
Vedle struktury DAG Maya definuje graf závislostí. Jeho zkratka je DG (Dependency Graph) a DAG je vlastně jeho podmnožinou. Graf závislostí může obsahovat cykly, hrany mezi jeho uzly popisují výměnu dat a jejich směr. Přenášená data hranami mohou být jak jednoduchá čísla, tak komplikovaná data jako je například povrch objektu. Uzly DG jsou v Maye použity téměř na vše. Je jimi popsáno například vytvoření modelu, deformace, animace, simulace a zpracování zvuku. Jedná se o složitou strukturu, která dokáže efektivně vyhodnocovat změny hodnot jen tehdy, pokud je to nezbytné. [Aut09]

3.1.4 Možnost vstupu

Vedle uživatelského rozhraní, Autodesk Maya nabízí ke komunikaci dva programovací jazyky: MEL a Python. Navíc poskytuje možnost programování zásuvných modulů v jazyku C++. Jak ukazuje obrázek 3.1, každá akce provedená v Maye je vykonána nějakou kombinací těchto interfaců.

MEL

Uživatelské rozhraní programu Maya je od první verze sestaveno z instrukcí a příkazů MEL (Maya Embedded Language) skriptu. Tento programovací jazyk vznikl výhradně pro použití v Maye. Jedná se o interpretovaný skriptovací jazyk, proto ho lze spustit přímo v textovém



Obrázek 3.1: Komunikace v Maye
Zdroj:[MT12]

vstupu Mayi. Z toho plyne i jeho nevýhoda, že může komunikovat s Mayou pouze pomocí definovaného rozhraní Command Engine (popřípadě interním použitím Pythonu). Dalším negativním aspektem jazyka MEL je, že nepodporuje objektově orientované programování.

Python

Python se v programu Maya vyskytuje od verze 8.5. Může spouštět stejné funkce jako MEL použitím rozhraní Command Engine. Lze říci, že Python je jako MEL skript interpretovaný jazyk. Jak poukazuje [Bur10], Python je spíše hybridem mezi interpretovaným a kompilovaným jazykem, protože při načtení modulu jsou jeho příkazy zkompilovány do bytekodů a ty jsou pak spouštěny. Python na rozdíl od MEL skriptu podporuje objektově orientované programování a jmenné prostory. Navíc je používaný od roku 1980, proto disponuje rozsáhlou knihovnou, vestavěnými funkcemi i širokou komunitou uživatelů, kteří přímo s programem Maya nemusí být nijak propojeni.

Balík funkcí PyMEL

Implementované příkazy Command Engine do jazyka Python jsou ekvivalenty příkazů programu MEL včetně všech jejich parametrů a návratových hodnot. To může být výhodou pro programátory, kteří znají MEL a chtějí se naučit Python pro vytváření Mayových skriptů. Pro zkušenější programátory však tento procedurální přístup může být na obtíž. Balík funkcí PyMEL vytvořil k podpoře objektově orientovaného programování v Pythonu vývojář Chad Dombrova. Tento balík byl později zařazen do základní instalace Mayi, objevuje se v ní od verze Maya 2011. PyMEL implementuje většinu funkcí Command Engine, které používá Python. Návratové hodnoty těchto funkcí už nejsou typu string, ale typu PyNode. Tento programový objekt zapouzdřuje funkce daného uzlu. PyMEL objektově komunikuje přímo s Maya API, a tak dovoluje integrovat ještě více funkcí než základní dva zmíněné přístupy. Díky

zaobalení API Mayi PyMEL v některých případech funguje rychleji a unikátní identifikátory uzlů dovolují snadnější odkazování na uzly Mayi.[MT12]

Programovatelné rozhraní jazykem C++

Programovatelné rozhraní (API) Mayi je nejflexibilnější cestou, jak do tohoto animačního programu přidat nejrůznější funkce. Uživatel tak může vytvořit příkazy, které se provedou mnohem rychleji, než při spuštění například kódu MEL skriptu. API navíc nabízí přístup k mnohem více funkcím. C++ je kompilovaný jazyk, proto každý vytvořený zásuvný modul musí být zkompilován pro každou platformu i verzi Mayi speciálně. Z toho plyne nepraktický průběh testování, kdy se každá změna kódu musí zkompilovat a do Mayi znovu načíst. Proto je řešením, pokud to je možné, využít pomalejší MEL skript pro otestování postupu a až poté ho implementovat jako zásuvný modul jazyka C++. Je třeba dodat, že zvládnout jazyk C++ je pro nezkušené programátory nejnáročnější z možností. Ti zkušení programátoři budou mít zase problém se v komplikovaném API zorientovat.

Programovatelné rozhraní jazykem Python

Při zakomponování Pythonu do Mayi byla do tohoto jazyku namapována řada tříd z API C++, díky kterým se množství funkcionalit tohoto API přesunula i do jazyku Python, kdy navíc počet takto zpřístupněných objektů každou verzí Mayi roste. Manipulovat s API programu Maya je možné jak z běžného skriptu, tak prostřednictvím zásuvných modulů.[MT12]

Propojení jazyků

Velkou výhodou, kterou Maya poskytuje, je propojení jazyků. Uživatelské rozhraní spouští MEL skripty či skripty jazyka Python a ty dokáží využívat zásuvné moduly. Zároveň MEL skript i skripty jazyka Python se dokáží navzájem spouštět i si předávat návratové hodnoty. Proto je možné zdrojové kódy nejrůzněji kombinovat nezávisle na tom, v jakém interface vznikly.

3.1.5 Spouštění skriptů

Uživatelské rozhraní

Celé uživatelské rozhraní Mayi, zkráceně GUI (Graphic User Interface), pracuje tak, že spouští příkazy jazyka MEL. Ty se vypisují do okna Script Editor přímo při práci s GUI. V MELu jsou vytvořeny i některé části samotného uživatelského rozhraní. Ty ostatní části lze měnit a vytvářet pomocí Maya API. Programátor má tak možnost vlastní skripty přímo do uživatelského rozhraní zakomponovat a vytvářet jeho nové části. Tím nejsnadnějším způsobem, jak skript do stávajícího GUI přidat, je uložení skriptu do Shelf menu, do ikonové nabídky programu.

Spuštění kódu

Jednořádkový příkaz jazyka MEL či Python je možné spustit přímo v okně Mayi v části zvané Command Line. Okno Script Editor nabízí textový editor pro víceřádkové kódy v jazyce MEL či Python. Editor nabízí barevnou odezvu na rozpoznané výrazy. Po dopsání kódu ho lze spustit a ihned pozorovat výsledek. V případě chyby Maya nabízí zpětnou vazbu v podobě chybových hlášek.

Globální funkce či objekty jsou v Maye uloženy po dobu běhu programu. Znovu je načíst lze i z externích souborů pomocí zadání buď absolutní cesty, či relativní dle seznamu složek, v kterých Maya prohledává. Příkaz v Shelf Menu pak obvykle obsahuje načtení daného skriptu a spuštění jeho funkce. Expression, Script Job a Script Node umožňují spouštění kódu jinak než přímým spuštěním kódu či interakcí s GUI.

Expression

Speciální kód nazývaný expression dovoluje spouštět uživatelský kód nepřímo. V momentě, kdy je expression vytvořen či upraven, je zkompilován a jsou vyhodnoceny chyby v syntaxi. Pokud chyby neobsahuje, je ihned spuštěn. Poté je proveden vždy, když se změní současný snímek animace nebo je spuštěn při přehrávání vždy při vykreslení nového snímku. Pomocí expression je možné upravovat i hodnoty částicového systému, kdy je kód spuštěn vždy pro každou částici exkluzivně. Expression se ukládají a načítají se scénou.

Kód expression podporuje MEL skript, kód vytvořený v Pyhonu zde není podporovaný. Je však možné ho spustit přes příkaz v jazyku MEL. Při používání atributů objektů nabízí expression syntaxe ještě přímý přístup typický pouze pro něj. Přímý přístup může být realizován buď ve formě označení uzlu a atributu pomocí tečkové notace. Expression je možné přiřadit konkrétnímu objektu, v takové situaci je možné přistupovat k atributům tohoto objektu i bez zmínění jeho jména. Při použití přímého přístupu atributů je Maya schopna takový kód vykonat i v momentě, kdy jsou tyto hodnoty jakýmkoliv způsobem změněny.[Aut15a]

Příkazy MEL spouštěny pomocí expression nejsou podporovány při dávkovém vykreslování. Proto je třeba před finálním vykreslením atributy, které ovlivňuje, převést do klíčových snímků pomocí funkce Bake Simulation. Hodnoty upravené přímým přístupem dávkové vykreslení zohledňuje.

Script Job

Pro vytvoření skriptu Script Job je třeba použít jazyk MEL či jazyka Python, kdy je při jeho vzniku zadána funkce nebo řetězec, který obsahuje skript používaného jazyka. Skript Job je připojen k nějaké události, podmínce či atributu, u kterých je sledována změna stavu. Pokud se stav změní, zadaná funkce či skript je proveden. Skript Job vzniká v rámci programu Maya, pokud není řečeno jinak, není vázán na scénu, v které vznikl. Jeho správa je prováděna pomocí unikátního id, které má každý Script Job přidělen.

Script Node

Script Node nabízí propojení skriptu se souborem scény Mayi s možností nastavit, jak a kdy bude skript spuštěn. Konkrétně skript může spustit: otevření scény, zavření scény, otevření

či zavření scény v GUI módu (druhým módem je myšleno dávkové vykreslování), před či po započetí vykreslování celého úkolu nebo každého snímku, při změně času a při dotázání. Při dotázání není automaticky spuštěn nikdy. Je však možné stejně jako ostatní typy spuštění zadat příkaz pro vykonání kódu daného objektu typu Skript Node. Z možností spuštění příkazu pro vytvoření Script Node ještě nabízí další interní spouštěcí události, které jsou pro uživatelské použití nepřístupné (Maya je bez varování automaticky nastaví na jinou spouštěcí událost). Skript je možné vytvořit jak v MEL skriptu, tak v Pythonu.[Aut10a]

3.1.6 Vzájemné ovlivňování uzlů

Předávání dat mezi uzly Mayi je klíčové pro její funkčnost. Maya u každého uzlu nabízí možnost vytvářet uživatelsky nastavitelné atributy typu Vector, Integer, String, Float, Boolean nebo Enum. Díky tomu lze vytvořit první článek řetězce spojení, který bude díky propojením specificky ovládat další uzly scény.

Vyvarování se skriptů

Pomocí skriptů je možné zpřístupnit a změnit hodnotu kteréhokoli atributu za použití nejrůznějších spouštěcích mechanismů. Častěji, než použitím skriptů, se uzly ovlivňují pomocí DAG a propojení atributů. O hierarchii uzlů a jejich ovlivňování pojednává část 3.1.2, následující text se zaměří na vytváření propojení mezi atributy uzlů. To je pro Mayu přirozenější a rychleji prováděné než ovlivňování pomocí skriptů. Maya také může lépe rozpoznat vazby mezi uzly (a tak i změny lépe zaznamenávat). Má také možnost s vytvořeným propojením dál pracovat. Proto, pokud je to možné, je propojení preferováno. Možností na pomezí obou přístupů lze označit ovlivňování atributu pomocí expression s přímým přístupem, které určitý typ propojení vytvoří.

Spojení

Pokud jsou atributy stejného datového typu, je možné mezi nimi vytvořit přímé spojení. Hodnota jedné atributy je pak vždy shodná s hodnotou druhou, dokud spojení není zrušeno. Spojení má vždy definován směr toku dat. Vzniká tak řídící atribut, který je možné libovolně měnit jakýmkoliv způsobem, a řízený atribut, který není možné měnit, dokud toto spojení trvá. Maya nabízí i zjednodušený přístup ke skupinám atributů, kdy je možné vedle základního spojení například translací v ose y vytvořit spojení mezi všemi translacemi pomocí jedné vazby. Pokud nestačí data překopírovat, ale je třeba s nimi vykonat ještě nějakou operaci, Maya nabízí pro práci s daty různé druhy DG uzlů. Lze tak provádět základní aritmetické operace jak mezi uzly, tak se zvolenými konstantami. Pomocí těchto uzlů je také možné vytvářet pozvolné přechody hodnot, podmínky a podobně.

Všechny další funkce, které atributy ovlivňují (jde především o ty určené pro animaci), pracují pomocí kombinace speciálních DG uzlů a jejich propojení. Dobrým příkladem je vytvoření klíčového snímku. Při jeho prvním výskytu je animovaný atribut propojen s nově vzniklým uzlem pro animaci pomocí animační křivky. Okna Mayi zobrazující hodnoty atributů, konkrétně Channel Box i Attribute Editor, barevně vyjadřují, jaký druh spojení daný atribut ovlivňuje.

Aby jeden atribut mohl ovlivňovat více vstupů, je třeba je speciálním uzlem spojit. Uzel pro míchání je pro tento účel často nejvhodnější, Maya ho v řadě případů při kolizi spojení vytváří automaticky. Příchozí spojení vchází do uzlu pro míchání spolu s novým spojením, následně je s nimi provedena zvolená operace. Na výstupu je pak už upravená jedna hodnota, která lze danému atributu bez problému připojit.

Spoje mezi DG uzly mohou tvořit smyčky. Není však podporováno zacyklení propojení jednotlivých atributů. Nesmí se tedy stát, že atribut, který je spojen s jiným atributem, jím bude přes libovolný počet uzlů ovlivněn. Do tohoto omezení je třeba zahrnout i vazby, které definuje DAG.

Omezení

Omezení jsou speciální propojení dvou a více uzlů, v jiných programech je možné se se shodnou funkcí setkat pod označením animační ovladače. Umožňují vytvořit závislosti mezi jejich transformacemi, zavést speciální limity objektů a automatizovat animační proces. [Aut10b] Ze vztahu tak vzniká označení obou objektů, objekt cílový (target object) a objekt omezený (constrained object). Omezení lze využít pro definování chování závislých objektů na animovaném objektu. Díky němu lze například sladit rotace či pozice jednoho objektu s druhým, je možné vytvořit vztah potomek-rodič. Díky možnosti omezení kombinovat a vypínat tak lze dosáhnout animace, kdy si dvě postavy předají objekt z ruky do ruky. Omezení může být praktické i ve velmi odlišné situaci, kdy je třeba rozpohybovat mechanické součástky, které jsou složeny z pístů, hřídelí a podobně.

Každé z omezení je realizováno pomocí speciálního uzlu, které z cílového objektu získá pomocí více propojení důležité informace. Na výstupu pak připraví atributy pro omezený objekt, které již propojí napřímo. Omezení pracuje buď s nejbližším místem na objektu, se směrem cílového vektoru nebo s pozicí jednoho či více objektů. Pozice v prostoru musí být vyjádřena bodem v prostoru, tak zvaný cílový bod. Ten je určen pozicí pivotu pro rotaci cílového objektu ve světovém souřadném systému. Pokud omezení používá více cílových objektů, cílový bod je na pozici váženého průměru jednotlivých pivotů. Uzel omezení definuje atribut pro vyjádření váhy každého řídícího objektu. Tato váha je datového typu float, může nabývat všech nezáporných hodnot a lze ji animovat. Používá se pro výpočet váženého průměru, při nastavení všech vah na nulovou hodnotu je omezení vypnuto. Toto vypnutí se děje skokově, po vypnutí se objekt přesune na lokální pozici, v které byl před vznikem omezení.

Některá omezení pracují s parametrem offset, který určuje, jaké hodnoty se ke standardnímu chování omezení mají přičítat. Tuto volně nastavitelnou hodnotu lze při vytvoření omezení automaticky vyplnit tak, aby se po vytvoření omezení současná transformace zachovala a použila jako výchozí. Alternativou je nechat omezení stávající hodnoty omezeného objektu přepsat. Toto přepsání lze využít i pro umístění objektu, kdy je omezení vytvořeno a ihned zase smazáno. Díky nulové hodnotě offset je při tomto úkonu objekt transformován do žádoucí pozice.

Ač je většina omezení realizována pomocí atribut transformačního uzlu omezeného objektu, jeho chování je realizováno ve světovém souřadném systému. Jinými slovy ovlivněné atributy ignorují umístění uzlu v hierarchii DAG. Při změně transformací rodičovských uzlů mění omezení hodnoty tak, aby byly transformace vykrácené. To se děje pomocí Parent In-

verse Matrix, která se posílá z omezeného objektu do omezení. Po smazání tohoto propojení se omezení začnou provádět pouze v lokálním prostoru.

Animační křivka

Animace pomocí klíčových snímků je realizována pomocí animační křivky. Uzel, který o animační křivce uchovává všechny údaje, pak pomocí spojení ovlivňuje daný atribut. Standardně se klíčové snímky vytváří ve spojitosti s pozicí na časové ose, kdy je daná hodnota přiřazena současnému snímku. Animační křivka však nemusí pracovat pouze s časem, k tomu účelu Maya nabízí vytvoření Driven Key.

Driven Key dovoluje vytvořit závislost mezi dvěma atributy, které budou propojeny pomocí animační křivky. Vznikne tak vztah dvou objektů, kdy řídící (driver) objekt ovlivňuje svými hodnotami řízený (driven) objekt. Ve všech ohledech se však s animační křivkou pracuje stejně jako s tou, která má na vodorovné ose časový údaj.

3.1.7 Základní deformace v programu Maya

Jak některé pohyby anorganických objektů, tak velká většina pohybů rostliny i zvířat se skládá z deformací materiálu, ze kterého jsou tvořeny. V počítačové grafice je proto velmi důležité to, jak tuto deformaci provedeme. Program Maya nabízí řadu druhů deformací. Ty je možné plynule zapínat a měnit jejich fungování, proto jsou vhodné pro animaci, dají se však použít i pro docílení základního tvaru objektu.

Deformátory je možné kombinovat pomocí řetězení vstupních operací v historii. Jak jsou deformace provedeny, určuje jejich pořadí. To Maya dovoluje upravovat i po vytvoření deformací pomocí seznamu vstupních operací (List of input operation). Deformátory umožňují s kostrou propojit objekty, které pracují s řídícími body. Jedná se o polygonální objekty, objekty typu NURBS i subdivision a o deformátor typu lattice.

Smooth skin

O kostře a jejím propojení s prostorovým objektem hovoří kapitola 2.2.1. Smooth skin umožňuje jeden řídící bod ovládat více kostmi, čímž umožňuje vytvořit plynulý přechod v místě ohybů. Váhu, která určuje vliv kosti na vrchol, je možné zadat buď přímo hodnotou, nebo pomocí nástroje štětec. Ten dovoluje kreslit přímo na objekt tahem myši či tabletu, a tím přidávat, ubírat či rozmělňovat váhy kostí v rámci více vrcholů najednou. Při označení vrcholů objektu namísto objektu celého lze fungování štětce omezit pro snížení chybovosti nepřesným kreslením. Štětec lze použít i plošně na celý objekt či na všechny označené vrcholy. Vhodné je při kreslení vah s kostí hýbat nebo kostru přímo naanimovat pro lepší odhalení nepřesně vytvořených vah a pro odladění funkčnosti. Váhy je možné nanášet i při jiných pozicích kostry, než v které byl smooth skin vytvořen. [KR05] ještě radí pro manipulaci vah všech kostí (až na poslední klouby hiearchie) pracovat tak, aby štětec vždy váhu pouze přidával, nikdy ji neubíral. V takovém případě totiž Maya sama musí rozhodnout, jak výsledné váhy přerozdělí. K tomu však dochází pouze tehdy, když je u smooth skin zapnuté interaktivní normalizování vah.

Váhy smooth skin je možné nejrůzněji kopírovat, exportovat a importovat ze souborů, s klouby a váhami provádět nejrůznější funkce. Maya podporuje možnost některé části kostry (nejčastěji ty koncové) vyloučit z fungování smooth skin tak, že v seznamu ovládaných kostí nebudou vůbec zahrnuty. Naopak je také možné do smooth skin přiřadit více hierarchií koster, v extrémním případě lze propojit s objektem jednotlivé, nespojené klouby. Maya navíc umožňuje připojit kterýkoliv DAG uzel, deformaci tedy může řídit například transformační uzel křivky nebo polygonálního objektu. Smooth skin má ve výchozím nastavení vypnut atribut Use Components. Po jeho zapnutí řídí deformaci i jednotlivé komponenty přiřazených objektů. Tato funkce je nejčastěji využita pro rozšíření funkce smooth skin svalovým systémem.

Pomocí změny měřítka kloubu lze měnit velikost pouze části objektu. Ovlivněné vrcholy tak mohou manipulovat s objemem objektu. Změnou měřítka nebo i posunem kloubu lze natahovat a zkracovat části kostry při stylizované animaci. Posun kloubu je také využitelný například při animaci pulzování více neforemných struktur.

Smooth skin je realizován pomocí uzlu skin cluster. Výsledný tvar objektu předává útvarovému uzlu pomocí vstupu inMesh. To znamená, že definuje celou jeho strukturu nezávisle na tom, jaká data byla v útvarovém uzlu původně. Celý Skin cluster pomocí spojení s klouby získává jejich současnou pozici. Protože má v atributu bindPreMatrix zaznamenanou pozici kostí při vytvoření Smooth Skin, dokáže porovnáním obou matic vypočítat, jak se má daná deformace provést.

Blend Shape

Tato funkce dokáže plynule interpolovat pozici vrcholů deformovaného objektu podle pozic vrcholů cílových objektů. Vrcholy funkce Blend Shape porovnává dle jejich indexu a pracuje s jejich lokální či světovou pozicí v prostoru. Aby funkce pracovala správně, nesmí být přidán či odebrán jediný komponent deformovaného objektu či některého cílového objektu. Při kombinaci s jinými deformátory by měl být blend shape vykonáván jako první. Blend shape se hojně používá na tvorbu mimiky, kdy je vytvořena sada výrazů, které se pak na hlavní objekt namapují. Další využití je při vytváření korekcí, tedy objektů, které upravují chování smooth skin v kritických místech.

Soft Modificator

Soft modifikátor je druh deformátoru, který umožňuje provádět transformace určitých vrcholů objektu. Transformace je určena transformacemi objektu handle tohoto deformátoru v prostoru lokálně globálním, tedy obě pozice mají vliv na výslednou deformaci. Neovlivňuje ji však pozice deformovaného objektu. U funkce soft modificator je možné nastavit centrum působení deformací, dosah deformace a pomocí křivky i průběh změny vlivu s rostoucí vzdáleností od centra deformace.

Lattice

Už vícekrát byl v práci zmíněn deformátor typu Lattice. Ten vytvoří kolem deformovaného objektu obálku, která je dle nastavení rozdělena ve třech osách. Na průsečících jednotlivých

řezů jsou vytvořeny řídící body, jejichž posunem se objekt deformuje tak, aby byl Lattice i při změně svého tvaru stále obálkou. Posun řídících bodů je sledovaný ve světových souřadnicích.

Ostatní deformátory

Deformátor Wrap dovoluje vytvořit vazbu mezi cílovým a deformovaným objektem na bázi vrcholů. Deformovaný objekt je ovlivněn pozicí vrcholů cílového objektu ve světovém souřadném systému. V momentě vzniku se tedy vrcholy obou objektů musí co nejpřesněji překrývat. Deformátor cluster dovoluje hýbat vrcholy objektů a animovat je. Jejich pozice závisejí na světových souřadnicích objektu Cluster Handle, který transformace realizuje. Poslední skupinou deformátorů, které budou v této práci zmíněny, jsou nelineární deformátory. Ty používají specifickou funkci, kterou řídí několik atributů a pozice objektu handle vůči deformovanému objektu ve světových souřadnicích. Daná deformace ovlivňuje jednotlivé vrcholy deformovaného objektu a vytváří tak z objektů nejrůznější tvary.

3.1.8 Pokročilé deformace v programu Maya

Všechny zmíněné deformátory pracují s transformacemi jednotlivých komponent bez přidané logiky. Nedá se tedy předpokládat, že by byly dostačující pro vytvoření komplexní simulace realistických deformací. Lze je spíše vnímat jako základní kameny, z kterých se taková simulace dá vytvořit. [KR05] a [KR06] uvádí dva přístupy k řešení tohoto problému, které se nazývají korekce (correctives) a svalový systém.

Korekce

Korekce jsou sada blend shape objektů, které se na deformovaný objekt aplikují automaticky v momentě, kdy je třeba dále upravit deformovaný tvar vzniklý pomocí Smooth Skin. Aby Blend Shape deformace fungovala správně, korekce musí být vytvořeny na nedeformovaném objektu. Vznik takové korekce je třeba buď provádět na nedeformovaném duplikátu objektu, což je velmi nepřesné a nepraktické, nebo využít skript, který deformace provedené funkcí Smooth Skin zase odečte.

Druhým problémem je nastavení toho, kdy se daný Blend Shape má spustit. U korekcí kloubů, které se ohýbají pouze v jedné ose, je řešením jednoduše sledovat danou osu a dle té například pomocí Driven Key spouštět daný Blend Shape. Spouštění korekcí, které řídí rotace kloubů s více stupni volnosti, zejména klouby kyčle a ramena, je mnohem složitější problém. Kombinací tří atributů pro rotaci je totiž možné získat stejný výsledek vyjádřený různými hodnotami. Proto je třeba spouštěcí mechanismus připojit k nějakému pomocnému systému, který dokáže jednoznačně určit směr kosti i její rotaci.

Svalový systém

Objekt, který co nejvěrněji napodobuje chování reálného svalu, je základem svalového systému. U takovéhoto svalu musí být možné ovládat pozici počátku a konce, tedy jeho úpony. Při jejich manipulaci se sval bude smršťovat a natahovat, na což by měl reagovat i jeho tvar takovým způsobem, aby byl vždy zachován jeho objem. Pokud jsou tyto podmínky splněny, je třeba sval správně umístit vzhledem k modelu a připojit k animační kostře. Správná

funkčnost pak spočívá v důkladně vytvořených vahách funkce smooth skin. K té lze totiž svaly připojit jako ovládací objekt a pokud má smooth skin pozitivně nastaven atribut Use Components, budou ovládací body svalu zohledněny při deformaci.

[KR06] popisuje tvorbu svalu pomocí tří kruhových křivek pro určení počátku, středu a konce svalu. Křivky pak tvoří povrch svalu typu NURBS pomocí funkce Loft. Křivky na krajích objektu pak řídí upnutí svalu a prostřední křivka je automaticky ovládána pro určení objemu objektu. Ještě realističtějšího výsledku je možné dosáhnout funkcí Soft a Spring Body, které přidají dynamické vlastnosti svalu tak, aby reagoval na rychlé pohyby a otřesy.

3.1.9 Objekty vhodné pro rigging

Maya nabízí tak velké množství funkcí, že obsáhnout je všechny v rámci jedné práce, není prakticky možné. Nicméně by bylo vhodné zmínit ještě průřezově další objekty a funkce, které jsou užitečné při riggingu.

Lokátory

Pro základní označení bodu v prostoru je v Maye připraven objekt typu lokátor. V náhledu je vykreslován jako kříž. Jedná se tedy o transformační uzel a velmi jednoduchý útvarový uzel, který se ani nevykresluje. Používá se pro řízení dalších funkcí, které potřebují definovat bod v prostoru.

Follicles

Follicles jsou prostorové objekty, které svou pozici však mohou mít spojenou s nějakým objektem, který tvoří plochu. Jejich umístění je v takovém případě definováno pomocí texturovacích souřadnic, proto lépe fungují s objekty NURBS než s těmi polygonálními, které často nedefinují texturovací souřadnice uniformě. Díky objektům Follicle je možné sledovat pozici i na deformovaném povrchu a připnout tak k určitému místu objektu jiný objekt. Maya pomocí objektů Follicle řeší především upnutí vlasů, chlupů a vousů (od toho vznikl jejich název folikul), lze je však využít jinými způsoby. Pozici objektu Follicle je možné animovat nebo řídit pomocí omezení pro určení nejbližšího bodu povrchu od bodu v prostoru. Pokud upnutý není, může se chovat jako běžný prostorový objekt.

Dynamické křivky

Funkce, která je stejně jako objekt Follicle spojená originálně se simulací vlasů, umožňuje automaticky animovat křivku. Takovou křivku je možné vytvořit automaticky nebo použít již vytvořenou křivku a podle té vytvořit křivku dynamickou. Původní křivka může být zachována celá, nebo může zůstat pouze její transformační uzel. Tento uzel nenáležící dynamické křivce je v obou možnostech používán k provádění posunů, na které reaguje simulace. Při jeho animaci je pak křivka dynamicky ovládána.

Tvar dynamické křivky řídí follicle, který je s touto křivkou automaticky vytvořen. Je nastaven jako rodič transformačního uzle, který je používán pro posun křivky. Díky tomu samotný pohyb follicle ovlivňuje její dynamickou simulaci. Vzhledem ke směru křivky jsou

definovány konce: základna (base) a vršek (tip). Follicle dovoluje zvolit, ke kterému z konců bude křivka upnuta.

Simulace je řízena pomocí uzlu Hair Systém Shape, který může definovat dynamické vlastnosti jak jedné křivce, tak celé skupině křivek. Dovoluje nastavit atributy řídící simulaci a kolize s objekty scény i mezi křivkami navzájem.

3.2 Animační rig

3.2.1 Analýza animačních zásad

Od rozvoje animovaného filmu se zdokonalovaly animační techniky, které jsou v různých formách často používány dodnes, u filmů s 3D modely nevyjímaje. Jako základní poučka pro animaci bylo v [TJ81] představeno dvanácti animačních zásad, ty jsou popsány i v [LR12] vzhledem k prostorové počítačové grafice.

Natažení a smrštění

Objekty se pro vytvoření dojmu flexibility a života deformují, zachovávají si při tom však svůj objem. Může se jednat o zploštění míče při nárazu, u postavy například o protažení ruky při zátěži. Technik takové deformace počítačového modelu je více, vyžadují však speciální nastavení a kontrolery jako nadstavbu nad základním ovládáním.

Předjímání

Postava vytváří pohyb, který připraví postavu i diváka k následující akci. V základním pojetí jde o pohyb opačným směrem, než bude vykonáván pohyb hlavní. Vytváří pocit plánování akce. Zde je třeba, aby animátor vhodně pracoval s animační křivkou, která mu takovýto druh pohybu snadno umožní.

Inscenace

Vytvoření vhodného pohledu a nasvícení scény tak, aby byl pohyb zachycen co nejpochopitelněji, samozřejmě vyžaduje speciální pozornost. Je třeba pohyb naplánovat tak, aby vyhovoval jak prostředí, tak veškeré akci postav a objektů ve scéně. Také se musí brát v úvahu to, čemu v kterou chvíli divák věnuje pozornost. V tomto ohledu má animátor 3D modelů oproti tvůrci kreslené animace značnou výhodu, protože snadno může měnit pozici kamery v prostoru, a tak různě experimentovat či později své rozhodnutí změnit.

Přímo v před & Od pózy k póze

Existují dva základní způsoby, jak vytvářet pohyby. Animátor může vyjít z prvního snímku, posunout se o jeden či více snímků a vytvořit novou pózu. Tento způsob je v literatuře pojmenován Přímo v před (Straight Ahead) a je vhodný pro mechanické a dynamické pohyby, které navazují vždy na poslední pózu a špatně se dají předvídat, jak v pokročilém čase mají vypadat. Od pózy k póze (Pose to Pose) zase popisuje animaci jako vytváření klíčových

pozic po celé délce animace. Až poté se animátor zabývá vzniklými mezerami, které postupně vyplňuje mezipohyby. Výhodou této techniky je snadné experimentování s rychlostí pohybů a snadnější plánování.

Překrývající se akce & Pokračování pohybu

Uvedení objektu do pohybu vyžaduje energii, kterou je pak opět třeba utlumit po dokončení pohybu. U volně spojených částí objektu tak při změně pohybu dochází k překrývání akce. Jedná se například o pohyb antény po zabrzdění auta či rotaci zápěstí po rychlém zastavení mávající ruky. Tato technika se nazývá překrývající se akce (Overlapping Action) a u řady počítačových pohybů, především těch mechanických, lze simulovat automaticky pomocí simulace základních fyzikálních pochodů.

Pokračování pohybu (Follow-through) hovoří o hmotnosti, přesněji hybnosti objektů. Čím těžší objekt je, tím více síly je třeba k jeho zastavení. Například odpálení míčku holí tak nekončí hned po nárazu hole do objektu, ale pokračuje dále. Na hráče působí hybnost rukou i hole, což vyžaduje energii a čas, aby pohyb ustal.

Zrychlení & Zpomalení

Řada akcí potřebuje čas k provedení změny pohybu. Po tento čas se plynule přechází od jednoho pohybu k druhému, k zastavení (Slow In či Ease In) nebo ke zrychlení (Slow Out či Ease Out). Pohyb řízený křivkou pozvolnou změnu nabízí a je možné ji ovlivňovat délkou a směrem tečen v klíčových snímcích.

Oblouky

Většina živých organismů díky omezení své kostry tvoří pohyby po dráze tvořené z oblouků, oblý tvar by měla například trajektorie pevně zvoleného bodu zápěstí při pohybu ruky. U počítačové animace se může vyskytnout problém v případě, že je využita inverzní kinematika, kdy animátor ovládá končetinu pomocí posouvání kontroleru v prostoru. Kontroler se při interpolaci bude lineárně posouvat a vznikne tak nechtěná kolize s tímto pravidlem.

Vedlejší činnost

Animace hlavní činnosti je stěžejní, posouvá děj dále dopředu a to je, na co by se divák měl zaměřit. Vedle toho vedlejší činnost přidává animaci hloubku, [LR12] ji nazývá okno do podtextu scény. Hlavní akce může být scéna se zaměstnancem a rozčíleným nadřízeným. Hlavní akce zaměstnance může být přikyvování křičícímu vedoucímu a vedlejší zapití prášku na uklidnění.

Zbarvení vedlejší činnosti může přidat změnu charakteru, s kterým postava provádí opakovanou činnost, pokud se změní například emoční stav postavy. Opakování činnosti je v počítačové animaci snadné, lze pouze zkopírovat klíčové snímky animace a mírně je upravit tak, aby se odstranila dokonalost opakované činnosti.

Načasování

Je samozřejmostí, že se animátor snaží vytvořit pohyby, které odpovídají svou rychlostí tomu, v jakém čase by je člověk opravdu prováděl. Časování akce by zároveň měl vést smysl záběru, svůj význam mají i pauzy mezi akcemi. Pomocí posouvání klíčových snímků může animátor počítačového modelu s časováním experimentovat a ihned se přesvědčit, co svému účelu poslouží nejlépe.

Nadsazení

Provádění pohybů, jejich časování, kompozice, předjímání a i další techniky se dají zveličit pro dosažení vtipnější, přímočařejší nebo nadsazené animace. [LR12] však upozorňuje na příliš časté užívání této techniky. Měla by se používat jen za docílením jasného účelu, nikoliv pouze pro získání vzhledu klasického kresleného filmu. Úpravou klíčových snímků, zejména hodnot atributů, lze s nadsázkou pracovat a tvořit různé verze nadsazení.

Obratnost v kresbě

Je jasné, že kreslená animace kladla na výtvarníky obrovské nároky na zručnost a zkušenost. Zkušený animátor byl schopný nakreslit postavu tak, aby se sama sobě dokonale podobala ve všech možných úhlech a výrazech. Mohlo by se zdát, že tento bod je na počítačovou animaci nepoužitelný. Animační rig však často nabízí širokou škálu výrazů a pozic tak, že při nevhodném použití může být postava k nepoznání od svého předchozího výrazu. Rig může obsahovat omezení různých extrémních póz, ty však mohou být animátorovi na obtíž a je spíše na něm, aby rig správně používal.

Přitažlivost

Každá postava, kladná i záporná, by měla být pro diváka zajímavá, aby stál o to se na ni dívat. Při správném použití časování, kompozice, zajímavých dynamických pohybů, vzhledu postavy, kontrastujících tvarů a rytmu lze vytvořit velmi zábavnou a přitažlivou postavu.

3.2.2 Druhy požadavků na animační rig

U každého modelu se rigger musí rozhodnout, které funkcionality rigu vytvoří a jaký typ ovládacích prvků použije. Při jeho rozhodování by měly hrát roli požadavky animátora, který s rigem bude pracovat, standardy obecné i případně standardy animačního studia. Zároveň není praktické implementovat prvky, které nikdo nepoužije i z důvodu zachování přehlednosti rigu. Určit, které ovládací prvky budou viditelné stále a které se budou zviditelňovat až po zapnutí, není také banální problém.

Při tvorbě animačního rigu musí být dopředu jasná stylizace animace. Na modely totiž mohou být kladeny požadavky na realistické vyznění pohybů. Takový model nepotřebuje tolik ovládacích prvků, je náročný především na nastavení fyzikálních simulací. Druhou cestou je přístup, kdy pohyby modelu mají připomínat kreslenou animaci, která často nezachová proporce modelu. Ten se pak může různě natahovat, prohýbat či jinak deformovat. Rig takového objektu obvykle dává animátorovi mnohem větší kontrolu nad celým modelem a prováděnými deformacemi.

Kontrolery

Pro ovládání rigu je možné využít transformační uzel kteréhokoliv DAG objektu, na který se navážou pomocí hierarchických vztahů či pomocí omezení funkce rigu. Pokud je relevantní, aby daný uzel ovládal rig pomocí jiného atributu než těch transformačních, je možné mu novou atributu přidat. Naopak ty atributy, které nedávají smysl, aby byly používány, je možné zamknout a zneviditelnit.

Objekty pro vytvoření kontrolerů jsou nejčastěji ty, které nejsou vykreslovány, zobrazují se jen v náhledu na scénu. Z těch nejjednodušších je to lokátor, zajímavější tvar pak nabízí křivka. Tvar křivky je možné využít k naznačení, kterou konkrétní část rigu daný kontroler ovládá, a dokonce lze pomocí něho vyjádřit, zda slouží pouze k translaci, k rotaci nebo k nějaké kombinaci druhů transformací.

Jak zmiňuje [KR06], především u mechanických modelů je možné pomocí skriptů nechat označovat animátora přímo geometrii modelu, s kterou chce manipulovat. To v praxi vypadá tak, že rig nepoužívá mnoho křivek, animátor klikne například přímo na zápěstí modelu pro přesunutí jeho ruky. Toto nastavení nabízí animátorovi přirozenější manipulaci s rigem, na druhou stranu se jedná o poměrně neobvyklé ovládání, jehož fungování nemusí být vždy jednoznačné.

Rig často znemožňuje animátorovi označit jiný objekt, než ten, který má animovat, aby nějakým způsobem funkčnost rigu neporušili. Většina pomocných objektů je proto zneviditelněna a přesunuta do skupiny v DAG struktuře, která je jasně označena, že neslouží k ovládání rigu. Geometrii deformovaného objektu animátor vidět musí, a pokud není použita zmíněná metoda přímého označování, je označení samotné geometrie znemožněno. Maya nabízí minimálně dva způsoby, jak toho docílit. Prvním je vložit geometrii do vrstvy, která má Display Type nastaven na Reference. Podobnou možností je toto nastavení vytvořit přímo útvarovému uzlu, který stejné nastavení nabízí v Object Display, v záložce Drawing Overrides.

Pick walk

Technika pick walk dovoluje uživateli Mayi pohybovat se v DAG. Pomocí šipek je možné označit DAG uzel, který je v hierarchii umístěn v nějakém vztahu s právě označeným objektem. Pomocí šipek vlevo a vpravo je možné pohybovat se mezi sourozenci, šipky nahoru označí přímého rodiče. Šipka dolů označí přímého potomka, který se nachází na prvním místě.

V rámci rigu není normálně možné pick walking používat, protože vztahy mezi kontrolery nejsou vytvořeny pouze DAG hierarchií. [KR06] však zmiňuje, že někteří animátoři toto chování rádi používají, a tak je vhodné vytvořit skript, který pick walk v rámci rigu dovolí přenastavit.

Nástroje pro animátory

Je diskutabilní, zda se dají jako součást rigu označit i nástroje pro animátory, které jim pomáhají pracovat s klíčovými snímky. [KR05] a [KR06] takové nástroje zmiňují, konkrétně se jedná o skript, který dokáže uložit a načíst ze souboru pozice kontrolerů. Pro animátory

tak vzniká banka s obrázky, které jim pomáhají vždy zopakovat daný výraz postavy. Další užitečné skripty pro animaci dovolují kopírovat animační křivku a zrcadlit hodnoty tak, aby byla animace použitelná i pro končetiny druhé poloviny těla.

3.2.3 Příprava rigu

Důkladný návrh animované postavy neobsahuje jen to, jak má postava vypadat, ale i přímo nákresy stěžejních pohybů, které má být schopna vyjádřit. Spolu s obrázkovým scénářem jsou tyto informace základem pro plánovací fázi před započetím riggingu. Rigger by měl v první řadě posoudit, zda připravený model bude schopný všechny požadované akce provádět. Měnit geometrii modelu s již rozpracovaným rigem může být totiž často problém, který rig může poničit, a tak vyžadovat jeho částečné přepracování. Další částí přípravy je plánování samotné implementace rigových funkcí i jejich kontrolerů. To, čeho má být model schopen, může pomoci riggerovi určit i rozbor způsobu dodržování animačních zásad v připravovaném audiovizuálním díle. Velkou pozornost si především žádá obličej postavy.

Obličej

Tím, co je lidský obličej schopen vyjádřit, se velmi podrobně zabývá [PE02]. Kódový systém obličejových pohybů, FACS (Facial Action Coding Systém), rozlišuje jednotlivé pohyby obličejových svalů a rozděluje je na číselně označené akční jednotky. Jejich kombinací je možné popsat jakýkoliv obličejový výraz a pohyb očí a hlavy. Před započetím prací na obličeji postavy by měl rigger seznam výrazů projít a vytvořit vlastní, pomocí kterého bude možné implementaci naplánovat. Zároveň je třeba zohlednit akce, které daná postava má být schopna vykonat vzhledem k jejímu obličeji i vzhledem k jednotlivým scénám audiovizuálního díla. Některé akční jednotky je možné vyjádřit různou intenzitou jednoho deformátoru, proto seznam není vůbec přepisem akčních jednotek. Seznam by měl tedy zmiňovat akční jednotky, které je možné pomocí daného ovladače vyjádřit, stejně jako typ použitého deformátoru. Alternativou k blendshape, který posouvá vrcholy deformovaného objektu po přímce, může být i kloub, který je schopen pracovat s rotací vrcholů.

Jak píše [KR05], v této fázi je vhodné pracovat se symetrickým modelem, je-li to možné. Díky tomu bude možné druhou část obličeje pouze překopírovat, a tak ušetřit mnoho času. Pro získání drobných vrásek a záhybů také nemusí být nezbytné pracovat se složitou geometrií. Pomoci mohou výškové textury vrásek s automaticky ovládanou hloubkou.

Je třeba ještě zmínit, že tento popsaný rigerský přístup je nazýván odspodu nahoru, kdy je třeba pro docílení jednoho výrazu použít více pohybů. Přístup shora dolů má připravené jednotlivé, kompletní výrazy, které však nedávají animátorovi mnoho prostoru a špatně se kombinují.

3.2.4 Zátěž při použití rigu

Výslednou použitelnost rigu může ovlivnit nejen jeho praktičnost a pokročilost funkcí, ale důležitým faktorem je také výsledná zátěž, kterou na animační program vytváří. Dříve, když se pracovalo na méně výkonných systémech, bylo třeba používat v náhledu pouze modely rozdělené na jednotlivé objekty, protože výpočty pro deformaci pomocí smooth skin program

příliš zatěžovaly a animátor si tak nemohl vytvářenou animaci plynule přehrávat. Podobný problém může v současnosti nastávat, když je ve scéně použito více postav s náročnými rigy, jeden velmi komplikovaný rig a podobně. Chybou může být také to, že rig neumožňuje některé postradatelné funkce pro animátora vypnout, například výpočty dynamiky.

Kapitola 4

Implementace

4.1 Testovací postava Zloděj

Výsledkem celého rigovacího procesu je zpřístupnění souboru deformací počítačového modelu. Při testování rigu je nezbytné právě deformace sledovat a posoudit výsledek i jejich ovládání. Kapitola seznámení s problematikou vysvětluje, že výsledný deformovaný tvar je také silně ovlivněn geometrií i topologií modelu. Proto je třeba zvolit vhodný model, který je pro animaci připraven. Jako zdroj informací pro modelování byly dodrženy zásady, které zmiňuje[Tin12].

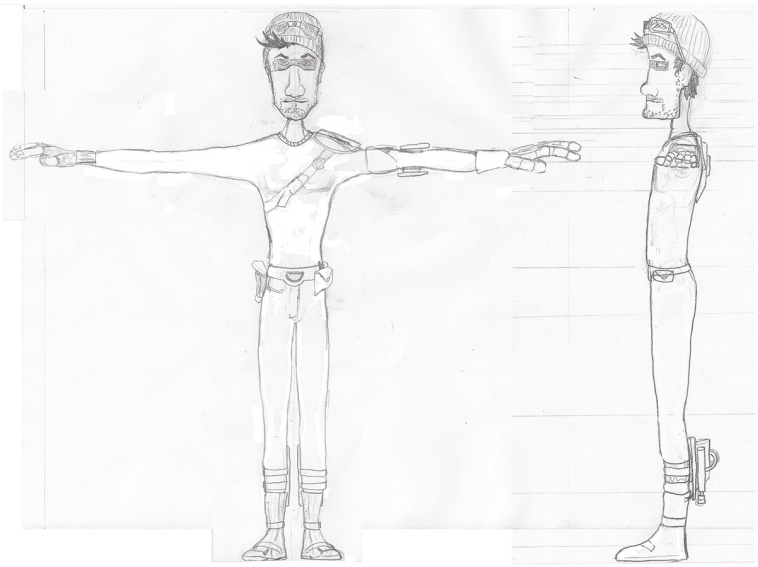
4.1.1 Návrh

Nejčastěji rigovaný objekt, který zároveň obsahuje mnoho problémů k vyřešení, je model lidské postavy. Rig se stává ještě složitější, pokud by postava měla být schopna animační nadsázky. S tou je často spojena i výtvarná stylizace, kterou by návrh takové postavy měl obsahovat. Pokud by postava měla být i co nejvíce univerzální a použitelná mimo podmínky velkého animačního studia, její aplikace v animační sekvenci by neměla vyžadovat složité scény a ideálně ani další postavy. Z těchto důvodů vznikl návrh stylizovaného sci-fi zloděje, s kterým je možné vytvářet nenáročné skeče. (Obrázek 4.1)

4.1.2 Tvorba modelu

Do programu Maya byl vložen nákres postavy tak, aby bylo možné při pohledu z boku a zpředu podle něj model vytvořit. Tvar těla vznikl z kvádru pomocí extrudování ploch a přidáváním hran a hranových smyček. Obličej vznikl složitější, ale přesnější metodou, extrudací hran a spojováním vrcholů. Díky tomu bylo možné mít kontrolu nad tím, kudy hranové smyčky obličeje povedou. Zvlášť důležité bylo, aby v místě očí a úst smyčka hran tvořila kruhy. Pro deformaci obličeje vyjadřující úsměv bylo zase třeba, aby hrany vedly od střední části nosu přes tvář kolem úst nad počátek brady. Zvláštní pozornost si žádalo ucho, které vzniklo jako samostatný model a poté bylo vrchol po vrcholu spojeno s geometrií hlavy. Stejným způsobem byla celá hlava připevněna k modelu těla.

Tělo bylo vymodelováno bez oblečení, v místech ohybu bylo třeba zvýšit hustotu vrcholů, aby při deformaci neklesala geometrická přesnost modelu. Směr smyček hran bylo třeba



Obrázek 4.1: Návrh modelu Zloděj
Zdroj: vlastní

sledovat především v místech dlaně a ramena. Při vzniku oblečení byla struktura těla využita. Duplikací a následnou úpravou modelu těla zloděje vznikl model pro čepici, škrabošku, triko, rukavice, pásek i kalhoty. Tvar vlasů, obočí a bot byl vytvořen pomocí extrudace základních primitiv.

Struktura texturovacích souřadnic vznikla pro místa, které nejsou zakryté jinou geometrií, tedy pro odhalené části kůže a pro oblečení. Bylo použito rovinné a automatické mapování celých objektů nebo jen jejich částí. Výsledný plášť vznikl sešíváním menších částí tak, aby model obsahoval co nejméně švů a aby tyto švy byly umístěny na co nejméně exponovaných místech.

4.2 Autorig Brig

Pro aplikaci představených rigovacích postupů a funkcionalit programu Maya vznikl program Blue-Rig, zkráceně Brig. Brig stejně jako ostatní autorigy co nejvíce usnadňují rigovací proces, proto nabízí uživatelské rozhraní, které implementované funkce ovládá a spouští. Architektura programu odděluje grafickou složku od tříd pro ukládání a práci s daty a od samotných funkcí, které vytvářejí jednotlivé části rigu. Tyto funkce lze spustit jak pomocí grafického rozhraní, tak vykonáním jednoduchého příkazu přímo v Maye. Aby bylo možné porovnat výhody a zápory obou jazyků, které Maya nabízí, byly použity oba, Python i MEL. Maya zprostředkovává komunikaci mezi oběma jazyky, proto bylo možné přistoupit k takovémuto řešení.

4.2.1 Uživatelské rozhraní

Grafické ovládání je nedílnou součástí každého autorigu, proto bylo vhodné tuto aplikační vrstvu pro Brig také vytvořit a zároveň vyšetřit možnosti, které Maya ohledně uživatelského rozhraní poskytuje. Grafické rozhraní je celé vytvořeno v jazyku Python s využitím standardní knihovny maya.cmds pro komunikaci s Mayou. Funkce jsou rozděleny do jednotlivých oken dle toho, v jaké části rigovacího procesu se uživatel nachází.

Obecné okno

Třída Default Window obsahuje základní funkce a proměnné, které pomocí dědění zprostředkovává dalším oknům aplikace Brig. Tato třída umožňuje ukládat základní informace o okně, jeho vytvoření, zobrazení a funkci pro vytvoření tlačítek Apply and Close, Close a nepovinné tlačítko Apply v dolní části okna. Zpřístupňuje ji soubor `defaultwin.py`.

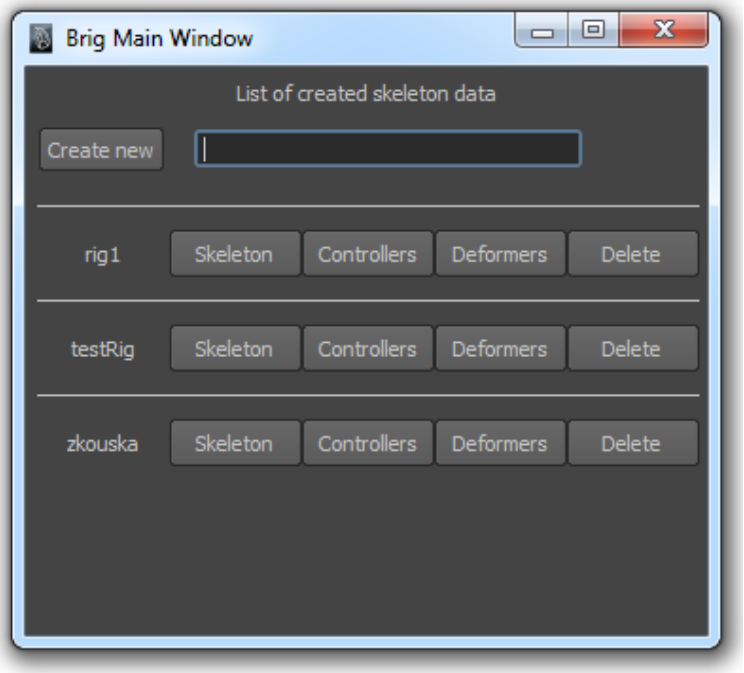
Okno Brig Main Window

Hlavní okno nabízí přehled vytvořených či rozpracovaných rigů, které je možné dál editovat pomocí dalších oken. Toto okno zároveň umožňuje vytvořit rig nový, spustit další okna programu Brig, konkrétně Assign Joints, Create Controllers a Create Deformations. Poslední tlačítko Delete umožňuje uložená data daného rigu vymazat. Vypisovaná data tomuto oknu poskytuje třída SkeletonData, jejíž instanci toto okno dokáže vytvořit při založení nového rigu. Toto okno vytváří třída BRigMainWindow pomocí kódu v souboru `brigwin.py`. (Obrázek 4.2)

Okno Assign Joints

Aby Brig mohl správně pracovat, potřebuje znát umístění každého kloubu vzhledem ke geometrii modelu. Okno Assign Joints umožňuje přiřadit jednotlivé klouby nově vytvořenému rigu pomocí grafických ikon. Jak ukazuje obrázek 4.3, díky siluetě postavy je možné přiřadit jednotlivé klouby rigovacímu systému kliknutím na modrou ikonu kloubu. Pokud je kloub již přiřazen, je ikona šedivá. Takto zbarvenou ikonou je možné kloub označit či přiřazení zrušit. Ikony ruky a nohy otevírají pro přesnější práci další zobrazení, v kterém je každý kloub dobře vidět. Páteř je jediný kloub, který může pojmout více kloubů zároveň. Toto zobrazení automaticky reaguje i na smazání kloubu ze scény.

Okno Assign Joints obsahuje tlačítka Mirror Joints, Save Joints, Load Joints, Erease Joints a Rename for MB. Pomocí tlačítka Mirror Joints je možné zrcadlit přiřazení kloubů i klouby samotné. K tomu slouží speciální okno, které umožní uživateli průběh funkce specifikovat. Tlačítko Save Joints dovoluje pozici kloubů i jejich přiřazení uložit do externího souboru s příponou brs (Brig Skeleton), který je možné opět načíst tlačítkem Load Joints. Obě tlačítka otevírají běžné systémové okno pro lokalizaci souboru ve složce pro kostry programu Brig. Základní instalace programu obsahuje v této složce uloženou výchozí kostru, což dává uživateli alternativu k procesu přiřazování kloubů. Klouby načtené kostry je možné jen posunout na správné místo v modelu a tím končit přiřazovací fázi. Přiřazené klouby umožňuje smazat tlačítko Erease Joints. Program Motion Builder vyžaduje přesný název



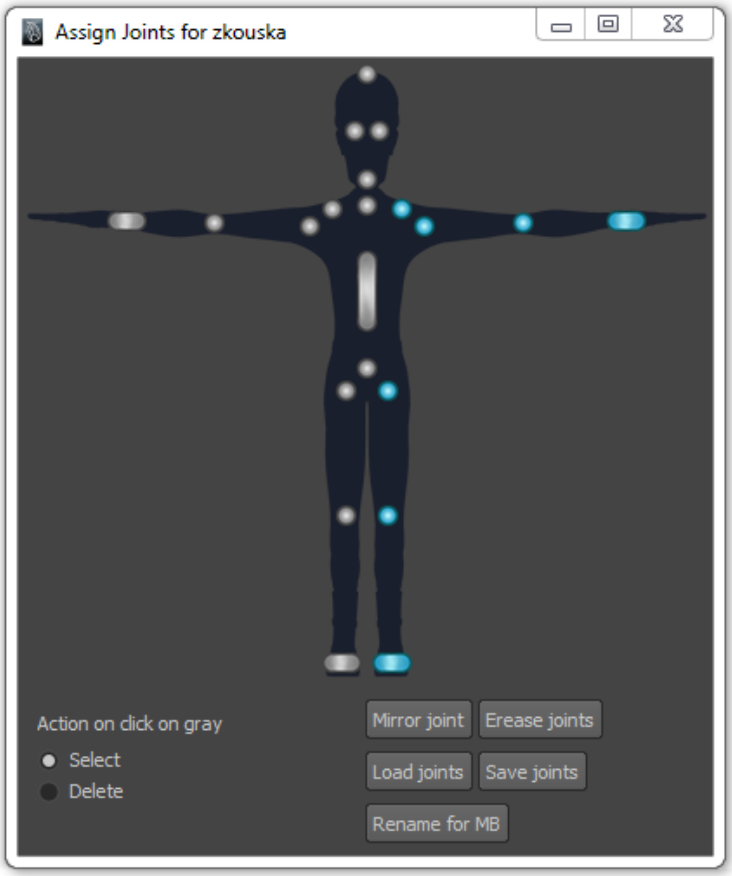
Obrázek 4.2: Hlavní okno programu Brig
Zdroj: vlastní

pro každý uzel kostry, aby dokázal správně s kostrou pracovat a navázat na ni data ze systému motion capture. Pomocí tlačítka Rename for MB je možné automaticky přejmenovat přiřazené klouby tak, aby bylo možné model pro Motion Builder exportovat.

Většina funkcí třídy AssignWindow zprostředkující okno Assign Joints komunikuje s třídou SkeletonData pro uchování a získání dat. Předává ji například informace o právě přiřazených kloubech a dotazuje se, zda je právě vykreslovaný kloub již přiřazen. Obecněji řečeno, všechny operace s kostrou přenechává právě jí. Její nejsložitější metodou je funkce pro zobrazení všech tří obrazovek s grafikou. Ta je překreslována při každé změně v přiřazených kostech. Protože při načítání celé kostry pomocí tlačítka Load Joints by k takové události došlo mnohokrát za krátkou chvíli, obsah okna je možné překreslit pouze po 0.2 sekundy dlouhém intervalu od posledního volání. Třída AssignWindow a třída MirrorJWindow pro ovládání okna s nastavením funkce zrcadlení jsou obsaženy v souboru `assignwin.py`.

Okno Create Controllers

Vytváření kontrolerů kostry je prováděno pomocí okna Create Controllers. To ukazuje nabídku dostupných rigovacích skriptů podle toho, které klouby byly přiřazeny a které kontrolery byly již vytvořeny. (Obrázek 4.4) Díky seznamu skriptů je možné jednotlivé části rigu vytvářet i mazat. Pro zpřehlednění návaznosti jednotlivých rigovacích úkonů je zde umístěno tlačítko Orient joints, které spouští stejnojmenné okno programu Maya. Okno Create Controllers je stejně jako okno Assign Joint schopno automaticky reagovat na změny přiřazených

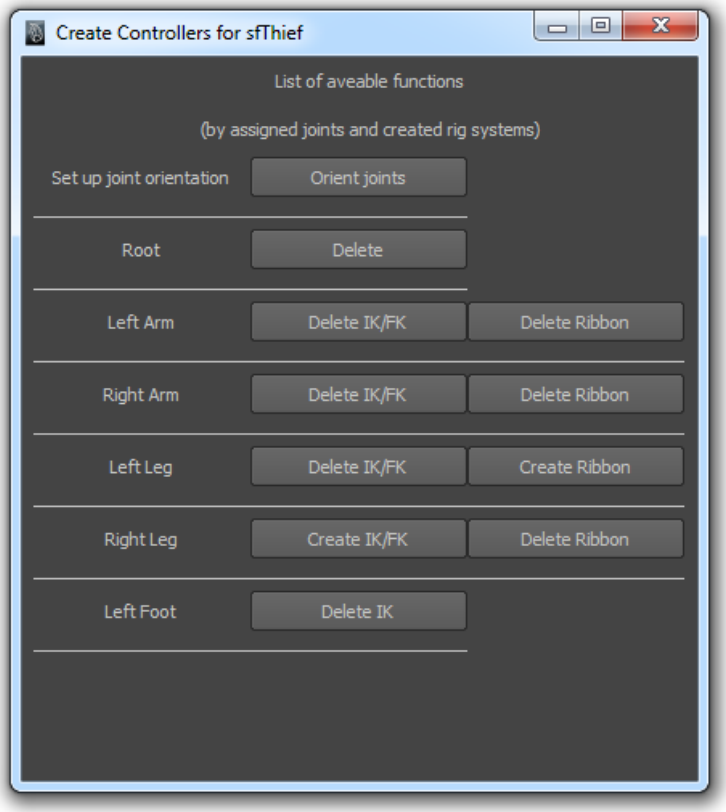


Obrázek 4.3: Okno Assign Joints
Zdroj: vlastní

kloubů.

Tlačítka Create IK/FK a Create Ribbon otevírají okna pro specifikaci prováděné funkce. Okno otevírající tlačítko Create IK/FK dynamicky zpřístupňuje a zakazuje některé nabízené volby dle nastavení nadřazených hodnot. Okno tlačítka Create Ribbon je nastavené tak, aby dynamicky dokonce přidávalo a ubíralo jednotlivé elementy okna na základě nastavení klíčových prvků.

Třída ContWindow okna Create Controllers přistupuje k informacím o kostře prostřednictvím třídy SkeletonData a k informacím o částech rigu třídou SystemDataNode. Třída ContWindow řeší samotné rozeznání, které funkce mají být uživateli zpřístupněny, provádí přípravu dat pro dané funkce a identifikaci, pro kterou část kostry byl daný skript spuštěn. Třídy IKFKWindow a RibbonWindow pro nastavení parametrů funkcí využívají událost elementů grafického rozhraní, která volá funkci při změně hodnoty daného parametru. Díky tomu je možné dynamicky editovat, mazat a vytvářet zobrazovaný obsah oken. Všechny tři zmíněné třídy jsou obsaženy v souboru `rigwin.py`.



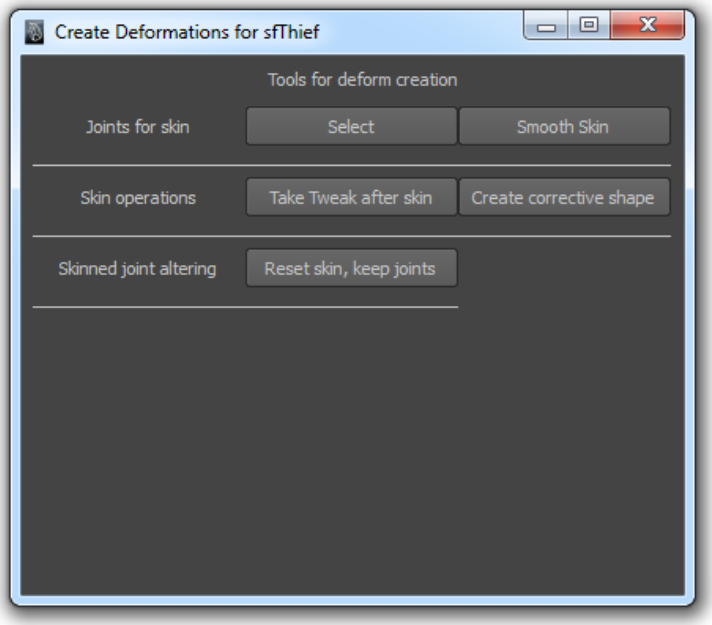
Obrázek 4.4: Okno Create Controllers
Zdroj: vlastní

Okno Create Deformations

Soubor několika nástrojů poskytuje okno Create Deformations. První řádek okna umožňuje automaticky označit kosti rigu, které mají být použity pro skinning. Při vytváření kontrolerů může totiž nastat situace, že je původní část kostry nahrazena jinými klouby, zároveň nemá smysl zahrnovat koncové klouby všech větví kloubové hierarchie. Ze selekce je vyloučen i kořen hierarchie a kloub specifikující pozici paty modelu. Označené klouby je pak třeba pomocí smooth skin navázat na rigovaný model, proto okno Create Deformations nabízí tlačítko pro vyvolání standardní nabídky Mayi pro vytvoření právě smooth skin. Je třeba zmínit, že pro správné použití selekce musí mít smooth skin nastavenou volbu Bind to na Selected joints.

Další pár tlačítek tohoto okna umožňuje vytvořit korekce modelu po deformaci smooth skin. První tlačítko zahájí úpravu označeného modelu a to druhé korekci vytvoří a přiřadí modelu jako nový objekt blendshape. Poslední tlačítko tohoto okna s popiskem Reset skin, keep joints pracuje s označenými klouby. Pokud tyto klouby provádí prostřednictvím smooth skin nějakou deformaci, je tato deformace zrušena a stávající pozice kloubů je použita pro smooth skin jako výchozí. Třída tohoto okna pojmenovaná DeformWindow pracuje pouze

se selekcí a spouští specifické metody k usnadnění práce s deformátory. Třída je uložena v souboru `deformwin.py`. (Obrázek 4.5)



Obrázek 4.5: Okno Create Deformers
Zdroj: vlastní

4.2.2 Správa dat

Maya pro své programovací jazyky nenabízí přímou konkrétní funkcionalitu, která by nabízela ukládání dat. Vzhledem k volnosti, kterou Maya programovacím jazykům nabízí, je tento problém možné vyřešit více způsoby. Program Brig ukládá informace o kostře i o částech rigu do uzlu typu Unknown. Tento uzel nemá v Maye specifickou roli, a tak nebude matoucí ho použít pro účely ukládání dat, navíc se nejedná o DAG uzel, proto bude ve scéně více skrytý. Maya dovoluje každému uzlu přiřadit uživatelsky definované atributy. Je možné tak upravit a používat uzel, aby se choval jako asociativní pole o proměnných datových typech. Každý uzel je obsluhován instancí třídy, která s ním provádí veškeré operace. Jméno uzlu, jména atributů a jejich hodnoty musejí obsahovat veškerá data, které instance třídy může potřebovat, aby při načtení scény bylo možné všechny data obnovit. Tento přístup pro ukládání dat využívají obě třídy, SkeletonData a SystemDataNode. Obě jsou napsány v jazyku Python za použití knihovny PyMEL pro komunikaci s Mayou.

Uložení přiřazených kloubů

Třída SkeletonData definována v souboru `skeletdata.py` obsahuje řadu konstant pro definici pojmenování jak datového uzlu, tak pro pojmenování jednotlivých atributů. Každý datový

uzel má tak pevně určené jméno dle prefixu a jména rigu, jehož data uchovává. Díky tomu je možné v konstruktoru třídy datový uzel dohledat a v případě jeho absence vytvořit uzel nový. SkeletonData využívá pro jeden rig datové uzly dva, jeden pro uložení přiřazených kloubů páteře a druhý pro evidenci pevně definovaných kloubů kostry. Evidence jednotlivých kloubů je prezentována pomocí konkrétně pojmenovaných atributů datového typu message. Řada různých uzlů Mayi včetně kloubů obsahuje atribut message, který s takovýmto atributem lze propojit. Toto spojení poskytuje informaci pouze o jménu objektu. Oproti možnosti uložení jména kloubu jako textového řetězce má toto řešení výhodu v tom, že je kloub stále dohledatelný i při změně jeho jména a je možné evidovat jeho smazání (či jeho znovuvytvoření pomocí příkazu krok zpět).

Třída SkeletonData umožňuje vytvářet záznam o kloubu, rušit ho, zjišťovat existenci přeřazení konkrétního uzlu a jeho název. Dále je možné všechny přiřazené klouby smazat či přejmenovat tak, aby splňovaly požadavky pro snadný import do programu Motion Builder. Pomocí statických metod lze smazat oba datové uzly dle názvu rigu a je také možné získat názvy všech vytvořených rigů scény podle vyhledání uzlu dle prefixu a jeho následném vyloučení z textového řetězce. Další funkce, kterou tato třída nabízí, dokáže vrátit pole jmen přiřazených kloubů. Je možné ji spustit v režimu, v kterém z výsledku vyloučí klouby, které se nemají použít ke skinningu.

Zrcadlení přiřazení je další funkcionalitou, která dokáže pracovat s datovým uzlem. Skript pomocí pozice kloubu root stanoví střed, díky kterému je schopen ve zvolené ose dohledat nepřiřazený kloub na opačné straně od přiřazeného kloubu.

Tvorba textového řetězce z informací z datových uzlů a parsování textového řetězce je též implementována v této třídě. Skript postupně projde všechny atributy obou datových uzlů. Díky tomu dohledá klouby a následně jejich pozici ve světových souřadnicích. Při prasování je třeba provést opačný proces, klouby uložit do datového uzlu a správně je spojit v DAG hierarchii.

Jako poslední je třeba představit funkci, pomocí které je možné registrovat externí metodu, která se má provést při změně spojení některého kloubu. Pro každý atribut je tak vytvořen Skript Job, který dané metody spouští. Script Job je vytvořen při registraci metody a přitom je navázán na existenci určitého grafického prvku, takže při jeho ukončení je zrušen i samotný Skript Job. Tímto způsobem je umožněno aktualizovat okna grafického rozhraní při každé změně datového uzlu.

Uložení částí rigu

Třída SystemDataNode uložený v souboru `ridgata.py` vytváří datový uzel specifický pro každý rig, část těla a použitou třídu skriptu. Vytváří rozhraní pro uložení všech permanentních částí vytvořeného rigu. Do jednoznačně pojmenovaného datového uzlu tato třída vytváří atributy s předdefinovanými prefixy, které slouží pro určení typu uložené hodnoty. Díky němu je možné rozeznat uložené kontrolery, hlavní kontroler, pomocné objekty, klouby používané pro skinning, uzel expression, omezení a i klouby kostry, které se mají při používání skinningu vyloučit. Další dva prefixy určují vytvoření Script Job skriptů s různými spouštěči. Pořadové čísla všech skriptů jsou po jejich vytvoření vždy uložena na konec textového řetězce atributu pro jejich evidenci.

Instance této třídy si ukládá pouze informace nutné k identifikaci svého datového uzlu. Díky této skutečnosti je možné kdekoliv v kódu tuto instanci vytvořit a pracovat s danými daty, není třeba mezi metodami odkaz na instanci předávat. To umožňuje její použití v jazyku Python a zároveň v jazyce MEL. Většina skriptů pro tvorbu částí rigu je totiž napsána v jazyce MEL. Ten pomocí příkazu `python` dokáže tuto třídu používat, datový tok mezi nimi je však omezen, a tak by nebylo možné MEL skriptu požadovanou instanci předat.

Při vytvoření instance je vždy zkontrolován výskyt specifického Skript Nodu ve scéně. Tento Skript Node je unikátní pro celou scénu a stará se o to, aby při otevření scény byly vytvořeny šechny uzly typu Script Node, které jednotlivé rigy ve scéně ke svému fungování potřebují. Při vytvoření části rigu využívající Script Job je jeho zadání parsováno a uloženo do datového uzlu ve formě textového řetězce. Právě Script Node po otevření scény pomocí statické funkce této třídy přečte uložené textové řetězce a dle instrukcí vytvoří všechny potřebné Script Job skripty.

Funkce této třídy umožňují smazat veškeré objekty příslušné části rigu, přičemž jsou nejdříve smazány veškeré Skript Job skripty díky evidenci jejich pořadových čísel v datovém uzlu. Poté jsou smazány všechny objekty uložené v datovém uzlu tak, aby se zrušily transformace kostry vzniklé kontrolery odstraňovaného rigu.

Pro úplnost je třeba dodat, že funkce třídy SystemDataNode umožňují snadno přidávat objekty a skripty Script Job a vracet název hlavního kontroleru. Protože některé rigy vytvářejí nové klouby, které mají nahradit při skinningu části základní kostry, tato třída vlastní statickou metodu pro průchod všech datových uzlů rigu. Tato metoda pracuje zároveň s polem kloubů, které poskytla třída SkeletonData, a tak je schopna připravit výslednou množinu kloubů pro skinning.

4.2.3 Skripty pro automatizaci rigovacího procesu

Brig nabízí základní množinu funkcí, které automatizují časté rigovací procesy a které bez použití skriptového nástroje stojí mnoho monotónní práce. Implementované skripty byly vybírány s ohledem na to, aby řešily vždy nějaký unikátní problém, každý z nich tak využívá jiné funkcionality programu Maya, pracují především s kostmi, s omezeními i s deformátory. Vzhledem k zaměření této práce nebyly implementovány animátorské nástroje. Pro úpravu výsledku funkce skinning Brig nabízí skript pro vytváření korekcí namísto použití svalového systému. Skript pro implementaci uživatelsky nastavené funkce pick walking byl zavrhnut z důvodu toho, že upravuje základní fungování zkratek programu Maya.

Funkce byly vytvořeny jak v jazyku Python, tak za použití jazyka MEL. Je možné je spustit jak pomocí uživatelského rozhraní, které skriptu poskytne potřebná data, tak pomocí příkazu, který získá potřebná data z parametrů funkce a ze seznamu označených objektů scény.

Tvorba kontrolerů

Všechny použité kontrolery v programu Brig mají některé společné vlastnosti. Jsou tvořeny NURBS křivkami a jejich transformační uzel má skryté atributy, jejichž uživatelská editace nedává vzhledem k zaměření kontroleru smysl. Viditelné hodnoty jsou nastaveny tak, aby

ve výchozím nastavení kostry měly nulovou hodnotu. Tvar, který křivka tvoří, by měl napovídat tomu, k jakým transformacím je daný kontroler určen a jeho pozice musí intuitivně specifikovat, kterou část modelu ovládá. Kontroler tak může mít tvar běžného kruhu, může však nabývat i složitějších tvarů jako jsou hvězda, krychle či koule.

Soubor `controllers.mel` obsahuje balík funkcí, které pomáhají popsané vlastnosti kontrolerů provádět. Dokáží tak upravovat atributy pro kontroler nabízející pouze posunutí, pouze rotaci, či obě možnosti. Kontroler ve tvaru koule, který se vykresluje podobně jako kloub, je možné vytvořit pomocí tří na sebe kolmých kruhů typu NURBS, které jsou přiřazeny pod jeden transformační uzel. Kontroler ve tvaru krychle je vytvořen pomocí křivky prvního stupně. Taková křivka tvoří rovné spojnice mezi řídícími body, které je možné do tvaru krychle poskládat. Soubor `brHelpers.py` obsahuje metodu pracující s označenou křivkou, která byla libovolně upravena. Tu tento skript převede na příkaz jazyka Python, pomocí kterého ji lze vytvořit. Díky tomu je snadné vytvářet prostřednictvím několika příkazů i složitější tvary křivek. Toho například využívá funkce uložená v `rootContCreator.py`, která je takto schopna vytvořit kontroler ve tvaru hvězdy pro ovládání kořenu rigu.

Vytvoření objektu Follicle

Na [Car13] bylo představeno několik rigovacích technik, které využívají objekt Follicle k vytvoření vazby mezi bodem na povrchu objektu a druhým objektem. Vzhledem k tomu, že Maya nenabízí nástroj, pomocí kterého by se Follicle dal řízeně na konkrétním místě povrchu vytvořit, je vhodné automatizovat proces, který k takovému nastavení vede. Vstupní hodnoty vytvořeného skriptu jsou cílový objekt, na který bude Follicle vytvořen, a lokalizátor, kterým bude označován libovolný typ objektu, jehož pivot bude řídit požadovanou pozici pro vytvoření objektu Follicle. Ten vznikne na spojnici nejbližšího bodu povrchu cílového objektu a právě pivotu lokalizátoru. Skript nejdříve jedním příkazem vytvoří uzel follicle a follicle shape, jejichž atributy určující posun a rotaci je třeba propojit dalším příkazem. Follicle shape je pak propojen s cílovým objektem, je vytvořen uzel closestPointOnSurface (popř. closestPointOnMesh pro cílový objekt typu mesh), který je propojen jak s lokalizátorem, tak s cílovým objektem. Výstupní hodnoty uzlu closestPointOnSurface jsou již v souřadnicích U a V, které objekt Follicle potřebuje k určení své pozice. Než se však tyto hodnoty propojí, je třeba výstupní U a V normalizovat rozpětím hodnot U a V, které cílový objekt obsahuje. Při takovémto nastavení vznikne Follicle, který vždy zaujímá takovou pozici, aby se nacházel na cílovém objektu co nejblíže lokalizátoru. Tento skript je implementován v souboru `createClosestFollicleDynamic.mel`.

Některé aplikace však požadují to, aby Follicle nebyl řízen lokátorem, ale po vytvoření naopak sám lokátor ovlivňoval. Proto vznikla druhá verze tohoto kódu, která hodnoty lokalizátoru nepropojuje, ale jen kopíruje. Na konci běhu této metody je uzel closestPointOnSurface smazán a objekt Follicle je nastaven jako rodič lokalizátoru. Tento skript je popsán v `createClosestFollicle.mel`.

Dopředná kinematika

Dopředná kinematika je v animaci velmi využívána ať už jako hlavní způsob animace, nebo jako alternativa k inverzní kinematice, kdy se mezi těmito druhy technik dá plynule přepínat. Animace pomocí dopředné kinematiky je možná přímo pomocí rotace jednotlivých

kostí modelu. Rigy však zpravidla nedovolují animátorovi přímý přístup k takovýmto strukturám a i v tomto případě vytvářejí nad kostmi sadu kontrolerů. K tomuto typu ovládání je vhodný často jednoduchý tvar kruhu se středem uprostřed ovládaného kloubu. Pozice tohoto kontroleru bude naopak definována pozicí kloubu. To poskytne animátorovi přehled, jakým směrem je kloub s kostí otočen.

Soubor `fkControllers.mel` obsahuje skript, který vytvoří křivku ve tvaru kruhu o uživatelsky nastavitelném poloměru. Skript tento kontroler umístí a otočí tak, aby se shodoval s pozicí ovládaného objektu. Skript obsahuje modifikátor, který ovlivní provedení výsledného kontroleru. Pomocí parametru je možné nastavit, aby byly pro ovlivnění objektů použity omezení. V takovém případě jsou nejdříve atributy transformačního uzlu kontroleru upraveny pro ovládání rotace. Poté vznikne omezení Point, kdy je pozice kontroleru řízena ovládaným objektem. Ovládaný objekt je pak pomocí omezení Orient ovlivňován rotací kontroleru.

Při odlišném běhu programu skript ověří, zda se ovládaný objekt nachází v DAG mimo kořen scény. Pokud ano, umístí ovládací prvek do hloubky, v které se cílový objekt nachází. Atributy tohoto transformačního uzlu jsou upraveny pro ovládání rotace a ovládaný objekt je nastaven jako potomek kontroleru. Celý proces pracuje v cyklu pro všechny objekty vstupního pole.

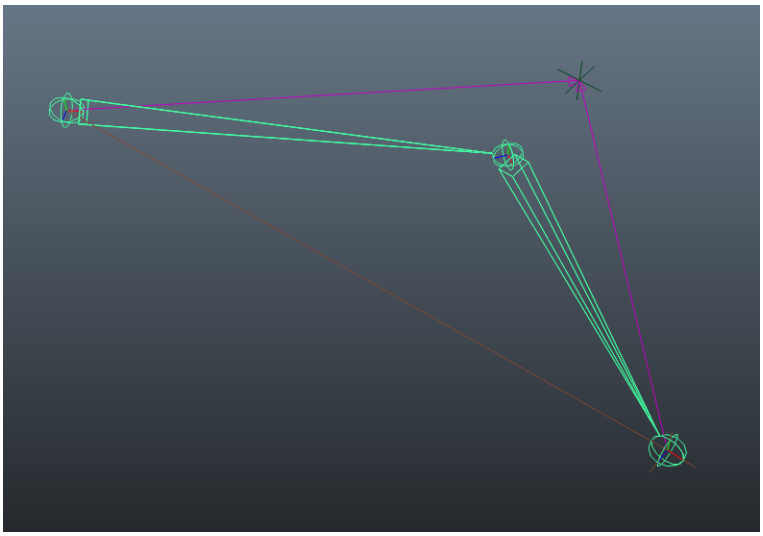
Inverzní kinematika

Vytvořit systém inverzní kinematiky nad zadaným řetězcem kloubů není v Maye nic složitého. Často je však třeba ještě přidat další objekt, který definuje plochu, v které se bude řetězec kloubů ohýbat. Objekt se se systémem inverzní kinematiky propojí omezením zvaným Pole vector, jehož samotné vytvoření není také komplikované. Problém přichází ve chvíli, kdy je třeba tento nový objekt umístit do prostoru tak, aby se při vytvoření tohoto omezení řetězec kloubů nepohnul. Pohyb kloubů by znamenal změnu umístění kloubů vůči modelu, což je při riggingu nepřijatelné. Proto je třeba objekt před vytvořením omezení umístit do plochy, v které se již klouby řetězce nachází.

Program pracuje se třemi klouby řetězce, které definují končetinu. Klouby je možné programu přiřadit pomocí parametru, verze programu pracujícího se selekcí objektů však obsahuje funkci, která pomocí orientace v DAG hierarchii určí automaticky začátek, konec a střed řetězce. Nezáleží tak, v jakém pořadí jsou klouby označeny. Díky tomu je skript schopen automaticky vytvořit systém inverzní kinematiky pomocí příkazu Mayi.

V této fázi je třeba určit pozici lokátoru tak, aby vznikl v rovině, kterou dané tři klouby tvoří. Na [Car13] bylo představeno elegantní řešení tohoto problému, které využívá funkce Mayi. Proto je skript rychlý a zároveň snadný implementovat. Tento skript tak vytvoří křivku o třech kontrolních bodech. Každý bod umístí do středu kloubu tak, aby v prostředním kloubu byl prostřední kontrolní bod. Pak pomocí funkce moveVertexAlongDirection přesune tento bod tak, že neopustí plochu, kterou s ostatními body křivky tvoří. Posun proběhne o hodnotu, která je funkci předána ve formě parametru. Na pozici prostředního bodu je pak vytvořen lokátor, křivka je smazána a bez posunu je vytvořeno omezení Pole vectore s novým lokátorem jako ovladačem. Pomocí objektu anotace jsou ještě vytvořeny dvě šipky, které lokátor graficky zvýrazní a naznačí propojení se systémem inverzní kinematiky, jak ukazuje obrázek 2. Anotace jsou pomocí omezení propojeny s klouby i s lokátorem a pomocí

Drawing Overrides jsou nastaveny na typ zobrazení zvaný reference. Díky tomu není možné křivky v náhledu na scénu omylem označit. (Obrázek 4.6)



Obrázek 4.6: Inverzní kinematika ruky s pole vectorem
Zdroj: vlastní

Pomocí parametru skriptu je možné vytvořit funkcionalitu, která ovládá natahování ruky. Uzel distanceBetween dynamicky počítá vzdálenost mezi první kostí řetězce a kontrolerem řetězce. Tato dynamická hodnota je uzlem multiplyDivide dělena konstantou, kterou skript získá sečtením délky obou kostí řetězce. Atribut Scale X kloubů těchto kostí je pak upravován pomocí výstupu uzlu Condition. Ten předává hodnotu jedna, pokud je výsledek dělení uzlu multiplyDivide menší než jedna. Pokud je tento podíl větší, je tato hodnota brána za výstupní. Toto nastavení tak dynamicky zapíná natahování řetězce pouze v momentě, kdy není schopen dosáhnout na kontroler. Všechny popsané skripty jsou součástí souboru `IK_with_poleVector.mel`.

Kombinace obou kinematik

Způsob animování pomocí inverzní kinematiky je v některých případech maximálně užitečný, existují však pohyby, které se mnohem lépe animují pomocí dopředné kinematiky. Díky tomu je často žádoucí, aby rig končetin, především ruky, obsahoval přepínač, který animátorovi umožní rozhodnout se, kterou techniku pro danou animaci chce použít. Při konkrétním nastavení metody pro tvorbu inverzní kinematiky je možné vytvořit navíc i systém dopředné kinematiky. Na hlavním kontroleru inverzní kinematiky přibude atribut, který řídí interpolaci mezi jednotlivými kinematikami. Přepínat mezi nimi umožňuje atribut základního ovladače inverzní kinematiky. Tento atribut je pouze propojen s atributem kontroleru, čímž je docílena popsaná funkcionalita.

Je známo, že animátoři se nechtějí zabývat přepínáním mezi dvěma systémy kontrolerů, ideálně by chtěli jednoduše používat oba zároveň. V Maye je takové chování problém, protože popsané nastavení vytváří cykly, kdy jedna hodnota je ovládána hodnotou, kterou sama

ovlivňuje. Pokud taková situace nastane, výpočet atributů se chová nepředvídatelně a celé nastavení není k použití. Brig představuje řešení tohoto problému pomocí skriptů Skript Job. Ty na základě selekce určitého kontroleru automaticky přepínají mezi kontrolery inverzní a dopředné kinematiky, přičemž podle vytvořených změn vždy aktualizují transformace neaktivních kontrolerů tak, aby jejich transformace odpovídaly kontrolerům aktivním.

V první řadě je třeba vyřešit, jak neaktivní kontrolery přesunout do nové pozice, která odpovídá těm aktivním. Změna pozice kontrolerů dopředné kinematiky pouze překopíruje rotaci kloubů v řetězci inverzní kinematiky do hodnot pro popis rotace kontrolerů. Ohledně ovlivnění kontrolerů inverzní kinematiky je problém složitější. Koncový kontroler lze jednoduše přesunout na pozici posledního kloubu řetězce, ovšem pro určení pozice kontroleru je třeba vytvořit pomocnou strukturu. Tou je při jednodušším řešením křivka, která se chová podobně jako ta, která byla použita pro určení pozice lokátoru při jeho vzniku. Je ji však třeba dynamicky aktualizovat podle změn rotací kloubů. První a koncový bod je ke kloubům připevněn pomocí deformátoru Cluster, prostřední bod je přesunut do předem nastavené vzdálenosti od prostředního kloubu. Posun je možné provést opět v ploše určené třemi klouby řetězce pomocí funkce moveVertexAlongDirection. Na pozici takto vypočítaného bodu je možné lokátor přesunout.

Představené řešení nalezení pozice lokátoru má nevýhodu v tom, že vypočítaná pozice lokátoru nebere zřetel na jeho původní pozici. Skript programu Brig tak ještě toto řešení rozšiřuje o funkci, která lokátor přesune na nejbližší bod původní pozice lokátoru a plochy, kterou tvoří řetězec kloubů. Na pomocnou křivku je dynamicky navázána plocha, která se vytváří podle pozice bodů křivky. Na ploše je vytvořen objekt Follicle pomocí uživatelského skriptu, který se automaticky přesouvá po ploše dle pozice lokátoru. Pro přesun lokátoru je tak možné použít pozici objektu Follicle.

Dynamické přepínání řeší sada skriptů Sricpt Job. První z nich je spouštěn vždy, když nastane změna označení objektů scény. Tento skript vyšetří, který objekt je označen, a podle toho dokáže předat kontrolu nad řetězcem kostí označenému kontroleru. Následně spustí skript pro přesun neaktivních kontrolerů.

Skript pro přesun neaktivních kontrolerů je automaticky spouštěn pomocí dvou skriptů Skript Job vždy, když se změní hodnoty rotace některého z kloubů řetězce. Ten podle atributu přepínajícího mezi kinematikami ovlivní neaktivní kontrolery a pokud je zapnuté automatické klíčování a nějaký klíč již kontroler obsahuje, spustí skript pro vytváření klíčových snímků kontrolerů.

Skript pro vytváření klíčových snímků kontrolerů je třeba spouštět i ve chvíli, kdy uživatel vytvoří klíč ručně. Pro tento účel je u každého kontroleru vytvořen atribut `key_flag`. Při vytvoření klíčového snímku kontroleru je vytvořen klíčový snímek i tomuto atributu. Pokud se jedná o první klíčový snímek, vytvoří se spojení mezi novou animační křivkou a tímto atributem, což lze sledovat pomocí skriptu Skript Job, a tak spustit danou funkci. Ta vytvoří klíčové snímky všem kontrolerům řetězce a klíčový snímek atributu `key_flag` zruší, aby bylo stále možné sledovat další vytvářené klíčové snímky kontrolerů.

Všechny skripty mají nastavené chování i pro hodnoty, kdy skript interpoluje mezi ovládáním inverzní a dopředné kinematiky. Díky tomuto nastavení je možné udržovat hodnoty kontrolerů tak, aby byl vždy aktuální a aby při animaci nezáleželo na tom, která sada kontrolerů je aktivní. Prakticky však nelze docílit úplné shody, protože interpolace kontrolerů dopředné kinematiky neprobíhá stejně jako interpolace inverzní kinematiky. Proto je stále

třeba o atributu přepínání kinematiky vědět a při animaci s ním počítat. Provedení všech popsaných metod je implementováno v `IK_with_poleVector.mel`.

Ovládání chodila

Při využití inverzní kinematiky pro ovládání nohou se při přesunu jejich kontrolerů chodidlo otáčí společně s rotací kostí lýtka díky jejich vztahu v DAG struktuře. Chování chodidla, na kterém postava stojí, je však odlišné. Chodidlo nereaguje na pozici lýtka, je stále zarovnáno se zemí, při chůzi se ohýbá ještě úplně jiným způsobem, který ovlivňuje samotnou konfiguraci řetězce nohy.

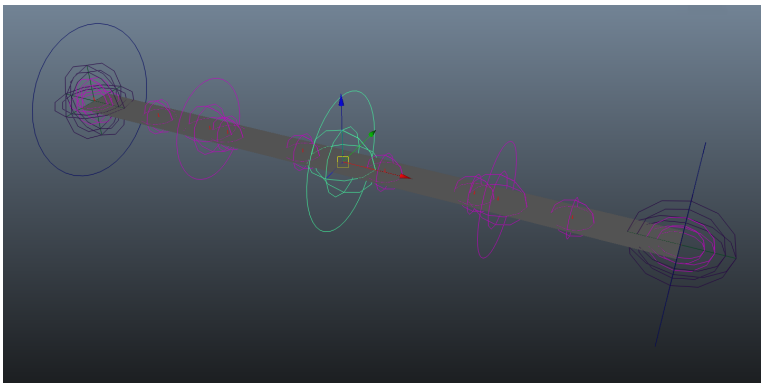
Skript uložený v `ikFootCreator.py` pracuje s kontrolerem řetězce inverzní kinematiky a klouby, které jsou skriptu předány v pevně stanoveném pořadí. Požadované pořadí je při použití verze skriptu pracující se selekcí možné nechat vypsat při spuštění kódu se špatným počtem označených objektů. Na počátku běhu skriptu je vytvořena pomocná kostra, která má opačně nastavené vazby DAG struktury oproti kostře vstupní. První kloub je umístěn na pozici paty, druhý se nachází ve špičce chodidla, třetí na pozici ohnutí prstů chodidla a čtvrtý je v místech kotníku. Tato struktura je propojena se vstupními klouby pomocí omezení Parent, kterým ovlivňuje pomocný kloub jemu předcházející kloub řetězce. Například pomocný kloub paty tak ovládá vstupní kloub špičky. Kloub kotníku je ponechán bez omezení, protože je již ovlivňován inverzní kinematikou. Místo něho je proto ovlivňován samotný kontroler inverzní kinematiky. Celý pomocný řetězec kloubů je ovládán pomocí nové křivky, která má díky popsané konfiguraci kontrolu i nad kontrolerem inverzní kinematiky. Základní vazba je zprostředkována pomocí omezení Parent, ovládání jednotlivých částí pomocné kostry pak ovlivňuje atribut Foot Roll. Skript s různými hodnotami tohoto atributu propojí pomocí funkce Driven Key hodnoty rotací pomocných kloubů. Výsledkem je tak kontroler, který animátorovi při animaci chůze pomůže ovlivňovat chodidlo i celou nohu.

Vytvoření systému Ribbon

Ribbon, česky stužka, nepatří k nejběžnějším rigovacím postupům, jeho aplikace však řeší nejrůznější problémy. Tento přístup, který byl představen na [Car13], se skládá ze tří a více úrovní kloubů. Vkládá se do hierarchie mezi dva klouby nějaké končetiny, nejběžněji při cartoon animaci, kdy se kosti pro zdůraznění pohybu nepřirozeně prohýbají. Druhé nejběžnější využití systému ribbon je aplikace pro ovládání tvaru rtů, ta však není prostřednictvím programu Brig přístupná.

Pozice kloubů vyšší úrovně ovlivní pozici všech kloubů nižší úrovně, v každé nižší úrovni je kloubů stejně nebo více než v té předchozí. Klouby první úrovně se prováží s původní hierarchií rigu, klouby poslední úrovně jsou použity ke skinningu. Klouby ostatních úrovní dávají animátorovi možnost zohýbat spojnici mezi klouby vyšší úrovně díky tomu, že jejich pozice ovlivňuje všechny klouby nižších úrovní, tedy i klouby pro skinning. Provázanost úrovní kloubů zajišťuje objekt NURBS plane, podle jehož tvaru je tato technika pojmenována. Tvar této stužky dané úrovně se řídí klouby té samé úrovně pomocí skinningu. Stužka typu NURBS plane je lineární s nejmenším počtem řídících vrcholů, který je určen počtem kloubů dané úrovně. Pro docílení nelineárního chování je stužka upravena funkcí Rebuild, kdy je původní stužka zachována, a tak je mezi stužkou lineární a stužkou vzniklou funkcí

Rebuild vytvořena závislost. Tato nově vzniklá stužka je kubického řádu, je tedy plynulejší. Na ní jsou vytvořeny objekty Follicle pomocí předem popsané funkce, které ovládají pozici kloubů nižší úrovně. Další úrovně celého systému opakují popsaný proces s využitím nových kloubů a stužek. Celá struktura je vidět na obrázku 4.7



Obrázek 4.7: Struktura systému Ribbon
Zdroj: vlastní

Vytvořená funkce souboru `ribbonCreator.mel` vytvoří systém Ribbon mezi dvěma označenými objekty. Jedná se však spíše o ukázku, protože hlavní funkce aplikace vyžaduje mnohem více parametrů než jen dva objekty, na které se nový systém Ribbon naváže. Parametrem této funkce jsou maximální a minimální poloměr nově vzniklých kloubů a kulovitých kontrolerů, který se podle těchto parametrů interpoluje mezi úrovněmi. Dalším parametrem funkce je pole, které definuje počet kloubů každé úrovně, a pole popisuje klouby, které se v běhu programu mají přeskočit a nevytvořit. Tak je možné upravit i výsledné rozestavení kontrolerů celého systému.

Je jasné, že funkce pracuje v cyklu, kdy postupuje od první úrovně níže. Klouby i řídící body stužky rozmisťuje lineární interpolací po spojnici dvou vstupních objektů a využívá popsanou proceduru pro vytvoření objektu Follice a pro přichycení kloubů nižší úrovně ke stužce. Klouby nejvyšší určí správnou orientaci kloubů, která se aplikuje na nové klouby všech úrovní. Díky tomu skript dokáže vytvořit systém Ribbon kdekoliv v prostoru pod jakýmkoliv sklonem. Kód využívá skript pro vytvoření kulových kontrolerů na všech ovládacích vrstvách systému Ribbon.

Funkci je možné pomocí dvou dalších parametrů spustit v režimu, kdy vznikne systém upravující měřítko kloubů nejnižší úrovně, tedy těch, které jsou používány pro skinning. Skript je nastaven tak, aby vznikal dojem zachování objemu končetiny při jejím natažení či deformaci. Funkce vytvoří křivku, která je řízena stužkou nejnižší úrovně. Při jejím vzniku je poznamenána délka křivky do uzlu multiplyDivide. Do něho je i zapojena aktuální délka křivky tak, že uzel tyto dvě hodnoty dělí. Tak je vypočítán faktor natažení, který je připojen do pomocného uzlu typu Unknown. Ten pro každý kloub nejnižší úrovně definuje konstantu pro upravení tohoto faktoru. Tyto konstanty je možné pomocí tří druhů klíčových slov parametru funkce interpolovat tak, aby se hodnoty výsledného faktoru zmenšovaly s přibližováním se buď k prvnímu objektu, ke druhému nebo s přiblížením ke středu. Tento výsledný faktor ovlivňuje přímou změnu pomocí uzlu Expression atributy ScaleY a ScaleZ

všech kloubů nejnižší úrovně. Popsanou funkcionalitu je možné prostřednictvím speciálních atributů uzlu Unknown vypnout nebo upravit tak, aby systém reagoval pouze na natažení končetiny a zanedbávaly se deformace, které vznikají ohýbáním systému Ribbon.

Anulování deformací skinningu

Při tvorbě kostry se všechny klouby mohou zdát správně umístěny, ovšem až v momentě, kdy se klouby začnou používat ke skinningu, může dojít k odhalení nepřesností. V takových případech je třeba s kloubem na správné místo pohnout tak, aby daný posun neovlivnil deformaci objektu, který je pomocí skinningu s kloubem spojený. K tomuto účelu Maya nabízí nástroj Move Skinned Joints Tool. Pokud je však kloub součástí již vytvořeného rigu, nemusí být vždy možné tento nástroj použít kvůli vytvořeným omezením. V takovém případě je možné buď danou část rigu zrušit, nástroj použít a rig vytvořit znovu. Druhou možností je použít skript, který změní matici uzlu, která výslednou deformaci způsobuje. Pro základní popis fungování Smooth Skin je třeba nejdříve vyjádřit předpis, který definuje váhu každé kosti pro daný vrchol:

$$\sum w_i = 1 \tag{4.1}$$

Kde w_i je váha kloubu indexu i. Předpis požaduje, aby součet všech vah kostí pro jeden vrchol byl vždy jedna. Výslednou deformaci vyjadřuje rovnice:

$$v' = \sum w_i * v * B_i * M_i \tag{4.2}$$

Daná rovnice označuje v jako souřadnice vrcholu objektu, v kterých se vrchol nacházel v době vytvoření Smooth Skin. Matice M_i popisuje aktuální světovou matici (World Matrix) kloubu indexu i. Tato matice vyjadřuje transformace kloubu ve světových souřadnicích. Matice B_i je inverzní světová matice (World Inverse Matrix) daného kloubu v době vytvoření Smooth Skin a v' značí výsledné souřadnice vrcholu.

V momentě, kdy je v Maye vytvořeno propojení kloubů a objektu pomocí funkce Smooth Skin, je pro každou kost v uzlu skinCluster, který řídí danou deformaci, vytvořen pro každý kloub záznam o jeho aktuální inverzní světové matici. Jak ukazuje rovnice 4.2, v době vytvoření spojení pomocí Smooth Skin je výsledek násobení matic B a W jednotková matice, proto nedochází k žádným deformacím.

Skript uložený v souboru `resetjoints.mel` pracuje s označenými klouby, s kterými provádí stejný proces, jako který probíhá při vzniku Smooth Skin. U daného kloubu nalezne objekty skinCluster, s kterými je propojen. V těchto objektech je vyhledán atribut bindPreMatrix pro příslušný kloub a následně je přepsán maticí worldInverseMatrix daného kloubu, což prováděné deformace způsobené daným kloubem anuluje.

Vytváření korekcí

Pro úspěšné vytvoření korekce deformovaného objektu pomocí Smooth Skin, je třeba být dobře seznámen s problematikou vrstvení deformátorů. Korekce je v Maye prováděna pomocí deformátoru Blend Shape, který každý vrchol deformovaného objektu porovnává s pozicí vrcholu vzorového objektu. Pokud by tedy byl Blend Shape umístěn jako poslední článek při deformování objektu, při jeho použití by byly všechny ostatní deformace přepsány vzorovým

objektem. Proto je třeba, aby byl naopak výsledek deformací Blend Shape deformátoru ovlivněn Smooth Skin. To však znamená, že vzorový objekt musí vznikat na modelu, který není Smooth Skin deformován. Tato situace je však velmi nepraktická, protože tvorba korekcí vyžaduje editaci nevhodně deformovaného místa v oblasti ohybu.

Základní řešení tohoto problému vytvořené v souboru `correctives.mel` pracuje s třetím deformátorem jménem Tweak. Ten je vytvořen automaticky při vzniku Smooth Skin a registruje uživatelsky vytvořený posun vrcholů deformovaného modelu. Uživatel tak může vytvořit korekci na samotném deformovaném modelu pomocí základních nástrojů, které pracují s transformacemi vrcholů. Tyto transformace jsou uloženy do deformátoru Tweak. Spuštění skriptu s označeným deformovaným objektem pak způsobí to, že je vyhledán nejdříve skinCluster objektu. Ten je dočasně vypnut. V této fázi vznikne duplikát objektu, který ovlivňuje pouze deformátor Tweak. Tento duplikát je použit jako vzorový objekt pro deformaci Blend Shape. V případě, že se jedná o první korekci, je deformátor Blend Shape nově vytvořen a umístěn tak, aby jeho deformace byla aplikována před deformací Smooth Skin. V opačném případě je duplikát přiřazen jako další vzorový objekt stávajícího deformátoru Blend Shape. Na závěr je vyhledán a vynulován uzel Tweak deformovaného objektu a Smooth Skin opět zprovozněn. Skript ještě spustí okno pro vytváření Driven Key s připraveným uzlem Blend Shape v kolonce Driven.

Celý popsaný proces je funkční, uživatelsky však není přívětivý. Deformátor Tweak, pomocí kterého se nastavují korekce, je umístěn tak, že se provádí před deformací Smooth Skin. Díky tomu manipulátory pro úpravu vrcholů nefungují předvídatelně, protože vlastně upravují vrcholy, které nebyly Smooth Skin ještě ovlivněny. To se projevuje tak, že pokud jsou například přesouvány vrcholy ohnutého kolena v ose x, díky následné deformaci pomocí Smooth Skin má každý z vrcholů osu x natočenou jiným směrem. Díky tomu je velmi náročné korekci vytvořit tak, jak je zamýšleno.

Brig tento problém řeší tak, že uzel Tweak uživateli dovoluje přesunout tím způsobem, aby byl prováděn až po deformaci Smooth Skin. To dovoluje uživateli vytvořit korekci pomocí manipulátorů, které již fungují ve světových souřadnicích. Tuto situaci však nelze vyřešit popsaným skriptem, protože Tweak obsahuje posun bodů po aplikaci Smooth Skin, vzniklý Blend Shape však tyto body upravuje před ní. Prakticky dojde k tomu, že upravené body jsou také deformovány Smooth Skin a vzniklý Blend Shape neodpovídá tvaru, který uživatel vytvořil.

Soubor `substractSkin.mel` obsahuje funkce pro odečtení deformací Smooth Skin. Díky tomu je možné z jakkoliv uživatelsky upraveného tvaru vypočítat tvar pro deformátor Blend Shape, který bude následně možné pomocí dané deformace Smooth Skin zpátky získat. K tomu je třeba vypočítat inverzní funkci aktuální deformace Smooth Skin. Pokud je deformace Smooth Skin vypočítána pomocí rovnice 4.2, její inverzní rovnice má pak znění:

$$v = v' * (\sum w_i * B_i * M_i)^{-1} \tag{4.3}$$

Použitím této rovnice je možné upravit každý vrchol deformovaného duplikátu objektu a tak získat požadovaný tvar. Zmíněný soubor obsahuje veškeré funkce pro práci s maticemi, protože rozhraní skriptu MEL pro takovéto úkony žádné metody nenabízí.

Toto řešení umožňuje použít i nástroj Sculpt Geometry Tool, který nepracuje s deformátorem Tweak, ale upravuje vrcholy objektu přímo. Před započetím vytvářením correctives je

však vytvořen duplikát, který uchovává informace o tom, jak má nijak neupravený objekt vypadat. Tato informace je pak po vytvoření korekce posunuta a všechny body deformovaného objektu jsou pomocí duplikátu vráceny na své původní místo.

4.2.4 Struktura a instalace

Všechny popsané skripty se nachází ve složce brig. Ta navíc obsahuje soubor `__init__.py`, který ji dovolí načíst příkazem `import` jako balík (package) jazyka Python. Externí grafika je umístěna v podsložce img. Druhá podsložka je nazvaná skeletons. Ta v základní instalaci programu obsahuje jeden soubor, který uchovává data pro vytvoření výchozí kostry skriptem. Tato složka je programem Brig při načítání i ukládání kostry vždy použita jako startovní. Předpokládá se tedy, že uživatel do ní bude ukládat i další soubory s kostrami. Po nahrání složky brig do některého umístění, které Maya kontroluje při načítání skriptů, je možné program Brig spustit kódem jazyka Python:

```
import brig
brigWin = brig.BRigMainWindow()
brigWin.create()
```

Jediný skript, který není možné spustit z grafického rozhraní programu Brig, je skript pro získání příkazu k vytvoření křivky. Po označení křivky je možné zadat kód jazyka Python:

```
import brig
brig.getCurveToStringSel()
```

Kapitola 5

Zhodnocení a výsledky

5.1 Vstupy programu Maya

5.1.1 Možnosti jazyků

Program Maya nabízí mnoho možností pro tvorbu uživatelského rozhraní. Lze tak docílit jak grafického ovládání pomocí výchozích komponent, tak je možné použít i vlastní grafické prvky. Velkou výhodou je i to, že lze na existenci okna navázat existenci skriptu Script Job. Díky tomu není třeba takové skripty evidovat a spravovat je. V programu Brig se dokonce podařilo docílit dynamicky měnícího se prostředí, které dokáže reagovat na změny stavu konkrétních prvků. Lze tedy říci, že tvorba uživatelského prostředí Mayi je nesvazující.

Autodesk Maya dává skriptům možnost pracovat s rozhraním pro načítání a ukládání souborů. K lokalizaci souboru ve složkové struktuře počítače nabízí vytvoření okna s nastavitelnými parametry, jako jsou: volba výchozí složky při otevření okna, filtr souborů, seznam filtrů souborů s uživatelsky nastavitelným popisem. Výsledný dojem s prací se soubory je díky tomu stejný, jako při používání běžného programu.

Nízká rychlost skriptů, které pracují s velkým množstvím dat, je u obou interpretovaných jazyků MEL a Python opravdu znatelná. Velkou výhodou je proto alternativa v podobě tvorby zásuvných modulů, jejichž rychlost práce není omezována dynamickou interpretací. Zároveň je třeba zmínit, že Maya nenabízí možnost výpočet uživatelsky zastavit, pokud ji programátor přímo sám neimplementuje. Při vstupu do nekonečné smyčky se musí proces systémově ukončit.

5.1.2 Porovnání jazyků MEL a Python

Pro větší programové struktury je jednoznačně vhodnější používat jazyk Python. Mimo to, že podporuje objektově orientované programování, nabízí možnost vytvářet balíky (packages), díky kterým se mnohem lépe předchází konfliktům při používání funkcí. Autorovi přijde vhodnější i syntaxe využívající odsazování. Kód se tak stává přehlednější a tvorba odsazení je přitom snadnější než použití složených závorek.

Použití balíku funkcí PyMEL je pro objektově orientované programování opravdu výhodou. Psaní takového kódu je snadnější a intuitivnější. Zorientovat se však v metodách, které

lze volat prostřednictvím samotných PyMEL objektů, je místy obtížné. Nakonec mnoho funkcí pro získání dat objektu pracuje se samotným balíkem PyMEL namísto dotazování vyšetřovaného objektu.

Skript vytvořený v jazyku MEL je jednoznačně výhodný při spouštění jednořádkového kódu, který využívá základní příkazy. Při jeho tvorbě není třeba nejdříve importovat balíky pro komunikaci s Mayou. Při načítání externích skriptů jazyka MEL se nemusí kontrolovat, zda byla vážně načtena poslední verze skriptu. Nastavit opakované načítání souborů v jazyce Python může být pro programátora, který se příliš v jazyce Python neorientuje, nečekanou překážkou.

Předností, kterou je třeba zdůraznit, je propojení obou interpretovaných jazyků. Oba jazyky je možné pomocí speciálních příkazů plnohodnotně používat. Pro předávání dat je možné použít čtení návratových hodnot. V případě předávání parametrů funkci jiného jazyka je nutné sčítat textové řetězce tak, aby jazyk dokázal přeložit výsledný text. Při používání textových řetězců jako parametrů funkcí je třeba věnovat zvýšenou pozornost na správné používání uvozovek a escape znaků.

5.1.3 Pracovní workflow

Textový editor Mayi pro vytváření skriptů není pro tvorbu delšího kódu vhodný, a tak je praktické využít nějaký externí editor. Ten může vytvářet skripty ve složce, kterou Maya při importu skriptů prohledává. Díky tomu je možné skripty spouštět několikařádkovými příkazy buď v textovém editoru Mayi, či pomocí tlačítka prostředí Shelf.

Alternativou tohoto řešení je vytvoření přímé komunikace mezi Mayou a externím textovým editorem. Jak popisuje [Aut15b], k tomuto účelu lze použít program Wing od firmy Wingware. Ten dovoluje vytvořit položku v základním kontextovém menu, pomocí které je označený text odeslán na nastavený port. Mayu je možné nastavit tak, aby přijímala vstupy z tohoto portu a automaticky je spouštěla. Díky tomuto propojení je ještě jednodušší externí textový editor používat.

5.2 Výsledky testování autorigu Brig

5.2.1 Testování autorigu Brig

Autorig Brig byl k otestování předložen Gerardu Sebastianu Verronovi, který se po dobu devíti let živí jako vedoucí technického oddělení v různých filmových studií. Gerardo Verrone byl po krátkém seznámení s rozložením funkcí programu schopen tyto funkce používat a vytvořit animační rig. Zároveň rozeznal ty nejobtížnější úlohy, které autorig Brig ření. Celý jeho názor na vzniklý program je možné prohlédnout si v příloze B. Vzhledem k tomu, že je tento tester argentické národnosti, tato příloha je psána v anglickém jazyce.

5.2.2 Možnosti rozšíření

Neexistuje autorig, který by dokázal vyřešit libovolný problém, který mu je zadán. Proto animační studia svůj systém pro tvorbu rigu stále aktualizují a přidávají mu nové funkce. Autorig Brig uživateli nabízí řešení těch problémů, které jsou pro ruční řešení nemožné či

nejnáročnější. Aby bylo možné použít autorig pro vznik rigu celého, je třeba implementovat další skripty, které se postarají o vznik chybějících kontrolerů postavy.

Skript pro tvorbu korekcí není optimalizovaný. Maticové výpočty probíhají přímo v interpretovaném jazyku, což celý běh programu velmi zpomaluje. Nejvhodnějším řešením problému by tak bylo použití zásuvného modulu pro maticové výpočty.

5.3 Výsledky testování vygenerovaného rigu

Autorig Brig byl použití k vytvoření rigu testovací postavy lupiče. Tu lze pomocí vzniklých kontrolerů snadno ovládat a používat ji k animaci s nadsazováním. Nohy i ruce postavy se mohou natahovat a kroutit. Při tom je zachován objem objektu, jak radí představené animační zásady. Grafický přehled póz, které je snadné díky vzniklému rigu docílit, obsahuje příloha **??**.

Vzhledem k tomu, že byly implementovány skripty řešící nejběžnější problémy jako pohyb rukou a nohou, či vznik korekcí, vzniklé nástroje je možné prohlásit za velmi univerzální. Samotný Brig nenabízí možnost rigování vícerukých či vícenohých postav, je však možné ho k tomuto účelu využít. Stačí pro jednu postavu vytvořit více rigových systémů.

Funkce autorigu pro přejmenování jednotlivých kostí dle názvosloví programu Motion Builder byla také otestována. Program importovanou animační kostru bez problému přijal a dále s ní pracoval. Bylo tak možné přesně následovat postup, který je popsán v příloze A, a postavu zloděje rozhýbat pomocí dat z motion capture.

Kapitola 6

Závěr

Tato práce popisuje širokou problematiku čtenáři, který je sice znalý některých pojmů počítačové grafiky a programování, ale který o samotném riggingu mnoho neví. Text tak vytváří informační základnu, na které je možné stavět, poměrně obsáhle popisuje fungování programu Autodesk Maya, ve kterém je rig vytvářen a rigovací skripty spouštěny, V implementační části předkládá konkrétní problémy a jejich řešení. Ty jsou voleny tak, aby byly co nejrůznorodější a aby jejich řešení byla aplikovatelná na co nejvíce odlišných problémů. Z textu by mělo být čtenáři jasné, co rigging je a jakým způsobem je ho možné provozovat. Pro čtenáře, který se v problematice orientuje, by práce měla poskytovat materiál pro rozšíření a ukotvení znalostí programu Autodesk Maya. Představené méně obvyklé rigovací techniky mohou být také zajímavým zdrojem informací.

Autodesk Maya nabízí dva interpretované programovací jazyky, MEL a Python, pomocí kterých je možné s programem pracovat. Praktická část této práce s oběma jazyky pracuje, a tak je možné diskutovat jejich výhody a zápory. Díky tomu lze konkrétně popsat, jak oba komunikují s programem Maya, jakou logickou strukturu dokáží vytvořit a jakým způsobem komunikují mezi sebou. Protože program, který byl v implementační části vytvořen, používá uživatelské rozhraní, může tato práce zhodnotit i to, co Autodesk Maya při stavbě rozhraní umožňuje a k jakým úkonům skripty opravňuje. S tvorbou uživatelského rozhraní je spojeno i ukládání uživatelských dat přímo do scény programu. Práce tuto méně obvyklou úlohu též rozebírá a předkládá její jasné řešení pro co možná nejrůznější případy.

Je samozřejmostí, že bylo provedeno testování popisovaných postupů na konkrétních případech, díky čemuž lze prohlásit předkládaná řešení za úspěšná a univerzální. Pro otestování rigovacích postupů byl použit model zloděje, který vznikl pro účely této práce. Tomuto modelu byly pomocí vzniklých skriptů vytvořeny kontrolery. Jejich prostřednictvím bylo následně možné postavu ovládat přesně tím způsobem, který vzniklé kontrolery dovolují. Testování autorigu provedl také profesionální rigger Gerardo Verrone, který byl schopný autorig úspěšně použít. Jeho kladné hodnocení se nachází v příloze B.

Možností navázání na výsledky práce je mnoho. Program, který byl při tvorbě této práce vyvinut, lze rozšiřovat o další funkcionality s využitím již vytvořených datových struktur a grafického prostředí. Vhodné rozšíření by bylo například vyšetření obličejové animace, která není v práci téměř vůbec zmíněna. Zajímavou úlohou jsou také pokročilé techniky pro spouštění korekcí.

Práce obsahuje v příloze A kapitolu, která průřezově popisuje práci se systémem motion capture. Tato část je zpracována co nejpodrobněji a nejpopisněji, aby bylo možné ji následovat krok po kroku. Lze ji označit za návod, který neznalému uživateli zprostředkovává relevantní funkce potřebných programů a vysvětluje, jak konkrétní úkony přesně provést. Tento návod popisuje celý proces od nastavení systému pro záznam pohybu, přes získávání a čištění dat až po jejich aplikaci na obecný model.

Literatura

[Aut07] Autodesk. Querying the scene graph, 2007. http://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/Maya-SDK/files/GUID-0B85C721-C3C6-47D7-9D85-4F27B787ABB6-htm.html, stav z 5. 1. 2015.

[Aut09] Autodesk. Dependency graph (dg) nodes, 2009. http://download.autodesk.com/us/maya/2009help/index.html?url=Dependency_graph_plugins_Dependency_Graph_DG_nodes.htm,topicNumber=d0e641207Asts-Object-space-world-space-and-tangent-space-htm.html//, stav z 5. 1. 2015.

[Aut10a] Autodesk. Script, 2010. http://download.autodesk.com/us/maya/2011help/Nodes/script.html/, stav z 5. 1. 2015.

[Aut10b] Autodesk. What are constraints?, 2010. http://download.autodesk.com/us/maya/2010help/index.html?url=CSCo_Types_of_constraints.htm,topicNumber=d0e324613/, stav z 5. 1. 2015.

[Aut15a] Autodesk. How often an expression executes, 2015. http://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Maya/files/GUID-9E3669B0-43B6-4B93-9494-C20439B87FD0-htm.html, stav z 5. 1. 2015.

[Aut15b] Autodesk. How often an expression executes, 2015. http://mayamel.tiddlyspot.com/#[[How%20can%20I%20have%20Wing%20send%20Python%20or%20mel%20code%20to%20Maya%3F]] , stav z 5. 1. 2015.

[Aut15c] Autodesk. Move tool, 2015. http://knowledge.autodesk.com/support/maya-lt/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/MayaLT/files/Basics-Tools-Move-Tool-htm.html, stav z 5. 1. 2015.

[Aut15d] Autodesk. Object space, world space and tangent space, 2015. http://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Maya/files/

Asts-Object-space-world-space-and-tangent-space-htm.html//, stav z 5. 1. 2015.

[Ber13] Roman Berka. Cloth modeling and animation - prezentace, 2013.

[Bra12] Dustin D. Brand. Human eye frames per second, 2012. http://amo.net/NT/02-21-01FPS.html/, stav z 5. 1. 2015.

[Bur10] Owen Burgess. The low-down on the python pyc files, 2010. http://mayastation.typepad.com/maya-station/python/page/3/, stav z 5. 1. 2015.

[Car13] Josh Carey. Anomalia - advanced rigging course, 2013.

[JS10] Jiří Žára Petr Felkel Jiří Sochor Bedřich Beneš. *Moderní počítačová grafika*. Computer Press, 2010.

[Kel10] Eric Keller. *Mastering Autodesk Maya 2011*. Wiley Publishing, 2010.

[KR05] Karim Biri Kiaran Ritchie, Jake Callery. *The Art of Rigging, Volume 1*. CG Toolkit, 2005.

[KR06] Karim Biri Kiaran Ritchie, Jake Callery. *The Art of Rigging, Volume 2*. CG Toolkit, 2006.

[LG02] Jim Lammers and Lee Gooding. *Maya 4 - Kompletní průvodce*. SoftPress, 2002.

[LR12] Eric Luhta and Kenny Roy. *How to cheat in Maya*. Elsevier, 2012.

[Mog07] Cory Mogk. Maya 2008 features detailed, 2007. http://area.autodesk.com/blogs/cory/maya_2008_features_detailed/, stav z 5. 1. 2015.

[Mot12] META Motion. Acceleglove product details, 2012. http://www.metamotion.com/hardware/acceleglove_productdetails.html/, stav z 5. 1. 2015.

[MT12] Adam Mechtey and Ryan Trowbridge. *Maya Python for Games and Film*. Elsevier, 2012.

[Nog11] Pedro Nogueira. Motion capture fundamentals. *Faculdade de Engenharia da Universidade do Porto*, 2011.

[PE02] Joseph C. Hager Paul Ekman, Wallace V. Friesen. *Facial Action Coding System*. A Human Face, 2002.

[RM00] Paul Read and Mark-Paul Mayer. *Restoration of Motion Picture Film*. Butterworth-Heinemann, 2000.

[Tin12] Brian Tindall. The art of moving points, 2012. http://hippydrome.com/, stav z 5. 1. 2015.

[TJ81] Frank Thomas and Ollie Johnston. *The Illusion of Life*. Walt Disney Production, 1981.

[tox08] toxik. Maya history, 2008. `http://www.toxik.sk/maya-startup-window-history/`, stav z 5. 1. 2015.

[Wik15] Wikinews. Autodesk to buy alias, 2015. `http://en.wikinews.org/wiki/Autodesk_to_buy_Alias/`, stav z 5. 1. 2015.

[YS00] Yamaguchi M. Arai K.-I. Takahashi K. Itagaki A. Wako N. Yabukami S., Kikuchi H. Motion capture system of magnetic markers using three-axial magnetic field sensor. *IEEE*, 2000.

Příloha A

Návod k použití a aplikaci dat motion capture

A.1 Sběr dat

A.1.1 Popis školního systému

Fakulta elektrotechnická na ČVUT v Praze disponuje systémem motion caputre od firmy OptiTrack. Záznamové zařízení i s nainstalovaným programem Motive se nachází v budově E na Karlově náměstí v laboratoři virtuální reality (VRlab). Jde o poměrně malou místnost o přibližných rozměrech, kde šest kamer umístěných ve stejné výšce obklopuje obdélníkový prostor ze tří stran. Prostor, který dokáží zaznamenávat lze přibližně popsat rozměry 2 x 2,3 metru. Kamery jsou na pohyblivých ramenech, dají se nasměrovat pro co nejoptimálnější pokrytí scény zornými poli. Systém motion capture je připraven pro práci v kombinaci se stereoskopickým plátnem tak, aby spolu tvořily interaktivní rozhraní, které bude schopno reagovat na pozici pozorovatele. Nicméně nic nebrání tomu, použít toto zařízení k záznamu pohybu. K jeho zpřesnění je možné dvě z kamer umístit na stativy na takové místo, aby byl prostor pokryt ze všech čtyř stran a aby byla scéna snímána z různé výšky.

Kamery OptiTrack typu s250e jsou zapojeny do modemu, do kterého posílají data pomocí ethernetového kabelu, který je i napájí. Každá z kamer obraz už sama předzpracovává, aby se snížil počet dat při posílání vysokofrekvenčního záznamu. Kamery mají vlastní osvětlení kolem objektivů a dokonce i display na zobrazení čísla kamery, které jim ovládací systém přiřadil. Pro záznam pomocí infračerveného spektra je vhodné co nejvíce potemnit záznamovou místnost, aby se zabránilo rušivým odrazům jiných objektů, než jsou markery na actorovi.

Práce v programu Motive je rozdělena do čtyř základních kroku, kterým odpovídají nabízené layouty, tedy funkce a rozložení oken v programu. V hlavním menu pod položkou Layout se dají přepínat, stejně tak jako pomocí klávesových zkratek ctrl + 1, 2, 3 nebo 4. Následující podkapitoly dodržují toto rozložení. V těchto částech bude vždy nejdříve popsáno prostředí s nejdůležitějšími funkcemi a pak bude popsáno krok za krokem, jak postupovat pro zaznamenání a zpracování pohybů actora. Tato část bude obsahovat i několik připomínek a rad ohledně programu Motive, které se vztahují konkrétně k verzi 1.5.0 64-bit.

A.1.2 Kalibrace kamer

První layout zvaným Camera calibration je vhodný pro nastavení pohledu kamer, režimu nahrávání a pro odfiltrování nežádoucích odrazů, které může systém zaměnit s markery. Okno Camera Preview zobrazuje přehled toho, co právě připojené kamery snímají. Horní menu tohoto okna nabízí nástroje pro práci s kamerovým výstupem. První sada tlačítek zleva umožňuje změnu pohledu, z nichž to první přepne okno do Perspective view. Okno s tímto pohledem se však v tomto prvním layoutu již nachází. Další tlačítko v menu Camera Preview přiblíží pohled na označený výstup kamery; zároveň pokud je pohled již přiblížen, toto tlačítko pohled zase oddálí. Třetí tlačítko zleva umožňuje procentuální přiblížení na výstup kamer. Pro změnu přiblížení funguje také kolečko myši.

Další sada tlačítek nacházející se za oddělující svislou čárou dovoluje vytvářet masky tak, že označené místo dané kamery bude při trackovacím procesu ignorováno. První tlačítko vytvoří masku automaticky, to druhé vytvořené masky zruší. Další tři tlačítka umožňují uživateli vytvořit masku pomocí kruhu, obdélníku a pomocí tahu myši. Zmíněné funkce pracují s těmi kamerami, které jsou označeny. Označení kamer je možné provést v seznamu v okně Cameras nebo označením přímo výstupu kamer v okně Camera Preview.

Poslední sada dvou tlačítek přidává do výstupu kamer grafické informace. První nabídka dokáže přidat síť, zaměřovací kříže zaznamenaných bodů a lze také zobrazit obdélníkový obrys objektů. Poslední tlačítko přidává informace k obrazu pomocí popisků. Pracuje s informacemi jak kamer, tak trackovaných bodů, které jsou ve výstupu kamer zaznamenány.

Nastavení kamer je možné upravit v okně Cameras a jeho podokně Properties. Okno Cameras nabízí seznam používaných kamer. Ten se dá v menu tohoto okna upravovat tlačítky pro přidání a ubrání kamery. Atributy u každé z kamer nabízí možnost nastavit rychlost snímání scény za sekundu (FPS), expozici (EXP), hranici pro rozpoznání objektu (THR) a sílu světla kamery (LED). Hodnoty je možné upravit jak kamerám jednotlivě, tak při výběru celé skupiny je možné je měnit plošně. V nabídce Properties je možné nastavit filtr, tedy zda kamera bude pracovat v infračerveném spektru či v režimu viditelného spektra. Video Type nastaví, zda kamery posílají nezpracovaný záznam, nebo hledají ve scéně trackovací body. Horní menu okna Cameras nabízí několik presetů vhodných při různých fázích kalibrace. Tento preset upraví všechna zmíněná nastavení tak, aby kamery pracovaly co nejvhodněji pro zvolenou činnost.

Nastavení pohledu kamery

Pokud kamery nemíří na scénu ideálně, je třeba kamery natočit, přitom nezáleží na natočení kamer kolem jejich pohledové osy. Výstup tedy klidně může být třeba vzhůru nohama. Kamery by měly svým pohledem pokrýt co největší prostor tak, aby vždy alespoň dvě z kamer byly schopny vidět marker umístěný ve scéně. Pozice a rotace kamer tak definuje prostor, na kterém budou actorovy pohyby zaznamenávány.

Při otáčení a umisťování kamer je vhodné zobrazit jejich výstup pomocí presetu Aiming. Preset pracuje pouze s označenými kamerami, je tedy vhodné označit celou skupinu kamer. Při volbě pohledu každé kamery je dobré její výstup v programu přiblížit a na monitoru sledovat, k tomu účelu slouží ikona bílého okna pod hlavním menu programu.

Kalibrace scény

Přestože program po nasbíraní dat sám vyzve k jejich uložení do souboru, samotný projekt, který tyto data sdružuje, po celou dobu práce uložen být nemusí. Proto je vhodné z důvodů organizace souborů již v této fázi projekt uložit a vznikající soubory ukládat do stejné složky.

Při přepnutí kamer do presetu Aiming jsou vidět body, které systém interpretuje jako markery. V tuto chvíli by se na základě výstupu kamer měly skrýt všechny takto rozpoznané objekty. Pokud jejich skrytí není možné, měly by být v programu zakryty maskou. Záložka Calibration v okně Camera Calibration nabízí automatické vytvoření masky, jedná se o stejnou funkci jako v okně Camera Preview.

Okno Camera Calibration uživatele dále provede celý procesem. Jeho první krok se spustí tlačítkem Start wanding. Nastavení OptiWand 500 mm odpovídá nástroji, kterým škola disponuje. Jedná se o tyč, která má na konci napříč připevněnou půlmetrovou lištu se třemi markery. Vzhledem k tomu, že systém zná proporce této tyče, při pohledu na ni v různých místech scény dokáže určit vzájemnou pozici všech používaných kamer. V této chvíli by tedy měl operátor v prostoru scény s OptiWand pohybovat. Systém tak bude průběžně sbírat údaje o scéně. V podokně Calibration Engine je při tomto procesu průběžně zobrazována kvalita vzorků. Sbírání je třeba ručně přerušit příkazem pro spuštění výpočtů. Při tomto procesu je kvalita popisována pomocí výrazů Low, Medium, High, Very High, odchylka v jednotkách pixelu a kvalita kalibrace Good, Great, Excellent, Exceptional. Pro získání kvalitních výsledků je třeba počkat několik minut. Poté je třeba kalkulace přerušit ručně. To, že data jsou již dostatečně přesná indikuje zezelenání pozadí a hláška Ready to Apply, nicméně k lepším výsledkům je vohdné výpočet nechat pracovat déle. Se školním systémem není složitá docílit nejvyššího slovního ohodnocení i všech položek a pixelovou odchylku jak kamer, tak metody (Ave) držet kolem jedné desetiny. Po stisknutí tlačítka Apply Result je vyvoláno okno s údaji o měření. Poté je možné data jménem Wanding Timeline uložit do souboru s příponou tim. Jeho načtení a opětovné použití se však při testování programu nezdařilo.

Dalším krokem je volba os scény, aby systém věděl, kde se nachází podlaha a kam mají směřovat ostatní dvě osy. Okno Camera Calibration toto umožňuje v záložce Ground plane. V tomto kroku by měl operátor umístit kalibrační pomůcku zvanou Calibration Square do scény a vhodně ji natočit pro určení směru os x a z. Provedení tohoto nástroje od firmy OptiTrack obsahuje dvě vodováhy, které je třeba zakrýt, protože jejich kovová konstrukce odráží světlo jako markery, a tak systém mate. Po stisknutí tlačítka Set Ground Plane se rozmístění kamer v Perspective View ihned přenastaví a celá kalibrace je dokončena. Program ihned uživatele vyzve k uložení kalibrace do souboru s příponou cal, který je možné do příštího projektu opět načíst pomocí tlačítka Open v hlavním menu. Samozřejmě je zde podmínka, že se s kamerami od poslední kalibarce vůbec nepohnulo.

A.1.3 Create

Layout jménem Create nabízí pouze možnost vytvořit rigid bodies, objekty skládající se z více markerů, které jsou navzájem vůči sobě nepohyblivé. Dají se vytvořit dvěma způsoby, v obou případech se však objekty musí nacházet ve scéně, aby je kamery viděly. Tlačítko Create From Visible předpokládá, že pouze jeden rigid body je systémem pozorován. Operátor tedy musí postupně rigid bodies objekty do scény nosit. Přinést do scény všechny rigid

bodies najednou umožňuje druhé tlačítko Create From Selection, kdy operátor jeden po druhém souboru markerů v perspektivnim pohledu označuje, a tak rigid bodies vytváří. Rigid body je v perspektivním pohledu prezentováno zástupným markerem, který se nachází v těžišti bodů tohoto objektu. Po označení rigid body lze měnit jeho nastavení, jako ho přejmenovat, nastavit Defection. To v milimetrech určí odchylku jednoho markeru, která již další trackování rigid body přeruší. Položka smoothing zas vyhladí výsledný pohyb rigid body, odstraní tedy šum.

A.1.4 Capture

Možnost zaznamenávat pohyby umožňuje layout Capture. Svou nabídkou je velmi podobný poslednímu layoutu, podrobnější popis prostředí obsahuje následující kapitola. V této fázi je totiž nejdůležitější tlačítko se znakem pro spuštění záznamu, tedy červený kruh na bílém kruhu. Při jeho stisknutí systém začne zaznamenávat pohyby markerů ve scéně. Záznam se ukončí stejným tlačítkem, kterým se spustil, automaticky se uloží do složky s názvem Session a datem nahrání. Tento soubor má koncovku tak. Vytvořený záznam je možné pak na časové ose procházet nebo přehrát při přepnutí okna Timeline z Live na Edit. Toto nastavení je však jediným rozdílem mezi tímto a layoutem Edit.

A.1.5 Edit

Layout Edit nabízí ve své levé části okno Project. V něm se nachází menu, pomocí kterého lze vytvořit objekt Rigid Body a skupinu zvanou Marker Set. Ta sdružuje skupinu markerů, které se budou vždy pohybovat shodně s pohybem daného kloubu. Vznikne tak Marker Set například pro předloktí, pro zápěstí a podobně. Data v okně Project jsou rozdělena do dvou částí, Assets a Takes. Assets nabízí seznam všech objektů Marker Set a Rigid Bodies, které byly použity ve všech projektech, které byly od spuštění programu Motive otevřeny. Marker Set se tedy neukládá do projektu, nýbrž do záběrových souborů. Pro každý Marker Set je třeba vytvořit konkrétní markery, které bude obsahovat. Když je Marker Set označen, je ve spodní části okna Project záložka Marker Set zobrazující jednotlivé markery. Zde se kliknutím do prostoru textového pole dají nové Markery vytvářet. Pro indikaci toho, zda je Marker Set a Rigid Body aktuálnímu záběru přiřazeny, je použito zaškrtávací pole a ztučnění jména.

Nabídka Takes v horní části okna Project obsahuje složky, které byly do projektu načteny. Ty ve svém jméně obvykle obsahují slovo Session. Uvnitř těchto složek je možné nalézt soubory s příponou tak, které obsahují data se záznamem pohybu. Při dokončení nahrávání pohybů jsou tento soubor a případně i celá složka vytvořeny a přidány do tohoto seznamu. Všechny vytvořené skupiny Marker Set a Rigid Bodies je třeba nejdříve do aktuálního záběru přidat. Při kliknutí pravým tlačítkem na danou skupinu se zobrazí nabídka. V té je třeba zvolit možnost „Add Asset to Current Take“.

Pro zahájení další práce je třeba nasbíraná data zpracovat. To lze provést pomocí tlačítka Trajectorize, které se nachází v menu okna Project. V této chvíli se nasnímaným markerům přiřadí rozeznaná Rigid Bodies. Ostatní markery je třeba označit a pomocí programu rozeznat je v každém časovém úseku zaznamenaného záběru. K tomu je třeba přepnout spodní část okna Project do režimu Labeling View. To lze provést pomocí pravého tlačítka, kterým

je třeba kliknout do prázdného místa prostoru seznamu záložky Marker Set. V levé části tohoto zobrazení se tak vypíše seznam skupin Rigid Bodies a jednotlivých markerů skupin Marker Set, které byly záběru přiřazeny. Pravá část pak obsahuje nerozeznané markery záběru. V tomto zobrazení je používaná ikonka s písmenem G, která značí, že marker má někde v časové ose takzvanou trhlinu (gap), které je třeba věnovat pozornost. Takový marker se tedy nachází jen v některých částech časové osy záběru. Dalším indikátorem je světlost nápisu jména markeru, která značí to, zda daný marker obsahuje nějaká data o pohybu v prostoru.

Následující postup vyžaduje, aby byly neidentifikované markery ze seznamu, který se nachází v pravé části Labeling View, uživatelem označeny jako některý marker ze seznamu, který se nachází v levé části tohoto zobrazení. Toto přiřazení je třeba provádět technikou Drag and Drop, kdy je pomocí držení levého tlačítka myši neidentifikovaný marker přesunut nad marker z levého seznamu. Po uvolnění levého tlačítka je marker přiřazen. Je třeba říci, že program nabízí i jiné způsoby, jak markery přiřazovat, ty se však v testované verzi programu nechovají správně. K identifikaci a označení nerozeznaného bodu je možné využít okno Perspective View, v kterém jsou markery vidět ve vzájemném kontextu. Okno Perspective View zobrazuje jeden konkrétní úsek časové osy, proto se v něm nemusí vyskytovat všechny nepřiřazené markery seznamu. Toto zobrazení zároveň nabízí způsob, jak jednotlivé markery rozeznat. Ty markery, které jsou zobrazeny jako bílé koule, jsou již identifikovány. Oranžové koule jsou markery ze seznamu těch neidentifikovaných. Tímto způsobem je třeba všechny neidentifikované markery přiřadit těm identifikovaným. Při tomto procesu se některé markery spojí do jednoho. Pokud se v záběru vyskytují markery, které jsou ve skutečnosti jen chybou systému, je třeba tyto objekty označit a klávesou delete smazat.

Pro nalezení každého markeru je možné využít časovou osu, která jeho výskyt v čase zobrazuje. V té je poměrně neintuitivní změna v zobrazení. Aktuální čas se mění při kliknutí do dolní části časové osy, v které jsou čísla popisující horizontální osu. Pokud se myší klikne přímo do grafu, dojde k označení markeru. Fungování kolečka myši má dva režimy dle kontextu (tedy označení) různých částí okna Timeline. Pokud kontext odpovídá oknu Timeline, ale neodpovídá samotnému grafu, dochází k posunu zobrazeného výseku časové osy. V případě, že je kontext v grafu samotném, dochází k přiblížení či oddálení pohledu na časovou osu, kdy je střed změny měřítka určen pozicí myši.

Po identifikování všech bodů je pravděpodobné, že animační křivka některých bodů obsahuje stále mezery či jiné nesrovnalosti. Pro práci s animační křivkou program Motive nabízí sadu nástrojl. Ty se nachází v okně Edit Tools, které lze nalézt vpravo od časové osy. V základním rozložení je poměrně malé a může být třeba nezobrazené záložky odkrýt pomocí šipek či zvětšení okna. Záložky tohoto okna jsou: Trim Tails, Gap Fill, Smooth a Swap Fix. Úvodem je ještě třeba zmínit, že při těchto operacích v testované verzi programu nefunguje funkce krok zpět.

Funkce Trim Tails automaticky zkracuje animační křivku kolem trhlin, jinak řečeno, všechny trhliny v zaznamenaných datech zvětšuje. Data mohou být těsně kolem trhlin v animační křivce zkreslená, proto je třeba mít nástroj, které tyto nepřesnosti dovoluje odstranit. Je možné i ruční odstranění dat. Pomocí tahu myši se zamáčknutým levým tlačítkem je možné data označit a data pomocí klávesy delete smazat. Při tomto úkonu je třeba dbát na to, aby byl kontext zaměřen přímo na časovou osu. Pokud tomu tak není, klávesa delete smaže celý marker, s kterým uživatel pracuje.

Gap Fill díry v datech zacelí pomocí nastavené interpolace. Tento nástroj nabízí tlačítka pro snadný pohyb po časové ose, kdy aktuální snímek přesune k výpadku v datech. Typy nabízených interpolací jsou: konstantní, lineární či kubická interpolace. Poslední možnost nabízí interpolaci založenou na vzorku pohybu jiného markeru. Druhý označený marker slouží jako předloha pohybu markeru, který byl označen jako první.

Záložka Smooth umožňuje odstranit ze záznamu šum, kdy podle nastavené maximální frekvence vibrací provede funkci, která hodnoty bodů vyhladí. Čím nižší je tedy hodnota tohoto atributu, tím je vyhlazení silnější.

Jen zřídka kdy se stane, že jsou data jednoho markeru obsažena v tom druhém. Pokud tato situace nastane, je možné použít nástroj Swap Fix. Pomocí tahu myši se zamáčknutým levým tlačítkem je možné označit problém. Následně je třeba označit se shiftem marker, s kterým je třeba zaměnit data, a stisknout tlačítko s popiskem Apply swap.

A.1.6 Ukládání a export

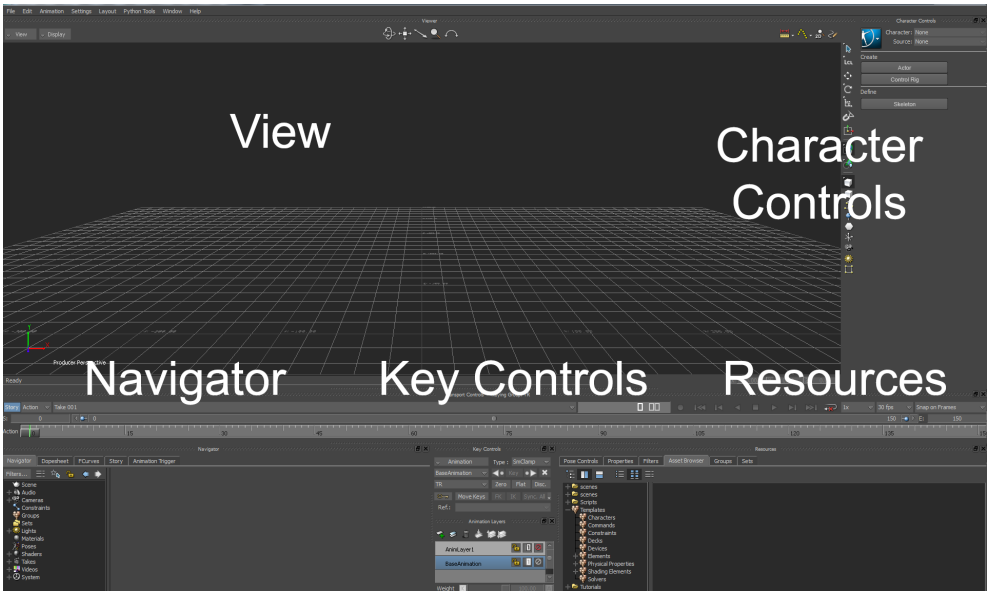
Projekt se ukládá do souboru s příponou ttp. Při zaznamenání záběru jsou vytvořená data automaticky uložena do složky, která nese v názvu slovo Session a datum vytvoření. Ta obsahuje soubory s příponou tak, které nesou samotná data o záběru, tedy data o pohybech markerů, jednotlivé Marker Sety a Rigid Bodies. Tyto složky je možné do libovolného projektu načíst pomocí příkazu Open v hlavním menu.

Pro export dat k dalšímu využití slouží položka Export tracking data, která se nachází v hlavním menu. Je vhodné data uložit jako soubor s příponou c3d. Uloží se tak všechny markery ve scéně, jednotlivé Rigid Bodies exportovány však nejsou. Je třeba je tedy chápat jako objekty, které uživateli při práci v programu Motive pomáhají s přiřazením markerů (program dokáže Rigid Bodies automaticky identifikovat) a pro snadnější dopočítání animační křivky těch markerů Rigid Body, které obsahují trhliny.

A.2 Aplikace pohybů v programu Motion Builder

A.2.1 Prostředí programu Motion Builder

Vzniklá a vyčištěná data je možné použít k navázání na animaci kostry v programu Motion Builder od firmy Autodesk. Ten při svém prvním startu dovoluje uživateli vybrat z několika druhů ovládání. Jednou z možností je nastavení, které je shodné s ovládáním programem Autodesk Maya. Díky tomu je možné používat stejné klávesové kombinace pro pohyb ve scéně a klávesové zkratky pro vyvolání všech druhů manipulátorů. Největším rozdílem oproti ovládání v Maye je způsob zrušení označení. To se provádí dvojklikem do míst scény, kde se nenachází žádný objekt. Základní rozložení pracovní plochy nabízí pohled na scénu jménem View, na jehož pravém okraji se nachází manipulátory a jiné nástroje. Vpravo od okna View se nachází okno Character Controls. Spodní části okna programu obsahuje okno Navigator, které je úplně vlevo. Uprostřed je okno Key Controls a vpravo Resources. Označení těchto oken je vždy nad nimi vycentrováno na střed. (Obrázek A.1)



Obrázek A.1: Okno programu Motion Builder
Zdroj: vlsatní

A.2.2 Použití exportovaných dat

Program Motion Builder dovoluje importovat soubory s příponou c3d jako skupinu markerů. Tu je možné přesouvat, rotovat a měnit její měřítko pomocí ikony síťové koule. Pokud jsou načtena data, která jsou již programem Motive vyčištěna, je možné vložit do scény objekt zvaný Actor. Tento objekt, který má tvar postavy, zprostředkovává aplikaci markerů na animační kostru. Objekt Actor se vytvoří pomocí tlačítka Actor v okně Character Controls. V této fázi je nejjednodušší načtené markery pomocí zástupné ikony umístit pomocí všech tří transformací. Při tomto procesu je třeba, aby markery kyčle a hlavy odpovídaly pozici vůči objektu Actor tak, jak byly markery rozmístěny na těle herce při záznamu pohybů. Pozice rukou a nohou je možné upravit pomocí rotace. K tomu je třeba označit přímo požadovanou část objektu Actor a spustit nástroj pro rotaci.

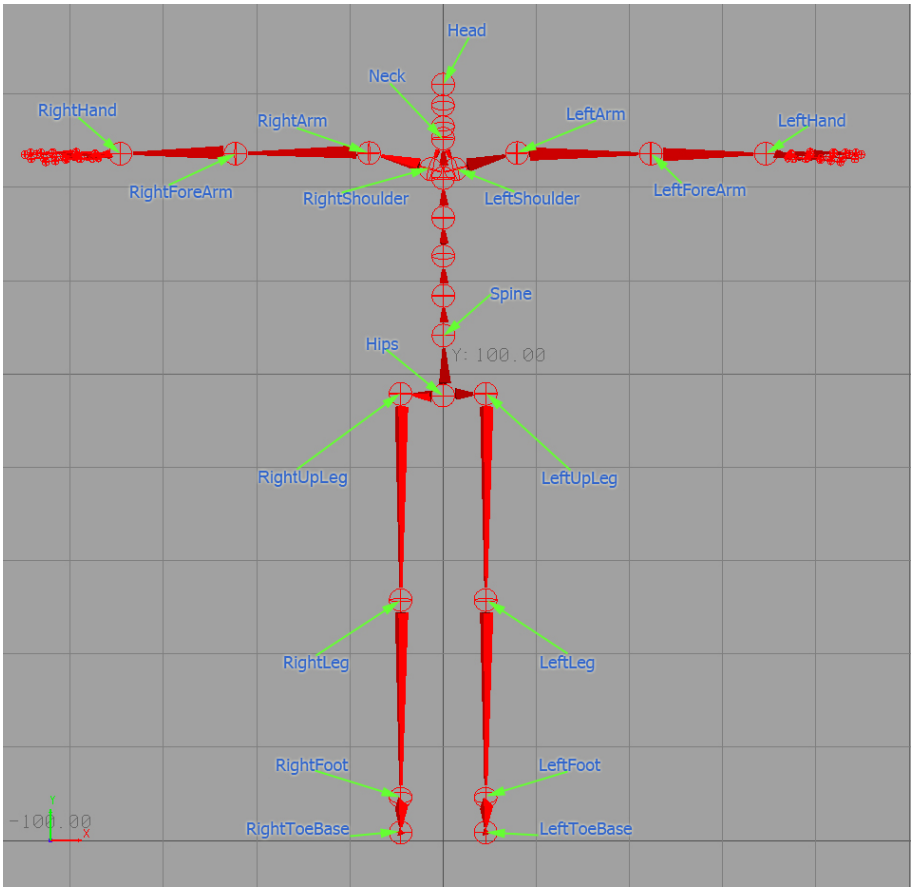
Přiřazení jednotlivých markerů objektu Actor je možné po vytvoření skupiny Marker Set. To umožňuje tlačítko okna Navigator, které se v jeho pravé části zobrazí po označení objektu Actor. Po vytvoření nové skupiny Marker Set je možné přiřadit načtené markery částím objektu Actor, který mají řídit. Pokud bude k přiřazování markerů použit pohled na scénu, může být vhodné pro tento úkon vypnout vykreslování samotného objektu Actor. Tuto funkci nabízí modrá ikonka v okně Character Controls. V této chvíli je již možné pomocí funkce Drag and Drop přesouvat markery ze seznamu v okně Navigator na modré ikonky, které jsou umístěny na obrázku objektu Actor ve vedlejší záložce. Pokud dojde ke špatnému přiřazení, je možné modrou ikonku označit a ze seznamu přiřazených markerů ji pomocí klávesy delete ze seznamu odstranit. Při přiřazování markerů je třeba, aby root, tedy ikona v oblasti pánve, obsahovala alespoň tři markery.

Následující krok je nezvratný, proto je vhodné rozpracovanou scénu uložit. Propojení

pohybu markerů a pohybu objektu Actor je spuštěno zaškrtávacím polem s popiskem Active, které se nachází nad systémem pro přiřazování markerů. Jakmile je pole zaškrtnuto, objekt Actor začne být řízen pomocí markerů. V této fázi, pokud markery byly správně přiřazeny, je možné si animaci prohlédnout a zkontrolovat pomocí časové osy. Pokud v přiřazování byly nějaké chyby, objekt Actor se pokroutí. V takové situaci je třeba načíst uložený soubor a chybu v přiřazování odhalit. Pokud je třeba provést jen malé korekce objektu Actor a markerů, je třeba pole Active odškrtnout a provést požadované transformace. Tlačítko Snap pak markery s objektem Actor opět propojí. V nabídce, které se při této akci zobrazí, je jen třeba zvolit volbu označenou zkratkou TR.

A.2.3 Aplikace pohybů na model

Aby bylo možné vložit kostru do programu Motion Builder, pro zjednodušení práce je vhodné, aby jednotlivé klouby modelu byly pojmenovány podle předem daného názvosloví (anglicky podle naming convetion). Toto názvosloví je znázorněno na obrázkuA.2.



Obrázek A.2: Názvosloví programu Motion Builder
Zdroj:<http://mocappys.com/wp-content/uploads/2010/08/Skeleton.jpg?220566>

Program Motion Builder standardně pracuje se soubory s příponou fbx. Je tedy přirozené,

že právě tento formát, do kterého zvládají export programy Autodesk Maya i Autodesk 3D Studio Max, je využíván pro přenos informací mezi programy. Pokud je do tohoto formátu uložena kostra nebo model s kostrou, které jsou propojené pomocí Smooth Skin, je možné takový soubor do programu Motion Builder vložit následujícím způsobem. V okně Resources se nachází seznam složek, které jsou označeny jako zdroj různých dat, které je možné do scény vložit. Do tohoto seznamu je třeba přidat složku, která model ve formátu fbx obsahuje. Pravým tlačítkem je třeba kliknout pod poslední složku a v kontextovém menu zvolit možnost Add favorite path. Po jejím označení je třeba vybrat soubor s modelem ze seznamu a pomocí techniky Drag and Drop ho přenést do scény. To otevře další nabídku, kde je třeba zvolit možnosti FBX merge a následně No Animation.

Model je třeba zvětšit a umístit, aby se co nejvíce shodoval s objektem Actor. Provést změnu měřítka je poměrně složité. V okně Navigator je v záložce Navigator seznam objektu ve scéně. V něm je i kořen animační kostry právě importovaného modelu. Při kliknutí na něj pravým tlačítkem je možné zvolit funkci Select Branches. Ta označí všechny klouby celé animační kostry. V okně Resources se pak může v záložce Properties upravit všem kloubům plošně hodnota Scale. Tuto změnu je třeba provádět osu po ose. Následující úkony, jako jsou rotace a posun, lze již vykonávat standardně označením vždy konkrétního kloubů a použitím manipulátoru.

V momentě, kdy se importovaný objekt shoduje svou pozicí a velikostí s objektem Actor, je možné provést úkon Characterize. V okně Resources, v záložce Asset Browser se ve složce Templates a v podložce Characters nachází objekt Character. Ten je třeba pomocí techniky Drag and Drop přenést na kořen importované animační kostry, která se nachází v seznamu objektů scény v okně Navigator. Tento úkon vytvoří nový objekt Character, který se nachází opět v okně Navigator v záložce Navigator. Dvojklikem na tento objekt se otevře nová nabídka v okně Character Controls. V té pak stačí nastavit objekt Actor jako zdroj pohybů. Seznam zdroje pohybů se nachází pod položkou, která je označena heslem Source.

Aby bylo možné exportovat pohyby, které byly použity na vložený objekt, je třeba tyto pohyby aplikovat na transformace kloubů. V hlavním menu v položce Animation se nachází funkce Plot All, která pohyby kloubů trvale vytvoří. Pak stačí jen scénu uložit v hlavním menu pomocí příkazu Save As. V okně Saving Option je třeba zatrhnout ten záběr ze seznamu, který načtené pohyby obsahuje.

Příloha B

Zpráva riggera Gerarda Verroneho

January 2, 2015

To Whom it concerns,

My Name is Gerardo Sebastian Verrone. I have a University Degree in Image and Sound

Design thought by the University of Buenos Aires.

http://www.uba.ar/ingles/download/studyingatuba/undergraduatedegrees

/SoundandImageDesign.pdf

I have over nine-years experience working for Films, Games and Broadcast and run through all the aspects of the game production. Currently specialized on rigging, I've been developing numerous animator friendly character rigs for several games.

I was amazed how Radek managed to assimilate all the needed knowledge to create his first Automatic Rigging System for Maya. He has as well learnt Python scripting and was able to create very useful modules in few months. Radek is a passionate artist constantly evolving. He has a wide-range of talents that makes him a real asset for any production. Looking as his tool is noticeable the great attention to details and organization.

By following the instructions on how to install and start the script we can see at first glance a pretty minimalistic window but, do not feel intimidated by the absence of buttons as all the best will come later.

It's time now to state a name to be used as a prefix for the rig. This function is pretty cool as we will be able to use more than one character in the same scene avoiding any collision with other character's names.

Once we have designated a name and pressed the "create new button" four new buttons appears on the interface that symbolize the main areas of the rigging construction phases. A new window with the T pose also appears on the screen.

At first I was little struggled to discover how all works but later and by reading the documentation on how to use it I discovered how easy is to setup a full character rig with this fancy tool.

My first good impression was about the interface with the Character in ”T Pose”. It was really funny and comfortable at same time to discover that the interface change the focus on different parts of the character while you click on the wrist or the ankle. The way to map each joint with the corresponding body part on that interface is awesome. The load/save joints is a must have function on this type of autorigging system as we can start by using a pre-made skeleton, save it in the middle of the process and continue from there in another day. This is also helpful to let is prepare in no time a bunch of crawl riggs that have minor tweaks. The “rename for MB” process makes it much powerfull for production. It could be silly to mention but, most of the autorigging system did not remember were to look for the saved skeletons but, this tool has this properly coded and once you press the “Load joints” button it jumps directly to the directory that stores all the pre-made skeletons.

Now that we have our skeleton in place it’s time to discover how the “Controllers” button works. I did not traverse all the python code yet but, I discovered that we do not need to select any joint at this or any future step as the tool seams to store in a “magic” way all the names of joints and any element that is created at all the steps. With no more delay I pressed the “Controllers” Button and a new window with plenty of possibilities appear on the screen.

This main module not only gave us the possibility to create the controller and the main body parts including arms and legs with auto IK/FK functionality and ribbons to make it all elastic but also gave us the awesome possibility to delete any specific part and re build it from scratch without breaking the rig.

I do really love the way ribbons are created and how they work. It’s amazing for the rigger to be able to add or delete them and to also re-create it with more joints on the shoulder part or the arm body part!

As a side note, the Auto IK/FK system works like a charm. I was playing with it for some long minutes and trying to debug in my head how it was done as the automatic behaviour is quite unique and accurate. Most of the rigs made by hand or by using automatic rigging systems like this, comes with a tool that gave the animator the possibility to indicate the switching mechanism to go from one system (IK) to the other (FK) but Radek has solved in an amazing and automatic way with no more intervention on the hands of the animator. This is one of the most remarkable things of this rig.

Jumping to the “Deformers” section we find two buttons that acts as a quick lunch operations to automatically select all the joints that will be used to smooth bind the character and the smooth skin button that alow us to pick the character body mesh. Once both things are selected the tool will automatically create the smooth bind.

There is a second section on the Deformer stage that’s dedicated to the Corrective Blend Shapes. This is a really useful part of the tool as we can extract a copy of the model at any pose, tweak the shape with any tool available in Maya and map that finished and corrected mesh as blend shape that will be driven with set driven keys. I will be delighted to test a further iteration of the script were the set driven key is mapped automatically avoiding the need to do some of those steps by hand. For example that part can replicate the set driven key window giving us the possibility to indicate who drives the blend shape and in which channel and once the “ready” button is pressed the tool can create the zero key with no blend shape applied and the actual pose with the blend shape applied in full. It’s not a big deal but something that helps a lot once you’re working.

Lastly, we found the “Reset skin, keep joints” button that do a “magic trick” and gave us the possibility to move and tweaks joints without altering and already skinned character. This is a functionality that only AAA studios had developed for their self. Considering how useful it is I still thinking why Autodesk have never develop such a useful tool inside Maya. Great Work Radek!

All in all, and considering the tool is in development phase as part of “Career Thesis“ it looks solid, well done and includes a lot of surprises that will delight many riggers out there. As a final note I must stress that Radek is a Proactive TD and always finding solutions to any problem, his constant smile makes you comfortable. He is not only a gifted artist but a great hearted person always here when you need him. I really hope to work with him some day.

You can contact me for any further details. animatorstudio@hotmail.com

Best regards.

Gerardo Sebastian Verrone

Technical Director

Buenos Aires, Argentina.

Příloha C

Ukázka rigu a jeho fungování