Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Informatika a výpočetní technika

Počítačová grafika

Grafický výstup elektronického tlumočníka znakové řeči

Vypracoval: Jiří Dobrý

Vedoucí diplomové práce: Prof. Ing. Václav Skala, Csc.

Rok a místo vydání: Plzeň, 2000

Sign Language Interpreter

Abstract

There are five human senses connecting each of us with the surrounding world. Deaf and hard of hearing people are missing one of these senses. Their communication is strongly affected, because most of them are not able to communicate verbally. In this article a graphic system for the computer visualisation of the sign language (the means of communication between deaf people based on hand movements) will be described. The system visualises an animated human model in real time and is able to translate written text into the sign language (with the limitations of the dictionary). In the following paragraphs an introduction to the sign language will be presented as well as an explanation of the geometric model of human being, controlling of animation and real time visualisation using OpenGL API.

1 Úvod 6

Teoretická část 7

2 Znakový jazyk 7

2.1 Historie znakového jazyka 7

2.2 Hlavní složky znakové řeči 8

2.2.1 Znakovaná čeština 8

2.2.2 Vlastní znakový jazyk 9

2.3 Rozšiřování znakové řeči 9

2.4 Komunikace 9

2.5 Struktura jednotlivých znaků 10

3 Anatomická struktura kostry člověka 12

3.1 Základní typy kloubních spojení horní končetiny 12

4 Lineární algebra 15

4.1 Maticový model OpenGL 15

4.2 Geometrické transformace 15

4.3 Inverzní matice transformace 16

5 Vizualizace lidské postavy počítačem 16

5.1 První virtuální herci 17

5.1.1 Stick figures 17

5.1.2 Surface model 17

5.1.3 Volume model 18

5.1.4 Multi-layered model 19

6 Techniky počítačové animace 19

6.1 Klíčování (key-frame animation) 20

6.1.1 Techniky key-frame animace 20

6.1.2 Získávání klíčových snímků 21

6.2 Procedurální animace 22

7 Interpolace 23

7.1 Problém interpolace 23

7.2 Typy interpolace 24

7.2.1 Lineární interpolace 24

7.2.2 Spline interpolace 24

8 Vlastní vizualizace 26

8.1 Grafické akcelerátory 27

8.2 Grafická knihovna OpenGL 28

Praktická část _____ *29*

9 Systém pro interpretaci znakové řeči počítačem _____ **29**

10 Volba metod _____ **29**

11 3D model pro animaci znakové řeči _____ **30**

11.1 Hierarchická struktura modelu _____ 30

11.2 Design modelu _____ 32

11.3 Datové struktury modelu a formáty souborů _____ 33

11.3.1 Datová struktura modelu _____ 33

11.3.2 Segment a jeho parametry _____ 33

11.3.3 Formáty souborů _____ 34

12 Animace _____ **36**

12.1 Geometrické transformace a řízení pohybu _____ 37

12.2 Omezení pohybu kloubů _____ 38

12.3 Deformace segmentů _____ 38

12.3.1 Koeficienty ohybu _____ 39

12.3.2 Vlastní algoritmus deformace _____ 40

12.4 Datová struktura slovníku _____ 41

12.4.1 Datové struktury _____ 41

12.4.2 Paměťová náročnost a diskový prostor _____ 42

12.5 Alternativní modely _____ 42

13 Vizualizace _____ **43**

13.1 Algoritmus vizualizace _____ 43

13.2 OpenGL _____ 43

14 Dosažené výsledky _____ **45**

15 Poděkování _____ **46**

16 Reference _____ **47**

Přílohy _____ *49*

A Uživatelská dokumentace _____ **49**

B Instalace _____ **56**

C Programátorská dokumentace _____ **57**

D Ukázky výstupu _____ **66**

1 Úvod

Každý člověk komunikuje se svým okolím. Komunikace je jednou ze základních lidských potřeb. Sluchově postižení lidé jsou ochuzeni o jeden z nejdůležitějších smyslů – sluch. Je jim tak velmi zkomplikována možnost komunikace a to především se slyšícími. Pro svou vlastní komunikaci (mezi neslyšícími) i pro komunikaci se slyšícím okolím (rodiče, sourozenci a další) používají speciální formu komunikace, znakovou řeč. Znaková řeč je však velmi komplikovaná zvláště pro slyšící osoby, a tak je komunikace neslyšících často omezena na úzký okruh lidí. Takto se neslyšící odtrhávají od ostatního světa, žijí v uzavřených komunitách a přestávají se snažit komunikovat se slyšícím okolním. Je pro ně potom většinou velmi komplikované řešit každodenní problémy - například komunikace s úřady. Řešení problému by však nemělo být jenom na neslyšících. Cílem všech lidí by mělo být vyjít si vzájemně vstříc a snažit se najít společnou formu komunikace vyhovující všem zúčastněným.

Tato práce popisuje návrh a implementaci systému interpreta znakové řeči[1], který se snaží pomoci řešit problém komunikace mezi neslyšícími a okolím. Systém využívá spojení moderních metod počítačové grafiky a animace k vytvoření modulu pro syntézu znakové řeči počítačem. S jeho využitím může vzniknout řada aplikací, které budou prospěšné jak lidem, kteří jsou sluchově postižení, tak i lidem, kteří s nimi přicházejí do kontaktu, ať už se jedná o rodiče, sourozence neslyšících, tlumočníky, učitele znakové řeči nebo kohokoliv jiného, kdo se s neslyšícími setkává.

Zde navrhovaný systém lze použít mnoha způsoby. Hlavní využití by mělo být pro slovník znakové řeči, jehož 4 rozměry (3D model + animace) jsou nesrovnatelné s knižním vydáním. Jistou obměnou by mohl být multimediální výukový program znakové řeči určený pro slyšící, nebo jakýkoliv výukový program se znakovou řečí jako jedním z výstupů, určený pro výuku neslyšících. Také by bylo možné připojení tohoto systému na titulkování filmů v televizi, kde se až doposud objevují pouze skryté titulky. Neslyšícím se tak zpřístupní informace ve formě pro ně mnohem přijatelnější.

[1] Tento projekt byl podporován projektem MŠMT ČR - projekt VS 97 155.

Teoretická část

2 Znakový jazyk

Český znakový jazyk je základním dorozumívacím jazykem neslyšících v naší zemi. Je to přirozený a plnohodnotný komunikační prostředek tvořený specifickými vizuálně-pohybovými prvky, tj. tvary rukou, jejich postavením, pohyby, mimikou, pozicemi hlavy a horní části trupu. Má základní atributy jazyka, kterými jsou znakovost, systémovost, dvojí členění, produktivnost, svébytnost a historický záměr, a je ustálen jak po stránce lexikální, tak i gramatické.

2.1 Historie znakového jazyka

K rozvoji vizuálně pohybových znaků přispěl významným způsobem abbé Charles Michel de l´Eppé (1712 – 1822), který založil první ústav pro nemajetné hluchoněmé v Paříži 1770. Vycházel od přirozených posunků a snažil se znakovou řeč zlepšit po gramatické stránce. Jeho znakový jazyk šířili jeho žáci po celé Evropě. Do USA jej přenesl na počátku 19. století T. H. Gallaudet. Na tomto základě se vizuálně pohybový jazyk v různých variantách dále rozvíjel prostřednictvím učitelů vzdělávaných na Gallaudetově koleji v USA.

Rozvoj vlastního znakového jazyka začal v Evropě se zakládáním ústavů pro hluchoněmé v 18. století. V těchto ústavech měli neslyšící mnohem větší možnost komunikace, což způsobilo prudký rozvoj vlastního znakového jazyka, komunikačního prostředku, který nejlépe vyhovoval jejich smyslovým, tj. zrakovým, dispozicím a umožňoval jim naplňovat jednu ze základních biologických potřeb člověka, potřebu informací.

K rozvoji znaků u nás významně přispěl V. Frost v 19. století. V některých ústavech byly znaky potlačovány, v jiných respektovány a používány, podle možností a situace. Velkým mezníkem byl Milánský sjezd v roce 1880, kde učitelé hluchoněmých přijali usnesení, že hluchoněmá osoba, která netrpí ještě jinou vadou, má být vyučována metodou orální. Toto usnesení bylo potvrzeno i na následujícím sjezdu v Paříži v roce 1900. V České republice se situace změnila až v roce 1998 na

základě zákona č. 155/98 Sb., o znakové řeči a změně dalších zákonů, kde je v §7 stanoveno právo neslyšících na používání znakové řeči.

2.2 Hlavní složky znakové řeči

V dorozumívání neslyšících tvoří vizuálně-pohybové projevy (není myšlen jen vlastní znakový a znakovaný jazyk) dominantní složku. Neslyšící mají obvykle pochopitelnou tendenci převádět všechna sdělení, pocity, rozpoložení do viditelné, zřetelné a tím i pro neslyšícího sdělitelné podoby.

Hlavní komunikace ve znakové řeči je založena na posunkování, které se skládá z pohybu prstů, rukou a paží. Některé znaky mohou být spojeny s určitou částí těla, především obličeje. Podrobněji bude struktura znaků popsána níže.

Velice důležitá je i doprovodná mimika, která dává znakové řeči citový přízvuk a vyjadřuje tím jemné odstíny jednotlivých mluvených slov. Používá se pro doplnění základní informace.

Prstová abeceda se používá tam, kde chybí vhodný znak. Pro každé písmeno české abecedy existuje jeden znak znakové řeči, obvykle posunkovaný jednou rukou. Sestavení slova z těchto znaků se nejčastěji používá pro vyjádření jména nebo názvu.

2.2.1 Znakovaná čeština

Znakovaná čeština je vizuálně pohybový jazyk, který téměř doslovně překládá češtinu do vizuálně pohybových znaků. Mnozí neslyšící znakované češtině rozumějí, ale sami ji vůbec nepoužívají. Rozsah znakové zásoby ve znakované češtině je větší, než ve vlastním znakovém jazyce neslyšících. Při složitějších sdělení mnozí neslyšící nerozumějí danému tématu. Slova, pro něž nemají neslyšící znak, vyznačí prstovou abecedou nebo parafrázují.

Znak je vizuálně pohybový prvek, který je základním stavebním kamenem tohoto jazyka. Skládá se z řady přirozených posunků nebo bývá složen z jiných znaků. Zásoba znaků se neustále rozšiřuje, obohacuje a mění.

2.2.2 Vlastní znakový jazyk

Vlastní znakový jazyk je v současnosti nejvýhodnějším komunikačním prostředkem neslyšících. Je to souhrn pohybových a mimických výrazových prostředků vnímaných převážně vizuálně, které byly vytvářeny obvykle skupinou neslyšících. Má svoji gramatickou strukturu, která je nezávislá na mluveném slově. V praxi se běžně používá v neformální komunikaci mezi neslyšícími, přičemž je doprovázen výraznou mimikou a současným užíváním mluvené řeči, obvykle s vynecháním hlasu. Vzory stylistiky vlastního znakového jazyka jsou v tabulce 1.

Mluvené slovo	**Znakový jazyk**
Co budeš dělat dnes večer?	Večer + dnes + co ?
Máš s sebou peníze?	Má + peníze + ty ?
Jak se daří tvé matce?	Máma + zdravá ?
Kdo u vás vaří?	Kdo + vaří ?

tabulka 1

2.3 Rozšiřování znakové řeči

Vytváření nových znaků je velmi náročné. Při tvoření nového znaku je nutné, aby se neslyšící seznámil s pojmem a obsahem daného slova. Na základě této prvotní informace se pak nechává prostor pro vytvoření nového znaku samotnými neslyšícími. Ve znaku jsou zdůrazněny podstatné vlastnosti nového pojmu tak, jak se jeví neslyšícím. Nemusí to být vždy vlastnosti, které by za charakteristické považovali slyšící. Bývají to většinou vlastnosti spíše nápadné než podstatné.

2.4 Komunikace

V kontaktu se slyšícími používají neslyšící hlasitou mluvenou řeč tak, jak se ji naučili ovládat. Obvykle je tato řeč disgramatická, obsahově chudá, s deformovanou melodičností, rytmem a nepřirozenou výslovností. Bývá často doprovázena výraznou mimikou. Taková řeč není vždy dobře srozumitelná.

Při dorozumívání se slyšícími používají neslyšící znakovou řeč zároveň s mluvenou tehdy, jestliže slyšící partner znakovému kódu rozumí nebo se snaží

rozumět. Neslyšící jsou ve svém projevu - ve snaze o komunikaci obvykle ochotni vyjít slyšícím vstříc, zvláště těm, k nimž mají důvěru.

Slyšící užívají v komunikaci s neslyšícími především mluvenou řeč. Měli by zřetelně vyslovovat s tváří přímo k neslyšícím, užívat především známých slov a srozumitelně formulovat věty. Pokud alespoň částečně znají znakový jazyk, užívají obvykle těchto znaků, a to tehdy, když neslyšící mluvené řeči dobře nerozumí. Mezi slyšícími, kteří jsou v častém styku s neslyšícími, je obvykle aktivní znalost nejčastěji používaných znaků, podle toho, jak si je osvojovali v průběhu své komunikace s neslyšícími.

Souběžné užívání mluvené a znakové řeči používají slyšící pro sdělování neslyšícím jen zřídka, protože málokteří slyšící ovládají znakový jazyk v dostatečném rozsahu. Tento způsob se užívá nejčastěji při tlumočení a výuce.

Vizuálně pohybová komunikace a její použití při výchově a vzdělání neslyšících je velmi diskutovanou otázkou. Jistou roli hraje fakt, že vizuálně pohybová komunikace je pro slyšící velmi těžko poznatelná a zapamatovatelná (s výjimkou slyšících dětí neslyšících rodičů).

2.5 Struktura jednotlivých znaků

Jak již bylo řečeno, znakový jazyk se skládá z několika důležitých částí. Nyní se zaměříme na nejpodstatnější část, kterou je znakování. Pod pojmem znakování rozumíme souhrn pohybů prstů, rukou a paží. Každý znak je složen z jednoduchých pohybů (např. „já“ = nataženým ukazováčkem pravé ruky ukážeme na svoji hruď) nebo jako kompozice existujících znaků (např. „Betlém“ = „Marie“ + „kostel“).

Tvary rukou jsou v tomto případě velmi důležité, protože některé znaky se liší od sebe jen málo při znakování, ale výrazně ve významu (např. „mám“ = totéž jako „já“, přičemž nataženy jsou dva prsty těsně u sebe – ukazovák a prostředník). Proto se také používají pro konstrukci znaků frekventované tvary ruky. Nejčastěji používané tvary ruky lze vidět na obr. 1. Tvar ruky se pak může měnit v průběhu jednoho znaku.

Pohyb paže většinou znázorňuje popis nějaké vlastnosti daného výrazu. Nebývá to vždy nejpodstatnější významová informace o daném slově. Ve většině případů jde spíše o vizuální metaforu, přičemž někdy je pro slyšícího člověka velice složité této metafoře porozumět.

Existuje také řada znaků, které využívají při své interpretaci částí těla nebo obličeje (např. znak červená ukazuje na rty).

Jednodušší znaky se většinou skládají z pohybů jedné ruky. Pro vyjádření složitějších se pak používá obou rukou. Existuje však většinou i varianta znakovaná jednou rukou, kterou lze použít v situacích, kdy druhá ruka není volná.

V celkovém souhrnu lze říci, že znakování se skládá z pohybu prstů, rukou a paží ve spojení s horní částí těla.

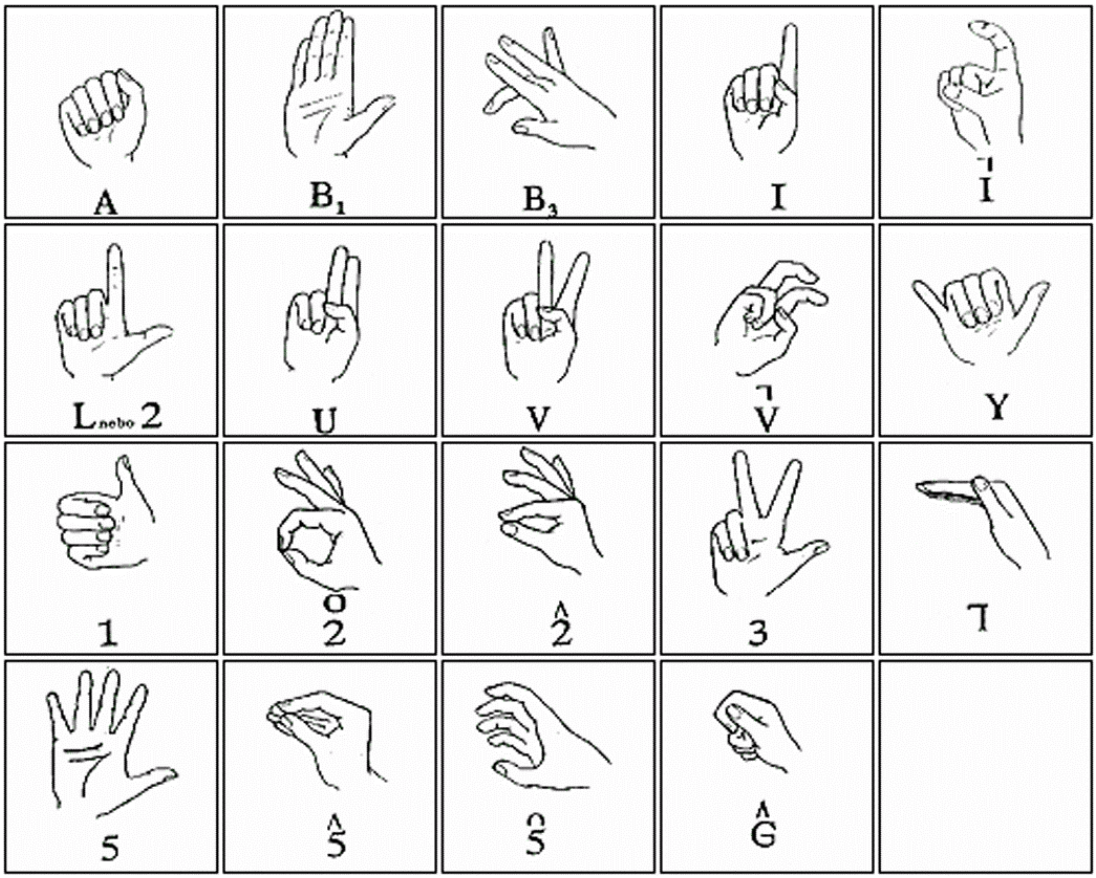
Ukázkou popisu několika znaků v knižním vydání slovníku znakové řeči jsou následující příklady. V knize Slovník znakové řeči [1] je řada znaků doplněna fotografiemi s pohyby schematicky naznačenými šipkami.

Znaky: „pondělí“ – natažená dlaň se prsty dotkne čela, pak brady

„já“ – nataženým ukazovákem pravé ruky ukážeme doprostřed prsou

„jablko“ – krouživý pohyb sevřenou pěstí pravé ruky po pravé tváři

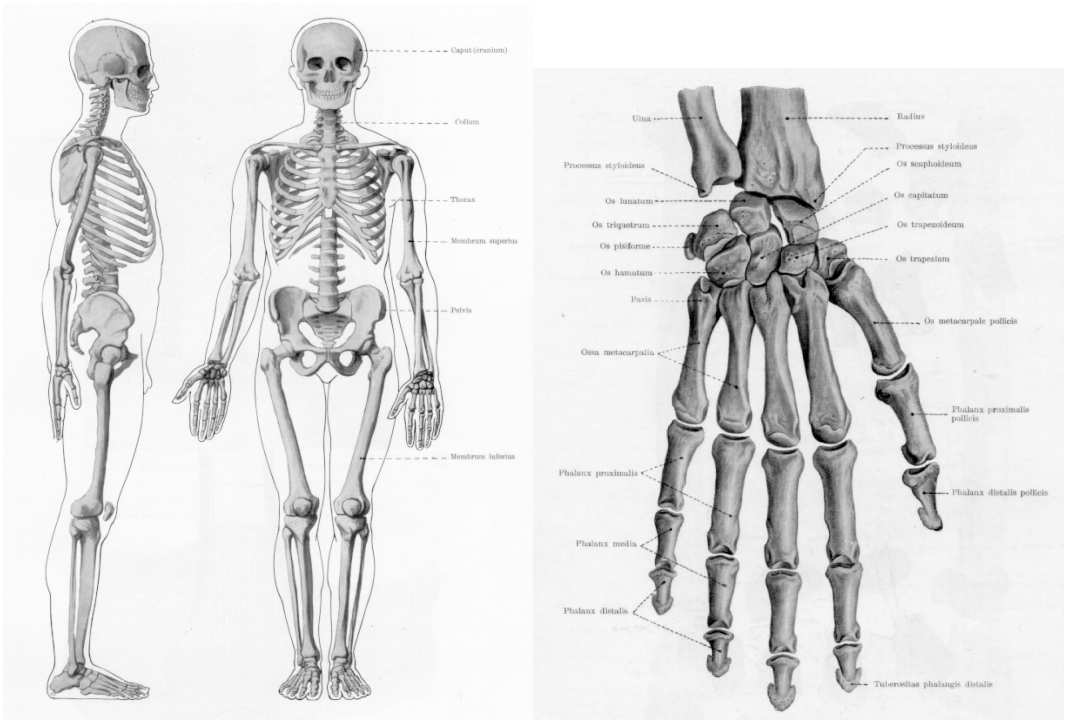
„kříž“ – naznačí se dvěma prsty (ukazovákem a prostředníkem) kříž



obr. 1

3 Anatomická struktura kostry člověka

Pro vytvoření modelu lidské bytosti je důležité znát alespoň základní informace o anatomické stavbě těla. V případě vizualizace tlumočníka znakové řeči je podstatná stavba kostry člověka a pohybové vlastnosti jednotlivých typů kloubních spojení. Parametry lidského těla popisuje publikace Biomechanical Basis of Human Movement [2]. Kostra člověka a kostra ruky jsou na obr. 2.



obr. 2

3.1 Základní typy kloubních spojení horní končetiny

- **Ball-and-Socket (rameno):**

Ball-and-Socket je nejpohyblivější kloubní spojení celého těla. Umožňuje pohyb ve třech rovinách. Základními pohyby jsou flexe – extenze, abdukce - addukce a rotace. Příkladem tohoto kloubního spojení je rameno (viz obr. 3a).

- **Hinge (loket):**

Kloubní spojení umožňující pohyb pouze v jedné rovině. Je také nazýváno uniaxiální a pohyby zakového typu kloubu jsou extenze a flexe. Toto geometricky nejjednodušší spojení můžeme nalézt v lokti a článcích prstů (viz obr. 3b).

- **Pivot (radioulnar):**

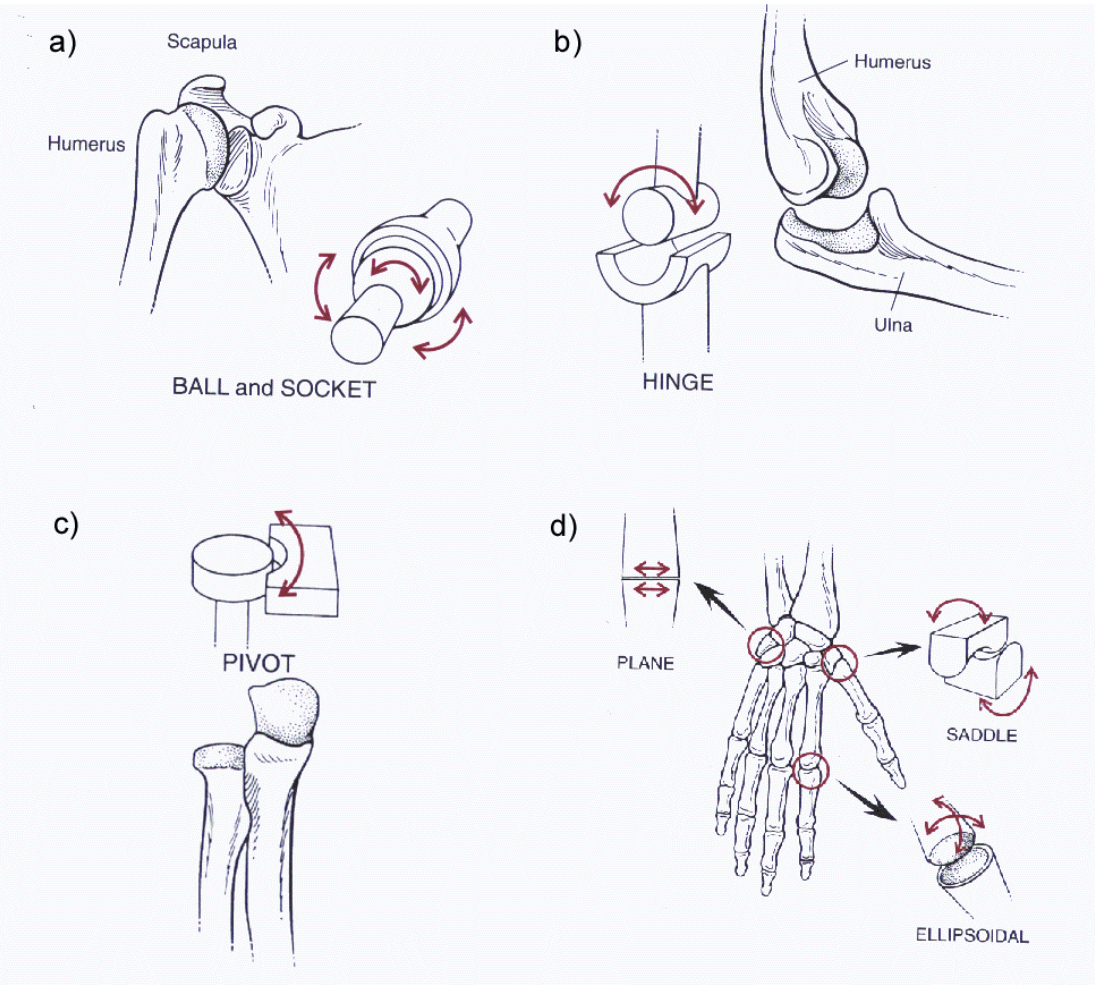
Pivot je také uniaxiální kloubní spojení. Umožňuje pohyb v jedné rovině a to v rovině kolmé na osu segmentu kostry. Pohyb se nazývá rotace. Toto kloubní spojení můžeme najít v předloktí (viz obr. 3c).

- **Ellipsoid (zápěstí, Metacarpophalangeal - prsty):**

Pohyb v elipsoidním kloubu je biaxiální. Možné jsou flexe – extenze a abdukce - addukce. Příkladem je zápěstí a připojení prstů ke dlani (viz obr. 3d).

- **Saddle (Carpometacarpal – palec):**

Toto je unikátní kloubní spojení vyskytující se pouze v místě připojení palce ke dlani. Umožňuje pohyby flexe – extenze, abdukce - addukce a také malou míru rotace. Funkcí je hodně podobné elipsoidnímu spojení (viz obr. 3d).



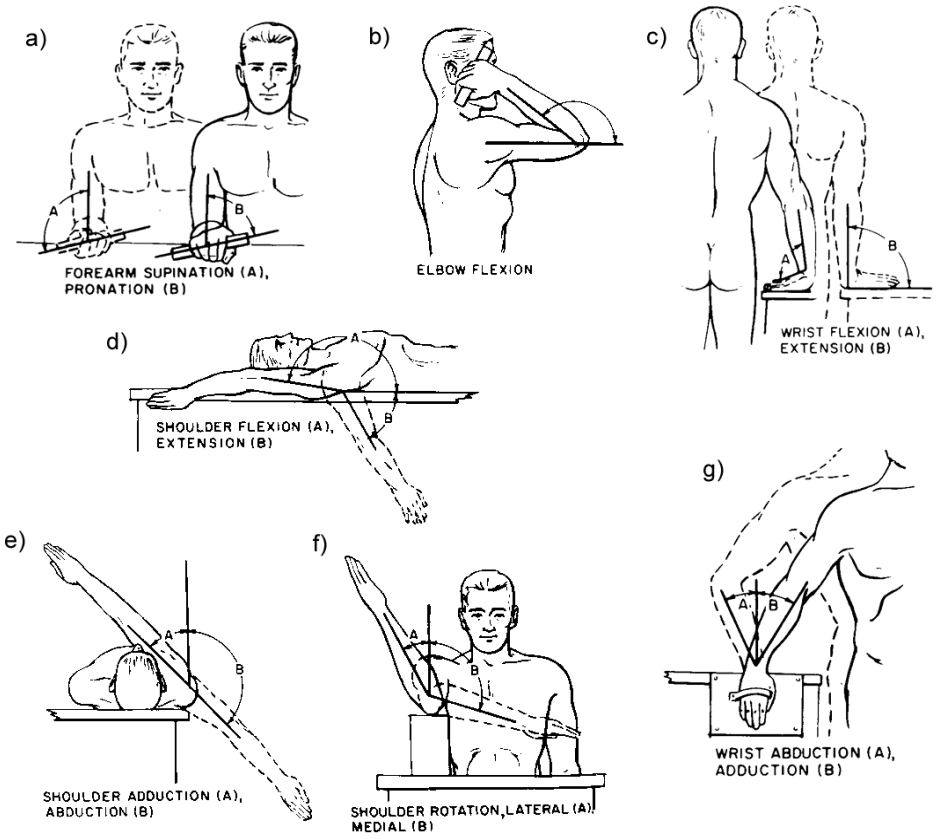
obr. 3

Lidské tělo je do pohybu uváděno velkým množstvím svalů. Pokud zahrneme všechny jednotlivé pohyby, dostaneme velmi komplikovaný systém.

Každý pohyb kloubů lidského těla je více či méně omezený. Omezení vyplývá jak z vlastní stavby kloubů, tak i z omezené pohyblivosti svalů. V tabulce 2 jsou uvedeny některé hodnoty omezení pohybů průměrného člověka. Na obr. 4 je ilustrativně naznačen význam uvedených parametrů.

Movement	**Range (deg.)**
Wrist flexion	90
Wrist extension	99
Wrist adduction	27
Wrist abduction	47
Forearm supination	113
Forearm pronation	77
Elbow flexion	142
Shoulder flexion	188
Shoulder extension	61
Shoulder adduction	48
Shoulder abduction	134

tabulka 2



obr. 4

4 Lineární algebra

V následujících odstavcích budou stručně popsány algebraické postupy, které jsou použity v systému pro vizualizaci znakové řeči. Předpokládá se základní znalost maticového počtu.

4.1 Maticový model OpenGL

Nejprve je třeba se zmínit o maticovém modelu použitého vizualizačního rozhranní. OpenGL transformuje souřadnice všech vstupujících elementů transformačními maticemi. Pro transformaci souřadnic ze souřadného systému objektu do souřadnic scény a pro rotace a translace segmentů je použita matice modelview.

Každý 3D vrchol *v* předaný OpenGL se transformuje podle vztahu (1).

$$x = M \cdot v \qquad (1)$$

kde *x* je transformovaný bod,

v je původní bod,

M je matice transformace.

OpenGL nabízí řadu operací, které aktuální matici M transformují. Maticově lze tyto operace popsat vztahem pro maticové násobení zprava (2).

$$M' = M \cdot T \qquad (2)$$

kde *M'* je nově získaná transformační matice,

M je původní matice,

T je matice aplikované transformace.

V následujícím vztahu (3) je ukázáno, jak se jednotlivé transformace akumulují a následně se jimi transformuje bod *v*.

$$x = M \cdot T_1 \cdot T_2 \cdot T_3 \cdot v \qquad (3)$$

Pokud tedy v OpenGL skládáme transformace, výsledné pořadí takto akumulovaných transformací je tedy opačné než ve kterém byly předány.

4.2 Geometrické transformace

Transformace, které jsou používány v systému lze popsat obecně v homogenních souřadnicích maticí 4×4 prvky (4).

$$M = \begin{bmatrix} & & & T_x \\ & R & & T_y \\ & & & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4)$$

kde R je submatice 3×3 prvky reprezentující rotace,

T je vektor translace.

4.3 Inverzní matice transformace

Výpočet deformace modelu v blízkosti kloubního spojení vyžaduje znát inverzní matici transformace. Tu lze jednoduše získat v případě, že matice transformace *M* je ortonormální. Pak totiž platí vztah (5).

$$M^{-1} = M^{T} \qquad (5)$$

Důkaz tohoto tvrzení je zřejmý. Matice M je ortonormální. Řádky tvoří navzájem kolmé jednotkové vektory. Pokud tedy vynásobíme MM^T, je zřejmé, že skalární součin normalizovaného vektoru se sebou samým je roven 1 a součin normalizovaných dvou vektorů navzájem ortogonálních je roven 0. Výsledkem je tedy jednotková matice *I*. V našem případě používáme inverzní matici rotace v kloubu, která splňuje podmínku ortonormality.

5 Vizualizace lidské postavy počítačem

Existuje řada metod jak zobrazovat modely reálného světa počítačem. Zobrazování lidských postav je jedním z nejobtížnějších problémů řešených v oblasti počítačové grafiky. Náročnost vychází z anatomické stavby těla. V době, kdy budeme schopni počítačem simulovat pohyb každého svalu, napínání kůže, realistické vlasy a další části těla přesně podle fyzikálních zákonů a pokud možno v reálném čase, bude tato oblast možná označena za uzavřenou, ale jistě se naskytne řada dalších úkolů k vyřešení. Zatím však, i přes velmi významné pokroky dosažené od počátků počítačové grafiky, je toto stále ještě budoucností. V historii se řada lidí pokoušela vytvořit vizuální simulaci lidské postavy. Výsledky každého z těchto pokusů byly úměrné své době a hlavně výkonu použité techniky. Pro dosažení maximální kvality při dostatečné rychlosti vykreslování byla objevena řada nových přístupů. V současné době se vrcholná díla počítačové animace objevují hlavně na filmovém plátně.

Většina nových filmů využívá počítačových triků a speciálních efektů k silnějšímu působení na diváka. Ovšem použití kompletního nahrazení živých herců postavami počítačem generovanými je zatím stále spíše v experimentálním stádiu. Kvalita animace a vizualizace takových virtuálních postav závisí na mnoha faktorech od designu počítačových modelů, přes řízení animace, nasvětlení scény až po vlastní generování výsledného obrazu.

5.1 První virtuální herci

Pokud se ohlédneme trochu do historie počítačové grafiky, nalezneme řadu pokusů o realistické ztvárnění virtuálních herců. Existuje několik hlavních přístupů jak vizualizovat modely lidských postav počítačem. Zde zmiňované metody jsou čtyři:

- **stick figures**
- **surface model**
- **volume model**
- **multi-layered model**

5.1.1 Stick figures

Jedním z prvních pokusů bylo použití takzvaných stick figures. Princip spočíval ve vykreslování postavy jako soustavy úseček se společnými body v kloubech. Tento přístup má však řadu omezení a vizuální výsledek se dosti oddaluje od realistického vyobrazení člověka. Jeden z prvních stick modelů byl navržen v roce 1970 panem Withrowem [3]. Další známý stick model, pojmenovaný Hipi, byl dokonce hrdinou filmu Dream Flight [4].

5.1.2 Surface model

Dalším způsobem jak zobrazovat lidské postavy je využití povrchové reprezentace (surface models). Kostra postavy je obalena povrchem tvořeným polygony nebo

zakřivenými povrchy (Beziérovy plochy, NURBS apod.). Řada takových modelů byla v minulosti navrhována především pro průmyslové simulace.

Například prvním z takovýchto modelů byl model prvního muže (first man model) sestavený ze sedmi segmentů. Vytvořil jej pan Fetter v roce 1968 pro testování ovládacího panelu letadla Boeing 747. Další známý model byl představen v roce 1969. Třetí modely muže a ženy byly vytvořeny tak, aby jejich složitosti mohly být přizpůsobeny konkrétní aplikaci. Čtvrté Fetterovy modely muže a ženy jsou založeny na jeho dřívější práci na téma vizuální efekty hemisférické projekce.

Jiný model, *Boeman*, byl modelem společnosti Boeing vytvořený pro simulování dosažitelnosti jednotlivých objektů v okolí člověka. Model samotný je složen z 23 spojených segmentů modifikovatelných délek.

Combiman se jmenoval model společnosti Aerospace Medical Research Center navržený pro testování designu kokpitu.

Výhody povrchových modelů spočívají v jejich realistickém vzhledu. Pokud je použit detailní model, může být výsledný statický obraz ve fotorealistické kvalitě. Problémy však nastávají při animování takových modelů. Model se totiž musí přizpůsobit pozici řídící kostry, aby nedocházelo k nespojitostem. Proto byly navrženy různé techniky pro ohýbání povrchových modelů v okolí kloubních spojení.

5.1.3 Volume model

Třetí metoda vizualizace 3D modelů lidských postav je objemová reprezentace. Modely takto vytvořené se skládají z objemových primitiv jako například válců, elipsoidů nebo koulí. Nejjednodušší modely jsou složeny z několika desítek objemových primitiv a neposkytují dobré vizuální výsledky, avšak při použití většího počtu primitiv, které dynamicky mění svůj tvar, může výsledek vypadat velmi realisticky.

Použití válců je výpočetně nejnáročnější z důvodu řešení napojování segmentů a odstraňování neviditelných povrchů.

Mnohem lepší je použití elipsoidů. Tento přístup navrhl Don Herbison-Evans. Popsal také algoritmus pro vykreslování kolidujících elipsoidů pomocí obrysových čar. Tento systém se nazývá NUDES (Numerical Utility Displaying Ellipsoid Solids).

Jako poslední je zmíněno použití koulí. Pan N. Badler z Pensylvánské univerzity vytvořil model člověka tvořený 310 koulemi. Počet zobrazovaných koulí je závislý na pohledu kamery i na vzdálenosti od objektu. Koule jako objemové primitivum má řadu podstatných výhod. Především je to jednoduchá detekce kolizí a také není potřeba zjišťovat přesné otočení, protože koule je v jakémkoliv 2D pohledu vidět jako kruh.

5.1.4 Multi-layered model

Multi-layered modely [7] se skládají z kostry (skeleton layer), mezivrstvy, která simuluje fyzikální chování svalů, kostí, podkožního tuku, a vrstvy kůže. Vzhledem k tomu, že celkový vzhled těla je silně závislý na aktuální struktuře svalové hmoty, podává tento model nejrealističtější výsledky. Hlavní výhodou je to, že jakmile je jednou takový model zkonstruován, závisí jeho stav pouze na kostře a vše ostatní se počítá automaticky. Svaly jsou např. reprezentovány soustavou dynamicky modifikovaných elipsoidů, které mají dány stavy v jednotlivých extrémních polohách a mezi nimi se jejich parametry interpolují.

Pro vizualizaci se používá vygenerování implicitních povrchů na základě geometrie dané mezivrstvou svalů. V některých případech se používá tzv. soft objects (nebo též metaballs).

6 Techniky počítačové animace

Existuje několik animačních technik, které produkují rozdílné výsledky. Výpočetní složitost, nároky na hardware, zkušenost a znalosti animátorů jsou jenom některé z důležitých aspektů ovlivňujících výslednou animaci.

Pro vytváření počítačových animací jsou používány dva základní principy:

- **animace užitím klíčových snímků**
- **procedurální animace**

6.1 Klíčování (key-frame animation)

Tato technika je založena na výpočtu mezisnímků (in-betweens), které interpolují klíčové snímky (key frames) dané animátorem. Ten obvykle vytvoří množinu snímků v důležitých bodech animace a zbylé mezisnímky se dopočítávají automaticky pro získání plynulého pohybu.

6.1.1 Techniky key-frame animace

Existují dvě rozdílné techniky animace užitím klíčových snímků.

- **Interpolace tvaru (Shape interpolation)**

První metoda vychází z předpokladu, že máme k dispozici kompletní startovní a cílový klíčový snímek. Interpolace se pak provádí na snímky jako takové. Pokud je například použit povrchový model, máme k dispozici ve dvou snímcích souřadnice všech bodů tohoto povrchu a interpolace je tedy obyčejná lineární interpolace souřadnic každého z bodů. Tato technika se někdy také nazývá image-based key−frame animation. Je to stará metoda, která byla používána hlavně pro animování čárových kreseb. Animátor zadal klíčové snímky a lineární interpolací byl získáván výsledný pohyb. Nevýhodou této metody je však velmi vysoká paměťová náročnost pro uchovávání klíčových snímků. Také je třeba si uvědomit, že animace je striktně dána počtem klíčových snímků a k vytvoření dalších póz je třeba další snímky doeditovat. Lineární interpolace v tomto případě také způsobuje řadu negativních efektů, pokud je třeba složitějších pohybů postavy. Hlavními nedostatky jsou pak například nedostatečná hladkost pohybu, nespojitosti v rychlosti nebo odchylky při rotaci. Podle některých autorů neexistuje uspokojivé řešení těchto chyb.

- **Interpolace parametrů (Parametric key-frame animation)**

Další technikou je generování mezisnímků interpolací parametrů modelu. Tato technika se také nazývá key-transformation. Model je specifikován množinou parametrů. Animátor nastaví hodnoty těchto parametrů v klíčových bodech animace a mezisnímky pak vznikají vykreslováním modelů vygenerovaných z interpolovaných parametrů. Parametry jsou v tomto případě geometrické transformace (např. translace, rotace kloubů modelu apod.). Tato metoda má velké výhody v mnohem menší náročnosti na paměť. Důležitá je také možnost snazšího generování nových

animací. Lze je totiž generovat částečně nebo i plně automaticky. Další výhodou je použití stejné animace na různé modely, které mají stejnou strukturu (např. stejnou hierarchii kostry). Je pak velice jednoduché přiřadit animaci k jinému modelu a doladit jen problematická místa (např. kolize způsobené různou délkou kostí).

6.1.2 Získávání klíčových snímků

Velmi důležitou otázkou v key-frame animaci je, jakým způsobem jsou klíčové snímky získány. Opět existuje několik metod, které se liší výpočetní složitostí, použitými technickými prostředky a nároky na zkušenosti animátorů.

Zde je výčet základních technik používaných převážně pro modely s pevným skeletem (bone models), pro které je typická key-frame animace interpolací parametrů. Parametry takových modelů jsou matice rotací v kloubech kostry.

- **Motion capture**

Motion capture je metoda snímání pohybu přímo z pohybujících se reálných objektů. K tomu je třeba speciální hardware, který převádí pozici reálných objektů ve scéně do numerické podoby. Tato data jsou předána počítači, který zaznamenává jednotlivé parametry. Z nich lze původní pohyb zrekonstruovat a použít pro animaci modelu. Nevýhodou je, že přesné zařízení pro snímání metodou motion capture je velmi drahé a při použití levnějšího hardware se nedosahuje dostatečně přesných výsledků. Animace pak musí být upravena do výsledné podoby animátorem. Přes zmíněné nevýhody jsou takto získané animace zatím nejrealističtější ze všech známých metod. Pohyb přesně odpovídá pohybu skutečného člověka a animace mají osobitý charakter.

- **Přímá kinematika (forward kinematics)**

Asi nejstarší a stále velmi používaná metoda je metoda přímé kinematiky. Animátor nastavuje pózy modelu ručně, přičemž může manipulovat s úhly v každém kloubu modelu. Přívlastek přímá zde značí fakt, že animátor nemá žádný nástroj pro automatické počítání úhlů v kloubech. Například při návrhu animace ruky sbírající předmět ze stolu, musí animátor nastavit pozici ruky blízko k předmětu. Při použití přímé kinematiky je třeba nejprve nastavit pozici ramena a poté lokte. Pokud ruka na

předmět nedosáhne je třeba opět změnit úhly v rameni a v lokti. Neexistuje zde mechanismus pro automatické spočtení úhlů ze zadané situace.

- **Inverzní kinematika (inverse kinematics)**

Velmi populární metodou hodně využívanou v moderních animačních programech je inverzní kinematika. Tato metoda umožňuje animátorovi mnohem rychlejší a pohodlnější návrh celé animace. Princip spočívá v automatickém dopočítávání úhlů rotací kloubů ze znalosti finální pozice koncového objektu jednoho řetězce segmentů v kostře. Animátor v tomto případě nastaví pouze 3D pozici koncového objektu (například ruky v řetězci nadloktí, předloktí, ruka) a systém za něj dopočte rotace úhlů (v rameni, lokti a zápěstí). Tento výpočet se provádí v reálném čase, takže animátor má stále přehled o výsledku. Tato technika je velmi užitečná pro rychlý návrh animací, avšak její nevýhodou je složitější implementace.

6.2 Procedurální animace

Animace užitím klíčových snímků je velmi rychlá a relativně jednoduchá metoda. Má však své velké nevýhody a to především omezení množství pohybů modelu na množinu pohybů, které lze komponovat z klíčových snímků. Pokud je například vytvořena animace chůze postavy po rovině a chtěli bychom animovat chůzi do kopce nebo po nerovném povrchu, musíme definovat nové klíčové snímky odpovídající danému terénu. V takových případech je řešením procedurální animace. V této metodě je pohyb modelu řízen algoritmicky. Například pro simulaci již zmíněné chůze po nerovném terénu, bude algoritmus zajišťovat pohyb všech důležitých kloubů v těle tak, aby postava držela rovnováhu, detekovala kolize s okolím, reagovala na vnější síly apod. Navržení a implementace realistické procedurální animace je velmi složité, ale lze tak získat velice dobré výsledky. Model může dynamicky reagovat na změny v okolí a speciální situace ve virtuálním světě bez větších paměťových nároků.

7 Interpolace

Volba interpolační funkce může silně ovlivnit výslednou animaci. Použití lineární interpolace pro metodu interpolace tvaru je celkem obvyklé, ale použití této nejjednodušší metody pro interpolaci úhlů v kloubech u parametrické interpolace je už méně vhodné a produkuje nepříliš dobré výsledky. Použití složitějších interpolačních metod je sice většinou zárukou lepší kvality, ale výpočetní složitost těchto metod je někdy nevyhovující zvláště v případech real-time animace.

7.1 Problém interpolace

Problém interpolace je možné formulovat následujícím způsobem: Je dáno $N + 1$ bodů P_i, kde $i = 0$ až N. Najděte křivku $Q(t)$ procházející těmito body.

Existují 2 extrémní řešení tohoto problému:

1. Spojit nezávisle páry bodů $\{ P_i, P_{i+1} \}$ - lineární interpolace.
2. Najít křivku, která prochází všemi body P_i tak, že nalezneme polynom $Q(t)$ stupně N. Pro velký počet N však křivka začne oscilovat a navíc je tato metoda velmi náročná na výpočet.

Mezi těmito dvěma extrémními přístupy existuje ještě další množina interpolačních křivek, které splňují jak základní požadavky na interpolaci, tak se i blíží svou hladkostí k polynomiální interpolaci. Můžeme totiž na interpolační křivku klást další požadavek, kterým je C^m spojitost, přičemž m je stupeň derivace, do kterého je spojitost zachována. Například C^2 spojitost zachovává spojitou nultou, první a druhou derivaci v každém bodě interpolační křivky. V některých případech stačí klást požadavek pouze na geometrickou spojitost G^m. Interpolační křivky, zachovávající spojitost do určitého řádu, se začaly ve větší míře používat v počítačové grafice pro návrhy designu karoserií automobilů. Obecně se tyto křivky nazývají spline a lze je určit několika způsoby. Nejznámější jsou tzv. Beziérovy křivky, B-spline, β-spline a další.

7.2 Typy interpolace

Základní interpolační metody používané v počítačové animaci jsou:

- **lineární interpolace**
- **spline interpolace**

7.2.1 Lineární interpolace

Nejjednodušší interpolací je lineární interpolace. Bod křivky *Q(t)* v čase *t*, pro *t* v int. $\langle t_i, t_{i+1} \rangle$ se vypočte podle vztahu (6).

$$Q(t) = (t_{i+1} - t) \cdot P_i + (t - t_i) \cdot P_{i+1} \qquad (6)$$

Tato interpolace je používána především pro key-frame interpolaci tvaru (shape interpolation). Jejími výhodami jsou rychlost a jednoduchost. Pro interpolaci parametrickou se však nehodí, protože např. u úhlů v modelu způsobuje příliš ostré přechody mezi klíčovými body.

7.2.2 Spline interpolace

Pro výše zmíněné nedostatky lineární interpolace se pro parametrickou key-frame animaci používají metody, které produkují hladší průběh animace zvláště v blízkosti klíčových snímků. Nejvhodnější typy spline křivek jsou kardinální (cardinal), Catmull-Rom a Kochanek-Bartels spline, které umožňují nastavení dalších parametrů.

- **Kardinální a Catmull-Rom spline**

Předpokládejme stejnou situaci jako byla v kapitole 7.1 Problém interpolace. Bod *V* interpolační křivky získáme výpočtem ze čtyř nejbližších bodů P_i (dva předchozí a dva následující) podle (7).

$$V = T \cdot M \cdot P^T \qquad (7)$$

kde *T* je matice $[t^3\ t^2\ t\ 1]$, *M* je matice spline a *P* je matice $[P_{i-1}, P_i, P_{i+1}, P_{i+2}]$. Takto generujeme body pro každý úsek interpolační křivky vždy mezi body P_i a P_{i+1}. Matice *M* má obvyklý tvar (8).

$$M_{car} = \begin{bmatrix} -A & 2-A & A-2 & A \\ 2A & A-3 & 3-2A & -A \\ -A & 0 & A & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad (8)$$

Parametr *A* se nazývá tension a ovlivňuje ostrost ohýbání spline. Speciální případ nastane, když *A=0,5*. Takový spline se nazývá Catmull-Rom spline.

- **Kochanek-Bartels spline**

Kochanek a Bartels [8] navrhli metodu, jak řídit lokální vlastnosti spline pomocí tří parametrů – tension (napětí), continuity (spojitost) a bias (sklon). Opět budeme uvažovat již známou situaci. Rozdíl od předchozích spline je v tom, že pro výpočet bodu *V* se použije pouze dvou nejbližších bodů P_i a P_{i+1} (jeden před a jeden za bodem *V*). Je však třeba zadat i tečné vektory v těchto bodech. Vztah pro výpočet *V* je obdobný jako u kardinálních spline (9).

$$V = T \cdot H \cdot L^T \qquad (9)$$

kde *T* je matice $[t^3\ t^2\ t\ 1]$, *H* je matice Hermitova a *L* je matice $[P_i, P_{i+1}, D_i, D_{i+1}]$. Hermitova matice má tvar (10).

$$H = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -3 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad (10)$$

Pro výpočet tečného vektoru D_i v bodě *Pi* se použije vztah (11).

$$D_i = a \cdot (P_{i+1} - P_{i-1}) \qquad (11)$$

Takto je definován kardinální spline. Catmull-Rom spline dostaneme, když dosadíme $a = 0,5$ (12).

$$D_i = 0,5 \cdot (P_{i+1} - P_{i-1}) = 0,5 \cdot \left[(P_{i+1} - P_i) + (P_i - P_{i-1}) \right] \qquad (12)$$

Zde je vidět, že tečný vektor je průměrem dvou tětiv, počáteční P_i - P_{i-1} a koncové P_{i+1} - P_i. Označíme nyní vektory počáteční derivace DS_i a koncové DD_i. Změnou těchto derivací lze ovládat lokální chování spline křivky v okolí bodů P_i. Jak již bylo zmíněno, používají Kochanek a Bartels tři různé parametry - tension (napětí), continuity (spojitost) a bias (sklon).

Tension je parametr, který ovlivňuje jak ostře se spline ohýbá v blízkosti vrcholu P_i. Tento parametr je v následujících rovnicích označen *t*.

Parametr continuity je značen c a ovlivňuje míru nespojitosti první derivace v bodě. Pokud si například představíme animaci míče odrážejícího se od desky, je zřejmé, že ve chvíli nárazu se mění velmi radikálně směr pohybu. Zde je výhodné nastavit nespojitost tímto parametrem.

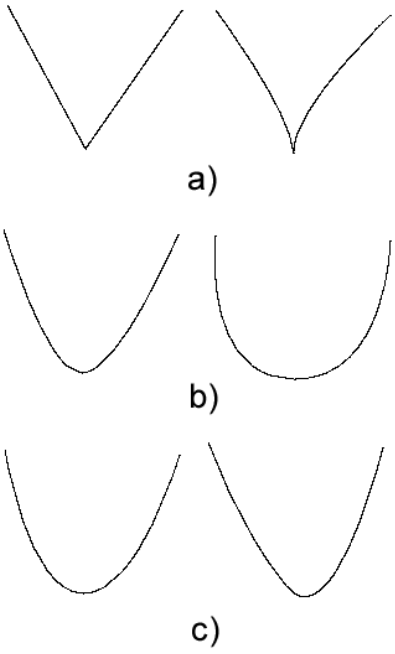
Posledním parametrem je bias označený jako b. Bias umožňuje ovlivnění směru tečny v bodě P_i. Jeho využití je pro animace, kdy je třeba ještě nějakou dobu pokračovat ve stejném směru i po projití bodem P_i.

Chování spline pro různé parametry t, c, b je znázorněno na obr. 5.

Výpočet počáteční a koncové derivace je dán vztahy (13) a (14).

$$DS_i = 0{,}5[(1-t)(1+c)(1-b)(P_{i+1} - P_i) + (1-t)(1-c)(1+b)(P_i - P_{i-1})] \quad (13)$$

$$DD_i = 0{,}5[(1-t)(1-c)(1-b)(P_{i+1} - P_i) + (1-t)(1+c)(1+b)(P_i - P_{i-1})] \quad (14)$$



obr. 5

8 Vlastní vizualizace

Vlastní vizualizací rozumíme finální převod 3D modelu na 2D obrazovku. To lze provést řadou způsobů. Jednotlivé metody se liší hlavně dobou, za kterou je výsledný obraz vyprodukován a kvalitou výstupu.

Tato dvě hlediska jsou zpravidla na opačných koncích. Rozlišujeme tak vizualizaci v reálném čase (realtime), u které je většinou omezena kvalita obrazu a

vizualizaci fotorealistickou, která naopak produkuje velmi dobré vizuální výsledky, avšak její časová náročnost je značná.

Realtime vizualizace se používá tam, kde potřebujeme interaktivitu systému nebo menší paměťové nároky. V běžných případech se dosahuje animačních rychlostí kolem 25 – 30 snímků za sekundu (FPS – frames per second), což je postačující vzhledem k rozlišovacím schopnostem lidského oka.

Oproti tomu fotorealistické zobrazování se používá například ve filmech, reklamních spotech a neinteraktivních animacích, protože v těchto případech nezáleží příliš na tom, jak dlouho trval celkový výpočet. Hlavním cílem je kvalita výstupu.

8.1 Grafické akcelerátory

S vývojem nových technologií ve výrobě hardware, vzrostla kvalita a výkon počítačových komponent, hlavně procesorů a pamětí. Velký dopad to mělo i na zobrazovací zařízení počítačů. Od prvních pouze znakových generátorů se postupně přešlo k černobílým, 4, 16 a 256 barevným grafickým adaptérům, až se začaly za standard považovat highcolor (32K, 64K barev) a truecolor (2^{24} barev) obrazovkové módy. Stejná situace se projevila i ve vzrůstajícím rozlišení obrazovky z počátečních 320×200 obrazových bodů až po dnes standardní 1024×768 a mnohdy i více. Kvalita obrazu tak dospěla do svého finálního stádia. Lidské oko totiž už téměř není schopné rozlišit více.

Rychlejší obvody a větší a levnější paměti byly také příčinou vzniku nového typu zobrazovacích zařízení na počítačích PC. S rozvojem 3D grafiky se začaly vyvíjet 3D grafické akcelerátory, hardware, který umožňuje personálním počítačům dosahovat v grafických aplikacích srovnatelného výkonu s grafickými stanicemi firem SGI a DEC. Princip urychlení pomocí těchto akcelerátorů spočívá v tom, že např. maticové výpočty transformací a zobrazení a základní grafické operace, jako je vykreslení bodu, čáry, polygonu, případně i s osvětlením a texturami, se přenesly do čipu grafické karty a tím se velmi uvolnilo zatížení procesoru. Existuje však řada výrobců takových zařízení a vzniká tedy otázka jakým způsobem tato zařízení využívat pokud možno jednotně. Byla proto navržena nová aplikační rozhraní (Microsoft Direct3D, 3DFx Glide) a z platformy SGI bylo převedeno na PC grafické rozhraní OpenGL.

Každé z těchto rozhraní má své výhody i nevýhody. Pro vědecké aplikace se však používá nejčastěji standardu OpenGL, který má svou historii na platformách SGI.

8.2 Grafická knihovna OpenGL

Jak již bylo řečeno výše, má toto rozhraní základy na platformě SGI. OpenGL je standard s velice striktně dodržovanými zásadami pro pojmenování funkcí a rozšiřování o nové části. Základní verze 1.0 obsahuje velkou množinu grafických operací pro vytvoření kvalitní vizualizace. Nové verze 1.1 a 1.2 jsou vždy vytvářeny rozšířením původní verze o nové prvky, které byly ve starší verzi použity jako tzv. extensions a ustálily se. Zařazení nových funkcí extensions do stávající verze OpenGL je otevřeno pro výrobce grafického hardware a pokud je dané extension hodně využíváno, může být přidáno do nové verze jako součást jádra.

Grafická knihovna OpenGL umožňuje kreslení základních grafických primitiv, kterými jsou bod, úsečka, trojúhelník, čtyřúhelník a polygon. Obsahuje kvalitní lokální osvětlovací model s Gouraudovým stínováním, podporuje texturování geometrických primitiv, alpha blending pro rendering průhledných objektů, stencil buffering, automatické generování souřadnic textur, používá geometrické transformace, projekce a má řadu dalších možností.

Většina dnešních výrobců grafického hardware podporují toto rozhraní a ve svých ovladačích dávají uživateli možnost maximálního využití výkonu v OpenGL aplikacích.

Praktická část

9 Systém pro interpretaci znakové řeči počítačem

Cílem této práce je vytvoření animovaného tlumočníka znakové řeči na počítači. Základní ideou je vytvoření systému, který je schopen zvládnout přehrávat v reálném čase sekvenci animací modelu člověka, vybraných ze slovníku obsahujícího řádově tisíce různých výrazů znakové řeči. V následujících kapitolách jsou podrobněji popsány použité metody a řešení konkrétních problémů.

10 Volba metod

Požadavek na rozsah slovníku a rychlost animace dává jako jednoznačné možné řešení realtime key-frame parametrickou animaci.

Použití animační metody deformace tvaru je nemyslitelné z důvodů paměťové náročnosti pro uchovávání všech póz modelu pro každý znak. Procedurální animace je pak příliš složitá na implementaci a navíc by vyžadovala popis každého znaku algoritmem.

Z vizualizačních technik je nejlepší použít realtime animaci povrchového modelu obsahujícího kolem 5000 trojúhelníků. Realtime animace spočívající v interpolování úhlů segmentů nám dává možnost rychlé rozšiřitelnosti slovníku a libovolného využití systému. Jediným problémem mohou být nároky na grafický hardware pro vizualizaci takového rozsahu scény. Dnešní běžné grafické akcelerátory však mají výkon několikanásobně vyšší. Výhodou povrchového modelu je také velká realističnost výsledku, při celkem nízkých nárocích na výpočet. Problém nastává pouze při deformaci segmentů modelu v blízkosti kloubů, avšak pro běžné pózy používané ve znakové řeči produkuje použitý algoritmus přijatelné výsledky. Multilayered model je sice lepší po vizuální stránce, ale jeho implementace je složitější a výpočetně mnohem náročnější.

Model má hierarchickou strukturu a využívá pro výpočet transformací knihovnu OpenGL. Pro interpolaci pohybu kloubů jsou použity spline křivky, které dávají dobré vizuální výsledky. Editor slovníku znakové řeči obsahuje omezení pohybu

kloubů odpovídající anatomickým možnostem průměrného člověka. Pro navrhování pohybu používá animátor metodu přímé kinematiky s možností rychlého výběru frekventovaných tvarů ruky.

11 3D model pro animaci znakové řeči

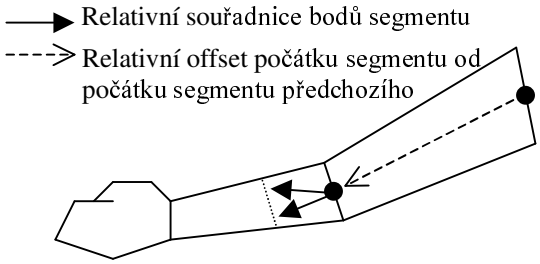
Pro vizualizaci tlumočníka znakové řeči bylo třeba vytvořit realistický 3D model lidské postavy. Takový model lze navrhovat a vytvářet v běžném 3D modelovacím software. Podrobněji je design modelu popsán v kapitole 11.2 Design modelu.

Model je navržen jako surface figure s povrchem tvořeným polygonální sítí. Lze si jej tedy představit jako trojúhelníkovou síť, která je rozdělena po částech na segmenty. Každý segment přísluší k jedné významné kosti v těle, resp. ke skupině kostí. Důležitá je také otázka, jakým způsobem jsou jednotlivé segmenty napojeny. V našem systému je každý segment reprezentován jako zvláštní objekt. Obsahuje tedy seznamy všech svých vrcholů, normál, souřadnic textur a trojúhelníků. Trojúhelníky jsou reprezentovány právě odkazy do zmíněných seznamů (vertexy, normály atd.). Z tohoto vyplývá, že v místě napojení dvou segmentů jsou hraniční body zdvojené. Jsou definovány pro každý segment zvlášť, čímž se sice mírně ztrácí na výkonu, ale struktura modelu tak zůstává mnohem přehlednější. Jakým způsobem se řeší situace při pohybu segmentů – jejich ohýbání, je uvedeno v kapitole 12.3. Deformace segmentů.

11.1 Hierarchická struktura modelu

Struktura modelu použitého v našem systému odpovídá struktuře lidského těla. Model má svou kostru, která je složena ze segmentů a hierarchicky je podobná struktuře kostry člověka. Geometrické parametry kostry jsou obsaženy přímo v parametrech segmentů modelu. Každý segment má souřadnice svých bodů vyjádřeny v relativních souřadnicích vzhledem ke středu kloubu, ke kterému přísluší. Například segment reprezentující předloktí náleží ke kloubu loket. Všechny body tohoto segmentu mají souřadnice vztaženy relativně ke středu lokte. Další informace, která je potřebná ke konstrukci kostry, je délka jednotlivých segmentů, respektive pozice počátku segmentu (střed kloubu) vzhledem k počátku segmentu předchozího.

K tomu slouží parametr *origin*, který vyjadřuje právě toto posunutí ve 3D souřadnicích. Situace je vidět na obr. 6.

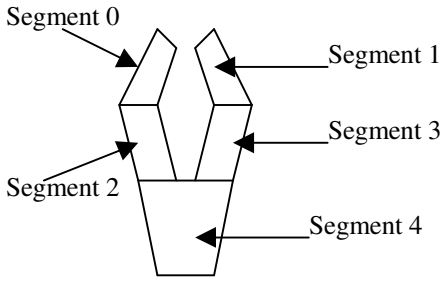


obr. 6

Celkovou hierarchickou strukturu je pak možné vystavět z parametrů segmentu, pokud známe závislosti pořadí napojení segmentů. Hierarchická struktura je velmi výhodná pro následnou vizualizaci v OpenGL. Pro segmenty lze nastavit transformační matice a jejich ukládáním do (resp. vybíráním z) maticového zásobníku vytvořit příslušnou hierarchii. Pro lepší názornost je uveden jednoduchý příklad. Mějme kostru (viz obr. 7) složenou z pěti segmentů.

Postup vizualizace lze popsat příkazy:

1. vykreslit segment 4
2. uložit matici do zásobníku
3. translace do origin segmentu 2
4. vykreslit segment 2
5. translace do origin segmentu 0
6. vykreslit segment 0
7. vybrat matici ze zásobníku
8. uložit matici do zásobníku
9. translace do origin segmentu 3
10. vykreslit segment 3
11. translace do origin segmentu 1
12. vykreslit segment 1
13. vybrat matici ze zásobníku



obr. 7

Taková sekvence příkazů je použita i pro náš systém. Hierarchie tak není reprezentována datovou strukturou tree, ale právě sekvencí příkazů pro zpracování segmentu (deformace, transformace a vykreslení) a ukládání do (resp. vybírání z) maticového zásobníku. Sekvence je v kódované formě uložena v souboru

cmdbuff.txt (viz kap. 11.3.3 Formáty souborů). Použité kódování je uvedeno v tabulce 3.

Kód	Operace
0	Deformace, transformace a rendering segmentu 0
1	Deformace, transformace a rendering segmentu 1
99	Deformace, transformace a rendering segmentu 99
100	Push matrix (uložení matice do zásobníku)
101	Pop matrix (vybrání matice ze zásobníku)

tabulka 3

Omezení na 100 segmentů v modelu je pro případ znakové řeči zcela postačující. Číselné označení segmentů je zcela libovolné, ale je třeba si uvědomit, že toto číslování je shodné s indexy segmentů ve slovníku znaků, a proto pokud se toto číslování změní, není možné použít již vytvořené slovníky. Segmenty jsou číslovány vzestupně od 0 podle svého pořadí v souboru objlist.txt (viz kap 11.3.3 Formáty souborů).

11.2 Design modelu

Vytvoření realistického 3D polygonálního modelu člověka není jednoduchá záležitost. Postup práce designu modelu lze rozdělit do několika kroků.

1. Prvním krokem je nafocení snímků reálného člověka a jejich digitalizace. Tyto snímky v hlavních (nárys, bokorys) a dalších pomocných pohledech se použijí jako předlohy a šablony designéra modelu i jako textury pro výslednou vizualizaci.
2. Hlavní práce spočívá ve vytvoření 3D polygonální sítě, která odpovídá reálnému objektu. Síť také splňuje určité požadavky kladené systémem, jako je rozdělení na segmenty, vytvoření jedné společné vrstvy bodů obou segmentů v každém kloubu a přidání další vrstvy u všech kloubů ve správné vzdálenosti, aby bylo možné použít algoritmus pro ohýbání segmentů při pohybu modelu.

3. Texturování objektu je posledním bodem designu. Ani tato část není jednoduchá. Každému trojúhelníku sítě se přiřadí textura a vrcholům souřadnice do ní.

V našem případě je použito poloautomatické rozdělení objektu na segmenty. Algoritmicky se dopočítají relativní souřadnice segmentů vztažené k počátku segmentu (středu otáčení) a relativní posuny jednotlivých segmentů. Dále se určí, které vrcholy jsou deformovány, a přiřadí se jim příslušné parametry.

11.3 Datové struktury modelu a formáty souborů

11.3.1 Datová struktura modelu

Datová struktura pro uložení 3D modelu tlumočníka znakové řeči je navržena tak, aby byla možná rychlá animace, vizualizace a jednoduchá deformace segmentů v okolí kloubů. Návrh vychází ze znalostí datových formátů komerčních 3D modelovacích aplikací a ze zkušeností práce s OpenGL.

11.3.2 Segment a jeho parametry

Základní parametry segmentu jsou

- **matice transformace a matice k ní inverzní**

Matice transformace je použita v několika situacích. Její hlavní význam je pro deformaci a transformaci segmentů. Přiřazení hodnot do této matice se provádí podle vztahu (15) (viz kap. 12.1 Geometrické transformace a řízení pohybu). Při konečné vizualizaci se tato matice předává OpenGL.

- **úhly natočení**

Tři hodnoty, které udávají orientaci segmentu. Soubor hodnot úhlů všech segmentů jednoznačně určuje stav celého modelu. Časový sled těchto souborů hodnot pak určuje animaci virtuálního tlumočníka.

- **počátek segmentu**

Bod, který udává parametry kostry. Jeho souřadnice jsou relativní vzhledem k počátku předchozího segmentu, na který je tento napojen. Je to také střed kloubu.

- **pole vertexů**

  Pole obsahující 3D souřadnice všech vrcholů daného segmentu.

- **pole normál**

  Pole obsahující 3D souřadnice všech normál daného segmentu.

- **pole souřadnic textury**

  Pole obsahující 2D souřadnice v textuře přiřazené vrcholům všech trojúhelníků daného segmentu.

- **pole trojúhelníků**

  Pole obsahující informace o každém trojúhelníku segmentu. V této položce jsou indexy odkazující do polí souřadnic vrcholů, normál, souřadnic textur a odkazy na příslušné textury, které se mají mapovat na daný trojúhelník.

- **pole koeficientů ohybu a seznamy vrcholů, které se deformují**

  Pole obsahující koeficienty ohybu každého bodu, na který je aplikována deformace, a dva seznamy indexů vrcholů, kterých se deformace týká. Jeden z nich je seznam všech vrcholů ohýbaných společně se segmentem, kterému vrchol přísluší a druhý obsahuje vrcholy, které se deformují se segmentem následujícím.

11.3.3 Formáty souborů

- **popis scény - slang.txt**

  Tento soubor je umístěn ve stejném adresáři jako hlavní aplikace. Obsahuje odkazy na všechny další soubory popisující model a scénu. Struktura souboru je následující:

<cesta k souborům modelu>
<jméno souboru obsahujícího seznam objektů>
<jméno souboru s maticemi bázových transformací>
<jméno souboru se sekvencí příkazů zpracování>
<jméno souboru obsahujícího popis parametrů světel>
pozice.kamery.x pozice.kamery.y pozice.kamery.z
úhel.kamery.heading úhel.kamery.pitch

- **seznam objektů – objlist.txt**

Tento soubor obsahuje seznam názvů všech segmentů, které je třeba načíst ze souborů modelů (*.slm). Na první řádce v souboru je počet segmentů a na dalších řádkách následují názvy souborů *.slm, které se načítají do scény.

- **matice bázových transformací – base.txt**

Pro uchování nastavení orientace souřadné soustavy všech segmentů slouží právě tento soubor. První řádka obsahuje počet záznamů. Submatice 3×3 je uvedena na třech řádcích vždy po třech hodnotách.

Počet.segmentů
$M1_{1,1}$ $M1_{1,2}$ $M1_{1,3}$
$M1_{2,1}$ $M1_{2,2}$ $M1_{2,3}$
$M1_{3,1}$ $M1_{3,2}$ $M1_{3,3}$
$M2_{1,1}$ $M2_{1,2}$ $M2_{1,3}$
$M2_{2,1}$ $M2_{2,2}$ $M2_{2,3}$
$M2_{3,1}$ $M2_{3,2}$ $M2_{3,3}$
…

- **sekvence příkazů (hierarchie) – cmdbuff.txt**

Hierarchická struktura je nahrazena posloupností sekvenčně prováděných příkazů a operací se zásobníkem. V souboru cmdbuff.txt je uložena právě tato sekvence příkazů. Opět je na první řádce uveden počet položek a na každé další jedna celočíselná hodnota zakódované sekvence.

- **parametry světel – lights.txt**

Systém podporuje tři světla. Nastavení parametrů pozice a difúzní složky je uloženo zde. První řádka obsahuje počet položek – musí zůstat 3. Další dvojice řádek udávají parametry světel.

3
L1.position.x L1.position.y L1.position.z
L1.diffuse.r L1. diffuse.g L1. diffuse.b
L2.position.x L2.position.y L2.position.z
L2.diffuse.r L2.diffuse.g L2.diffuse.b
L3.position.x L3.position.y L3.position.z
L3.diffuse.r L3.diffuse.g L3.diffuse.b

- **Formát souborů SLM**

Soubory SLM obsahují veškeré geometrické informace jednotlivých segmentů. V těchto souborech jsou uloženy souřadnice vrcholů, normál, textur, seznamy trojúhelníků modelu, informace o texturách a materiálech, koeficienty deformace a

seznamy deformovaných vrcholů. Pokud je na jedné řádce více položek, jsou v souborech SLM tyto položky odděleny tabulátorem.

Struktura souboru je následující:

1 řádka hlavička
<číslo předchozího segmentu>
souřadnice.počátku.x souřadnice.počátku.y souřadnice.počátku.z
<počet vrcholů segmentu>
souřadnice.vrcholu.x souřadnice.vrcholu.y souřadnice.vrcholu.z
…další souřadnice vrcholů
<počet normál segmentu>
souřadnice.normály.x souřadnice.normály.y souřadnice.normály.z
…další souřadnice normál
<počet souřadnic textur>
souřadnice.textury.x souřadnice.textury.y
…další souřadnice textur
<počet trojúhelníků segmentu>
index.vrcholu index.vrcholu index.vrcholu
index.normály index.normály index.normály
index.texcoord index.texcoord index.texcoord
index.materiálu
…další trojúhelníky
<počet koeficientů deformace>
koeficient.deformace any.real any.real sousední.segment
…další koeficienty
<počet vrcholů deformovaných s tímto segmentem>
index.vrcholu
…další indexy vrcholů
<počet vrcholů deformovaných s následujícím segmentem>
index.vrcholu
…další indexy vrcholů
<počet materiálů>
jméno.materiálu
any.real any.real any.real
any.real any.real any.real
any.real any.real any.real
any.real
opacity
jméno.souboru.textury
…další materiály

12 Animace

Nejen detailní 3D model, ale také vlastní animace a její řízení jsou velmi významné argumenty ovlivňující vizuální výsledek. V této kapitole jsou popsány problémy týkající se animace, jejího řízení, řešení omezení pohybu lidských kloubů v editoru znaků, ohýbání segmentů v blízkosti kloubů a další, které s nimi tématicky souvisí.

Jak již bylo řečeno, náš systém využívá pro animaci virtuálního tlumočníka algoritmus klíčových snímků s interpolací parametrů modelu a následnou zpětnou

syntézou. Parametry jsou v tomto případě hodnoty úhlů rotací v segmentech podle jednotlivých os. Pro výpočet mezihodnot těchto parametrů je použita spline interpolace, která produkuje velmi plynulé pohyby (podobné pohybům člověka), při relativně nízké výpočetní složitosti.

12.1 Geometrické transformace a řízení pohybu

Pohyb lidského těla je velmi složitý. Realistická simulace všech kostí, svalů a kůže v reálném čase je stále nemožná na běžném hardware. Proto je nutné přejít k určitým zjednodušením. Kostra člověka je nahrazena geometrickým modelem složeným ze segmentů. Každý z nich má ve svých datech uloženu transformační matici, která reprezentuje obecnou orientaci v prostoru, komponovanou z matic transformace souřadnic mezi souřadnými systémy scény a segmentu, a dalších tří matic rotace podle vztahu (15):

$$M = B \cdot R_1 \cdot R_2 \cdot R_3 \cdot B^{-1} \qquad (15)$$

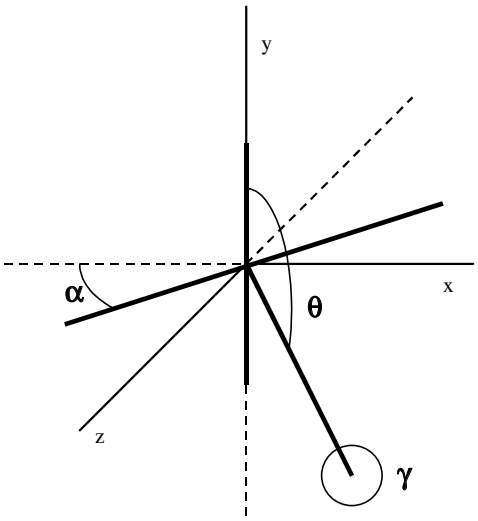
kde M je výsledná matice transformace

B, B^{-1} jsou matice reprezentující transformace mezi jednotlivými souřadnými systémy scény a segmentu (osy rotací nejsou vždy rovnoběžné s osami scény)

R_1, R_2, R_3 jsou matice rotací. V každém kloubu jsou maximálně 3 možné rotace.

Stav segmentu tak lze popsat třemi parametry – hodnotami úhlů v kloubech. Z těchto parametrů se zkonstruují matice rotací R_1, R_2, R_3. Vzhledem k tomu, že se náš model sestává z přibližně 50 segmentů, stačí k reprezentaci stavu celého modelu kolem 150 hodnot. Schematicky je ukázána rotace ve třech osách na obr. 8.

Mezi jednotlivými klíčovými pózami (stavy) modelu se pak interpolací dopočítávají hodnoty úhlů, z nich se konstruují matice transformací podle (15) a ty se předávají do OpenGL.



obr. 8

12.2 Omezení pohybu kloubů

Žádný kloub v lidském těle však není schopen naprosto volného pohybu ve všech osách. Pohyby lidských kloubů jsou omezené. Model který používáme má různé stupně volnosti pohybu segmentů tak, aby jeho omezení odpovídalo možnostem pohybu reálného člověka. Aby byl model lehce ovladatelný a schopen reprezentovat libovolné pózy, byl zvolen jednoduchý princip omezení pohybu. V editoru znaků je umožněno nastavit limity. Nastavují se minimální a maximální hodnoty úhlů v kloubech. Tak lze pohyb omezit nebo dokonce zakázat v jednotlivých osách.

Existuje ještě jiný princip omezení pohybu v kloubech, který je přesnější než námi použitý. Je to vytvoření 2D mapy dosažitelnosti. Princip spočívá ve vytvoření bitové mapy, ve které souřadnice x, y reprezentují normalizované hodnoty úhlů α a θ ve sférickém systému a hodnota v bitové mapě udává, zda je tato kombinace úhlů možná či nikoliv. Konstrukce takové mapy je však složitá a v našem případě nepřináší přílišné výhody. Většinou stejně závisí správné naanimování pohybu na zručnosti animátora.

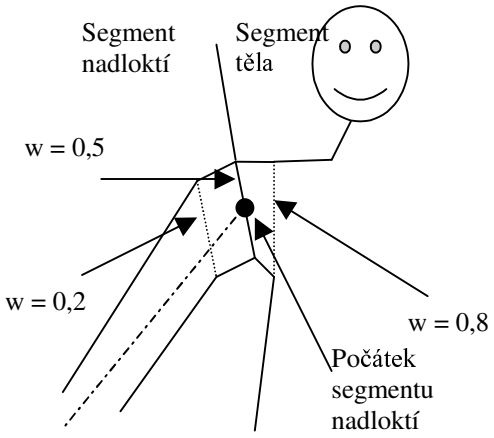
12.3 Deformace segmentů

Jedním z nejkomplikovanějších problémů při vizualizaci modelů lidských postav je řešení ohýbání kloubů. Při použití multi-layered modelů je situace řešena bez větších komplikací, avšak u modelů povrchových je to velmi často diskutovaný

problém [9] [10]. V minulosti byly často používány techniky, které se snaží deformaci segmentů obejít. Řešení například spočívá v umístění koule do každého z kloubů, čímž se zajistí hladký povrch modelu v každé fázi pohybu. Jiné řešení nechává jednoduše segmenty protínat. Tyto techniky mají velkou nevýhodu při texturování. Protínání nedeformovaných objektů pokrytých texturou nedává dobré vizuální výsledky. V naší práci se snažíme řešit ohýbání deformací segmentů v blízkosti kloubů.

12.3.1 Koeficienty ohybu

Ohýbání je založeno na myšlence lokální deformace segmentů. Vrcholy, které jsou společné dvěma segmentům, nebo s takovými body sousedí, mají speciální parametry. Jsou to koeficient ohybu a odkaz na segment, se kterým bod sousedí. Hodnoty koeficientů ohybu byly stanoveny experimentálně tak, aby ohýbané části modelu vypadaly přijatelně ve všech pozicích, které je schopen kloub obsáhnout. Používáme dvě vrstvy ohýbaných bodů: vrstva bodů sdílených dvěma segmenty (hodnota koeficientu *w* se přibližuje 0,5) a druhá vrstva, která obsahuje body, které sousedí s první vrstvou (koeficient *w* má hodnoty kolem 0,2 nebo 0,8 podle dvou případů popisovaných níže). Situaci popisuje obr. 9.



různé hodnoty koeficientů ohybu

obr. 9

12.3.2 Vlastní algoritmus deformace

Nechť model je v základní pozici. Vrcholy všech segmentů mají souřadnice vztaženy relativně ke středu otáčení segmentu. Transformační matice všech segmentů jsou jednotkové matice. Žádné deformace nejsou aplikovány. Nyní změníme jednu z matic (například v rameni). Rotace v rameni změní transformační matici segmentu nadloktí. Vrcholy, které budou touto změnou ovlivněny jsou vrcholy společné pro segmenty nadloktí a tělo a také vrcholy, které jsou s nimi sousedící. Je třeba rozlišit dvě situace.

- **Deformace vrcholů náležejících k otáčenému segmentu (nadloktí)**

V tomto případě se všechny body tohoto segmentu transformují podle transformační matice segmentu. Transformace některých bodů jsou však ovlivněny váhami (koeficient ohybu). Body, které mají váhu rovnu 0 jsou plně otočeny a body s váhou 1,0 zůstávají ve stejné pozici jako před otočením. Mezi těmito extrémními hodnotami se plynule přechází podle hodnoty koeficientu - parametru *w*. Pravidlo pro transformaci je popsáno vztahem (16).

$$M' = M \cdot w + I \cdot (1 - w) \qquad (16)$$

kde *M'* je nová inverzní matice transformace,

M je původní inverzní matice transformace,

w je koeficient ohybu,

I je jednotková matice.

Výsledná matice M' je následně normalizována a body s váhou *w* jsou touto maticí transformovány. Nyní je na místě vysvětlit, proč byla použita inverzní matice, a ne přímo matice transformace. Každý bod segmentu je při zobrazování pomocí OpenGL transformován maticí MODELVIEW. Do této matice nastavíme hodnoty z matice transformace segmentu a tím bez vlastního přepočítávání všech následujících segmentů v hierarchii získáme příslušné otočení. Je tedy jednoduché si nyní představit, že pokud například chceme, aby bod, který se deformuje, zůstal v původní pozici jako před otočením segmentu, je třeba jej vynásobit inverzní maticí transformace segmentu s váhou *w = 1,0*. Následnou aplikací OpenGL transformace se tak dostane do původní pozice.

- **Deformace vrcholů náležejících k předchozímu segmentu v hierarchii (tělo)**

Situace vrcholů příslušejících k segmentu těla je poněkud komplikovanější. Nejprve je třeba přepočítat jejich souřadnice do souřadného systému s počátkem ve středu otáčení (kloub ramena), protože jejich souřadnice jsou vztaženy k počátku segmentu těla. Poté už můžeme aplikovat běžnou rotaci opět podle vztahu (16). Po této transformaci je však ještě třeba aplikovat transformaci podle transformační matice následujícího segmentu, protože vrcholy segmentu těla nejsou při předání do OpenGL transformovány transformační maticí nadloktí. Výsledné souřadnice vrcholů opět translací převedeme do původní souřadné soustavy (vzhledem k počátku segmentu těla). OpenGL poté aplikuje transformaci tohoto segmentu a zobrazí výsledek.

12.4 Datová struktura slovníku

Datová struktura reprezentující slovník znakové řeči je konstruována tak, aby bylo možné do ní uložit jakýkoliv znak. Obsahuje všechny klíčové snímky animace.

12.4.1 Datové struktury

Každý znak se skládá ze dvou řetězců (jméno znaku a řídící řetězec pro další využití), počtu klíčových snímků a pole klíčových snímků. Podle jména znaku se provádí sekvenční vyhledávání znaku ve slovníku a je-li nalezena shoda, předá se index znaku, který se má znakovat.

Pole klíčových snímků je alokováno dynamicky a obsahuje ve svých položkách jednotlivé klíčové snímky reprezentované dobou trvání v milisekundách a polem hodnot úhlů všech segmentů. Modul znakové řeči obsahuje metody pro přidávání a rušení jednotlivých položek slovníku. Celý slovník je pak pole pointerů na jednotlivé znaky. Setřídění podle abecedy se provádí při ukládání slovníku do souboru v editoru.

Datová struktura a správa slovníku je velmi jednoduchá, protože cíl této práce je zaměřen na vizualizaci. Pro další využití by bylo vhodné přístup do slovníku změnit, popřípadě nahradit standardním databázovým systémem. Bude však třeba dbát na rychlost, aby nebyla složitou správou databáze zdržována animace modelu.

12.4.2 Paměťová náročnost a diskový prostor

Každý snímek je reprezentován množinou všech úhlů natočení všech kloubů a dobou, která udává kolik milisekund trvá přechod z předchozího do tohoto snímku. Hodnoty úhlů jsou uloženy ve stupních jako 2 bytové celočíselné. Do souboru se pak ukládají všechny hodnoty úhlů všech segmentů. Celkem tak jeden klíčový snímek v souboru zabírá $2 \cdot 3 \cdot (18 \cdot 2 + 10) + 4 = 280$ B (2B na integer, 3 hodnoty úhlů, 18 segmentů paže a ruky, 2 ruce, 10 dalších segmentů, 4B integer – doba přechodu mezi dvěma snímky). Pokud se znak bude skládat z průměrně 10 klíčových snímků a budeme uvažovat slovník o 10 000 znacích, bude na disku zabírat kolem 28 MB.

V případě uložení slovníku v paměti, jsou nároky o něco větší kvůli zachování obecnosti datové struktury. Úspory na paměť jsou zaměřeny na ukládání na disk. Avšak i tak je rozsah slovníku v paměti přijatelný.

12.5 Alternativní modely

Nevýhodou popisovaného systému je úzká souvislost slovníku s 3D modelem. Pokud bychom totiž chtěli použít jiný 3D model, nebyl by obecně již vytvořený slovník znaků platný. Vyplývá to z geometrických vlastností. Problémy by se týkaly použití modelu s odlišnými délkami segmentů, jinými pozicemi počátků segmentů, rozměry hlavy a dalších částí těla. Model lišící se právě v takových parametrech by totiž při zachování stejných úhlů v kloubech zaujímal odlišné pozice. Je samozřejmé, že existují znaky, u kterých by tyto geometrické odlišnosti příliš nevadily. Například velkou část prstové abecedy lze použít pro velmi rozlišné modely se stále přijatelnými výsledky. Největší problémy ale nastanou v případě, kdy se znak ukazuje oběma rukama nebo rukou na určité části těla (např. na hlavě). U takových znaků je třeba přesně zachovat proporce modelu pro patřičnou animaci, protože i malé změny mohou způsobovat například kolidování rukou s tělem nebo hlavou, kolize levé a pravé ruky, nebo naopak u znaků, u kterých mají např. být ruce spojené, budou v naprosto odlišných polohách.

Řešení tohoto problému je velmi komplikované. Jednou z možností by bylo použití detekce kolizí částí těla navzájem doplněné o částečné deformace a inverzní kinematiku. Tento přístup je však velmi náročný jak implementačně, tak i výpočetně.

V našem systému se tedy spokojíme s omezením na daný model s možností modifikací modelu při zachování hlavních geometrických proporcí.

13 Vizualizace

13.1 Algoritmus vizualizace

V této kapitole shrneme algoritmus, který je používán pro vizualizaci modelu tlumočníka znakové řeči. Jednotlivé kroky byly podrobněji vysvětleny v předcházejících odstavcích.

- **Algoritmus:**

nastavení hodnot úhlů všech segmentů v konkrétním čase dané animace
vypočtení transformačních matic všech segmentů
for všechny příkazy z řídící sekvence příkazů
    case příkaz = ulož matici: glPushMatrix(GL_MODELVIEW);
    case příkaz = vyber matici: glPopMatrix(GL_MODELVIEW);
    case příkaz = zpracuj segment n:
    {
        deformování obou konců segmentu v závislosti na přísl. transf. maticích
        nastavení matice GL_MODELVIEW na hodnotu transformační matice seg
        vlastní rendering
    }
end for

13.2 OpenGL

- **Rendering**

Pro rendering modelu v OpenGL byly použity základní metody vizualizace v tomto API. Model se vykresluje jako množina osvětlených, otexturovaných trojúhelníků předáním souřadnic vrcholů, normál a textur. Použití urychlovací

techniky displaylist je nemožné, protože velká část scény je dynamická (deformace segmentů při pohybu kloubů).

Renderer umožňuje nastavení různých režimů zobrazení (viz příloha A Uživatelská dokumentace).

- **Lighting**

Osvětlovací model OpenGL je lokální. Znamená to, že trojúhelníky jsou osvětleny bez realistických stínů a bez vlivu okolních objektů. Pro každý trojúhelník OpenGL spočítá pouze základní osvětlovací rovnici a tou vyčíslí hodnoty barev všech vrcholů trojúhelníka. Uvnitř trojúhelníka se pak tyto hodnoty lineárně interpolují. Na osvětlení trojúhelníků mají vliv pouze světelné zdroje. Při použití osvětlovacího modelu OpenGL je třeba správně nastavit parametry textur tak, aby výsledná barva byla kombinací barvy textury a osvětlovacího modelu. Toto nastavení se provede předáním parametru GL_MODULATE funkci glTexEnvf.

- **Texturing**

Pro zvýšení detailnosti modelu při zachování téměř stejného výkonu je využito mapování textur. Každý objekt má pro každý vrchol určeny 2D souřadnice a referenci na texturu, která se na něj mapuje. Texturování pak zařizuje samo OpenGL, kterému se předají tyto souřadnice a nastaví se aktivní textura, která byla předtím do OpenGL také předána. Rozlišení textury vyvažuje hrubší strukturu polygonů a celkový vizuální výsledek je velice dobrý. Zvolená metoda deformace segmentů má také velkou výhodu právě pro texturování, protože jiné algoritmy produkují různé diskontinuity textury, což má velmi rušivý efekt ve výsledném obraze.

- **Alpha blending**

Alpha blending je technika sloužící k renderingu průhledných polygonů. V případě tohoto systému byla technika blendingu použita pouze okrajově a to pro realističtější zobrazení skel brýlí virtuálního tlumočníka.

14 Dosažené výsledky

Tato práce dokumentuje systém pro vizualizaci znakové řeči počítačem. Systém je vyvinut v prostředí Delphi 4.0 pod systémem Windows NT. Na počítači s běžným grafickým akcelerátorem (viz doporučená konfigurace) pracuje v reálném čase.

Součástí systému je editor (***SLEdit***) znaků, který umožňuje vytvoření téměř libovolné animace pohybu lidských rukou a uložení takové animace do souboru slovníku. Ovládání editoru je přizpůsobeno potřebám a možnostem znakové řeči. Při vytváření nových znaků je výhodou automatické omezení pohybu do značné míry odpovídající reálnému omezení lidských pohybů. Urychlení práce s editorem spočívá také v možnosti výběru tvaru ruky z databáze nejběžněji používaných tvarů.

Další aplikací je ukázkové využití modulu pro vytvoření jednoduchého tlumočníka z psaného textu do znakové řeči (***Tlumocnik***). Tato aplikace je schopna převádět psaný text v českém jazyce do vizuální podoby české znakové řeči, která je oficiálním dorozumívacím prostředkem neslyšících v naší zemi.

V tomto dokumentu jsou prezentovány algoritmy použité pro řízení pohybu počítačového modelu člověka animovaného technikou klíčování - interpolace parametrů a zobrazovaného jako povrchový model. Je popsána relativně rychlá metoda lokální deformace segmentů v blízkosti kloubních spojení při rotacích v kloubech, která dává dostatečně kvalitní výsledky.

Pro rychlý a kvalitní grafický výstup je použito grafické rozhraní OpenGL, které využívá výhody moderních grafických akcelerátorů.

Statické ukázky výstupu lze vidět v příloze D.

- **Doporučená konfigurace**

CPU:	Intel Pentium II 300 MHz
Memory:	64MB RAM
Graphics Card:	NVidia Riva TNT2 (nebo srovnatelná s podporou OpenGL)
OS:	Windows NT 4.0

15 Poděkování

Na tomto místě je třeba poděkovat všem, kteří přispěli k vytvoření této práce. V prvé řadě Jakubovi Marešovi a Pavlovi Samcovi za vytvoření 3D modelu, dále Prof. Ing. Václavu Skalovi, CSc., všem grafikům ze Západočeské univerzity v Plzni a grafické skupině University de Girona ve Španělsku za věcné připomínky a komentáře a rodině a přátelům za morální podporu.

16 Reference

[1] Gabrielová,D., Paur,J., Zeman,J.: Slovník znakové řeči, Horizont, Praha 1988

[2] Hamill,J., Knutzen,K.M.: Biomechanical Basis of Human Movement, Williams & Wilkins 1995

[3] Withrow,C.: A Dynamic Model for Computer-Aided Choreography, Computer Sci.Dept., Univ.of Utah, 1970

[4] MIRALab, University of Geneva: Dream Flight, 1982

[5] Thalmann,N.M., Thalmann,D.: New Trends in animation and visualization, John Wiley, 1991

[6] Thalmann,N.M., Thalmann,D.: Computer Animation Theory and Practice - Second Revised Edition, Springer-Verlag 1985, 1990

[7] Chadwick, J.E., Haumann, D.R., and Parent, R.E.: Layered construction for deformable animated character, Computer Graphics, 23, (3), 1989, pp.243-252

[8] Kochanek, D.H.U., Bartels, R.H.: Interpolating Splines With Local Tension, Continuity, And Bias Control, SIGGRAPH(84), pp. 33-43.

[9] Babski,Ch., Thalmann,D.: A Seamless Shape for HANIM Compliant Bodies, http://ligwww.epfl.ch/~babski/StandardBody/Deformation

[10] Daniel Thalmann, Jianhua Shen, Eric Chauvineau: Fast Realistic Human Body Deformations dor Animation and VR Applications, Computer Graphics International IEEE Society Press, June 1996, pp. 166-174

[11] http://www.itl.nist.gov/iaui/ovrt/projects/vrml/h-anim/jointInfo.html

[12] McDonald,J., Alkoby,K., Berthiaume,A., Chomwong,P., Christopher,J., Davidson,M.J., Furst,J., Konie,B., Lancaster,G., Lytinen,S., Roychoudhuri,L., Sedgwick,E., Tomuro,N., Toro,J., Wolfe,R.: An Improved Articulated Model of the Human Hand, pp.306-311, in WSCG2000 Int.Conf. proceedings, 2000

[13] Skala,V.(Ed): WSCG'2000 Int.Conf. proceedings, Univ. of West Bohemia, Plzen, Czech Republic, 2000

[14] DePaul University, Chicago: American Sign Language Synthesizer Project, http://asl.cs.depaul.edu/aslhome.html

[15] H-Anim – Humanoid animation working group, http://ece.uwaterloo.ca/~h-anim

[16] Computer Graphics Lab (LIG) at the Swiss Federal Institute of Technology: http://ligwww.epfl.ch

[17] Mathematics sources: http://mathworld.wolfram.com

[18] Harold V. McIntosh: Linear Algebra, http://delta.cs.cinvestav.mx/red/alglineal/algebra.html

[19] Jiří Žára, Bedřich Beneš, Petr Felkel: Moderní Počítačová Grafika, Computer Press Praha 1998

[20] Waite group Press: OpenGL Super Bible, http://www.itknowledge.com/reference/archive/1571690735/ewtoc.html

[21] OpenGL homesite: http://www.opengl.org

[22] OpenGL Manual Pages

[23] Mark Segal, Kurt Akeley: The OpenGL Graphics System: A Specification (Version 1.2.1)

[24] Way cool, way fast OpenGL techniques: http://reality.sgi.com/opengl/tips

[25] OpenGL Red Book

[26] Advanced Graphics Programming Techniques Using OpenGL, Siggraph'98 Course

[27] OpenGL Game Developers Mailing List

Přílohy

A Uživatelská dokumentace

Možnosti rendereru

Modul pro vizualizaci znakové řeči obsahuje renderer. Okno, které rendereru patří a kam se rendering provádí, obsahuje menu, přes které je možno měnit nastavení zobrazování. Menu se zobrazí při stisku klávesy F10, pokud je aktivní okno rendereru.

Jednotlivé položky menu jsou popsány zde:

- **Rendering mode**

Point - zobrazují se pouze body modelu

Wire - model se zobrazuje jako drátěný

Flat - model se zobrazuje jako množina plných trojúhelníků s konstantním stínováním

Gouraud - model se zobrazuje jako množina plných trojúhelníků s Gouraudovým stínováním

- **Camera movement**

Walk - se stisknutým levým tlačítkem myši se otáčí kamerou v horizontálním a vertikálním směru a s pravým tlač. se pohybuje kamerou vlevo, vpravo, vpřed a vzad ve stejné výšce nad podlahou

Fly - se stisknutým levým tlačítkem se otáčí kamerou v horizontálním a vertikálním směru a s pravým tlač. se pohybuje kamerou vlevo, vpravo, vpřed a vzad ve směru pohledu

UpDown - se stisknutým levým tlačítkem se otáčí kamerou v horizontálním a vertikálním směru a s pravým tlač. se pohybuje kamerou vlevo, vpravo, nahoru a dolu vzhledem k podlaze

Examine - stiskem levého tlačítka se určí bod, na který kurzor myši ukazuje a model se posune tak, aby tento bod ležel uprostřed obrazovky. Poté lze modelem

otáčet ve dvou osách při stále stisknutém levém tlačítku. Se stisknutým pravým tlačítkem myši lze měnit vzdálenost kamery od modelu.

- **Rendering options**

Model	- zapíná/vypíná zobrazování modelu tlumočníka
Skeleton	- zapíná/vypíná zobrazování kostry modelu
Textures	- zapíná/vypíná mapování textur
Lighting	- zapíná/vypíná osvětlovací model OpenGL
Backface culling	- zapíná/vypíná odstraňování odvrácených trojúhelníků
Bending info	- zapíná/vypíná zobrazování koeficientů ohybu v bodech
Normals	- zapíná/vypíná zobrazování normál
Floor	- zapíná/vypíná zobrazování podlahy

- **Reset view**

Nastavuje pozici a otočení kamery na počáteční hodnoty.

Editor znaků *SLEdit.EXE*

Aplikace SLEdit je určena pro management slovníku znakové řeči. Její možnosti jsou rozsáhlé. Základní funkce této aplikace je vytváření slovníku znakové řeči, umožňuje však i modifikaci parametrů kostry a nastavení omezení pohybu kloubů pro snazší manipulaci s modelem. Nejprve budou popsány funkce všech kontrolních prvků a poté jednoduché příklady jak postupovat při vytváření nových znaků a jak modifikovat parametry kostry.

- **Dictionary**

Load	- nahrání souboru slovníku z disku do paměti
Save	- uložení souboru slovníku na disk
Insert	- přidání nového znaku do slovníku na aktuální pozici. Pokud je zaškrtnuta možnost AutoSort, zatřídí se slovo po přidání na své místo podle abecedy. Při vytvoření nového znaku se do parametrů znaku uloží aktuální hodnoty.
Del	- vymazání znaku ze slovníku
Sort	- setřídění slovníku podle abecedy

AutoSort - pokud je zaškrtnuto, třídí se slovník podle abecedy s každou modifikací položky Sign

Listbox - v listboxu lze vybrat aktuální znak, se kterým se bude pracovat

Sign - řetězec reprezentující znak textovým popisem – je to jednoznačný identifikátor znaku a podle tohoto řetězce se vyhledává znak ve slovníku. Synonyma – stejné znaky s různými textovými reprezentacemi se vytváří tak, že se čárkou oddělí v řetězci Sign.

Control string - kontrolní řetězec. Zatím nemá zvláštní využití.

- **Keyframes**

Insert - vložení dalšího klíčového snímku do sekvence znaku

Del - zrušení klíčového snímku v sekvenci

Number of - zobrazuje aktuální klíčový snímek a celkový počet klíč. snímků ve znaku

Delay - nastavení doby, za jak dlouho má animace přejít do tohoto klíčového snímku ze snímku předchozího. Hodnota je v milisekundách.

- **Angles**

Ovládací globus - ovládací globus umožňuje pohybovat aktuálním segmentem ve dvou osách současně. Při stisknutém levém tlačítku pohybujeme prvními dvěma úhly (pokud je pro ně pohyb povolen) – abdukce, addukce, flexe a extenze. Při stisku pravého tlačítka měníme pohybem myši třetí úhel – rotace.

Úhly - pro přesné nastavení jednotlivých úhlů slouží tři SpinEdit komponenty. Ovládání pomocí ovládacího globu má totiž nevýhodu v tom, že je obtížné změnit pouze jeden z prvních dvou úhlů. SpinEdity pro jednotlivé úhly jsou upravené a nabízí možnost rychlé modifikace jednotlivých úhlů tahem myši v horizontální ose při stisknutém tlačítku.

Reset - toto tlačítko nastavuje všechny tři hodnoty úhlů na 0.

Limits - slouží pro zobrazení dialogu nastavení krajních mezí úhlů popsaného níže.

- **Segment**

First	- nastaví aktuální segment na první segment ruky (pravé nadloktí)
Prev	- nastaví aktuální segment na předchozí
Next	- nastaví aktuální segment na další
Last	- nastaví aktuální segment na poslední segment (levá dlaň)
L↔R	- přepnutí aktuálního segmentu z jedné ruky na druhou
ComboBox	- výběr aktuálního segmentu lze provést také v tomto ComboBoxu

- **Skeleton**

X,Z,Y	- tahem myši se stisknutým tlačítkem na těchto třech komponentách otáčíme lokálním souřadnicovým systémem segmentu. Měníme tak směry os rotace otáčením těchto systémů v osách scény.
R,G,B	- podobné jako předchozí s rozdílem, že tentokráte otáčíme lokálními systémy segmentů v jejich vlastních osách. Pokud je v rendereru zapnuto zobrazování kostry (skeleton), jsou osy souřadnicového systému označeny základními barvami Red, Green, Blue. Kontrolní prvky R, G, B umožňují otáčení podle příslušných os.
Reset	- pro uvedení lokálních os segmentu do počáteční polohy slouží tlačítko reset.
Undo	- návrat k předchozí hodnotě
L	- Lock způsobí natočení červené lokální osy do stejného směru jako je osa segmentu.
→	- kopírování nastavení lokálních systémů segmentu pravé ruky do levé ruky
Load	- nahrání orientace lokálních systémů segmentů z disku
Save	- uložení orientace lokálních systémů segmentů na disk

- **Animation**

First	- nastaví aktuální snímek na první ve znaku
Prev	- posune o jeden klíčový snímek zpět
Stop	- zastaví animaci a přepne z režimu přehrávání animace do režimu editace

Play - spustí přehrávání animace (přepne do režimu přehrávání animace)

Next - posune o jeden klíčový snímek dopředu

Last - nastaví aktuální snímek na poslední klíč. snímek ve znaku

- **Copy**

Copy sign - zkopíruje znak do dočasné paměti

Paste sign - zkopíruje znak z dočasné paměti

Copy pose - zkopíruje klíčový snímek do dočasné paměti

Paste pose - zkopíruje klíčový snímek z dočasné paměti (s ohledem na Hand only)

Basics - zobrazí dialog pro výběr z frekventovaných tvarů ruky

Left → Right - zkopíruje pózu levé ruky do pravé (s ohledem na Hand only)

Right → Left - zkopíruje pózu pravé ruky do levé (s ohledem na Hand only)

Hand only - pokud je zaškrtnuto Hand only, nekopíruje se nastavení ramene, lokte a zápěstí.

- **Mapa kostry**

Mapa kostry umožňuje rychlou selekci segmentu. Aktivní segment je vyznačen červeně, stejně jako segment, na kterém je kurzor. Výběr segmentu se provádí stiskem tlačítka. Po určité době bez pohybu kurzoru, nebo po opuštění mapy, segment nad kterým byl kurzor zhasne, zatímco aktivní segment zůstane vyznačen červeně.

- **Basics**

Přes tento dialog je možno velmi rychle nastavit tvar ruky přiřazením jednoho z frekventovaných tvarů. Stačí kliknout na příslušný obrázek. Pokud chceme frekventovaný tvar ruky změnit, lze kliknutím s podržením klávesy Shift uložit aktuální pózu aktuální ruky.

Load - nahrání frekventovaných tvarů ruky z disku

Save - uložení frekventovaných tvarů ruky na disk

Autoclose - pokud je tento CheckBox zaškrtnutý, dialog basics se zavírá po každém výběru z frekventovaných tvarů automaticky.

- **Limits**

Min - minimální hodnota příslušného úhlu

Max - maximální hodnota příslušného úhlu

Load - nahrání limitních hodnot úhlů z disku

Save - uložení limitních hodnot na disk

R→L - kopírování limitních hodnot úhlů z pravé ruky do levé

Jednoduchý příklad editace znaku

Nyní popíšeme vytvoření jednoho znaku znakové řeči v editoru SLEdit. Po spuštění editoru je třeba nahrát slovník, ve kterém chceme slovo vytvářet tlačítkem Load v sekci Dictionary. Poté zadáme text reprezentující znak v editovacím okně Sign. Budeme například vytvářet slovo „ahoj“. Do editovacího okna Sign tedy zapíšeme „ahoj“ a začneme vytvářet klíčové snímky. Nyní změníme počet klíčových snímků na 2 pomocí *insert* a *del* v sekci *keyframes*. Do každého z nich přiřadíme tvar ruky „U“ ze základních tvarů (basics – obr. „U“, zavřít basics, další klíčový snímek, opět Basics – „U“, zavřít). A oběma klíčovým snímkům nastavíme hodnotu delay na 500 ms. Nyní nastavíme v prvním snímku hodnoty úhlů ramene pravé ruky na 7,-21,73; úhly lokte na 116,0,0 a úhly zápěstí na 0,0,0. Levou ruku necháme v původní pozici. V druhém snímku nastavíme hodnoty pravého ramene na 44,-10,60; lokte na 64,0,0 a zápěstí 0,18,0. Nyní už můžeme celý znak přehrát stiskem tlačítka play v sekci animation.

Nastavení parametrů kostry

Směry lokálních os segmentů kostry lze měnit. Například osy rotací palce v základní poloze modelu je vhodné nastavit tak, aby se základní pohyby jako třeba flexe a extenze prováděly rotací kolem jedné osy. Pokud bychom použili osy rovnoběžné s osami scény, byla by tato rotace velmi složitá. K modifikaci os otáčení segmentů slouží v editoru sekce Skeleton. Tlačítka X,Y,Z umožňují rotaci lokálního souřadnicového systému segmentu podle os rovnoběžných s osami scény a tlačítka R,G,B pak rotace podle jednotlivých os lokálního systému. Pro přesné pochopení je vhodné si vypnout zobrazování modelu, zobrazit pouze skeleton a přiblížit si aktuální kloub.

Tlumočník znakové řeči *Tlumocnik.EXE*

Ukázková aplikace tlumočníka znakové řeči je velmi jednoduchá. Skládá se z okna rendereru a okna aplikace, které obsahuje Memo pro psaní textu a 4 tlačítka Přelož, Stop, Vymaž a Nahraj slovník.

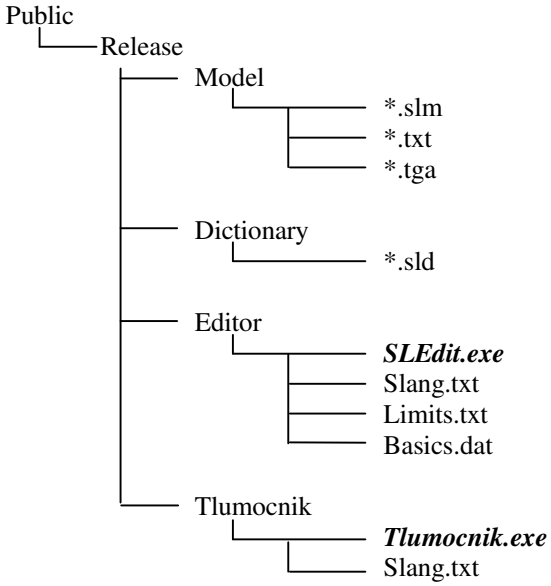
Memo box	- slouží zapsání textu, který chceme překládat
Přelož	- spustí překlad
Stop	- zastaví animaci
Vymaž	- smaže obsah memo boxu
Nahraj slovník	- zobrazí dialog pro nahrání slovníku znaků

Po spuštění aplikace se zobrazí dialog pro nahrání slovníku. Ukázkový slovník first.sld je v adresáři Public/Release/Dictionary a obsahuje pouze několik slov. Text Memo boxu, který se objeví po spuštění demonstruje základní funkce slovníku. Pokud slovo není nalezeno ve slovníku, systém jej rozloží na písmena a ta vyznakuje prstovou abecedou.

B Instalace

Aplikace ***SLEdit*** a ***Tlumocnik*** nemají žádný instalační program. Jejich instalaci je proto možné provést manuálně zkopírováním příslušného adresáře na pevný disk. Obě aplikace vyžadují operační systém Windows NT nebo Windows 98 (nejsou otestovány pro Windows 2000, ale i s tímto operačním systémem by měly pracovat bez problémů).

Na přiloženém CD jsou aplikace umístěny v adresáři *Public\Release*. Struktura adresářů je následující:



V adresáři *Model* jsou všechny soubory popisující scénu a 3D model tlumočníka. *Dictionary* je datový adresář pro uložení slovníků. *Editor* obsahuje aplikaci ***SLEdit*** a její datové soubory. V adresáři *Tlumocnik* je uložena aplikace ***Tlumocnik*** a příslušný datový soubor obsahující odkaz na 3D model a scénu.

Tuto strukturu lze zkopírovat na pevný disk. Pro modifikaci slovníku slouží aplikace ***SLEdit*** v adresáři *Editor* a pro prezentaci výstupu je určena aplikace ***Tlumočník***.

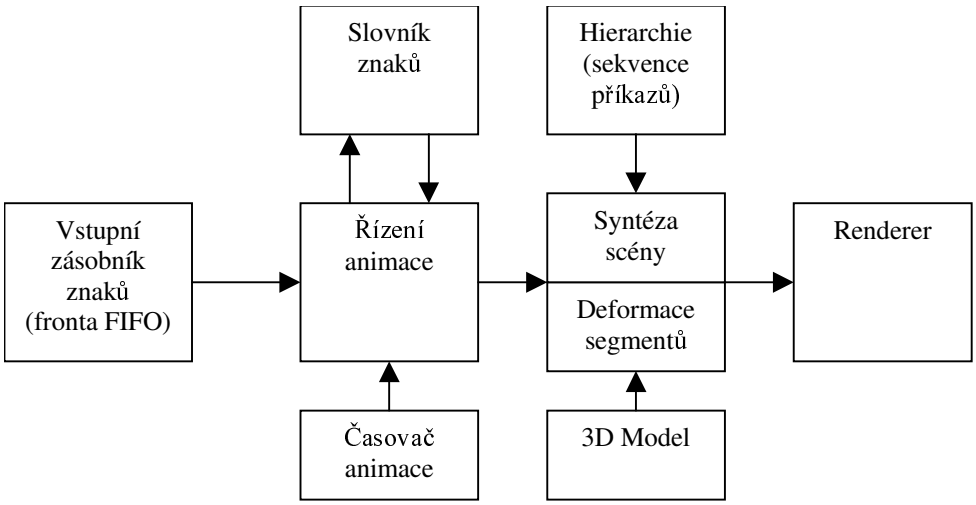
C Programátorská dokumentace

Popisovaný systém je vyvíjen v prostředí Borland Delphi na platformě Windows NT. Renderer, správa slovníku a řízení animace jsou spojeny do jednoho modulu (unity) SLI2. Interface tohoto modulu s okolím se skládá z několika propojovacích funkcí, které slouží ke spouštění animace, správě slovníku a nastavení parametrů zobrazení. Aplikace pak mohou využitím tohoto modulu zobrazovat znakovou řeč.

Modul SLI2

- **Schéma koncepce**

Modul není stavěn objektově. Všechny funkce interface mají strukturu běžných funkcí a procedur. Z objektového návrhu aplikace v Delphi jsou pro modul použity pouze třídy TForm pro vyrobení okna rendereru a TThread pro řízení rychlého renderingu nezávislého na okolním běhu aplikace. Vnitřní strukturu lze popsat schématem (obr. 10).



obr. 10

Slovo reprezentované indexem ve slovníku je předáno modulu a zařazeno do vstupního zásobníku (kruhové fronty FIFO). Poté v závislosti na časovači je slovo předáno řídícímu modulu animace, který si zjistí potřebná data ze slovníku a předá interpolované parametry do části pro syntézu scény. Scéna je syntetizována z 3D modelu, výstupních parametrů řízení animace a použitím sekvence příkazů zajišťujících hierarchickou strukturu. Výsledek syntézy je předán rendereru, který celou scénu zobrazí přes OpenGL.

- **Datové typy**

Veškeré datové typy jsou tvořeny tak, aby byly přehledné, a aby je bylo možné použít univerzálně. Často jsou používány struktury (*record*), které poskytují velkou přehlednost kódu. Datový typ je pojmenován unikátním jménem s předponou *T*, která je typická pro datové typy v Delphi (např. *TPoint3D*). Ke každé struktuře je vytvořen typ ukazatel na tuto strukturu a je pojmenován s předponou *P* (např. *PPoint3D*). Dále je většinou třeba vytvořit pole datových struktur. To je zajištěno deklarací typu s délkou pole omezenou na rozsah indexů v rozsahu typu *word*. Pro pole struktur je stanovena předpona *TA* (*TAPoint3D*). Pokud však chceme pole použít, zabírala by taková datová struktura příliš paměti při statické deklaraci proměnných, a proto použijeme typ ukazatel na pole struktur a proměnnou alokujeme dynamicky. Typ pointer na pole struktur má předponu *PA* (*PAPoint3D*).

Nejdůležitější datové typy jsou použity pro reprezentaci modelu. Uvedeme stručný přehled nejvýznamějších v základní podobě (bez ukazatelů a polí).

Geometrické typy

Souřadnice bodů, normál, textur a hodnoty úhlů v kloubech jsou ukládány do čtyř datových typů podle dimenze a typu jednotlivých položek.

```
TPoint2D = record x,y:TGLFloat;end;
TPoint3D = record x,y,z:TGLFloat;end;
TPoint4D = record x,y,z,w:TGLFloat;end;
TPoint3DI = record x,y,z:SmallInt;end;
```

Parametry pro ohýbání segmentu

Informace o tom jak deformovat body segmentů jsou uložené v datové struktuře *TBendw*. Každý bod segmentu má tuto informaci. Z floatových parametrů je použit pouze parametr a, který je koeficientem ohybu popisovaným v předchozím textu. Parametr *seg* odkazuje na segment, který řídí deformaci daného bodu.

```
TBendw = record
           a,b,t:TGLFloat;
           seg:integer;
         end;
```

Popis trojúhelníku

Trojúhelníky modelu jsou reprezentovány strukturou obsahující indexy vrcholů, normál, souřadnic textur a materiálů. Indexy odkazují do polí hodnot obsažených ve struktuře *TModel*.

```
TFace = record
        a,b,c:integer;    // indexy vrcholu
```

```
    na,nb,nc:integer; // indexy normal
    ta,tb,tc:integer; // indexy souradnic textury
    material:integer; // index materialu
  end;
```

Materiály

Model má seznam materiálů, které jsou mapovány na jeho trojúhelníky. Datová struktura materiálů obsahuje řadu položek, přičemž pouze některé jsou využity. Po nahrání všech modelů z disku do paměti se projdou všechny materiály a nahrají se textury. Pokud dva modely sdílí stejný materiál (rozlišuje se podle *id* materiálu), sdílí i stejnou texturu (nejsou duplicitně v paměti). Nahrávání textur bez duplicit zajišťuje zvláštní procedura.

```
TMaterial = record
        id:string[50];        // jednoznacny identifikator materialu
        diffuse:TColor3F;     // barva - nevyuziva se
        specular:TColor3F;    // barva - nevyuziva se
        emission:TColor3F;    // barva - nevyuziva se
        shininess:TGLFloat;   // lesklost - nevyuziva se
        opacity:TGLFloat;     // pruhlednost - pouziva se pro bryle
        texname:string[50];   // jmeno textury
        texindex:integer;     // index do seznamu textur
      end;
```

Model

Každý segment je reprezentován proměnnou typu *TModel*. Model obsahuje především veškeré geometrické informace segmentu, jako jsou souřadnice vrcholů, normál, textur, koeficienty ohýbání, informace o trojúhelnících, dále pak informace o texturách a materiálech, matice transformací a matice lokálního osového systému, úhly natočení v kloubu apod.

```
TModel = record
        prevseg:integer;          // odkaz na predchozi segment
        origin:TPoint3D;          // pocatek segmentu - rel souradnice vzhledem k predchozimu seg
        angles:TPoint3D;          // floatove uhly - naplni se pri interpolaci z IntAngles(integerove)
        IntAngles:TPoint3DI;      // integerove uhly - naplni se pri editaci
        basematrix:TMatrixf;      // matice transformace os segmentu
        invbasematrix:TMatrixf;   // inverzni matice k predchozi
        matrix:TMatrixf;          // transformacni matice - transformace a deformace
        num_vertex:integer;       // pocet vertexu v poli vertexu
        vertex_set:PAPoint3D;     // pole vertexu
        num_color:integer;        // pocet polozek barvy vrcholu - nevyuziva se
        color_set:PAColor3F;      // pole barev - nevyuziva se
        num_normal:integer;       // pocet normal v poli normal
        normal_set:PAPoint3D;     // pole normal
        num_texcoord:integer;     // pocet souradnic textury
        texcoord_set:PAPoint2D;   // pole sour. textury
        num_face:integer;         // pocet trojuhelniku
        face_set:PAFace;          // pole trojuhelniku
        num_bendw:integer;        // pocet parametru pro ohyb
        bendw_set:PABendw;        // pole param. ohybu
```

```
    num_deformprev:integer;    // pocet vrcholu ohybanych s timto segm.
    deformprev_set:PAInteger;  // pole vrcholu ohybanych s timto segm.
    num_deformnext:Integer;    // pocet vrcholu ohybanych s dalsim seg.
    deformnext_set:PAInteger;  // pole vrcholu ohybanych s dalsim seg.
    num_material:integer;      // pocet materialu tohoto segmentu
    material_set:PAMaterial;   // pole materialu tohoto seg.
  end;
```

Slovník a klíčové snímky

Datová reprezentace slovníku je jednoduchá. Každý znak (*TSign*)se skládá z množiny klíčových snímků v chronologickém pořadí a z dalších informací (text znaku, počet klíč. snímků a řídící řetězec). Klíčové snímky jsou pak reprezentovány datovou strukturou *TPose*, která obsahuje jednu celočíselnou hodnotu delay (dobu trvání přechodu z předchozího snímku do tohoto) a pole celočíselných hodnot úhlů ve všech kloubech.

```
TPose = record       // definovana staticka faze modelu – klic. snimek
    delay:integer;              // doba trvani snimku
    angles:PAPoint3DI;          // nastaveni uhlu
  end;

TSign = record                  // znak znakove reci
    StrWord:shortstring;     // slovni popis
    Speech:shortstring;      // ridici retezec pro pohyb ust
    num_pose:integer;        // pocet poz ve slove
    pose_set:PAPose;         // jednotlive pozy – klic snimky
  end;
```

- **Proměnné**

Camera

Parametry kamery jsou uloženy v této proměnné. Význam parametrů je komentován přímo v kódu.

```
Camera:record                   // Parametry pozorovatele
    Position:TPoint3D;          // pozice
    Angle:TPoint3D;             // uhly
    scrWidth,scrHeight:integer;  // rozliseni viewportu
    farclip,nearclip:TGLFloat;   // clipping planes
    parallax:TGLFloat;          // paralaxa pro stereo
    picdif:TGLFloat;            // picture difference pro stereo
    FOV:TGLFloat;               // FOV kamery
    focusStep:integer;          // pocitadlo kroku pri vstupu do examine rezimu
  end;
```

Mouse

Stav myši uchovává datová struktura *Mouse*.

```
Mouse:record                    // mys - ovladani modelu v okne
    ButtonDown:boolean;         // button mysi je stisknut
    Button:TMouseButton;        // button ktery byl naposled stisknut
    delta:TPoint2D;             // zmeny mysi od posledniho cteni
```

```
    act:TPoint2D;              // aktualni pozice mysi
    start:TPoint2D;            // pozice pri stisknuto tlacitka
    pickedpoint:TPoint3D;      // stredovy bod pri examine
  end;
```

Models

Nejvýznamnější proměnnou je samozřejmě models. Je to dynamické pole obsahující položky typu *TModel* ve kterých jsou uloženy všechny segmenty 3D modelu viruálního tlumočníka. Proměnná *num_model* udává počet segmentů celého modelu.

```
num_model:integer = 0;          // pocet modelu
models:PAModel;                 // geometrie vsech modelu
```

Textury

Pro uchování textur v paměti a pro jejich jednoznačnou registraci slouží následující proměnné. Význam je popsán příslušným komentářem.

```
num_tex:integer = 0;            // pocet textur
texs:PATextureRGB;              // vsechny textury
texnames:PAShortString;         // jmena textur
```

Execution buffer

Pole *execbuf* celočíselných položek obsahuje sekvenci příkazů, podle které probíhá rendering. Zajišťuje se tím hierarchie modelu.

```
num_execmd:integer;             // pocet prokazu v exebuf
exebuf:PAInteger;               // execbuffer (poradi vykreslovani modelu)
```

ROptions

V této struktuře je uloženo aktuální nastavení rendereru. Jednotlivé položky ovlivňují např. zapínání/vypínání textur, světel apod.

Měření času

Proměnná delay slouží k ukládání veškerých časových a synchronizačních informací. Z těchto položek se vypočítává parametr *t* pro interpolaci mezi dvěma klíčovými snímky.

```
Delay:record                    // pocitadlo casu
    Total:integer;              // pocet tiku mezi dvema fazemi
    Counter:integer;            // pocet tiku zbyvajicich do dalsi faze
    LastTime:double;            // naposledy zjistovana hodnota time
  end;
```

Slovník

Pole *Dictionary* je reprezentací slovníku znakové řeči. Jednotlivé položky jsou typu ukazatel na znak a celé pole je alokováno dynamicky. Rozsah slovníku je

stanoven pevně na max. 10000 znaků. Pole není za běhu udržováno konzistentní, což znamená, že po zrušení jednoho znaku se patřičné položce přiřadí hodnota *nil* a vznikne tak volná pozice. Setřídění pole probíhá při ukládání na disk předáním indexů na položky pole podle abecedy. Veškeré přístupy k jednotlivým znakům se provádí předáním indexu do tohoto pole. Vyhledávání v poli podle řetězce se provádí sekvenčně.

```
Dictionary:PAPSign;                    // Slovnik pohybu
```

Animace a interpolace

Pro animování je nutné uchovávat několik klíčových snímků. Minimální počet je dva, pro lineární interpolaci. Použitá spline interpolace pak vyžaduje přistupovat ke čtyřem klíčovým snímkům současně. Pro uchování předchozího a dalších tří následujících klíč. snímků slouží proměnná *ActPose*.

```
ActPose:array[-1..2] of TPose;
```

Vstupní fronta klíčových snímků

Spuštění animace se provede zařazením klíčového snímku, nebo celé sekvence do vstupní fronty. Tato fronta je kruhová FIFO a obsahuje položky složené ze dvou celočíselných hodnot, přičemž jedna udává index znaku ve slovníku a druhá číslo klíčového snímku. Vkládání znaku do fronty se provede sekvenčním vložením všech jeho klíčových snímků.

```
Say:record
    buffer:array[0..Say_Length-1] of
              record              // vstupni kruhova fronta pro znakovku
                Word,Pose:integer;
              end;
    head,
    tail:integer;              // index na zacatek a konec vstupni kruhove fronty
  end;
Talking:boolean;     // boolovska hodnota urcuje, zda probiha prehravani (rezimy playing/editing)
```

- **Metody**

Funkce jednotlivých metod jsou komentovány přímo v kódu. Navíc přesné pochopení všech postupů vyžaduje podrobnější zkoumání zdrojového kódu a proto zde budou uvedeny pouze funkce zajišťující rozhranní mezi modulem SLI2 a aplikací.

procedure RotateBase(m:PModel;angle:TGLfloat;x,y,z:TGLfloat)

Popis: Rotuje souřadný systém segmentu podle specifikované osy. Slouží pro změnu parametrů kostry.

Parametry: m – segment

angle – velikost úhlu rotace

x, y, z – osa

procedure LoadBase(filename:string;var Models:PAModel)

Popis: Nahraje nastavení lokálních souřadných systémů segmentů z disku.

Parametry: filename – název souboru

Models – pointer na pole všech segmentů

procedure SaveBase(filename:string;var Models:PAModel)

Popis: Uloží nastavení lokálních souřadných systémů segmentů na disk

Parametry: filename – název souboru

Models – pointer na pole všech segmentů

procedure SetActiveSegment(seg:integer)

Popis: Nastaví aktivní segment modelu. Aktivní segment je rendererem zobrazen červeně. Pokud nechceme aktivní segment zobrazovat, předáme funkci hodnotu –1.

Parametry: seg – číslo segmentu nebo -1

procedure SetBaseLength(f:TGLfloat)

Popis: Nastaví velikost zobrazovaných os souřadných systémů segmentů (pokud je zobrazován skeleton).

Parametry: f – velikost (default 0,3)

procedure SetAxisLength(f:TGLfloat)

Popis: Nastaví velikost zobrazovaných os rotace (pokud je zobrazován skeleton).

Parametry: f – velikost (default 1,0)

procedure BuffAddPose(Word,Pose:integer)

Popis: Přidá jeden klíčový snímek do vstupní fronty.

Parametry: Word – index slova ve slovníku

Pose – číslo klíčového snímku

procedure BuffAddSign(Word:integer)

Popis: Přidá jeden znak do vstupní fronty.

Parametry: Word – index slova ve slovníku

procedure BuffClear

Popis: Vymaže vstupní frontu.

procedure StopAnim

Popis: Zastaví přehrávání animace a přepne z režimu playing do režimu editace.

procedure SetModelPose(Pose:TPose)

Popis: Nastavení pózy modelu přímo (v režimu edit).

Parametry: Pose – klíčový snímek

procedure DictionaryGetPose(var DestPose:TPose; word:integer; pose:integer)

Popis: Vrací pózu ze slovníku.

Parametry: DestPose – výstupní proměnná
word – index znaku
pose – číslo klíčového snímku

procedure DictionaryDeleteSign(word:integer)

Popis: Smaže znak ze slovníku.

Parametry: word - index znaku

procedure DictionaryPutSign(SrcSign:TSign; word:integer;delete:boolean)

Popis: Uloží znak do slovníku.

Parametry: SrcSign – znak, který se ukládá
word – index na kterou pozici se znak má uložit
delete – udává, zda se má nejprve slovo slovníku smazat a poté vytvořit nové

procedure DictionaryGetSign(var Sign:TSign; word:integer;new:boolean)

Popis: Vrací znak ze slovníku.

Parametry: Sign – výstupní proměnná. Znak se do ní uloží.
word – index znaku ve slovníku
new – udává, zda se má pro výstupní proměnnou alokovat pamět

procedure DictionaryFree

Popis: Uvolní paměť celého slovníku.

function DictionaryGetInsertIndex:integer

Popis: Vrátí index pozice, která je ve slovníku volná.

function DictionaryFindSign(sword:string):integer

Popis: Vrátí index nalezeného slova ve slovníku. Pokud jej nenalezne, vrací hodnotu NOWORD.

Parametry: sword – hledané slovo zadané řetězcem

procedure DictionarySave(filename:string;WList:PAInteger;WLCount:integer)

Popis: Ukládá slovník do souboru na disk.

Parametry: filename – jméno souboru

WList – pole, které může být použito pro setřídění ukládaného slovníku. Pokud je naplněno indexy jednotlivých slov a parametrem WLCount se předá příslušný počet položek, setřídí se zapisovaný slovník podle tohoto pole.

WLCount – počet položek pole WList. Pokud je tato hodnota 0, slovník se ukládá tak jak je momentálně setříděn v paměti.

procedure DictionaryLoad(filename:string)

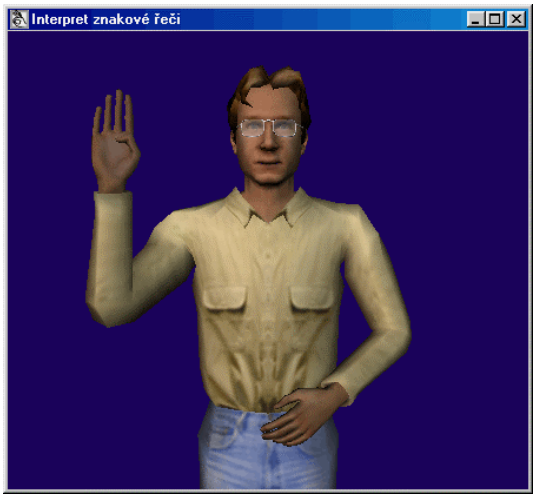
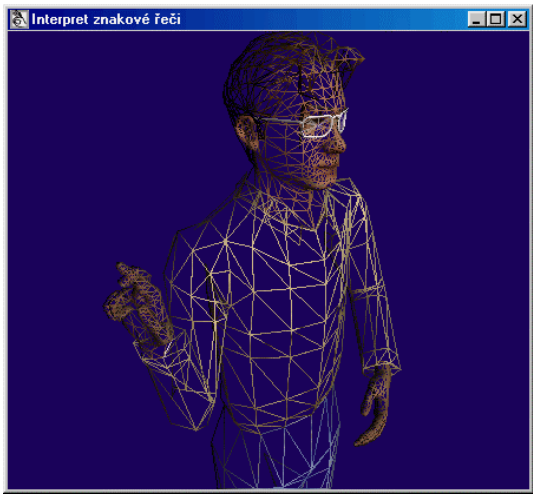
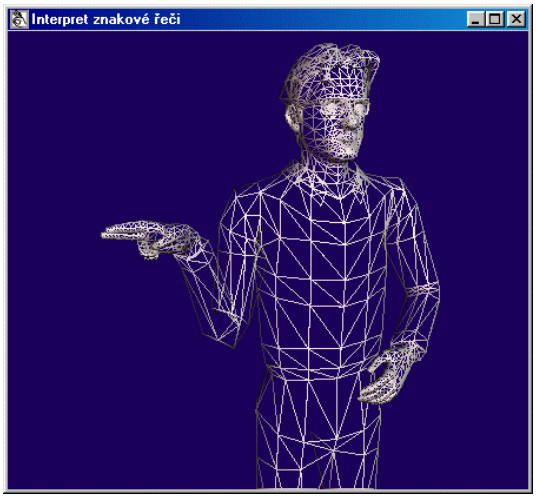
Popis: Nahraje slovník ze souboru do paměti.

Parametry: filename – jméno souboru

Interface pro OpenGL

Následující unity byly použity pro propojení prostředí Delphi s grafickými knihovnami OpenGL32.DLL a Glu32.DLL.

OpenGL.pas
last change : 03. October 1999
(c) Copyright 1999, Dipl. Ing. Mike Lischke (public@lischke-online.de)

Geometry.pas
Version 2.4
last change : 03. October 1999
(c) Copyright 1999, Dipl. Ing. Mike Lischke (public@lischke-online.de)

D Ukázky výstupu