STUDIA INFORMATICA 2001
Volume 22 Number 4 (46)

Piotr FABIAN
Politechnika Śląska, Instytut Informatyki

ANIMACJA POSTACI W SYSTEMIE WIZUALIZACJI JĘZYKA MIGOWEGO

Streszczenie. Artykuł przedstawia koncepcję jednego z elementów systemu wizualizacji gestogramów, modułu animacji. Omówione zostały założenia, problemy występujące przy realizacji modułu działającego w czasie rzeczywistym oraz przedstawiona została realizacja modułu z zastosowaniem bibliotek OpenGL i sprzętowej generacji grafiki.

AVATAR'S ANIMATION AS A PART OF A SIGN LANGUAGE VISUALIZATION SYSTEM

Summary. The article presents a general idea of one of elements needed in a system visualizing gestures of the Polish sign language. Next, presumptions for this module are presented, problems concerning the implementation of a real-time module and a solution using the OpenGL interface and hardware accelerated rendering.

1. Zarys problemu

System tłumaczący wypowiedzi w języku naturalnym pisanym na język migowy[1] złożony jest z dwóch głównych, niezależnych modułów: translatora dokonującego tłumaczenia zdania na ciąg gestogramów (symbolicznego zapisu języka migowego) [6] oraz modułu wizualizacji gestogramów w postaci animacji. Problem animacji został podzielony na kilka podproblemów. Tutaj zostaną opisane metody realizacji jednego z fragmentów drugiego

[1] System tworzony jest w Instytucie Informatyki Politechniki Śląskiej w Gliwicach w ramach badań nr 8 T11C 007 17 finansowanych przez KBN w latach 1999-2001.

elementu, związanego z zagadnieniami projektowania i wyświetlania animacji. Przyjęto następujące założenia:

- moduł translacji dostarcza zapisu gestogramów, zgodnie z notacją zaproponowaną przez prof. Szczepankowskiego,
- na podstawie gestogramu ustalane są położenia elementów manekina w kluczowych momentach animacji,
- położenia pośrednie są interpolowane w wybranej dziedzinie: położenia w przestrzeni lub kątów, z uwzględnieniem konieczności unikania kolizji,
- otrzymany ciąg położeń jest wizualizowany w postaci animacji postaci na ekranie monitora.

Aby cały system mógł znaleźć praktyczne zastosowanie, przetwarzanie wypowiedzi na animację musi być realizowane bez znaczących opóźnień. Celem opisywanych badań było opracowanie metody pozwalającej wygenerować animację na podstawie położeń manekina w wybranych momentach animacji. Realizacja animacji spełniającej podane wcześniej założenia wymaga rozwiązania kilku problemów:

- wyboru środowiska graficznego,
- definicji kształtu postaci zgodnej z anatomią człowieka,
- planowania ruchu,
- realizacji animacji z określoną prędkością.

2. Środowisko i ogólne założenia

Efekty w postaci realistycznie wyglądającej wizualizacji najłatwiej uzyskać nagrywając przy użyciu kamery gestykulującą osobę. Niestety, liczba możliwych wypowiedzi jest tak duża, że nierealne jest zarejestrowanie wszystkich dopuszczalnych zdań jako niezależnych sekwencji. Konieczne byłoby zastosowanie mniejszych jednostek podstawowych i zestawianie z nich animacji. Tu z kolei problemem stają się przejścia między różnymi sekwencjami, tzn. pośrednie ułożenia rąk i dłoni między końcem jednego gestu a początkiem następnego, dla których nagrania nie byłyby dostępne. Duża liczba koniecznych nagrań i pracochłonność ich katalogowania powodują, że bardzo trudne byłoby zrealizowanie całego słownika. Problem uzupełniania animacji płynnymi przejściami jest również poważny.

Ponieważ system wizualizacji wymaga działającego modułu animacji, w opisywanym tu podejściu z góry przyjęto założenie modelowania sztucznej postaci (manekina) w trzech wymiarach i generowanie płaskiego obrazu na ostatnim etapie, bez odwoływania się do gotowych kadrów z sekwencji filmowych. Tym samym problem animacji dzieli się na dwa podproblemy: generowania sekwencji parametrów sterujących, opisujących położenie

manekina w różnych chwilach czasu i wizualizacji tego manekina. Sekwencja parametrów sterujących jest w dużym stopniu niezależna od sposobu wizualizacji, która może być realizowana np. techniką śledzenia promieni, opisem w języku VRML, sprzętową generacją grafiki trójwymiarowej itp.

3. Wymagania dotyczące rozdzielczości przestrzennej i czasowej

Jedynym sposobem uzyskania animacji na monitorze komputera jest wyświetlanie obrazów przedstawiających kolejne kadry tej animacji z taką prędkością, aby obserwator miał wrażenie płynnego ruchu i mógł rozpoznać wystarczającą do zrozumienia gestu liczbę szczegółów. Zagadnieniami parametrów tak realizowanej animacji zajmowano się ponad 100 lat temu, w okresie powstawania kinematografii. Stwierdzono, że już 16 kadrów wyświetlanych kolejno w ciągu sekundy daje wrażenie płynnego ruchu, na skutek bezwładności wzroku człowieka i ograniczonej prędkości przetwarzania informacji wzrokowej. Później przyjęto większą liczbę, 24 kadrów na sekundę (ang. frames per second, fps) dla kinematografii i 25 do 30 dla telewizji [7].

Kolejnym parametrem jest rozdzielczość wyświetlanego obrazu. W przypadku komputera jest ograniczona możliwościami karty graficznej i monitora. Jak wynika z dalszych rozważań, parametry te są wystarczające na potrzeby omawianego systemu.

Czasami programom telewizyjnym, np. niektórym dziennikom, towarzyszy tłumacz języka migowego. Można przyjąć, że na potrzeby odbiorców takich programów, osób niesłyszących, forma taka jest odpowiednia i wystarczająca, by założyć podobne parametry tworzonej animacji. Postać tłumacza, mimo że zajmuje niewielką część ekranu, ma wystarczające dla odbiorców rozmiary. Orientacyjne rozmiary postaci stosowane w telewizji wynoszą od 190*250 pikseli do 280*340 pikseli, przy standardowej rozdzielczości obrazu telewizyjnego PAL 768*576 pikseli. Przy takich rozmiarach drobnymi szczegółami są nawet palce, będące na ekranie obiektami o średnicy rzędu kilku pikseli.

Definicja manekina nie musi więc obejmować małych szczegółów, ponieważ przy założeniu podobnego rozmiaru postaci jak w programach telewizyjnych, i tak nie zostałyby one odwzorowane na ekranie, natomiast wpłynęłyby na obniżenie wydajności animacji.

Jak wcześniej wspomniano, w telewizji i kinematografii stosuje się ok. 24 fps. Co najmniej taką wartość należy zapewnić w animacji, chociaż akceptowalne są niższe wartości, do 16 fps.

4. Opis pozycji manekina

W wyniku translacji zdania w języku naturalnym moduł animacji otrzymuje gestogram w zapisie prof. Szczepankowskiego [6]. Format ten dokładnie jest opisany w innym opracowaniu dotyczącym systemu translacji [8]. Gestogram definiuje pozycje i ruchy podczas pokazywania znaku migowego poprzez określenie położenia rąk (np. z boku ciała, z przodu), ułożenia dłoni (np. palce w górę), ruchu. Jednak niektóre znaki nie mogą być precyzyjnie opisane w takiej notacji [1]. Kreślone w powietrzu kształty są na tyle złożone, że wymagają precyzyjnego opisu, np. w postaci definicji trajektorii ruchu. Poniżej przedstawiono parametry jednoznacznie opisujące pozycje manekina.

4.1. Parametry

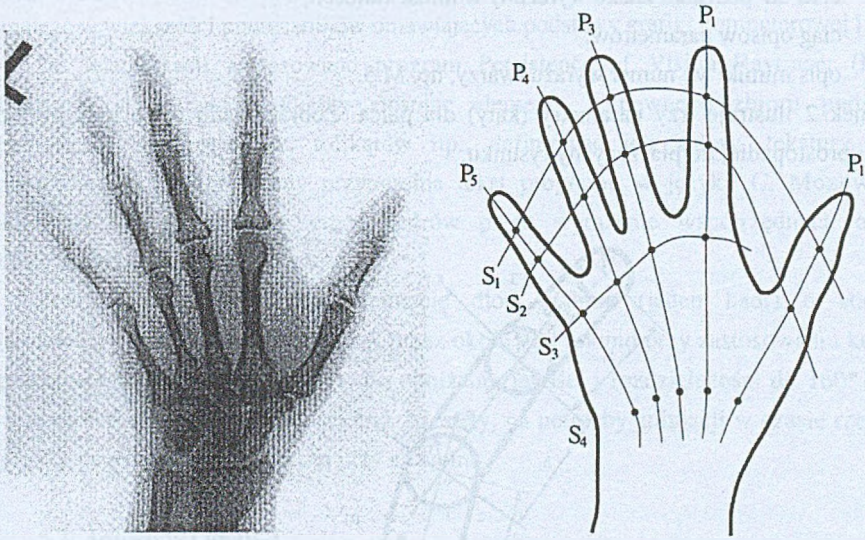
Założono, że manekin ma stałe rozmiary szkieletu. To założenie jest ważne, ponieważ moduł detekcji i usuwania kolizji wymaga identycznych parametrów manekina co moduł animacji. Z dobrym przybliżeniem można przyjąć, że wszystkie wymagane pozycje manekina da się uzyskać zmieniając jedynie kąty między elementami szkieletu w poszczególnych stawach. Jednoznaczny opis pozycji wymaga więc podania tylko tych kątów. Wyznaczenie kątów na podstawie gestogramu opisującego pozycję manekina jest złożonym zadaniem, gdyż gestogram definiuje jedynie położenie i orientację dłoni, nie opisując dokładnie położenia ramienia i przedramienia. Różne położenia tych elementów manekina prowadzą do tej samej pozycji dłoni. Zagadnienie doboru brakujących parametrów, tak aby ustawienie manekina odpowiadało gestogramowi i było realizowane w naturalny (optymalny) sposób, jest nazywane problemem kinematyki odwrotnej. Problem ten był rozwiązywany dla wielu szczególnych przypadków w robotyce [4]. W module animacji jest on rozwiązany w uproszczony sposób.

Zależnie od stawu może on posiadać jeden, dwa lub trzy stopnie swobody. Każdy stopień swobody oznacza możliwość obrotu wokół jednej z trzech wzajemnie prostopadłych osi kartezjańskiego układu współrzędnych w trójwymiarowej przestrzeni.

Na rys. 1 przedstawiono zdjęcie rentgenowskie dłoni i oznaczono stawy w poszczególnych palcach.

Palce na rysunku oznaczono przez P1 (kciuk) do P5 (mały palec), stawy od S1 (najbliższy końcom palców) do S4 (wewnątrz dłoni, traktowany jak nieruchomy). Kciuk ma jedynie trzy stawy. Stawy S1 i S2 mają jeden stopień swobody, stawy S3 po dwa stopnie swobody. Przyjęto, że w neutralnej pozycji cztery palce P2 do P5 są wyprostowane i leżą w jednej płaszczyźnie. Kciuk jest dodatkowo odchylony od tej płaszczyzny, co uwzględnia model manekina.

Dla animacji gestów tylko niektóre stawy człowieka mają znaczenie. Model manekina uwzględnia: szyję, stawy barkowe, łokcie i nadgarstki. Wszystkie stawy mają w modelu trzy stopnie swobody, chociaż w rzeczywistości stawy w łokciu i nadgarstku są ze sobą powiązane, tzn. nie jest możliwy niezależny obrót wokół osi przedramienia dla tych dwóch stawów. Jednak przyjęcie trzech stopni swobody upraszcza sposób opisu pozycji, w żaden sposób nie zmniejszając jego ogólności.



Rys. 1. Zdjęcie rentgenowskie dłoni; Px – palce, Sx - stawy
Fig. 1. X-ray photo of a hand; Px – fingers, Sx - joints

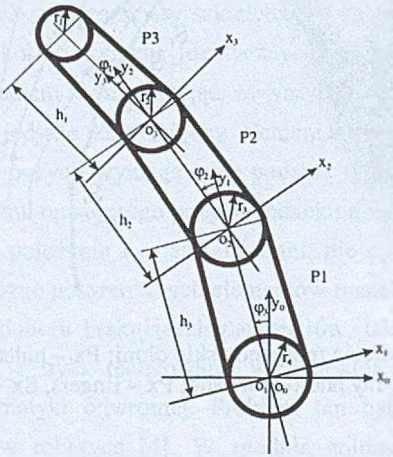
4.2. Precyzyjny opis gestu

Przyjęto, że w wewnętrznej reprezentacji modułu animacji gestogramy będą przedstawiane jako sekwencje „kadrów" opisujących gest. Gestogramom statycznym odpowiada opis w postaci ciągu parametrów opisujących kąty we wszystkich opisanych wcześniej, uwzględnianych w animacji stawach manekina. Dla gestogramów dynamicznych opis jest sekwencją opisów statycznych dla kluczowych momentów pokazywania znaku migowego. Momenty te powinny zostać tak dobrane, aby możliwa była prosta interpolacja parametrów w wybranej dziedzinie (kątów bądź położenia w przestrzeni), nie wymagająca dodatkowych analiz w obrębie jednego znaku migowego. Odpowiednie dane są uzyskiwane w wyniku przeprowadzenia translacji gestogramu i zastosowania kinematyki odwrotnej. W przyszłości może okazać się celowe zapisanie tych danych w słowniku (nazywanym dalej wektorowym), aby uniknąć wielokrotnego analizowania tych samych danych.

Zgodnie z przyjętą koncepcją, stałe dla wszystkich animacji są rozmiary szkieletu manekina. Zmieniają się tylko kąty. Przyjęto, że opis jednego znaku migowego (dla jednego gestogramu) zawiera następujące informacje:

- Identyfikator gestogramu (numer, nazwa).
- Liczbę „kadrów" (1 dla znaków statycznych).
- Ciąg opisów poszczególnych kadrów, każdy opis zawiera:
  - czas od początku znaku wyrażony w milisekundach,
  - ciąg opisów parametrów,
  - opis mimiki M: numer wyrazu twarzy, np. M:5.

Rysunek 2 ilustruje trzy parametry (kąty) dla palca. Zobrazowano tylko kąty obrotów wokół osi prostopadłej do płaszczyzny rysunku.



Rys. 2. Wybrane parametry pozycji dla jednego palca (φ_1, φ_2, φ_3)
Fig. 2. Selected parameters describing the position of one finger (φ_1, φ_2, φ_3)

Węzły, dostarczane przez moduł kinematyki odwrotnej i definiujące kolejne kadry, nie muszą być rozmieszczone w równych odstępach czasowych. Powinny tylko zapewnić wystarczająco dokładne odwzorowanie przebiegu zmian parametru w czasie.

5. Techniki wizualizacji obiektów 3D

W celu porównania wydajności różnych technik wizualizacji obiektów 3D, jakim jest też animowany manekin, zrealizowano animację trzema metodami: techniką śledzenia promieni, z

wykorzystaniem interfejsu OpenGL i sprzętowej generacji grafiki oraz przy zastosowaniu języka opisu sceny VRML.

5.1. Technika śledzenia promieni

Jest to jedna z częściej wykorzystywanych obecnie metod tworzenia realistycznie wyglądających obrazów odwzorowujących trójwymiarową rzeczywistość. Opis metody można znaleźć w większości podręczników omawiających podstawy grafiki komputerowej [3].

Do wizualizacji zastosowano program Persistence of Vision Raytracer (POV-Ray). Pozwala on zestawiać obiekty stosując elementy z pewnego zbioru podstawowych elementów: sfer, stożków, trójkątów itp., definiować ich kolory, tekstury, położenie obserwatora itp. Opis sceny przypomina tekst programu w języku C. Możliwe jest też automatyczne generowanie ciągu kadrów przez wyrażanie współrzędnych obiektów i obserwatora jako funkcji czasu.

Zrealizowano przykładową animację dłoni. Obraz (jeden kadr) o standardowej rozdzielczości VGA generowany był przez około 200 sekund przy zastosowaniu komputera z procesorem Celeron II 633 MHz. Po obniżeniu jakości i rozdzielczości do 160*120 pikseli czas skrócił się do kilkunastu sekund. Niestety, na potrzeby animacji w czasie rzeczywistym grafika generowana jest kilkaset razy za wolno.

5.2. Interfejs OpenGL

Większość współczesnych kart graficznych ze sprzętową akceleracją operacji 3D, wyposażonych w specjalizowany procesor graficzny, obsługuje standard OpenGL. OpenGL jest interfejsem dostępu do sprzętowych możliwości karty graficznej [5]. Został opracowany w 1992 roku przez firmę SGI jako rozwinięcie biblioteki IRIS GL. Obecnie OpenGL jest standardem otwartym. Jego zmiany i rozszerzenia są zatwierdzane przez grupę OpenGL Architecture Review Board (ARB), której członkami są różne firmy zajmujące się sprzętem graficznym i oprogramowaniem.

W dużym uproszczeniu można stwierdzić, że OpenGL stanowi zbiór funkcji uruchamiających odpowiednie podprogramy procesora graficznego i / lub sterownika programowego obsługującego ten procesor. Dzięki specjalnie opracowanej architekturze karty graficznej, która może być traktowana jak oddzielny system komputerowy w dużej mierze niezależny od głównego procesora komputera, operacje graficzne są wykonywane znacznie szybciej, niż byłyby wykonywane przez główny procesor. Przy rozdzielczości ekranu 800*600 pikseli i trybie TrueColor (3 bajty na piksel) obraz opisywany jest przez 1,44 miliona bajtów. Jeśli wyświetlana ma być animacja z prędkością 25 fps, to w ciągu sekundy

należy obliczyć i przesłać 36 megabajtów informacji o kolorach pikseli. Taką, a nawet wyższą wydajność rzeczywiście oferują współczesne karty graficzne. W testach użyto m.in. karty Riva TNT2 M64 z pamięcią 32 MB. Interfejs OpenGL może być też realizowany w sposób programowy, jednak wydajność takiego rozwiązania w przypadku generacji animacji jest niezadowalająca.

OpenGL oferuje funkcje operujące na grafice 2- i 3-wymiarowej. Oznacza to, że w przypadku scen trójwymiarowych programista nie musi implementować algorytmów odwzorowujących przestrzeń na płaszczyznę i wyznaczających kolory pikseli na ekranie. Definiuje tylko obiekty w przestrzeni, położenie źródeł światła, obserwatora i pewną liczbę dodatkowych parametrów. Procesor graficzny sam dokonuje odpowiednich przekształceń i rzutuje obiekty na płaszczyznę, tworząc płaski obraz.

Efekt nie jest identyczny z wynikami śledzenia promieni ani nie jest całkowicie zgodny z prawami optyki. Jednak realizm utworzonej sceny jest wystarczający, aby obserwator nie miał wątpliwości, jaką bryłę przedstawia płaski rzut.

Animacja zrealizowana przy zastosowaniu OpenGL została przygotowana w środowisku MS Windows 98 i za pomocą kompilatora Visual C++ z biblioteką MFC.

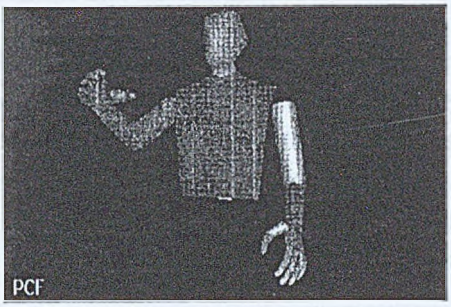
Rozmiary elementów manekina zostały zdefiniowane według dostępnych danych anatomicznych zaczerpniętych z [2]. Wszystkie parametry opisujące pozycję manekina zawarte są w jednym obiekcie, zawierającym kąty dla szyi i dwa zestawy danych dla rąk: lewej i prawej. Każda ręka jest oddzielnym obiektem, zawierającym wartości kątów dla stawu barkowego, łokcia, nadgarstka i dla palców, zgodnie z wcześniej przedstawionym schematem. Lewa ręka jest lustrzanym odbiciem prawej.

Animacja wymaga definicji ciągu kadrów węzłowych k_i. Kadr k_i definiowany dla chwili t_i zawiera wartości parametrów manekina. Kadry animacji mogą być też wyznaczane dla czasów leżących między t_i a t_{i+1}. Wtedy odpowiednie parametry są interpolowane.

Podczas pokazu animacji realizowane są następujące kroki:

```
„pobierz opis kadrów S=(k1, k2, ..., kn)"
dopóki czas <= kn.czas powtarzaj
   „wyznacz czas ti, jaki minął od początku animacji"
   „dokonaj interpolacji wartości kątów dla chwili ti"
   „wykreśl postać manekina"
   „obsłuż dodatkowe zdarzenia wpływające na animację"
koniec dopóki
```

Jak wynika z tego schematu, liczba wykreślanych kadrów nie jest z góry określona. Zależy od zapamiętanego w opisie gestogramu lub słowniku wektorowym całkowitego czasu trwania sekwencji i wydajności systemu. Rysunek 3 przedstawia przykładowy kadr z animacji manekina zrealizowanej z zastosowaniem interfejsu OpenGL. Uzyskano prędkość generacji od 50 do 90 pełnych kadrów na sekundę, co pozwala na zastosowanie tej techniki w systemie wizualizacji gestogramów.



Rys. 3. Przykładowy kadr z animacji
Fig. 3. Example frame from the animation

5.3. VRML

Inną możliwością wizualizacji ruchomych obiektów jest zastosowanie języka opisu scen trójwymiarowych VRML (ang. Virtual Reality Modeling Language). Można powiedzieć, że jest to trójwymiarowa wersja szeroko stosowanego w sieci Internet języka opisu stron HTML. Scena opisana w języku VRML może zawierać opis trójwymiarowych obiektów wraz z definicją ruchów. Możliwa jest też interakcja z użytkownikiem, wybierającym punkt obserwacji i wpływającym na postać sceny, jeśli opis w VRML przewiduje taką możliwość. Do wyświetlania scen opisanych w języku VRML stosuje się przeglądarkę VRML, która zwykle jest uzupełnieniem możliwości normalnej przeglądarki HTML. Rozsądna prędkość działania takiej przeglądarki wymaga zastosowania karty graficznej ze sprzętową akceleracją operacji graficznych. Możliwe jest wtedy uzyskanie płynnej wizualizacji i animacji z prędkością 25 fps lub wyższą.

Przykładowa animacja manekina zrealizowana w tej technice wykazała, że VRML może być alternatywną metodą wizualizacji, jednak z pewnymi ograniczeniami. Scena musi zostać wygenerowana wraz z opisem ruchu i dopiero potem możliwa jest jej wizualizacja. Aplikacja komunikująca się z użytkownikiem powinna mieć przez cały czas kontrolę nad animowaną postacią, co w tym rozwiązaniu nie jest bezpośrednio zapewniane. Jest to natomiast efektywny sposób dystrybucji generowanych animacji w sieci Internet, dający możliwość realizacji aplikacji tłumaczącej język naturalny na animacje w technice klient-serwer, wymagający od użytkownika jedynie instalacji przeglądarki VRML.

6. Uwagi końcowe

Opisano ogólne założenia modułu animacji manekina, wewnętrzną postać opisu gestogramu, zbiór parametrów opisujących pozycje i ruch manekina. Porównano kilka metod wizualizacji obiektów trójwymiarowych. Wskazano na metodę stosującą interfejs OpenGL jako podstawową dla ostatecznej wersji systemu translacji języka migowego i omówiono zrealizowaną przy zastosowaniu tej metody implementację modułu animacji.

LITERATURA

1. Hendzel J. K.: Turystyczny słownik języka migowego, Wyd. PTTK „Kraj", Warszawa 1989.
2. Staudt J., H Merker H. J.: Funktionelle Anatomie und Histologie in Wort und Bild, VEB Verlag Volk und Gesundheit, Berlin 1990.
3. Hermida A.: Przygody w świecie promieni, Intersofland, Warszawa 1994.
4. Craig J.: Wprowadzenie do robotyki. Mechanika i sterowanie, WNT, Warszawa 1995.
5. Wright R. S. jr, Sweet M.: OpenGL - księga eksperta, Wyd. Helion, Gliwice 1999.
6. Szczepankowski B.: Niesłyszący - głusi - głuchoniemi. Wyrównywanie szans. Wydawnictwa Szkolne i Pedagogiczne, Warszawa 1999.
7. Encyklopedia Britannica, wydanie internetowe (www.britannica.com), Britannica.com, Chicago 2001.
8. Fabian P., Francik J.: Synthesis and Presentation of the Polish Sign Language Gestures, 1st International Conf. on Applied Mathematics and Informatics at Universities, Gabčikovo, Slovakia 2001.

Recenzent: Dr inż. Maciej Bargielski

Wpłynęło do Redakcji 18 lipca 2001 r.

Abstract

This paper presents an animation module, which is the last part of a system visualizing gestures of the Polish sign language. The system translates sentences written in Polish language into symbolic sign representation and then presents a sequence of gestures in a form

of animation. Animation uses a model of a human, called avatar. Requirements concerning the time- and screen resolution, necessary for understandable animation are presented. Then parameters describing precise position of the avatar are specified. Three different approaches to the animation are presented and analyzed, namely ray tracing, OpenGL and VRML. Internal format of the data defining movements of the avatar and the usage of the OpenGL interface in the module are described in detail. Real time aspects of the solutions used in this module are discussed. Finally, tests of the module in a real system are presented.