Sign Language Interpreter

Jiří Dobrý, Václav Skala[1]
Department of Computer Science & Engineering,
University of West Bohemia, P.O.Box 314, Univerzitní 8, 306 14 Plzeň, Czech Republic
E-mail: jdobry@students.zcu.cz, skala@kiv.zcu.cz

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. Introduction

Everybody needs to communicate with the world around him or her. For the communication people use their senses. Deaf and hard of hearing people are unable to use the most common way of communication - a spoken language, because their ability to hear is limited and in most cases is none. Therefore they use special form of communication - the sign language. It is a language with its own grammatical rules, words and special expressions. Communication is then based on combining movements of fingers, hands and arms. Some of the signs consist of movements of hands in connection with different parts of the rest of the body (especially of the face). Sometimes grimaces and facial expressions are used too. A special form of communication with the deaf people is the lip reading. When speaking to them they watch the movements of the speaker's mouth and if they are trained in it, they can understand even without using the sign language or hearing the speaker.

The system described here uses computer graphics to simulate the real sign language speaker. There are many possibilities how to use the system. One of them is virtual sign language speaker in connection with TV subtitles. This virtual speaker will be able to translate written form of subtitles into more suitable form for the deaf people, into the sign language. Another possibility is to use the system as an interactive sign language dictionary. This four dimensional dictionary (3D space + time) cannot be compared with the book. Another very similar way of use is multimedia application for learning sign language. We can use all the advantages of modern multimedia systems and join it together with sign language. As the result we get very suitable way how to learn sign language especially for deaf children, their parents and people working with deaf and hard of hearing persons. Similar system to this is now being developed on the St Paul's University in Chicago [7].

2. Visualisation of humanoids

2.1. Model representation

There is a lot of methods for visualising objects of the real world by a computer. Human beings visualisation is one of the most complicated problems being solved in computer graphics. The difficulty is given by anatomical structure of the human body. One day we will be able to simulate movements of all the muscles, skin deformation, hair and a lot of other parts of human body according to

[1] This work was supported by the Ministry of Education of the Czech Republic - project VS 97155, Academy of Sciences of the Czech Republic project - A2030801

all the physical laws in real time. But it is still the future. There are systems producing very realistic human model visualisation (computer generated movies, commercials, etc.), but these techniques are time consuming and unusable for a real time system.
Many people were trying to visualise human models in the past and different techniques and algorithms were used.

a) Stick figures

One of the first experiments how to visualise the animation of humanoids was using the stick figures. The body was represented by a set of segments connected together in joints. However this approach has a lot of disadvantages and the visual results are not very realistic. The first well known model was designed by Withrow in 1970 [9].

b) Surface figures

Another method used in humanoid visualisation is the use of the surface model. Skeleton is covered by the surface consisting of polygons or curved surfaces (Bezier patches, NURBS etc.). Many of these models were designed for commercial use. The big advantage of such models is their realistic appearance. The animation of these models is complicated near segment joints, because some kind of shape deformation must be applied in order to get acceptable results.

c) Volume figures

The third method is the volume representation of the models. The models are built from volume primitives such as cones, ellipsoids or spheres. Visualisation of such a model is fast, but the visual results are not so good as in the case of surface models.

Our approach

For our purposes we have compared advantages and disadvantages of the methods. To achieve good visual results, the best way is to use a surface model with approximately 5000 triangles. Such a complex model can still animated in real time with the use of recent graphic hardware accelerators. Very important thing is bending the segments in joints when animating such a model. The bending algorithm we use will be described later. To get a high performance, relatively fast bending method is used.

2.2. Animation techniques

There are several animating techniques that produce different results. The computational complexity, special hardware requirements, knowledge and experience of the animators are some of the parameters affecting the final result. Two basic principles for controlling the animation are used.

a) Key-frame animation

Key-frame animation is based on computing the frames (so called in-betweens) between the frames that are set by animator. Animator usually sets only several frames and the application computes the in-betweens to get smooth movement of the animated object. Two different approaches are used for key-frame animation.

Techniques

Shape interpolation

The first approach we mention here is interpolation of the frames themselves. It is called image-based key-frame animation or shape interpolation. It is an old technique used already for animated line drawings. The animator sets the positions of all the vertices in 3D space and the application interpolates the vertex co-ordinates between two frames. Linear interpolation causes a lot of negative effects in more complicated movements if this method is used.

Parametric key-frame animation

Another technique is generation of in-betweens by interpolating the parameters of the model. This method is called parametric key-frame animation or key-transformation. The model is specified by the set of parameters. Animator sets the values of these parameters in the key-frames and the in-between frames are computed by

interpolation. But in this case only the parameters are interpolated. From the interpolated parameters we get new poses of the models, that we render to get the final image.

Interpolation

Interpolation is very important part of the animation. There are several methods of interpolation. Some of them are faster but produce bad visual effects and, on the other hand, there are very good interpolation algorithms that are time consuming and therefore they can't be used for real time animation. The most popular methods of interpolation are linear interpolation (used for the shape interpolation) and spline interpolation (for parametric key-frame animation). The most interesting splines for animation are cardinal, Catmull-Rom and Kochanek-Bartels splines, that allow more advanced controlling of the splines.

Obtaining the key-frames

Very important question in the key-frame animation is how to get the key-frames. Let's mention some of the most popular ways.

- Motion capture

Motion capturing is a method based on special hardware that grabs real character motions and converts the movements into digital form. The hardware for precise motion capturing is very expensive. The cheaper hardware is not very accurate and the animator must correct the data manually.

- Forward animation

The most common way of designing the animation is the forward method. Animator sets all the poses of the characters manually in the forward direction. The forward direction means in this case, that for example to reach the object on the table with the hand of the character, animator must set rotations for the character's shoulder properly and then for the elbow. If the hand can't reach the object, animator must rearrange all the rotations in the arm.

- Inverse kinematics

It is a very popular method in contemporary animation software. This method enables the animator to set only the terminal part of the bone chain to specific position and the software computes all the rotations automatically. In the case of the arm and the object, the animator sets only the position of the hand to the object and application computes interactively the rotations in wrist, elbow and the shoulder. This technique is very progressive for the design, but it is not so easy to implement.

b) Procedural animation

Key-frame animation is very good and relatively easy method for animation, but there are limitations in the use, because the movements are limited to the number of animations given by the number of the key-frames. Sometimes the application calls for the more advanced and flexible animation. The procedural animation is the solution. In procedural animation functions or special algorithms control the movements. For example the advanced algorithm for walking animation can consist of setting the rotations of all the segments of the legs, detecting collisions with the ground, keeping balance of the character and many other aspects. Procedural animation is more difficult to implement, but it is flexible to react automatically to situations in the virtual world.

Our approach

We use key frame animation for animating our virtual sign language speaker. The shape interpolation is not suitable for body animation, because the sign language has thousands of different movements and the shape interpolation requires storing all the key-frames in the 3D co-ordinates of all the vertices. Instead of shape interpolation we use parametric key-frame animation. The parameters are 1 - 3 angles representing rotations in the joints. The best method for interpolating the values of the angles when computing the in-betweens seems to be the spline interpolation, because it produces a smooth movement of the segments (similarly to human movements). The linear interpolation produces movement similar to movement of a robot. The spline interpolation

has better visual results and is not so much time consuming.

3. Anatomic and geometric skeletons

Human body is very complicated. Real simulation of all the bones, muscles and skin is still impossible in real time applications with contemporary graphic hardware. That is why some simplifications must be done. The human skeleton can be represented by geometrical model. Each important bone is substituted by one segment in the virtual character and the transformation matrix of the segment represents the direction of the bone in current body pose. The transformation matrix is given by composition of the following matrices.

$$M = B \cdot R_1 \cdot R_2 \cdot R_3 \cdot B^{-1}$$

where:

- M is the matrix of final transformation,
- B, B^{-1} are the matrices, that represent transformations between the co-ordinate space of the scene and the segment (the axis of rotations are not always parallel to scene axis),
- R_1, R_2, R_3 are the matrices of rotations. There are up to three different rotations in the body joints.

This model enables to represent the state of one segment with 3 values (3 angles). These three values are used for constructing the R_1, R_2, R_3. Our system consists of about 50 segments. The state of the whole model can be described by about 150 values. However not every part of the body is able to reach all the directions. The movements of human articulations are limited. The model we are using has different degrees of freedom in different joints in according to ability of a real human. Some of the limit values are taken from [3].

There are several joint types in human body with different mobility. To keep the model easy to control and able to reach all the poses, we use simple joint restrictions. The maximum and minimum angle values for all the joints are set according to [4]. The movement limitations of the arm are in the table 1.

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

Table 1 (movement limitations)

There is possibility to use 2D reachability map, where x, y co-ordinates represent alpha and theta angles in spherical co-ordinates and the value in the map says if this values of the angles are possible to reach or not. This approach is more suitable for complicated joints like shoulder. But for our purposes the construction of the 2D reachability map is too complicated and so we do not use this method.

4. Hierarchical structure and bending of the segments

4.1. Hierarchical structure

As it was already mentioned above the model consists of segments connected in joints. The whole skeleton has the tree structure. The nodes of this tree represent segments and each of the segments contains transformation matrix which affect this segment and also all the segments following in the hierarchy. There is a special system for controlling the order of transformations. In fact we don't use the tree data structure in the application. Instead of this we use stack buffer, that contains all of the operations in sequential order. The operations control the rendering and deformations of the whole model. There are three kinds of operations: Pop, Push and Deform&Draw. The Pop and Push are operations, that store and load the transformation matrices and The Deform&Draw is operation that applies deformation and renders one segment. This approach is very similar to philosophy of

OpenGL transformation matrices and so it is easy to implement and fast.

4.2. Bending of the segments

One of the most complicated problems of human model visualisation is realistic bending of the surfaces representing parts of the body near articulations. Several techniques for avoiding the bending were used in the past, but they don't bring good visual results. Some of them are based on very simple idea of putting spheres into the joints or simply letting the segments to intersect. These methods are not convenient for our needs. Instead of that we tend to use bending of the segments.

The bending is based on idea of local deformation of the segments. The vertices that are close to the joints have special parameter that we call bending weight and also the reference to the segment they are close to. The bending weights say how flexible is the vertex - how much it will be deformed. The vertices close to the joint are deformed by a simple rule, which takes into account the bending weights and the transformation matrices of the two segments closest to the joint.

Values of the bending weights have been obtained experimentally in order to get good visual result of the bent areas in all poses the joint is able to reach. We use two layers of the bent points. One layer contains points shared by two of the segments (usually the value for these vertices is approx. 0.5) and the second layer contains the neighbours of the first ones (vertices connected with the shared ones usually the weights are approx. 0.20 or 0.80 depending the two cases discussed later). The situation can be seen on figure 1.

Let the model is in basic pose. All of the points have co-ordinates relatively to the centre of rotation of this segment. All of the transformation matrices are identity matrices. No bending is performed. Now we change one matrix (for example rotation in the shoulder). The rotation of the shoulder changes the matrix of the upper arm segment. The vertices that will be deformed are those belonging to the upper arm segment and also the vertices belonging to the chest segment.

Two different situations in each segment are solved:

Points that are closer to the origin of the segment (upper arm segment)

In this case, the matrix of the segment they belong to transforms all the points. But the weights are included and so the vertices with the weight of 0.0 are fully rotated and those that have weight 1.0 are static (stay in basic position). The rule for rotation is described here.

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

where:

- M' is a new inverse matrix of the deformation
- M is an original inverse matrix of the deformation,
- w is a bending weight of the vertex,
- I is an identity matrix

The final matrix M' is then normalised and the vertex is transformed by this matrix. Now let us explain why we are using inverse matrix. All of the vertices are transformed by model-view matrix when displayed by OpenGL. We set the values to the model-view matrix from segment transformation matrix. Now it is easy to imagine that if we want to keep the vertex in its initial position, we must firstly transform this vertex by the inverse matrix of the segment and than after the transformation provided by OpenGL the vertex has its original position.

Points closer to the next segment in hierarchy (body segment)

Situation for the body segment vertices is more complicated. At first we must compute relative co-ordinates to the centre of rotation, because the vertices of this segment have the co-ordinates relatively to the segment origin and now we want to rotate them around different centre (the centre of the next segment). Then we must apply the inverse rotation as in the case above and finally full forward rotation controlled by the transformation matrix of the next segment. Then we must re-compute back the co-ordinates of the vertices to its original

system (relatively to its origin). The OpenGL then applies the model-view matrix transformation.

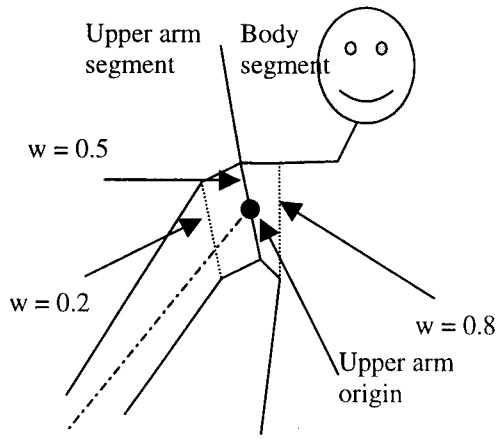


Figure 1 (different bending weights)

5. Rendering with OpenGL

OpenGL is very advanced graphic API, that provide all the important features of modern rendering techniques such as blending, texturing, local illumination, stencil buffering and many others, with maximal support of new hardware graphic accelerators. We use this API to get very fast rendering of our virtual character. The data structures in the application are suitable to be used with the OpenGL. We use the OpenGL matrix model when rendering hierarchical virtual character, because the OpenGL can take advantages of the hardware when computing geometric transformations. To describe all the features of OpenGL that we use in visualising the sign language interpreter is not the main point of this article.

6. Conclusion

The system for the interpretation of the sign language by computer was presented. The system has its own integrated database of signs with a special editor to manage this database. All modules and applications have been programmed in Borland Delphi 4.0 on Windows NT platform. Sign language interpreter is able to translate written text in the Czech language into the Czech sign language and visualise the animated virtual speaker of the sign language in real time on contemporary graphic hardware (tested on NVidia Riva TNT2). The system is easy to use, relatively fast and able to be included in different applications that need the Czech sign language as an output. Several examples can be seen on figures 2a, 2b, 2c and 2d.

7. Acknowledgements

The authors would like to thank to all who contributed to this work, especially to Jakub Mares and Digital World s.r.o. for the model design, colleagues at the University of West Bohemia in Plzen who have stimulated these thoughts, to anonymous reviewers of this paper as they shared some valuable insights on this problem solution. Their invaluable critical comments and suggestions improved the manuscript significantly.

8. References

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

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

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

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

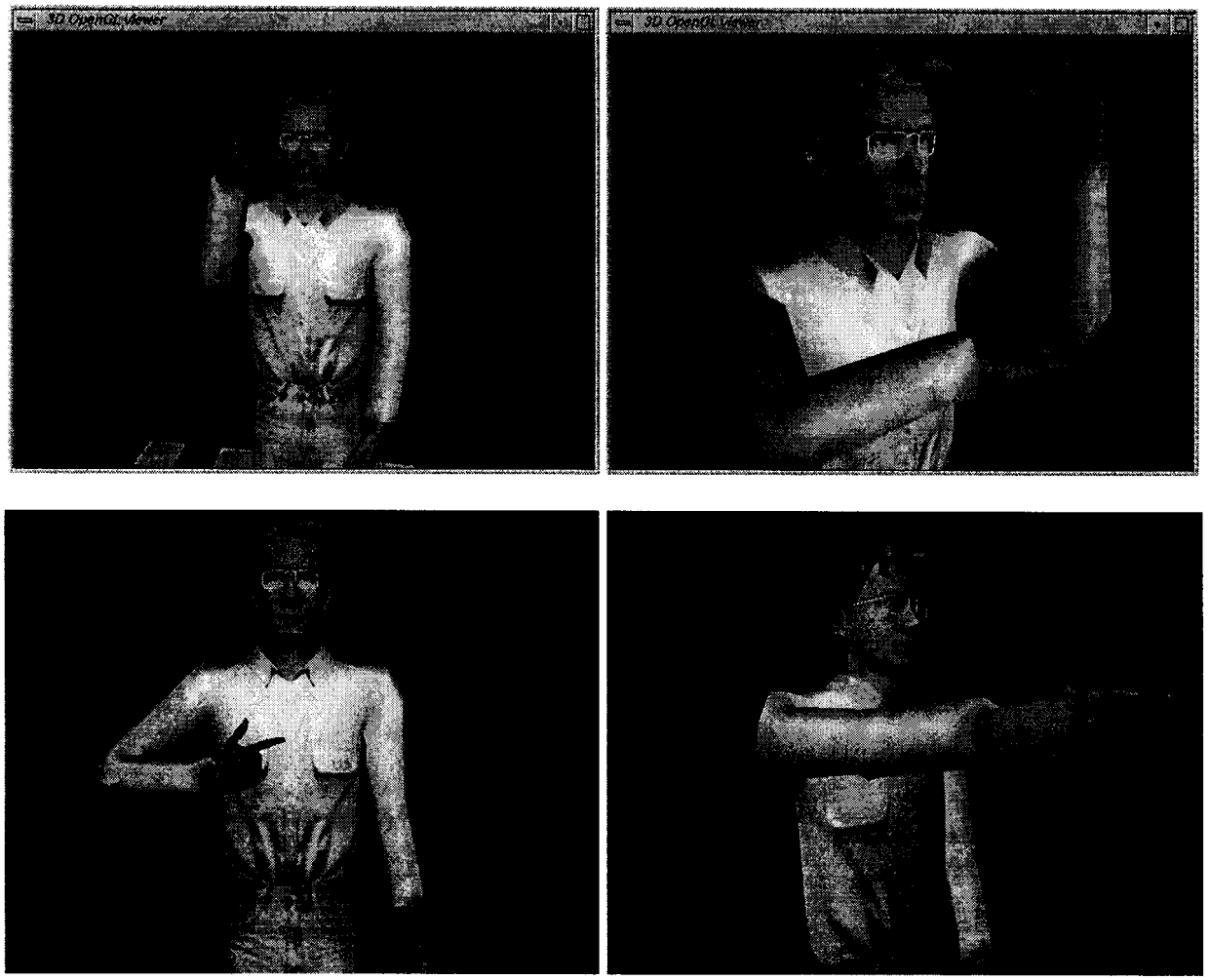
[5] Gabrielová,D., Paur,J., Zeman,J.: Slovník znakové řeči, Horizont, Praha 1988

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

[7] 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

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

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



Figures 2a, 2b, 2c, 2d