Application of 3D Facial Animation Techniques for Chinese Opera

Hao Zhou

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfilment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

August 2003

Your file Votre référence
ISBN: 0-612-83926-5
Our file Notre référence
ISBN: 0-612-83926-5

ABSTRACT

Application of 3D Facial Animation Techniques for Chinese Opera

Hao Zhou

Among China's rich cultural heritage is Chinese opera, a kind of folk art that combines singing, reciting, acting and dancing. Over more than 13 centuries, varied styles have evolved – the more famous being Beijing opera, Yue opera, Huangmei opera, Kunqu opera and Sichuan opera. In the 20th century with the popularity of movies and pop music, Chinese opera underwent a serious decline and some of these styles are on the verge of extinction. Recognizing this, in May 2001, UNESCO proclaimed Kunqu opera as one of the 19 "Masterpieces of the Oral and Intangible Heritage of Humanity". The Chinese government has also taken many initiatives for preserving and recording this art form, primarily using more conventional media like text, photographs, illustrations, sound, film and video.

In this thesis we propose the use of 3D computer animation techniques for the same purpose. More specifically, since exaggerated facial expressions and falsetto singing are distinguishing aspects of this art form, we have experimented with application of 3D facial animation techniques first. In order to ensure that we can capture all the necessary facial detail, we have chosen to use a point cloud representation for the face and a rather elaborate facial structure in comparison to, say, the structure proposed in the MPEG-4 standard. For increasing rendering performance, we have adapted the Q-Splat tree data structure devised by Stanford University's "Digital Michelangelo Project" for representing and rendering high precision scan data of ancient sculptures. Since a 3D scanner was not available to us during our research, for experimental purposes, we have converted triangle models of faces into point samples. They are then used for point cloud representation of faces and further for animation of facial expressions.

We have implemented a program to test our ideas on facial animation of Chinese Opera singers. The program accepts animation scripts and synthesizes the animation depicting Chinese Opera sequences. A few short video clips animating similar expression sequences from live video capture have been synthesized and these show that the approach is quite promising and could be extended with further research.

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I am deeply indebted to my supervisor, Prof. S. P. Mudur, whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis. Especially, I would like to give my special thanks to my wife, Ye Wang, whose patient love enabled me to complete this work. I also want to thank my colleague, Xin Li, who painted the point sampled faces of different Chinese opera characters.

A generous equipment grant awarded by NSERC, Canada to my supervisor for setting up a 3D scanning facility has provided the motivation for initiation of this research and will further help us continue experimenting with live 3D captures.

The triangle mesh models used in this research to capture the point-sampled representations for simulating 3D scanned data are downloaded from http://www.facegen.com/.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

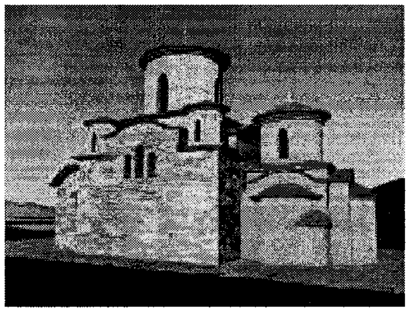
1. Introduction

In recent years there has been a strong fascination in the development and use of computer 3D animation. A distinct example is almost every Hollywood movie now carries computer animation. However, computer technologies are also being used towards other — some might say "nobler" — ends. One such application is the preservation and recreation of cultural and natural sites around the world. The United Nations Educational, Scientific and Cultural Organization's (UNESCO) World Heritage program is charged with the responsibility of preserving and documenting such sites. There are many ongoing efforts to use interactive visual — and sometimes immersive — computer technologies to help in this effort.

1.1 Preserving Heritage Projects

Electronic and more specifically digital documentation of heritage and culture ranging from ancient scriptures, monuments, relics, paintings, sculptures and various performing arts has been of significant importance for a large number of years now [1]. There have been a number of large digital archives projects, such as Prof. Robert Stone's (UK) Virtual Stonehenge, Darko Disoski's project — the Church of St. Pantheleimon, Ohrid, Republic of Macedonia, and Prof. Jiang Yu Zheng's work on the Museum of Terra-Cotta Warriors and Horses in Xian, China, etc (*cf.* Figure 1-1, 1-2, 1-3). [2]

Figure 1-1: Storm clouds approaching Virtual Stonehenge



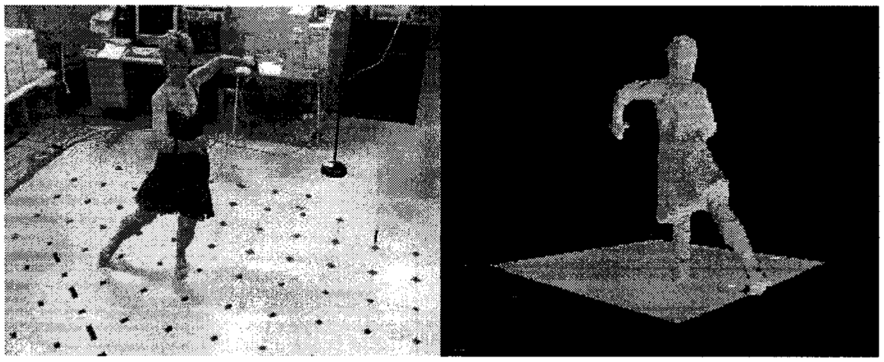
Figurel-2: Model of Church of St. Pantheleimon, Ohrid, Republic of Macedonia

Figurel-3: Digitizing the Terra-Cotta Warriors, Xian, China

Virtual recreations of archaeological heritage sites has received considerable attention as it has the potential for users to interactively walk through and obtain a near real experience [3, 4]. The Virtual Heritage Network of the Virtual Systems and Multimedia Society (VSMM) has this as its major focus. With the advent of 3D scanning devices, a number of projects have been initiated in high fidelity reconstructions of ancient sculptures [5][6] in 3D. For example, the national research council in Canada has been regularly digitizing art works and objects for quite a few years now. [7].

In the performing arts domain, there has been some effort in digital recreations of Hollywood actors. Miralab created a 3D animated model of Marilyn Monroe, and put her in various virtual situations (*cf.* Figure 1-4)[8]. Other efforts in use of 3D for performing arts include motion capture used in dance [9,10], and the CMU (Carnegie Mellon University) 3D room (Falmenco Dance *cf.* Figure 1-5) [11]. Other than this we are not aware of any significant efforts in using 3D animation for cultural art form documentation or preservation, not withstanding the fact that 3D animation has been in use in commercial cinema for quite a few years [12].

Figure 1-4: Virtual Marilyn Monroe of Miralab



Figurel-5: Virtual Spanish Dancing (Flamenco)

1.2 Overview of Chinese Opera

Among China's rich culture heritage, there are several worldwide renowned art forms; one of them is Chinese Opera. Chinese Opera has a history of more than 13 centuries, beginning with the court ballet of early *Tang* dynasty. In the long development of Chinese Opera, many different styles of opera appeared. Each style takes its name from the place where it originated or where it is popular. Among well known styles are *Beijing* Opera (actually the national form, *Beijing* city), *Yue* Opera (Zhejiang province), *Huangmei* Opera(*Anhui* province), *Kunqu* Opera(*Kunshan* Opera, popular mainly in Jiangsu Province), *Sichuan* Opera (*Sichuan* province) [13]. In the 20th century, with the popularity of movies and pop music, Chinese opera underwent a serious decline and some of these styles are on the verge of extinction. This has caught the attention of the entire world. On 18 May 2001, *Kunqu,* the 500-year-old Chinese Opera, was proclaimed by UNESCO (the United Nations Education, Science and Culture Organization) as one of the first 19 "*Masterpieces of Oral and Intangible Heritage of Humanity*". *Sichuan* Opera has also applied for similar recognition [14]. With help from UNESCO, a non-government association is making verbatim videotape recordings of ballads and operas of original performances [15].

Financial aid from UNESCO is far from enough. The Chinese government is also striving very hard to protect its traditional cultural and artistic forms by taking extensive actions to preserve publication of scripts, photographs, and audio and video recordings of opera performances. In order to protect the heritage of Beijing Opera, an eight-year videotaping project was launched in 1994. The project saw the completion of 582 videotapes, lasting over 500 hours by the end of August, 2002. Besides Beijing opera, some of China's local operas have also been videotaped in the same way [16]. Art institutes such as the Chinese Academy of Arts (CAA), a national artistic research institute responsible for saving, safeguarding, studying and passing on Chinese artistic heritage are deeply involved. There are today a number of websites which describe this art form and also include digitized videos showing Chinese opera performances (URL: http://www.shidaiguocui.com/wangshang_list.php, http://www.jingjuok.com/sp/shipin.htm). In addition, the government plans to support the opera theatres and the training of new performers, the revival of rarely performed operas, and the organization of festivals [17].

Chinese Opera is a kind of folk art, in the sense that it combines singing, reciting, acting and dancing. The distinguishing features of Chinese opera include (1) paucity of stage props, (2) falsetto singing, (3) symbolism in gait, and gesture (4) face painting, which is used to reveal character, (5) acrobatics, (6) stylized and colourful costumes, and (7) accompaniment by a

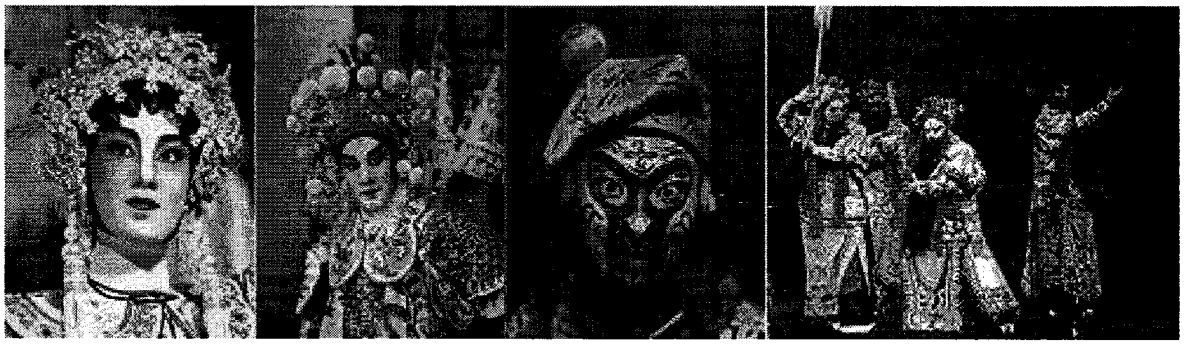


Figure 1-6: Still Frames from Chinese Opera

small orchestra [13]. Figure 1-6 shows some still frames from Chinese opera illustrating some of these aspects. For more dynamic aspects, the reader is referred to the digitized videos on the web sites listed earlier.

However, neither texts and pictures nor audios and videos can individually illustrate all these features to the desired level of fidelity. Also, while cross referenced multiple media content does indeed alleviate this problem to a certain extent, the holistic experience so inherent in the enjoyment of this art form is missed out. The authors believe that advanced technology of computer graphics, animation and multimedia has the potential to make it possible to simulate a Chinese Opera performance in virtual reality. A three dimensional virtual show accompanied by stereophonic sound could give spectators the full view of visual field, a near real experience that would otherwise be difficult to create using conventional 2D media. It is no doubt one of the best ways to restore Chinese Opera's fascination. Using techniques such as motion capture, it is possible to fit movement to a 3D sculpted model of the early performing artist and thus be able to recreate performance videos of the same artist through his images and audios. Certainly given the current state of art in realistic rendering and animation, this is ambitious. But over time, we do believe that the rate of technology developments in this field is such, that graphics and computing performance issues will assume lesser significance. Further, it is well established that more often than not, heritage applications tend to stretch technologies beyond their current frontiers [7].

To the best of our knowledge, ours is one of the first attempts to apply 3D animation techniques for documenting visual performing arts such as Chinese opera. 3D animation techniques not only provide simulation of the real opera scene but can also help in a unique way in the restitution of this distinct Chinese culture heritage.

From a research point of view in computer animation, human character animation and facial animation are interesting and challenging. The ability to model the human face and then animate the subtle nuances of facial expression remains a challenge in computer graphics.

In this thesis we describe the results of our research which mainly consists of a modest attempt in the application of 3D facial animation techniques to high fidelity documentation of Chinese opera singers' facial expressions. The goal of this research is primarily to explore the feasibility and use of appropriate facial animation techniques, to understand the difficulties in digitally documenting the finer aspects and nuances of Chinese opera and to evolve techniques that show promise. As a result of this research we have not produced any authentic documentation of Chinese opera. However, we do believe that our experiments have revealed the potential of this technology. With wider interest and adequate resources the techniques could be used to document and preserve this heritage.

The remainder of the thesis is organized as follows. Chapter 2 provides a comprehensive survey of facial modelling and facial animation techniques including a discussion of the MPEG-4 Facial Animation Standard. Chapter 3 describes the representation of faces as point sampled surfaces, their hierarchic organization and the associated techniques. In Chapter 4, we discuss the algorithms developed by us for animating different expressions and real time rendering. In Chapter 5, we provide some details of our program implementing the above and also give an analysis of the results based on some experiments that we carried out to synthesize animations corresponding to short sequences from live video capture. Finally in Chapter 6, we give our conclusions and discuss further work.

2. A Survey of Facial Modelling and Animation

Since the pioneering work of Frederic I. Parke in early 1970s[18], many research efforts have attempted to generate realistic facial modeling and animation. The ultimate goal for research in facial animation is a system that 1) creates realistic animation, 2) operates in real time, 3) is automated as much as possible and 4) adapts easily to individual faces [19].

2.1 Modelling Faces

Developing a facial model involves devising geometric descriptions and animation support capabilities that represent the faces of interest. A facial model is usually the sum of many parts and details. Much of the research focuses on the facial mask [18]. The facial mask includes the visible external skin surface of the face and often includes the front part of neck. But, a complete face model should add facial features such as eyeballs, eyebrows, eyelashes, teeth, tongue, etc.

2.1.1 Facial Geometry

The face has a very complex, flexible, three-dimensional surface. It has colour and texture variation and usually contains creases and wrinkles. For the purpose of facial animation, static models are not useful; the facial models must be constructed in ways that support animation. Various animation techniques, which are discussed in the next section, assume an underlying geometrical face representation that allows both effective animation and effective rendering. Typical representations are Volume Representation and Surface Representation.

Volume Representation includes constructive solid geometry (CSG), volume element (voxel) arrays, and aggregated volume elements such as Octrees[18]. However, the structural shape change required in facial animation is complex for voxel models.

Surface Representation includes implicit surfaces, parametric surfaces and polygonal surfaces [18]. The blending and constraint properties of implicit surfaces would be more difficult than other techniques, and real time animation with implicit surfaces is even more difficult. Parametric surfaces include *bivariate Bézier*, *Catumull-Rom*, *B-Spline*, hierarchical *B-Spline*, and NURB surfaces. These surfaces are generated by three functions of two parameters, one function for each of the spatial dimensions. These functions are based on quadric or cubic polynomials.

The majority of facial models that are in practical use are based on polygonal surface descriptions [18]. Polygonal surfaces meshes are constructed by a number of vertices and these vertices are then connected with triangular or quadrilateral polygons to form the desired surface. The polygons must be laid out in a way that allows the face to flex and change shape naturally. A polygonal model for facial animation must allow representation of all possible expressions after some modification. While 3D scanning techniques are extensively used for obtaining the face surface data as point clouds, the surface sample points are most often used to construct a polygonal (triangle) mesh closely fitting the original data.

In the last decade, various methods have been developed for generating triangular meshes out of point clouds. However, the triangulation process is time-consuming. Therefore, some researchers have attempted to directly use the point clouds model instead. Compared to triangle mesh, point clouds models have detailed geometry, fine textures and are easily captured by 3D scanner. Point cloud models have some drawbacks: scanned data usually has hundreds of millions of samples which are impractical to render; scanned image almost miss some highly occluded regions; the locations of points are often imprecise due to noise. In recent times, the use of point sampled surface is becoming increasingly popular. Rusinkiewicz and Levoy [20][48] developed their *Qsplat* hierarchical bounding sphere representation for efficient rendering of their very large 3D scanned models of sculptures as part of Stanford University's Digital Michelangelo project. Mitsubishi Electric Research laboratories have also developed a rendering system using SURFELS (Surface Elements) as Rendering Primitives [21]. Unlike classical surface discretizations, i.e., triangle or polygon meshes, or piecewise algebraic surfaces, SURFELs are point primitives without explicit connectivity.

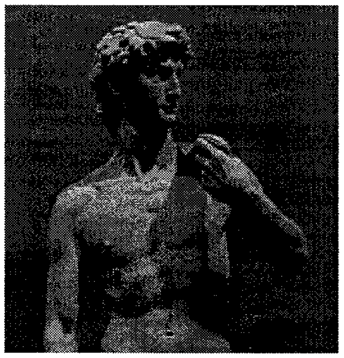


Figure 2-1: Point cloud model of David from Digital Michelangelo Project.

2.2 Facial Animation Techniques

Computer facial modeling and animation is a fairly well studied subject, with even textbooks written about the techniques in use [18]. It is virtually impossible to enumerate all of facial animation techniques here. In the rest of this section, we briefly introduce five fundamental categories. They are interpolation, parameterization, muscle-based, pseudo muscle-based and performance-driven animation.

2.2.1 Interpolation Based Animation

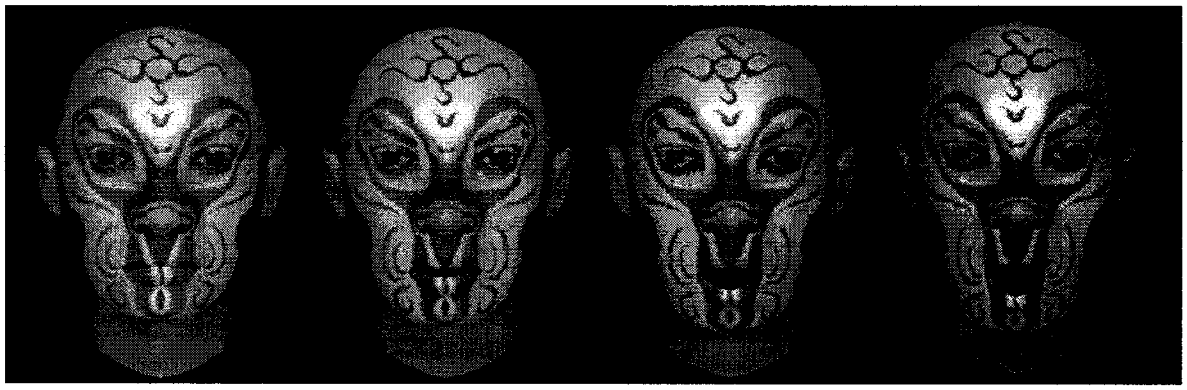
Interpolation is perhaps the most widely used of the techniques for facial animation[30]. Several films such as *Sextone for President* and *Don't Touch me* contain convincing facial animation achieved by digitizing the face in each of several different expressions and then interpolating between these expressions. The goal of interpolation is to create smooth motion between two *key-frames* or *key-poses* at extreme positions, over a normalized time interval. The key-frame or key-pose corresponds to the desired expression. Interpolation process is a computer algorithm which generates all middle frames between two key-frames. The following formula shows a simple linear interpolation method.

$$\text{(Intermediate frame)} = (1.0 - a)\,(\text{keyframe1}) + a\,(\text{keyframe2}) \quad 0.0 = a = 1.0$$

The intermediate frame is specified by the fractional interpolation coefficient a.

Interpolation can be extended in several ways. A cosine interpolation function or other variations can provide acceleration or deceleration effects in the progress of interpolation. Other interpolations such as Bilinear Expression Interpolation, n-Dimensional Expression Interpolation, Pairwise Expression Interpolation and Facial Region Interpolation are useful extensions [18].

Interpolation schemes are fast, but have limitations. The range of expressions is related to the number of the expression poses available. An expression that falls outside the bounds of the key poses set is unattainable. Also, each key pose requires an explicit geometric data collection or data generation effort and combination of independent face motions are difficult to produce.



a = 0 a = 0.3 a = 0.7 a = 1.0

Figure 2-2: Interpolation for in-between facial expressions

2.2.2 Parameterized Animation

In 1974, Parke completed the first parameterized facial model [22]. Parameterized models are powerful tools for facial image synthesis and animation because they only need to manipulate sets of criteria – the parameters – to create a sequence of images. There are two categories of parameters, *Expression Parameters* and *Conformation Parameters*. The more often addressed category are Expression Parameters, which include face's emotional aspects, such as eyelid opening, eyebrow arch, eyebrow separation, jaw rotation, etc. Conformation Parameters control individual facial conformation or shape. Conformation Parameters include jaw width, forehead shape, nose length and width and so on. These two categories are independent to each other. Most facial models have focused only on expression parameters [18].

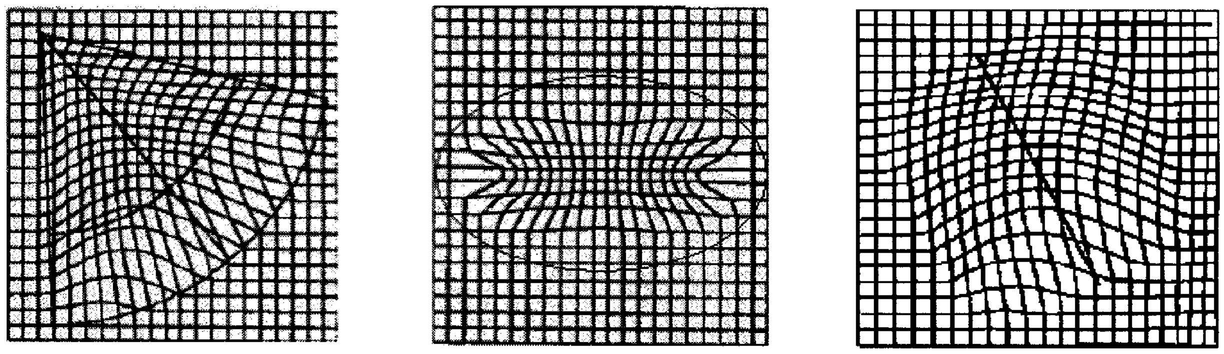
Developing facial parameter sets is essential to parameterized animation. These parameter sets are independent and parameterizations allow explicit control of specific facial configurations. Parameter sets are always based on facial structures or anatomy. Taking expression parameters as an example, most research develops parameter sets related to facial regions, such as eyes, mouth and facial mask. Useful parameters for the eyes may include eyelid opening, pupil size or dilation, eyebrow shape and positions, and direction in which the eyes are looking. Mouth parameters could include jaw rotation, width of the mouth, mouth expression (smiling, frowning, etc.), position of the upper lip and position of the corners of the mouth. With only a few parameters, it is possible to develop simple and interesting expressions. MPEG-4 Facial Animation is a standard which describes a 3D parameterized talking head oriented to low bandwidth requirements [23]. A more detailed introduction to MPEG-4 is provided in section 2.3.

Parameterization specifies any possible face and expression by a combination of independent parameter values. Combinations of parameters provide a large number of facial expressions with relatively low computational costs. However, parameterization always defines noticeable motion boundaries to avoid conflicts between two parameter sets. And, since parameter sets depend on facial regions, a complete generic parameterization is not possible. Another limitation is that manual tuning of parameters is needed and is tedious. This may often lead to coding of unrealistic expressions.

2.2.3 Skin and Muscle-based Animation

In 1981, Platt and Badler [24] used a mass-and-spring muscle model to simulate facial expressions. In their model, polygonal vertices of the face surface (the skin) were elastically interconnected with modelled springs. These vertices also were connected to the underlying bone structure of model using simulated muscles. The face expressions were manipulated by applying muscle forces to the elastically connected skin mesh. Their model consists of 38 regional muscle blocks interconnected by a spring network. The deformations of the spring meshes generate realistic facial animation.

A very successful muscle model was proposed by Waters [25]. Figure 2-4 illustrates some expressions created by Waters' muscle model. Waters described three primary muscles types in his work: linear, sphincter, and sheet (*cf.* Fig2-3). Linear muscle and sheet muscle are described as linear contraction, whereas the sphincter muscle is described as an elliptical contraction. In linear muscle, the surrounding skin is contracted towards the static node of attachment on the bone within a finite distance. In sphincter muscle, the skin tissue is squeezed toward an imaginary centre, like the tightening of a string bag. The squeezing can be described as occurring uniformly about a point of contraction. Sheet muscle is a broad float area of muscle fibre strands and does not emanate from a point source. As a result, it contracts to a localized node, rather than to a group of separated muscle fibre nodes. An example is seen in Billy, the baby in the movie “Tin Toy”, who has 47 Water’s muscles on his face.[26].



(a) (b) (c)

Figure 2-3: (a) Contraction of linear muscle, (b) sphincter muscle and (c) sheet muscle

In 1990, Terzopoulos and Waters [27] proposed an extension to the muscle-based model with a three-layer deformable lattice structure. Their three-layers of deformable mesh correspond to skin, fatty tissue, and muscles. Elastic spring elements connect each mesh node and each layer. Muscle forces propagate through the mesh systems to create animation.

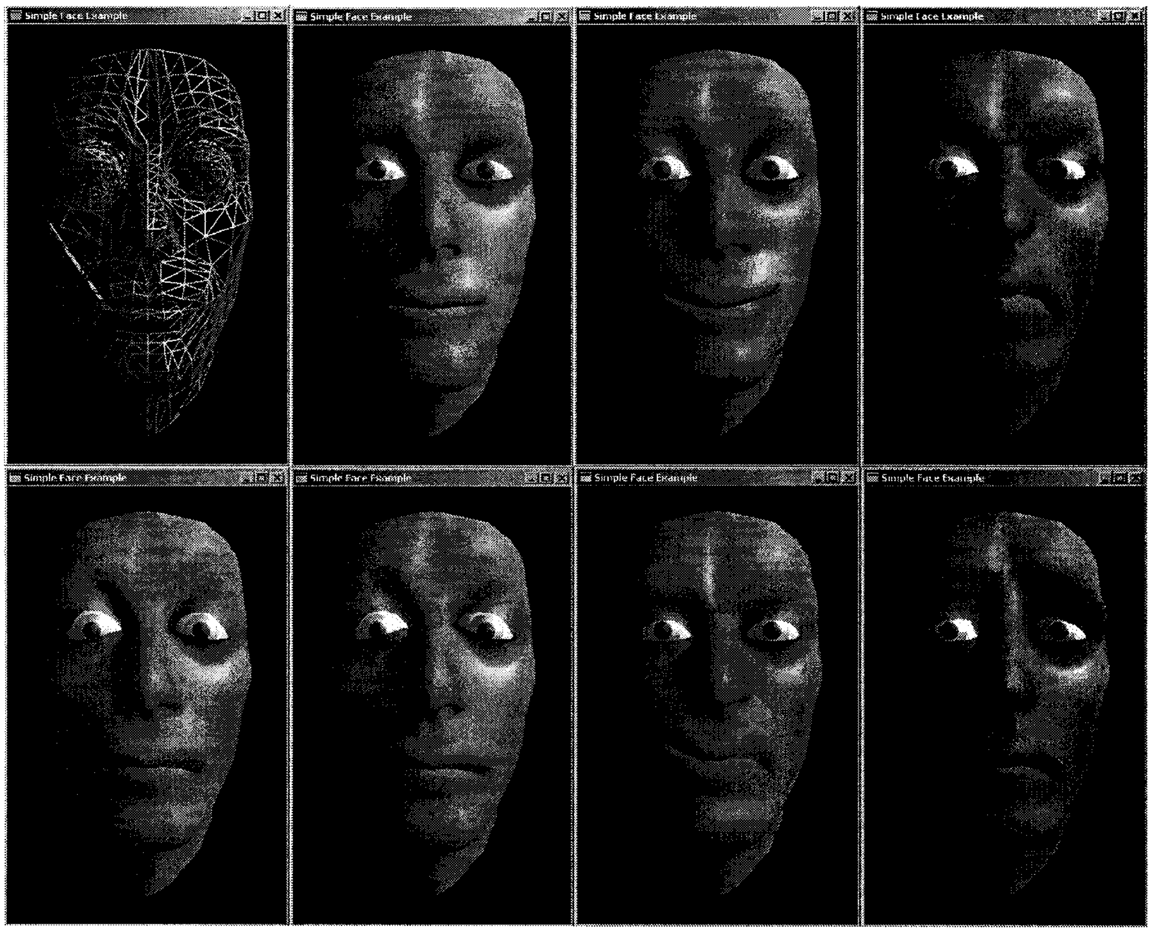


Figure 2-4: Images illustrating expressions created by Waters' muscle model

In most cases, polygon mesh is the primitive of surface representation. However, muscle-based movements when applied to polygonal models will lose some smoothness or flexibility of human face. Fixed polygonal models do not deform smoothly in arbitrary regions, and planar vertices cannot be twisted into curved surface without inserting new vertices. Some approaches try to use Parametric Surface Patches to overcome this problem. For example, Waite [28] used *B-spline* patches to model and animate the face with a muscle model. *Pixar* used *bicubic Catmull-Rom spline* patches to model the baby Billy's face in the animation of "*Tin Toy*" (Copyright 1988 Pixar Animation studios), and used a variant of *Catmull-Clark* [29] subdivision surfaces to model Geri (Geri's game, 1997, Pixar Animation studios). Some authors also classify these methods as Pseudo Muscle-based Animation which is described in the next section [30].

2.2.4 Pseudo Muscle-based Animation

With pseudo muscle-based facial animation, muscle actions are simulated using geometric deformation operators. These techniques include abstract muscle action [31] and freeform deformation. [32]

Abstract muscle action (AMA) is reported by Magnenat-Thalmann et al in 1988. The AMA procedures work on specific regions of the face. Each AMA procedure approximates the action of a single muscle or a group of closely related muscles. For example, the vertical jaw action is responsible for opening the mouth. It is composed of several motions: lowering the corners of the mouth, lowering the lower lip and parts of the upper lip, and rounding the overall lip shape. The AMA procedures are not independent, so the ordering of the actions is important.

Freeform deformation (FFD) is a technique for deforming solid geometric objects. Conceptually, a flexible object is embedded in a three dimensional cubic lattice with control points. As the control points are deformed, so is the embedded flexible object. It can be used to control shape change for surface primitives of any type of degree such as planes, quadrics, parametric surface patches, or implicitly defined surfaces. Coquillart [33] reported an extension of FFD that allows alternative control-point structures such as cylindrical lattices.

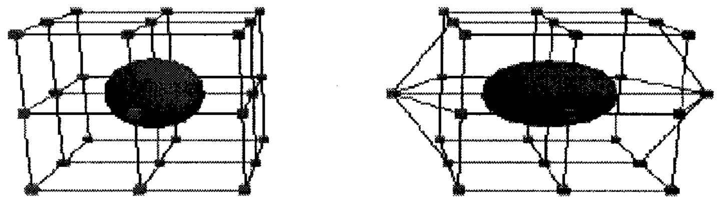


Figure 2-5: Freeform deformation.

Rational freeform deformations (RFFD) are an extension to basic FFDs which use rational basis functions in the formulation of the deformation. The rational basis functions incorporate weights for each control point in the parallelepiped control lattice. Kalra et al. [34] described interactive techniques for simulating facial muscle actions using RFFDs.

2.2.5 Performance-based Animation

Performance-based Animation involves capturing real human actions or tracking video images to drive synthetic characters. Therefore, these approaches rely on the aid of special data capture devices, such as data gloves, instrumented body suits, 3d scanners or digitizer etc. With increasing applications in industry, performance-based animation has become a popular method to achieve lifelike facial animation. Although there are a variety of methods in this category, the basic principles of performance-based animation are straightforward [18].

Accurate tracking of feature points or edges is often needed in performance-based animation. The simplest tracking method is to track markers placed directly on the subject's face. Coloured markers painted on the face or lips are extensively used to ease the process of creating facial expression or speech recognition. There are also some real time marker tracking devices, such as *Eyematic* Inc.'s face tracking system and *Courtesy* of Adaptive Optics.

Snakes, or deformable curves, are widely used to track intentionally marked-up facial features. The human face has many feature lines and feature boundaries, such as brows, furrows and lip margins. A technique commonly referred to *active contour, deformable curve* or *snake* can be used to track features over time. Snakes can also estimate face muscle contraction parameters from image sequences. In turn these parameters can be used to drive a facial animation. [35][36]

Optical-flow is another feature tracking method in image analysis. An example of optical-flow approach can be seen in Essa's work [37]. Optical-flow computations rely on a sequence of images at a pixel level of detail. Essentially, flow can be estimated by tracking the motion of pixels or of a small group of pixels from frame to frame. To be effective, optical flow requires as much textural detail as possible to be present in the region of interest; otherwise it becomes hard to determine where a group of pixels move.

Morphing is used in performance-based animation. Ulgen [38] uses 3D-volume morphing to obtain a smooth transition from a generic facial model to a target model. First, biologically meaningful landmark points are selected (manually) around the eyes, nose, lips, and perimeters of both face models. Second, the landmark points define the coefficients of the Hardy multi-quadric radial basis function used to morph the volume. Finally, points in the generic mesh are interpolated using the coefficients computed from the landmark points. An example uses a generic face with 1251 polygons and a target face of 1157 polygons. Manually, 150 vertices are selected as landmark points, more than 50 around the nose. The success of the morphing depends strongly on the selection of the landmark points. *Pighin* et al. employ a scattered data interpolation technique for a three-stage fitting process [39]. In the first stage, camera parameters (position, orientation, and focal length) are estimated. These are combined with manually selected correspondences to recover the 3D coordinates of feature points on the face. In the second stage, radial basis function coefficients are determined for the morph. In the third stage, additional correspondences facilitate fine-tuning. A sample generic mesh of under 400 polygons is morphed with 13 initial correspondence points, and 99 additional points for final tweaking. Morphing methods produce very realistic facial animations, but they share similar limitations with the interpolation approaches. Selecting corresponding points in target images or geometric models is manually intensive, dependent on viewpoint, and not generalizable to different faces.

Performance-based animation has been even popular in movies and 3D games because of its high realistic effects. A recent film "The HULK" gives us the great visual enjoyment of computer animation (not only facial animation but also body animation) once again [40].

As we shall see later, our animation of Chinese opera faces is also proposed using performance-based facial animation (also called expression mapping). We have also integrated key frame interpolation, parameterization and 3D geometry morphing techniques in our system.

Figure 2-6: Computer animation used in movie "*The HULK*"

2.3 MPEG-4 Facial Animation Standard

In this section we briefly describe the MPEG-4 facial animation standard because it defines a rich set of facial animation parameters. Although MPEG-4 is an audio/video compression and coding standard, it allows more realistic facial animation if one develops more complete and detailed control parameters based on MPEG-4's architecture. One example of MPEG-4 facial animation standard is the computer human interface developed by AT&T Labs-Research [41].

The MPEG-4 Facial Animation Standard specifies a face model in its neutral state, a number of feature points on this neutral face as reference points, and a set of facial animation parameters (FAP), each corresponding to a particular facial action deforming a face model in its neutral state [23].

Deforming a neutral face model according to some specified FAP values at each time instant generates a facial animation sequence. Because the FAPs are required to animate faces of different sizes and proportions, the FAP values are defined in FAP units (FAPUs). The FAPUs are computed from spatial distances between major facial features, like mouth width (MW0) on the model in its neutral state.

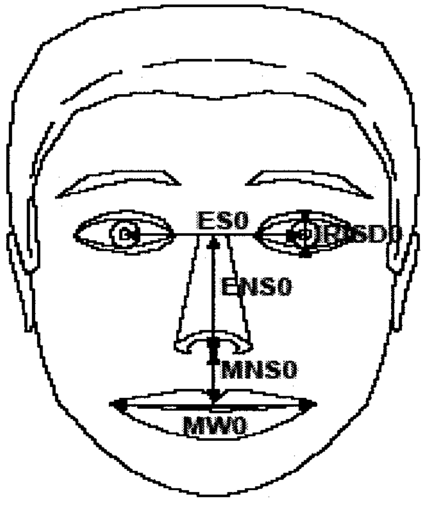


Figure 2-7: A face model and the feature points used to define FAPUs. Fractions of distances between the marked key features are used to define FAPUs.

The MPEG-4 Facial Animation standard specifies two sets of parameters, FDP (Facial Definition Parameters) and FAP (Facial Animation Parameters), to define the geometry and animation of 3D face models.

Facial Definition Parameters: These parameters allow the specification of the geometrical shape of the face model by means of face definition parameters (FDPs); moreover, texture data may also be sent to the decoder. The parameters allow the decoder to create a 3D model with specified shape and texture.

Facial Animation Parameter: These parameters allow the definition of face animation by means of FAPs. FDPs typically are transmitted only once, whereas FAPs are transmitted once for each frame. MPEG-4 defines

68 FAPs to specify face states during animation. Each FAP has predefined semantics and defines one degree of freedom.

2.3.1 Facial Definition Parameters (FDP)

FDPs are designed to enable the customisation of a proprietary 3D facial model together with the information about how to animate it. Whereas the first case allows a limited control on what the animated face will look like, the second case allows for full control of the animation results. MPEG-4 specifies 84 feature points on the neutral face. The main purpose of these feature points is to provide spatial references to key positions on a human face such as major muscles, bones, external facial organs, connection points between muscles and bones, etc. These 84 points were chosen to best reflect facial anatomy and movement ‘mechanics’ of a human face. Feature points are arranged in groups like cheeks, eyes, mouth, etc. The location of these feature points has to be known for any MPEG-4 compliant face model. Figure 2-8 shows a few examples of feature points:

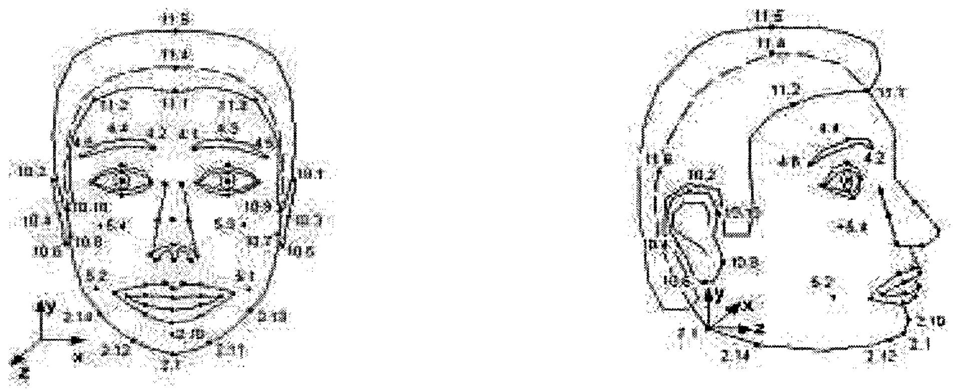


Figure2-8: Some Feature Points defined in MEPG4 standard

2.3.2 Facial Animation Parameters (FAP)

As mentioned before, FAPs were designed to allow the animation of 3D faces, reproducing movements, expressions, emotions, and speech pronunication. FAPs are based on the study of minimal perceptible actions and are closely related to muscle actions. The set of FAPs represents a minimal complete set of basic facial movements including head motion, tongue, eye, and mouth control, allows representation of most natural facial actions as well as exaggerated, non-humanlike actions (e.g., useful for cartoon-like animations).

Table 2-1: FAP groups [MPEG4 - 2]

Group	Number of
1: visemes and expressions	2
2: jaw, chin, inner lowerlip, cornerlips, midlip	16
3: eyeballs, pupils, eyelids	12
4: eyebrow	8
5: cheeks	4
6: tongue	5
7: head rotation	3
8: outer lip positions	10
9: nose	4
10: ears	4

The MPEG-4 FAP set includes 68 FAPs, 66 low-level parameters associated with lips, jaw, eyes, mouth, cheek, nose, and so on, and 2 high-level parameters associated with expressions and visemes (mouth shapes for speech animation). All facial animation parameters are categorized into 10 groups related to parts of the face (*cf* Table 2-1).

All low-level FAPs involving translational movement are expressed in terms of FAPUs. FAPUs are defined to allow the interpretation of FAPs on any facial model in a consistent way (shown in Fig. 2-7). Visemes and expressions are two high-level FAPs. The high-level FAPs allow two visemes/expressions to form a predefined set to be mixed together based on the values of two parameters. These parameters are the visemes/expressions selection parameter and the corresponding intensities. To avoid ambiguities, the visemes/expression FAPs can have impact only on low-level FAPs that are allowed to be interpolated at the time the visemes/expression FAP is applied.

Since the MPEG-4 standard has low bandwidth communication over the net as its primary motivation, fidelity and realism of expression are given much less importance. Hence we are not in a position to use the specification in an as is form for heritage documentation. At the same time, the specifications methodology is general enough for us to adapt it suitably in our application.

3. A Sampled Point Cloud Face Representation

3.1 Face Representation for Chinese Opera Singers

Given the vast body of earlier research in facial animation using polygonal mesh representations, we initially evaluated the use of a triangle mesh for the Chinese opera performers. In general, a polygon model is good for large, flat or subtly curved regions. Polygons have highly efficient rasterization with 3D graphics hardware. However, after a little study we came to the conclusion that we needed a representation with very fine control. Chinese opera is lifelike, exquisite and humorous. It has strong life flavor, but also has systemic and perfect patterns. Among these, facial expressions and fine facial movements highlight the opera performance. Highly elaborate painting of the face depicting specific characters and significant features is another aspect (*cf* Fig. 3-1).

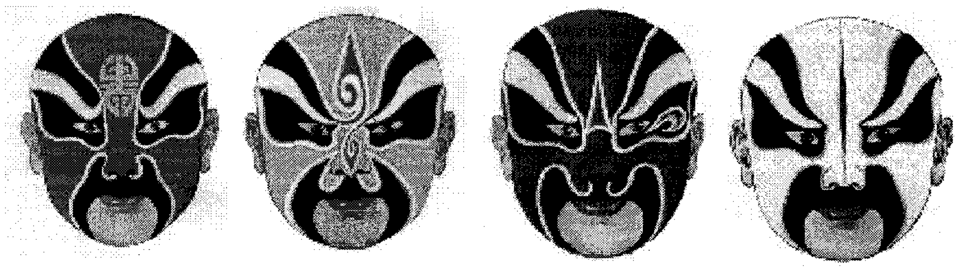


Figure 3-1: Some Painted Faces from Chinese Opera.

Minimalistic polygon models would exhibit unacceptable faceting artifacts, such as angular silhouettes. Processing many small triangles in a very high resolution model would lead to excessive floating point and rasterization requirements. Further, very detailed texture mapping of the polygons would have to be used to convey the painted face details with high fidelity. Texture maps however, have to follow the underlying geometry of the polygon model and work best on flat or slightly curved surfaces. Realistic face depiction would entail a large number of textures that have to be applied in multiple passes during rasterization. A curved surface representation using say, NURBS patches was not considered because of the inherent complexity in high quality manual modeling of faces having different expressions. Geometric surface techniques have been very successfully used in feature films such as PIXAR's Geri's game (subdivision surfaces) or the recent character of Gollum in Lord of the Rings: The Two Towers. However, we have chosen to use a point cloud representation directly without organizing these into an underlying surface representation like, say, a polygonal mesh.

Our motivation - once the techniques can be demonstrated to work – is that 3D scanning technology would make it, relatively speaking, easy to obtain high fidelity point cloud face models. Point cloud representations provide directly discretized point samples of the surface geometry. Furthermore, point clouds are known to work well for models with rich, organic shapes or high surface details, such as human face. In a preprocessing step, computation-intensive calculations such as texture mapping are done for each of the point samples. By moving surface property estimation [42] and texturing computations to a preprocessing step, rendering cost is dramatically reduced. Point cloud models however, have higher cost per pixel, but with very little preprocessing required in the rendering pipeline. In the near future, we expect that increasing processor speeds and graphics acceleration hardware support would allow rendering to be done with real-time performance. At present, we did not have access to a 3D scanner. Hence, we have chosen to use triangle mesh models available for download [52] and convert these into point cloud face models. In the rest of this chapter, we will describe the hierarchical face representation derived by us and our algorithms for constructing this hierarchical data structure from point samples.

3.2 Facial Feature Points and Facial Feature Regions

Our face structure is in many ways similar to that used in polygonal representations – face feature points (FFP), that are continuously transformed to obtain facial expressions. We have defined 870 feature control points on the face (shown in Figure 3-2). Based on these feature points, we group all the points on the face surface into 1480 face feature regions (FFR). We have used this large number of FFPs to address two aspects in our solution:

1) Our goal is to be able to capture the specificity in the facial expressions of Chinese opera singers. We therefore need much finer control over facial feature transformation specification.

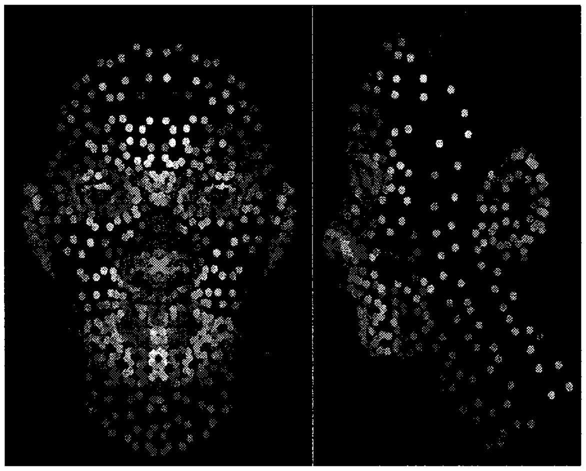


Figure 3-2: Facial Feature Points location

2) In a polygonal model, transforming a FFP automatically transforms the polygon faces incident to that point, correspondingly deforming the face surface. In a point cloud model, every point has to be suitably transformed. There has to be a well-defined mechanism to propagate the transformation specified at the feature point to all the other points in the immediately connected feature regions. The nonlinear surface deformation effect is thus obtained by displacing the feature points and using affine transformations applied to small triangular feature regions. This leads us to the requirement of a fine tessellation that maintains near planarity of the smallest element – feature regions.

Although confirming FFPs is a manual job, we can give some useful clues to do it. First, we decide the special feature points nose tip and chin tip because nose tip is most outstanding point and chin tip is right under it. Second, mouth contour and chin contour are easy to locate. Then, we locate the ears, eyes and eyebrows. Finally, it is not hard to place the rest other FFPs. We accept that identification of so many feature points from point cloud samples is a tedious task, but it provides us a mechanism to catch exquisite facial feature. And later, we shall consider building semiautomatic interactive FFP identification tools that assist in this process. Some related work has been done in Yin [43] and Lee's [44] work.

Each FFR is defined by choosing three boundary feature points. A sample point belongs to this FFR, if it is closest to this triangle and its projection falls within the bounding triangle.

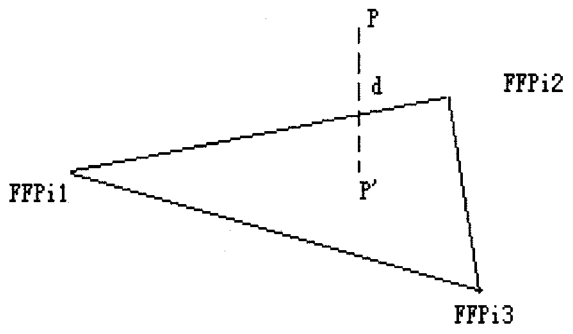


Figure 3-3: Point cloud related to FFPs

As shown in Figure 3-3, an FFR_i has 3 associated FFPs (FFPi1, FFPi2 and FFPi3). These 3 FFPs make up a 3D triangle T. If point P belongs to FFR_i, then its projection

P' = a* FFPi1 + ß* FFPi2 + (1 –a–ß) * FFPi3, with a,ß, 1 –a–ß? [0, 1],

And, P = P' + F * d, where F is the face normal of the FFP triangle, d is the distance from P to P'.

After we get the topology of FFP construction, our next task is to divide all the point clouds into facial feature regions. We first define a boundary cube volume for each FFP triangle. For those points that fall inside the cube, we compute their orthographic projection onto the FFP triangle plane. For each FFP triangle, we exclude the points whose projections are outside of the triangle. At last we group the points closest to the same FFP triangle into one FFR.

The simplest way to check whether a projection point (P') falls inside a triangle is area checking. As shown in Figure 3-4, if point P' is inside triangle ABC only when the summed area of triangles AP'B, BP'C and CP'A is equal to the area of triangle ABC.

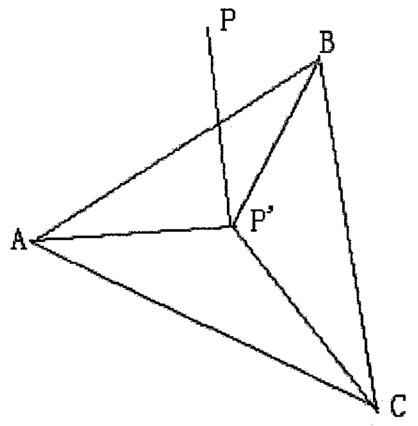


Figure 3-4: Projection of points onto a triangle plane

Because a triangle's area is a half of the parallelogram's area which share two edges with this triangle, we use the following formula to calculate a triangle's (3 vertices are (x_1, y_1, z_1), $(x_2, y_2, z2)$, (x_3, y_3, z_3)) area.

$$\text{area } S = 0.5 * \text{length} \left(\begin{vmatrix} i & j & k \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} \right)$$

3.3 Facial Action Units and Facial Animation Coding System

In order to enable simple analysis of facial animations, we also classify feature regions into 23 Facial Action Units (FAU) according to their biological locations (*cf* Fig. 3-5). Actions units are useful in developing Facial animation coding system (FACS)[50], which describes the relationship between facial expressions and basic facial units.

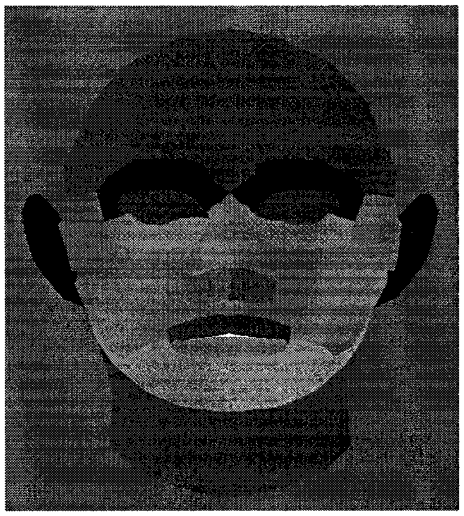


Figure 3-5: Facial Action Units in Pseudo-colour

FACS is a widely used technique to describe facial expressions [18]. Moreover, FACS can be used as a way to control facial movement by specifying the action units needed to achieve desired expression changes. Ekman and his colleagues [45] developed a well-known scheme for muscle-based facial animation. In our approach, we use 7 primary expressions and 14 modifiers to derive our FACS. With the help of FACS, we may combine some independent expressions, such as the eyelid movement and mouth movement.

The 1480 facial feature regions are arranged into a hierarchic structure shown in Fig.3-6.

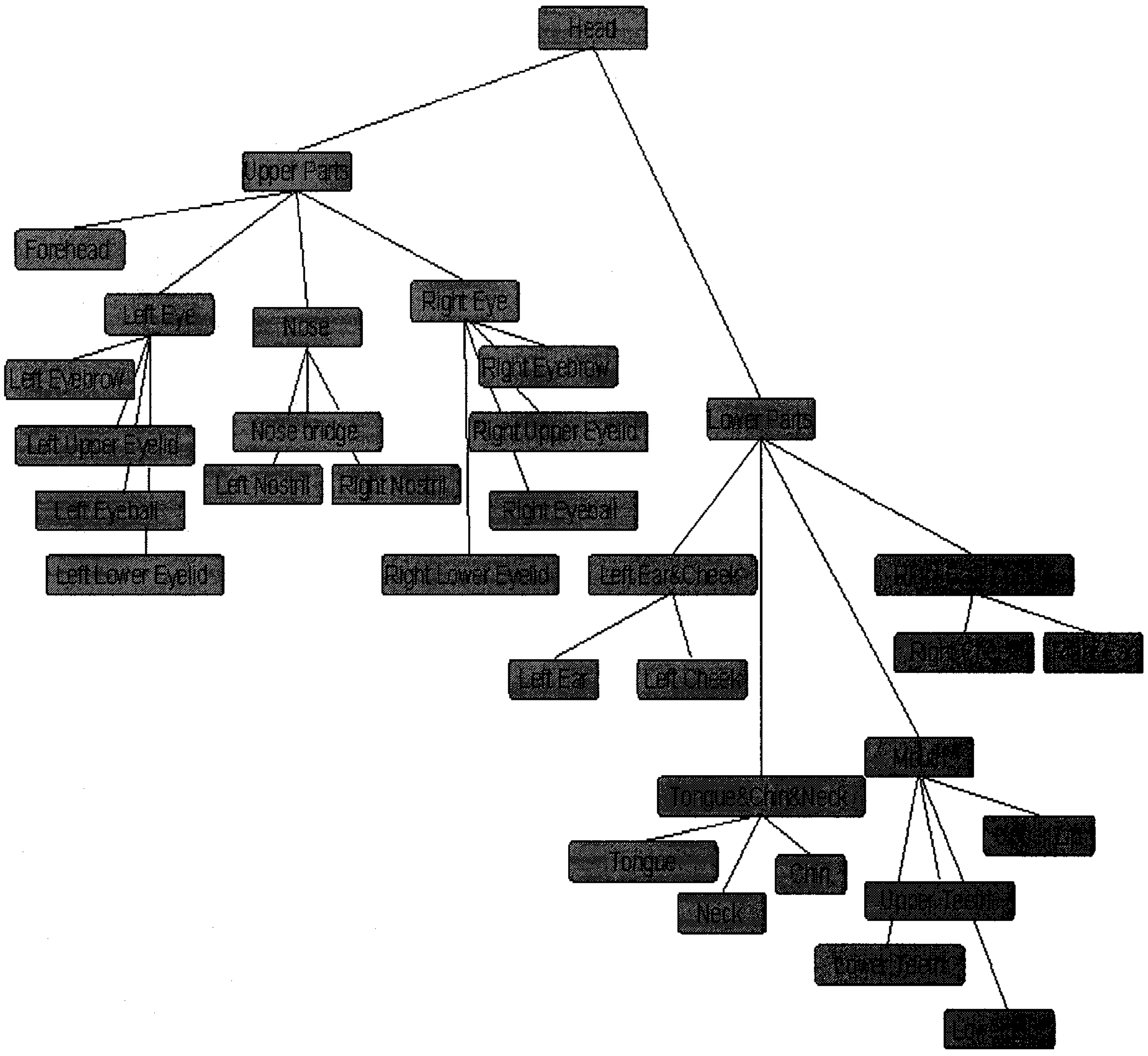


Figure 3-6: Face Structure Composed of Facial Feature Action Units

3.4 Hierarchical Representation Structure

Q-Splat point rendering system is a fast multi-resolution rendering system. Its advantages such as visibility culling, level-of-detail control, etc. enable it to render large meshes in real time. Therefore, we decide to derive our data structure from Q-Splat and extend some of the attributes to make it suitable for facial animation.

Presently, the generic process used to arrive at the data structure is as follows:

First of all, we identify facial feature points on the face model although it is tedious work.

Second, the whole face is divided into facial action units. In each unit, a set of distinct FFRs are identified as already shown in Table 3-1. We have chosen the FFP set to be comprehensive. For specific cases, we may achieve the required facial movement with fewer FFPs.

Table 3-1: Composition of Facial Action Unit from Facial Feature Regions

Action Unit	**#FFRs**	**#FFPs**	**Action Unit**	**#FFRs**	**#FFPs**	**Action Unit**	**#FFRs**	**#FFPs**	**Action Unit**	**#FFRs**	**#FFPs**
Forehead	58	87	Left ear	36	60	Left eye ball	44	70	Right cheek	69	85
Left eyebrow	28	34	Right ear	36	60	Right eye ball	44	70	Tongue	45	64
Right eyebrow	28	34	Upper lip	89	122	Left Lower eyelid	37	50	Chin	95	150
Left Upper Eyelid	43	60	Lower lip	85	111	Right Lower eyelid	37	50	Neck	48	66
Right Upper eyelid	43	60	Upper teeth	40	20	Nose Bridge	40	54			
Left cheek	69	85	Lower teeth	22	20	Left nostril	44	66	Right nostril	44	85

Third, we compute the bounding sphere of each FFR. In our approach, we take the pivot circle center of the FFP triangle as the sphere center and the radius is the longest distance from every point within FFR to sphere center. If FFPs are selected properly, the radius is the distance from one of 3 FFPs to the bounding sphere center (shown in Figure 3-7).

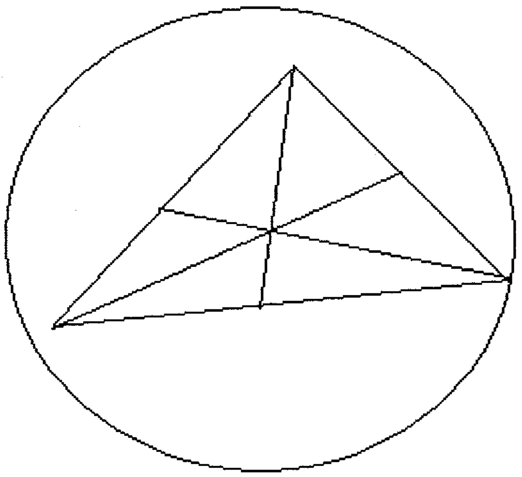


Figure 3-7: Bounding sphere of FFP triangle

Next, we group these spheres by action units, and form action unit sub-trees shown in Fig. 3-6 and thus build the facial structure tree. Since every FFR corresponds to a bounding sphere, we label these spheres with their facial action units. Therefore, in each action unit, there are about 30 to 150 FFRs. While a Q-Splat tree node has four children at most, the Q-Splat sub-tree for each action unit has 3 to 5 levels. Figure 3-8 shows the progress of building the action unit sub-tree. In this process, we keep clustering those spheres which are very close to each other and combining them into a larger sphere. At last, we get one Q-Splat sub-tree whose leaves cover all FFR bounding spheres.

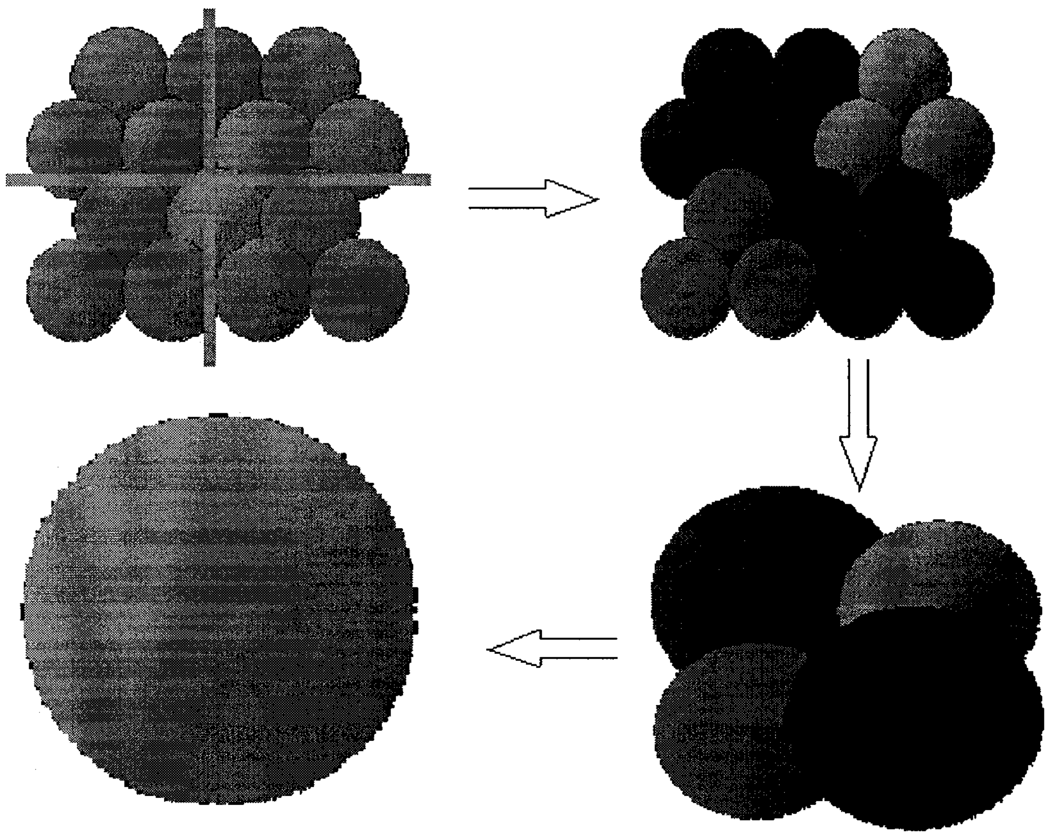
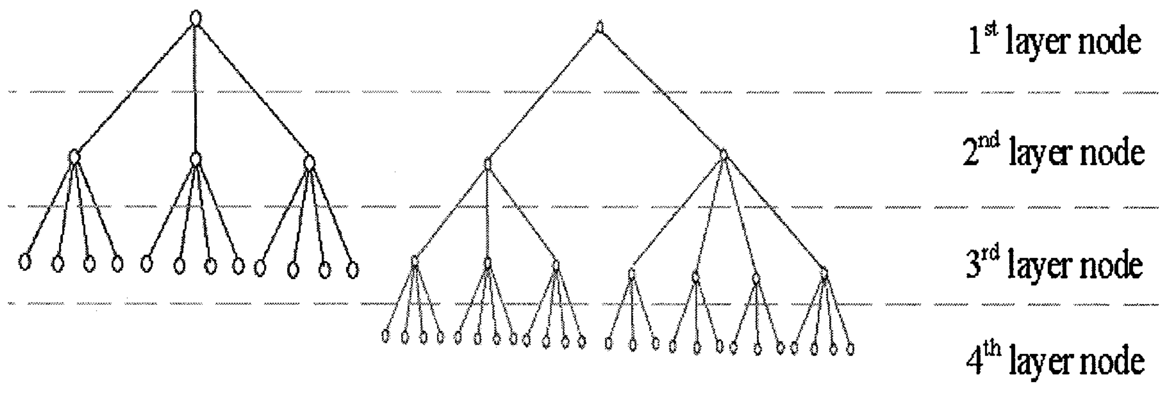


Figure 3-8: Building sub-tree of facial action unit

Q-Splat tree is not a balanced tree. But, in order to create a fast rendering data structure, we try our best to build action unit sub-trees with lowest height and each branch has almost same numbers of children. Fig. 3-9 shows the structure of the tree when the leaves are 12 and 25.

```
4 | 12        ..........................12  3rd layer node
  ------------
   4 | 3    0 ..........................3   2nd layer node
     --------
       0    3 .......................... 1  1st layer node

4 | 25        ..........................25 4th layer node
  ------------
   4 | 6    1 ..........................7  3rd layer node
     --------
   4 | 1    2 ..........................2  2nd layer node
     --------
       1    0 ..........................1  1st layer node
```



(a) **(b)**

Figure 3-9: Q-Splat tree structure (a) 12 leaves (b) 25 leaves

Furthermore, we combine these sub-trees as the structure shown in Figure 3-6. There are only 3 cases of combination because a Q-Splat tree node has at most 4 children. Figure 3-10 shows all the three cases of combination.

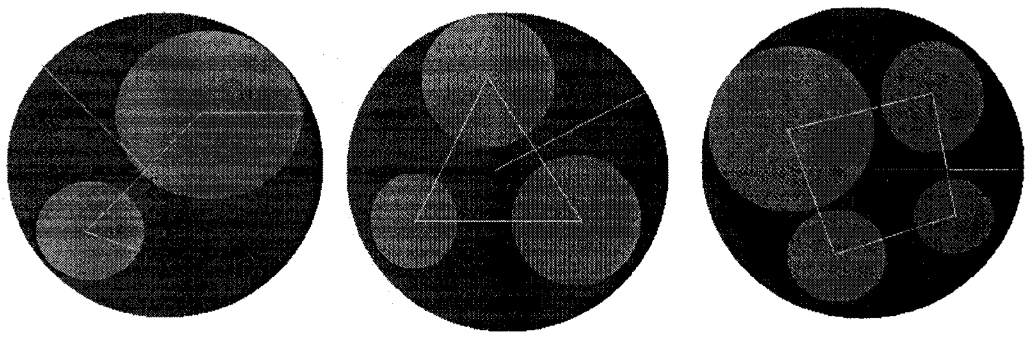


Figure 3-10: Three Combining methods

Finally, it is the process of subdividing the triangle to obtain the point samples. Each triangle is recursively subdivided into 4 triangles. FFR surface is also subdivided accordingly. When the process goes up to a given resolution, bounding sphere of the resulting triangles are considered as 3D points making up the point cloud. Figure 3-11 shows the hierarchical structure of subdivision of FFP triangle.

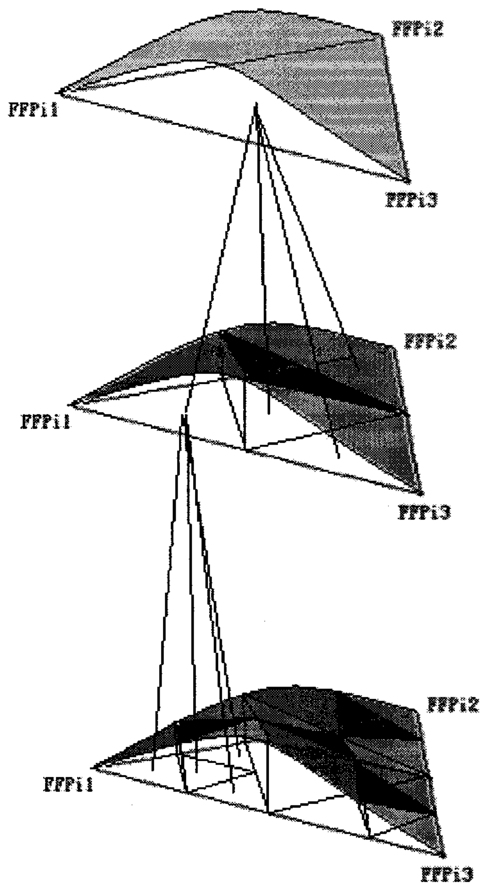


Figure 3-11. Hierarchic structure of point cloud organization

Each point is given a colour to depict the painted face of the character being modeled. Fig. 3-12 shows a few of the Chinese opera characters modeled as point clouds using the above process.

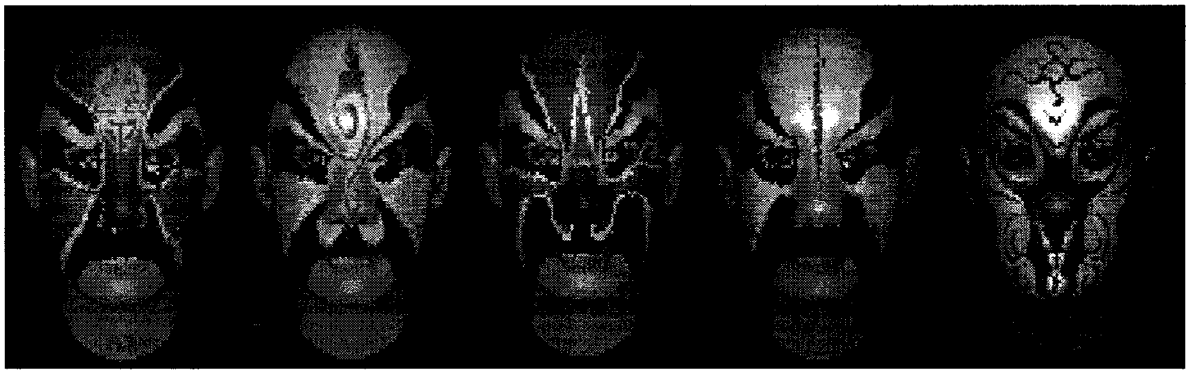


Figure 3-12: 3D point cloud models for a few Chinese opera characters

Our purpose is providing a reasonable method to build a tree structure for a face model based on the data from 3D scanner. However, in the absence of easy access to a live performer and a 3D scanner, we derived point cloud models from triangle meshes for experiment and demonstration purpose.

In the next chapter we describe how feature transformations are specified along with the tree nodes to obtain desired animations depicting facial expressions.

4. Algorithms For Facial Expressions

4.1 Facial Expressions in Point Sampled Surface Model

Every facial expression in a point sampled model of the face can be considered as a sequence of transformations that every point and associated normal undergoes. In its most general form, every point may have its own unique transformation sequence. However, in practice, groups of points will undergo the same transformation. For example, in the simple gesture of turning the head, most of the face points, excepting the neck points, will undergo the same rotation transformation. In what follows, we shall term an animated facial expression as a facial animation primitive.

4.1.1 Facial Animation Parameters

Since the computation for transforming all points is slow, feature regions are defined as transformation areas. Therefore, our facial animation is regarded as FFR deformation controlled by associated FFPs. If FFPs are defined appropriately, FFR deformation can be simulated by affine transformation. However, in some cases, some points move individually besides the FFR movements. To produce more realistic results, we can add point level movement to augment FFR deformation obtained using an affine transformation. FFR deformation and point movement will give us very fine control over the animation.

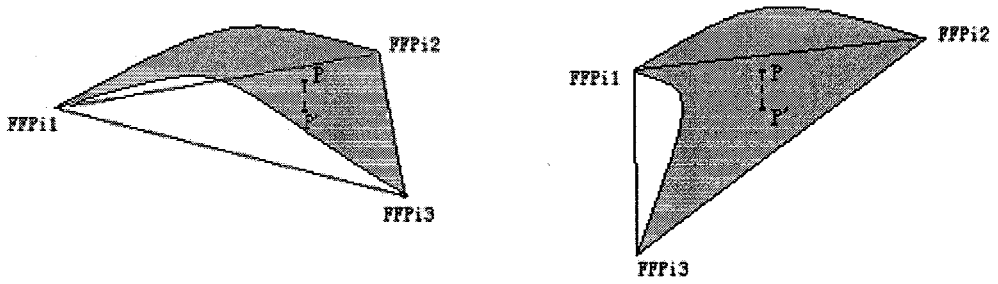


Figure 4-1 : FFR deformation and point movement

As illustrated in Fig. 4-1, the change in coordinates is an affine transformation of the associated FFP triangle, and the change in (a,ß, d) is point level movement. We define Facial Animation Parameter (FAP) as all the FFR transformations and individual point transformations which transform one face expression to another.

Our method for determining primary FAPs set for different set of facial animations is a combination of direct parameterization and key frame interpolation techniques. The FAPs for each facial animation primitive are determined as follows. We assume animation starts from the normal pose. We define an end pose, intermediate key poses and the time interval(s). Given this sequence: *(normalpose,keypose1,keypose2, ... , endpose),* we must obtain the corresponding FAP_i. A simple procedure for this is described in the following paragraph. The starting FAP for the normal pose has only to identity transformations at all nodes. During the rendering of this animation primitive, we linearly interpolate to obtain the in-between FAP_i for each of the frames.

For this we consider points subset {N} for each FFR, then try to derive the best fitting FAP such that the FFR subset $\{N\}_i$ transforms to the corresponding FFR subset $\{N\}_{i+1}$ of the successor pose. This is a problem well addressed in the Computer Vision community and is known as the pose estimation problem [46]. In our implementation we have chosen a simple but effective method. In the normal pose, with each FFR we have associated 3 FFPs. Their positions in two successive poses are used to derive an affine transformation for the point cloud within this FFR.

As mentioned earlier, the large number of FFPs that we have chosen for the face structure enables us to obtain non-linear deformation of the face surface by using piecewise linear (affine) deformations. This is the same as what happens in a high-resolution triangle mesh model. Every triangle undergoes an affine transformation; collectively it results in non-linear deformation of the face. The FAP gives us complete control in defining any facial animation to the level of detail desired. Also, with key poses being obtainable using 3D scanning technology, FAP for obtaining high fidelity in rendering the animation can be derived. For our experimentation work, as shown in Table 4-1, we have implemented a normal pose and 37 key expression poses. These include 7 primary expression poses, 14 modifier poses and 16 Phoneme poses. This list is only a basic FAP set.

Table 4-1: List of Animation Primitives Implemented

Expression (7)	Modifier (14)	Phoneme (16)
Anger	Blink Left Eye	Aah
Disgust	Blink Right Eye	B, M, P
Fear	Left Brow Down	Big aah
Sad	Right Brow Down	ch, J, sh
Smile (close Mouth)	Left Brow In	D, S, T
Smile (open Mouth)	Right Brow In	Ee
Surprise	Left Brow Up	Eh
	Right Brow Up	F, V
	Left Eye Squint	Ei
	Right Eye Squint	K
	Look Down	N
	Look Left	Oh
	Look Right	ooh, Q
	Look Up	R
		Th
		W

4.1.2 Construction of Facial Animation Parameters

As mentioned earlier, our facial animation method is based on a set of point cloud models. We build the facial animation parameters by comparing the point cloud models in different expressions to the normal face model. Before we compare any point cloud models, we should first mark FFPs on every model and build the correspondence between them. In section 3.2, we have discussed the method of Mapping FFP and Grouping FFR, so we do not repeat it here.

Facial Animation Parameters include FFR deformation and point movements. FFR deformation is a group movement caused by the displacement of the associated FFPs. Comparing the associated FFP in each expression to the normal face, the affine transformation of FFP triangle is the transformation of all points in the FFR. It may be noted that not all feature points have to be transformed for each facial movement. For each expression, we identify the FFPs that have to be transformed and derive the transformation. Unaffected FFPs have the identity transformation associated with them.

Point movement parameter describes the movement of the points inside each FFR. To keep correspondence to hierarchy data structure, point movement parameter is also rendered hierarchically. We build the Q-Splat tree structure for every point model using the method that we discussed in section 3.4. Then, we compute the point movement by comparing the tree nodes in the same position of the tree. Since the number of points in 3D scans of different facial expressions may not be same, some points may not find the matching points between two successive key poses. We therefore need to add/delete points when handling point level transformations during the animation. This is simple as we have the barycentric coordinates for each point within the FFP triangle.

Through point movements, we may get better realistic animation than those implemented with triangle models theoretically. However, if we have large number of FFPs, the actual displacement value for the points takes a very small value and can be omitted. FFP deformation could provide good and fast animation.

4.2 Extending the Range of Expressions

The primary set of FAPs is far from enough to create realistic facial animation. We need to improve our FAP set by obtaining more 3D scanned expressions and identifying FFP locations. Since computing an FFP-driven expression is a time-consuming work, we try to take advantage of all the FAPs we have got to extend the range of expressions. Parameter combination and expression interpolation which we have experimented with appear to be efficient ways with relatively lower cost.

4.2.1 Parameter Combination

Every FAP concerns a set of FFPs and FFRs, and these FFPs and FFRs belong to some FAUs. The relationship between FAP and FAU can be looked up through FACS. When two FAPs are totally independent, which means they do not affect the same FAU, they can be trivially combined to generate a new expression. One obvious example is FAP8 "*Blink Left eye*" and FAP9 "*Blink right eye*". FAP 8 works on FAU4 "*Left Upper eyelid*" and FAU8 "*Left Upper eyelid*", while FAP9 affects on FAU5 "*Right Upper eyelid*" and FAU9 "*Right Upper eyelid*". They are independent of each other. The combination of FAP8 and FAP9 leads to a new FAP "Blink both eyes" (shown in Figure 4-2).

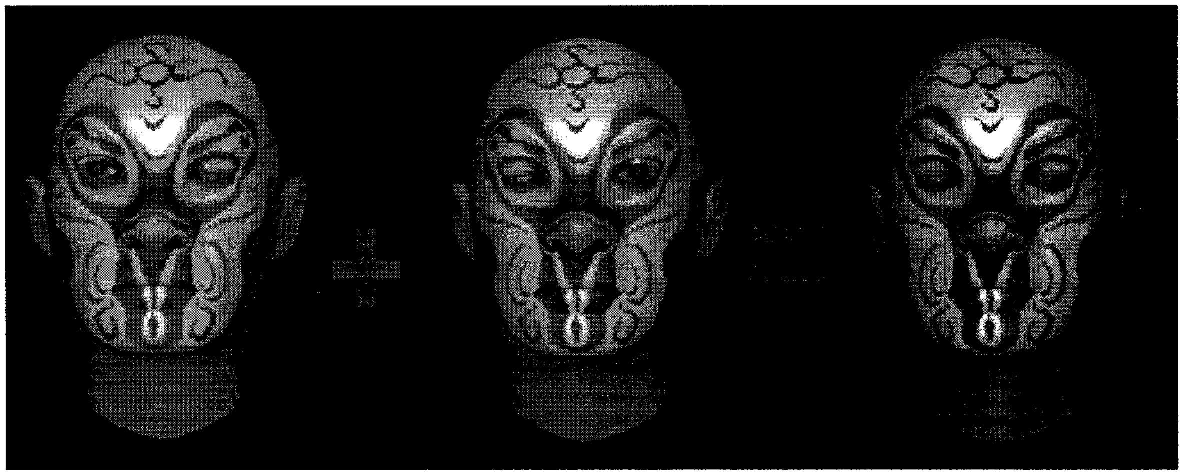


Figure4-2: Parameter combination

4.2.2 Key Expression Interpolation

Interpolation is widely used in our approach for implementing and controlling facial animation. As discussed in the last section, the pose of each expression is mainly determined by FFP positions and we have matched FFPs in different expressions. In-between poses can be derived through interpolating the corresponding FFPs positions. Our experiments also show that most interpolated in-between poses are reasonable.

Interpolation not only creates smooth animation by generating middle frames between two key frames but can also be used to generate new expressions. Normally, a single interpolation can interpolate two key expressions. A bilinear interpolation will blend four key expressions with two interpolation parameters. And, an n-dimensional interpolation allows blending in n-dimensional interpolation space. In a word, interpolation can be used to create a wide range of facial expression changes. Of course arbitrary, interpolation may result in unrealistic expression. Hence, the interpolation parameters have to be carefully chosen.

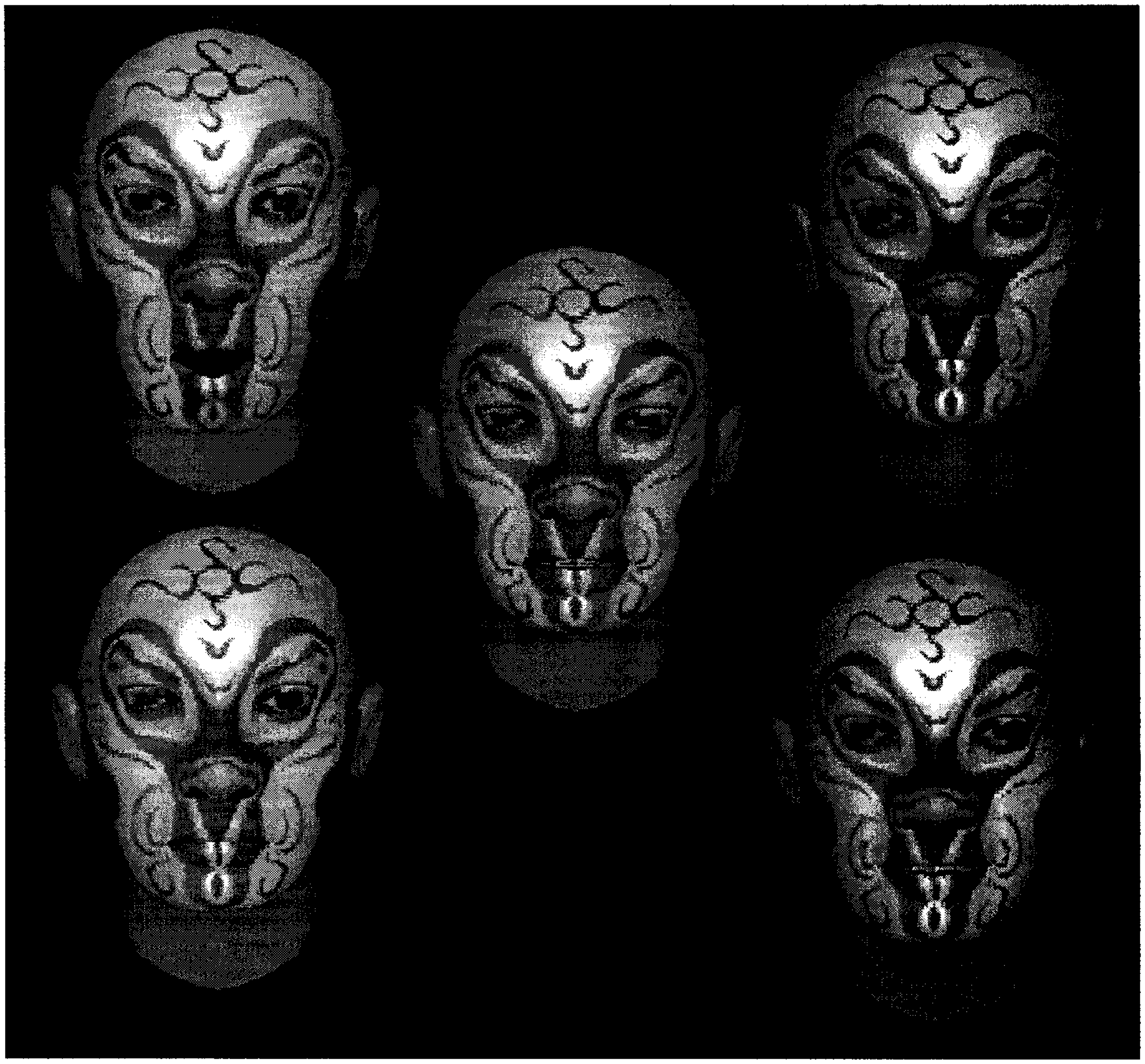


Figure4-3: Bilinear Interpolation of facial expressions

4.2.3 Video Clips Simulation

As an experiment, we spliced a short sequence involving the "Monkey King" character from a video (we only had access to rather low resolution video). We then hand coded a sequence of expressions with timing durations corresponding as closely as possible to the expressions of the "Monkey King" character in the video clip. Fig. 4-4 shows a few frames from a Chinese opera video and the corresponding 3D poses from animations created using our techniques. For the complete video clip synthesised as part of our research, please see http://www3.sympatico.ca/hao.zhou/thesis/thesis.htm.

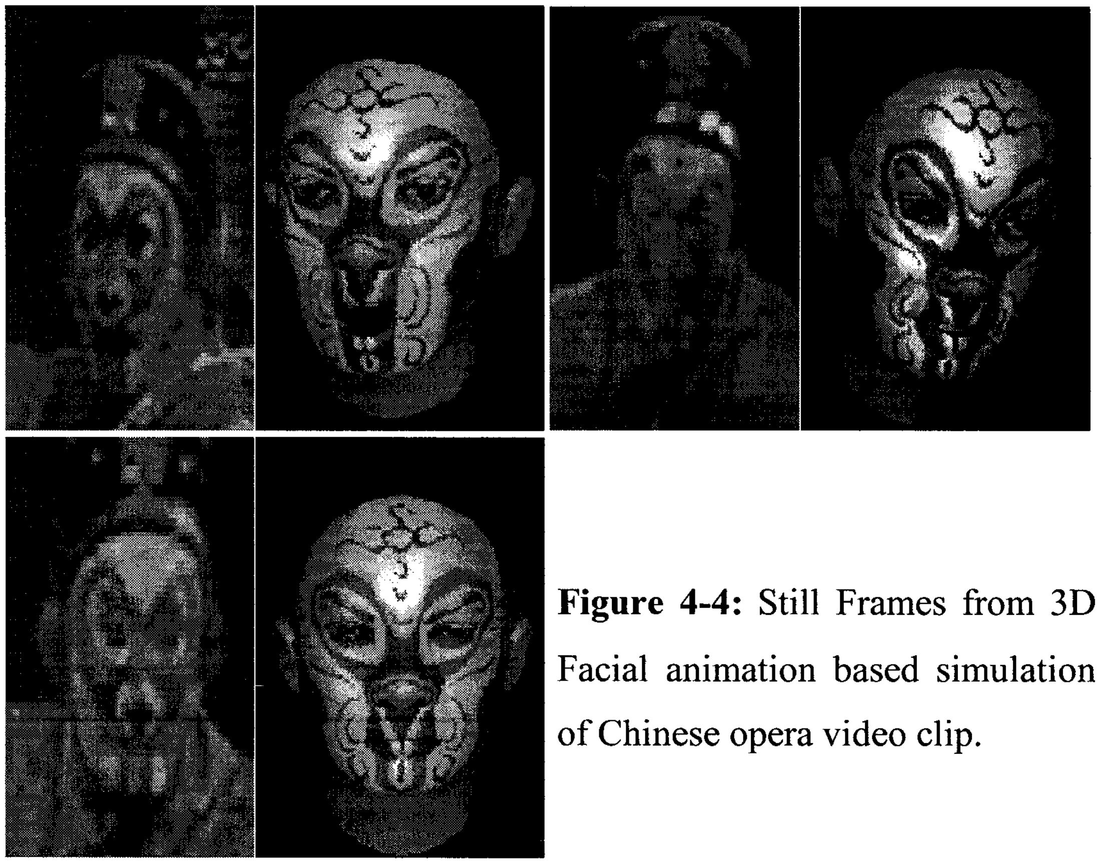


Figure 4-4: Still Frames from 3D Facial animation based simulation of Chinese opera video clip.

4.3. Rendering Point Cloud Models

Our rendering method is mostly based on the method used in the Q-Splat system. Q-Splat rendering is a recursive process aimed at display a static object represented as a hierarchical point cloud structure. We extended the rendering algorithm to be applicable for facial animation. To the best of our knowledge, ours is the first such attempt to extend the Q-Splat technique to support facial animation.

4.3.1 Basic Rendering Algorithm

In Q-Splat rendering, a traverse function starts from the root of the tree, and travels through the whole tree until it reaches a leaf node or the screen size of the node is small enough. Since only objects that fall inside the view frustum (the pyramid-shaped view volume made up by the near clipping plane, far clipping plane, and four intersecting infinite planes) need to be rendered. To increase the rendering speed, the Q-Splat system does frustum culling by testing each node against the planes of the view frustum. If a node lies outside, this node and its sub-tree are discarded and not processed further.

We added a transformation check in the traversal process. Before a node is drawn, we check the data structure whether there is any transformation parameter assigned to it. Transformation could be FFR deformation or point movement. If transformation parameter is not empty

(Identity), we set the transformation and then paint a splat. For a Splat, we have chosen a reasonable size 3D point primitive of OpenGL.

```
TraverseHierarchy(node)
{
    if (facial animation parameter is set)
        Set transformation;
    if (node not visible)
        skip this branch of the tree
    else if (node is a leaf node or benefit of further recursion is too low )
        draw a splat
    else for each child node
        TraverseHierarchy(child node)
}
```

4.3.2 Splat Shapes

Q-Splat does a good job of point cloud rendering. Q-Splat supports several splat shapes, square OpenGL point, small triangle, texture-mapped polygon, sphere, ellipse, etc. Each splat is drawn according to the splat node's position, size, color and normal. The following figures (from Figure 4-5 to Figure 4-10) show the point cloud model drawn by different splat shapes.

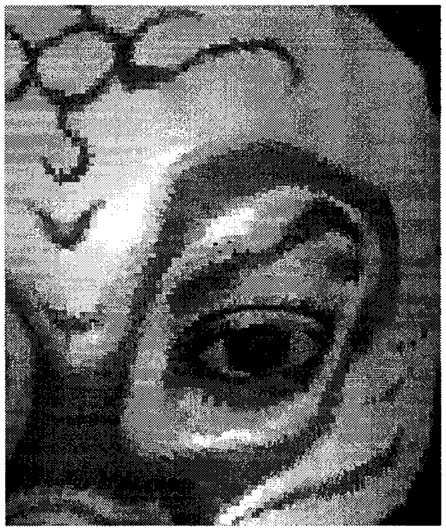


Figure 4-5: Point cloud model drawn using OpenGL points

Splat shape: ■

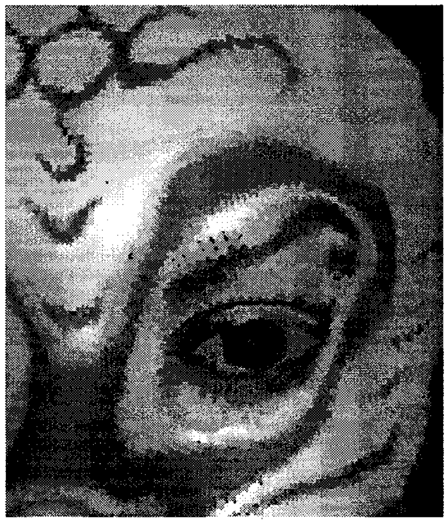


Figure 4-6: Point cloud model drawn using OpenGL round points

Splat shape: ●

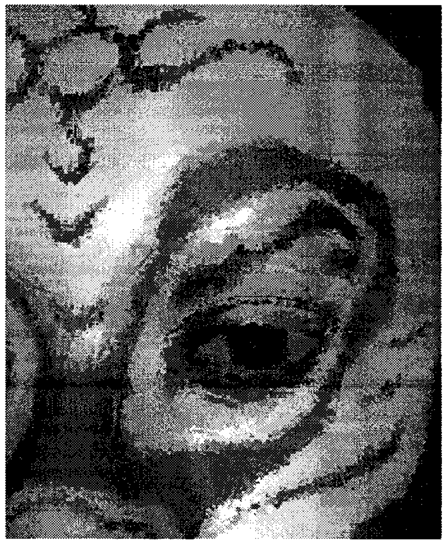


Figure 4-7: Point cloud model drawn using quads

Splat shape: ■

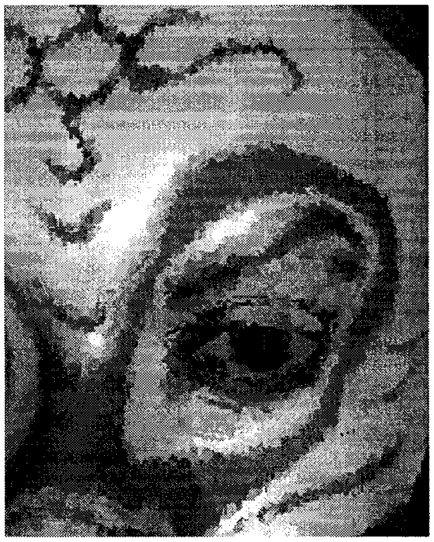


Figure 4-8: Point cloud model drawn using triangles

Splat shape:

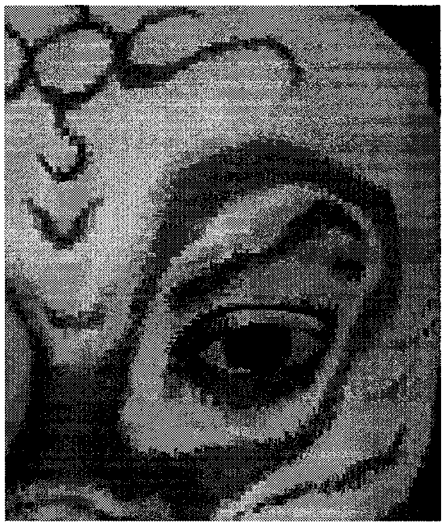
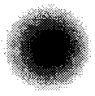


Figure 4-9: Point cloud model drawn using Gaussians texture splatting

Splat shape: 

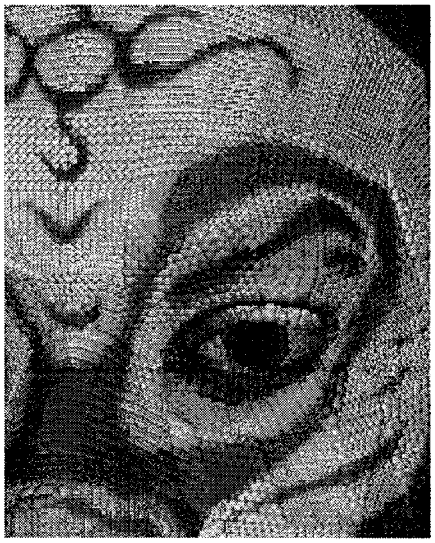
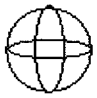


Figure 4-10: Point cloud model drawn using Spheres

Splat shape: 

In Figure 4-5 and 4-6, we may find some holes near the eyebrow tips area. To show them clearly, we use a circle to mark it out in Figure 4-11. The reason is some splat sizes become greater than the maximum size of OpenGL point size after we enlarge the 3d point model. According to OpenGL manuals, the diameter of rasterized points must lie within a support point size range. If the requested size is beyond the maximum point size, only the maximum size point is drawn. However, this problem is easy to fixed by changing the splat shape to quad if the point size is too big.

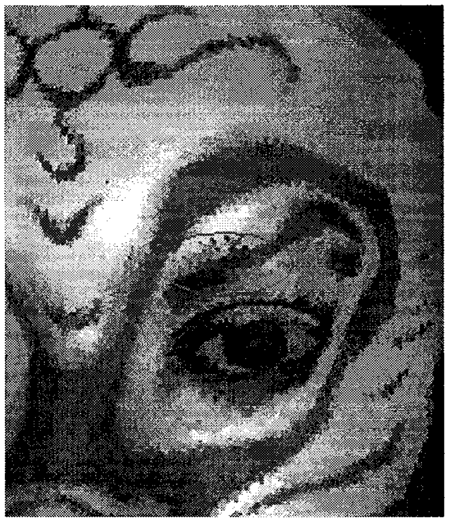


Figure 4-11: Holes appearing in highly zoomed-in image of point cloud model.

4.4 Real Time Animation

As an experiment, we implemented 3 segments of facial animation simulating some sample video clips. In first step, we separated the sound track from the video. Then, we cut the whole video into several actions.

Finally, we simulated these actions in 3D animation and reattached the sound again.

An animation script is used to describe the preset actions and their play time. The script format is as follows.

```
Animation Begin
    Audio Handle
    Action1
        FAP1
        X, Y, Z Rotation
        Play time1
        Display Message1
        Still Flag
    Action2
        FAP2
        X, Y, Z Rotation
        Play time2
        Display Message2
        Still Flag
    ……
    Actionn
        FAPn
        X, Y, Z Rotation
        Play time n
        Display Message n
        Still Flag
Animation End
```

Audio Handle identified which piece of audio will be played in this animation. In each action, FAP specified the facial animation parameter corresponding to the action; X, Y, Z Rotation is the rotation of the whole model in this action; Play time and display message are the total action time and the message text displayed in the message bar. Still flag indicates whether the model will keep a still pose or undergo a linear interpolation in the action period.

5. Implementation Details And Results

In this chapter, we will briefly describe the program we have implemented for facial animation. This includes data structure, system organization and animation control flow. Our system is developed with C++ and OpenGL.

5.1 Face Representation Structure

As we discussed earlier, our face representation structure is derived from Q-Splat tree. A Q-splat tree node contains the location and size of a sphere relative to its parent, a normal, the width of a cone of normals, a color, and a few bits used in representing the structure of the tree. Our structure is similar to Q-Splat tree, defined as follows:

```
class CQSplatTreeNode
{
        DWORD            m_Serial;
        float            m_Radius;
        float            m_Color[6][3];

        float            m_Alpha [3];
        float            m_Beta [3];
        float            m_d ;
        float           *m_NormRef;
        float           *m_PosRef;
        ......

        int               m_ChildNum;
        CQSplatTreeNode* m_Child[4];
};
```

```
class CQSplatTree
{
        CQSplatTreeNode * m_Root;
        int                          m_Palette;
        float                        m_SplatSize;
        SPLAT_TYPE          m_SplatShape;
};
```

In Q-splat tree, each node's position and size is related to its parent. Our approach is an FFP driven facial animation, so our tree node's position, size and outward face normal is mainly determined by FFPs assigned to this node. Through this structure, we can see that a node's position is controlled by 3 factors (*m_Alpha, m_Beta, m_d*) and its relative FDP's position (*m_PosRef*).

As we can see from the structure, colors are saved at every tree node. In our implementation, we can associate different color schemes for one point cloud model (*cf* Figure 3-12). For example, presently, there are 6 sets of color values in each node. In the rendering process, changing the color scheme will lead to a new face. Point cloud models are more efficient to control the color display because it is able to control the color at the point level. Figure 5-1 a hypothetical case of a painted face whose left part and right part of face are painted in different color schemes.

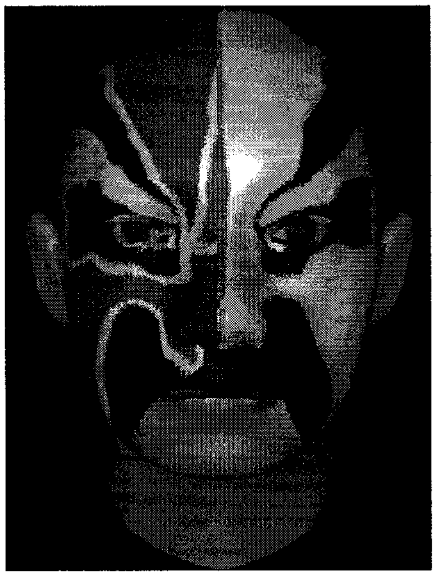


Figure 5-1: 3D point cloud models whose left part and right part of face are painted in different color.

5.2 FFP and FFR Structure

The data structure that we have used to describe FFPs is quite simple. It is a list of FFP positions and outward normals.

```
typedef struct _FFP
{
        float       m_Pos[3];
        float       m_Norm[3];
} FFP;
std::vector<FFP>        m_FFP;
```

FFR structure shows the relationship between the FFP and Q-Splat tree. Each FFR corresponds to 3 relative FFPs and a Q-Splat tree node, which stands for a piece of the point cloud surface. A FFR structure is defined as follows.

```
typedef struct _FFR
{
        WORD            m_Serial;
        WORD            m_FFPIndex[3];
        bool            m_Updated;
} FFR;
std::vector<FFR>                        m_FFR;
```

In the above structure, *m_Serial* refers to Q-Splat tree node serial number and *m_FFPIndex* specifies the index of relative FFP in the list. *m_Updated* indicates if the FFR needs repainting or not.

5.3 FAP Structure

As mentioned before, each FAP stands for a certain expression, and it includes two kinds of movements, FFR movements and Point movements. However, in our implementation we used a large number of FDPs, so point movement is very small and could be omitted. Therefore, our FAP structure consists of a subset of FDP displacements. We define the FAP structure as follows.

```
typedef struct _FAP
{
        std::map<WORD, FFP>             m_FFP;
} FAP;
std::vector<FAP>                        m_FAP;
```

From the structure, we can see each FAP is a set of new positions and normals of FFPs.

5.3 System Organization

Our system is simple as illustrated in Figure 5-2. CEngine is a frame module encapsulated all the sub-modules, data and functions needed for animation. CEngine consists of 4 embedded modules, CAudio, CCamera, CTtimer, Chead, and an Animation Process. CAudio module is in charge of reading the sound resources and managing the sound playing. CTimer module supplies a high resolution timer for facial animation. CCamera module initialises the 3D scene rendering, sets the viewport, projection and places object in the proper position. CHead module is the core module in the whole system. It includes the face representation data and the FFP, FFR and FAP information. The jobs of CHead module include rendering the face structure and affecting the facial expressions on the face models.

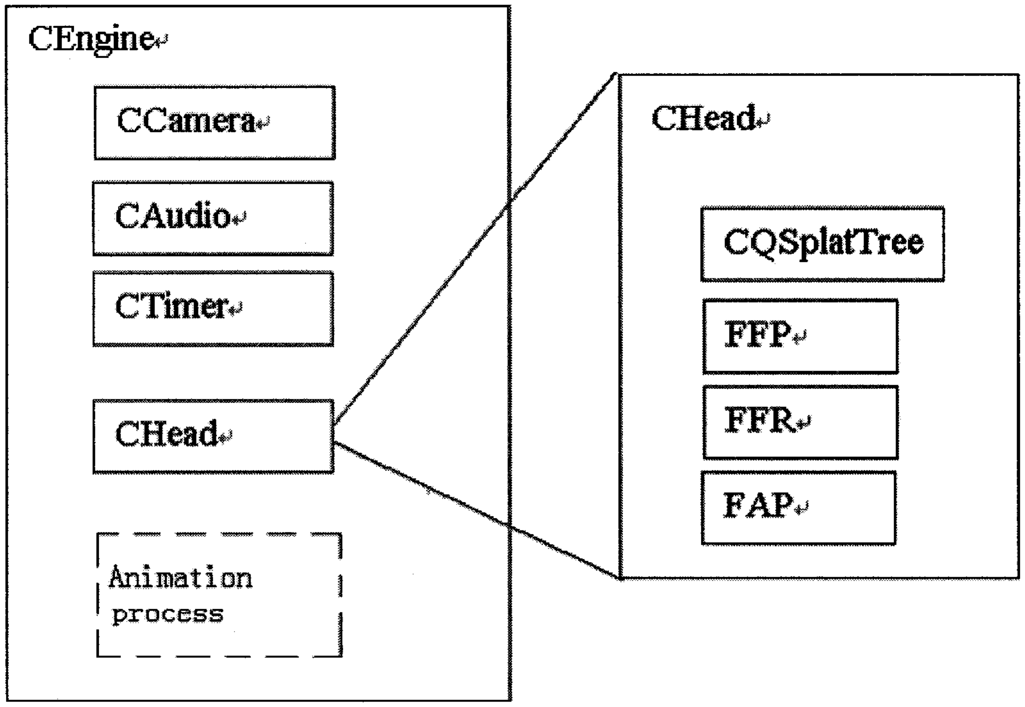


Figure 5-2: System organization

In our system, all animations are executed by setting an animation script. The work of getting the instructions from animation scripts and doing the corresponding operations is done by Animation Process which is resident in *CEngine* module. Animation Process is a powerful process. It can call *CAudio* to play the sound, inquire the time by communicating with *CTimer*, or ask *CHead* to affect one FAP upon the face structure to show a certain expression. The animation process keeps checking animation events while the system is idle. Once there is an animation event, Animation Process invokes corresponding modules to do the request operations. For example, when the animation process gets a sound handle from the script, it calls the *CAudio* module to play the sound.

5.4 High-Performance Timer

Windows system provides a WM_TIMER message and a set of API timer functions, such as SetTimer and KillTimer, to support timer event processing. The Operating system will put a WM_TIMER message in the message loop in a time interval specified by SetTimer function. Then, A timer proc will retrieve WM_TIMER message from the message loop and do the relative processing. However, this method is not suitable for real time animation because WM_TIMER message depends on system status. The message process would be delayed when the system is busy. Instead, we use a high-resolution performance counter to control the animation in our system. The performance frequency cannot change while system is running.

For each animation segment, a time slot is preset. We retrieve the performance counter value before and after first frame is rendered. The difference is the time spent on this frame drawing. Comparing to the preset time, we know how many frames we need to draw and how much percentage of the parameters should be set for the rest frames. The time length of drawing a frame is dynamically adjusted after the most recent frame is drawn because the time cost may have a bit of difference.

There are *QueryPerformanceFrequency* and *QueryPerformanceCounter* functions to get access to the system performance frequency and counter. But not all PC hardware supports high-resolution performance counter. In case hardware does not support, we use *timeGetTime* function to retrieve the system time, in milliseconds. The system time is the time elapsed since Windows was started.

5.5 Accelerating Animation Speed

The Windows system is known as a message driven operating system. Normally, there is a message loop dealing with the message queue that processes the events. A typical message loop is as follows.

```
MSG msg;
While ( GetMessage(&msg, NULL, 0, 0) > 0 )
{
        TranslateMessage(&msg);
        DispatchMessage(&msg);
}
```

We can see that a message loop consists of 3 functions. *GetMessage* keeps checking message queue. If the message queue is empty, message loop basically stops and waits for the next message. When an event causes a message to be added to the queue, *GetMessage* gets the message from the queue and passes it to *TranslateMessage*. *TranslateMessage* does a bit of processing on the message and then *DispatchMessage* looks up the current window procedure and dispatches the message. However, this process is not suitable for 3D real time animation. In a 3D animation, 3D scene rendering will occupy more CPU time and take longer time than other processes such as keyboard events, sound playing, etc. During the rendering process, the system is too busy to handle other messages and all messages that come in this period have to wait in the message queue. Therefore, some real time messages are not handled within the acquired time. That is the reason we see that sometimes the sound comes before the image changes.

To overcome this problem, we designed the message loop as follows.

```
 MSG  msg;
CEngine  m_Engine;
 bool get_msg;
 msg.message = WM_NULL;
 PeekMessage( &msg, NULL, 0U, 0U, PM_NOREMOVE );

 while( WM_QUIT != msg.message  )
 {
// Use PeekMessage() if the app is active, so we can use idle time to
// render the scene. Else, use GetMessage() to avoid eating CPU time.
   if( m_Active )
      get_msg = ( PeekMessage( &msg, NULL, 0U, 0U, PM_REMOVE ) != 0 );
   else
      get_msg = ( GetMessage( &msg, NULL, 0U, 0U ) != 0 );

   if( get_msg )
   {
      // Translate and dispatch the message
       if( accel == NULL || m_hWnd == NULL || TranslateAccelerator( m_hWnd, accel,
       &msg )  == 0 )
```

```
        {
            TranslateMessage( &msg );
            DispatchMessage( &msg );
        }
    }
    else
    {
        // Render a frame during idle time (no messages are waiting)
        if( m_Active )
            m_Engine.Process();
    }
```

As can be seen above, we add some checks before doing the rendering process to avoid the wasteful rendering.

Another measure we have taken to save CPU time is that we set a check point in the recursive rendering process. In our implementation, we set a check point before we go further to next layer. At the check point, we check the status variable in *CEngine* module. If an abort drawing message has been received, the recursive rendering process will stop and return.

5.6 Results

Our work is based on PC hardware with Windows 2000 Operating system. The desktop PC we used is DELL Precision WorkStation 350 series. The detailed configuration of the PC is: Intel Pentium4 Processor 2.80GHz, 1024MB System Memory, ATI Fire GL E1 Video Controller with 128MB Memory. In our experiments, we collected some useful result data as follows.

- The sampled point cloud model of normal face includes 88753 points. These points are distributed in a Q-Splat tree with 14 layers.

- The average rendering time for the normal face is about 0.068 second, the time to set FAPs and for rendering is 0.091 second.
- We synthesised 3 video clips for experiment. The lengths are 2.7, 2.4 and 10.6 seconds. The first clip has 5 actions; the second has 10 actions and the third has 15 actions.

6. Conclusions and future work

Chinese opera is a performing art, many styles of which are being revived by various groups with support from Chinese government and UNESCO. While conventional media like text, stills, audio and video are being used for this documentation purpose, we propose the use of computer animation techniques for the same. Specifically we attempt to demonstrate this by applying facial animation techniques. Taking into account the high fidelity requirements and the imminent availability of 3D scanning devices, we have devised an elaborate hierarchical facial structure using point sampled surface representations. All animations integrate direct parameterization and key frame interpolation techniques. We have experimented with a few characters and a comprehensive list of facial animation primitives. We have created a short animation sequence by hand coding the facial expressions corresponding to a short video clip. The overall results are satisfactory and quite encouraging. A paper based on the above work has been accepted for presentation at the international conference VSMM 2003 (International Conference on Virtual Systems and Multimedia) [51]

As mentioned earlier, our point cloud models are obtained by converting triangle meshes downloaded from http://www.facegen.com/. These converted point cloud models are noise-free which is quite different from directly scanned data. When we use scanned pointed cloud models in our future research, we need to filter out the noise due to the scanning process. This could be done for each FFR, after the FFPs are indentified.

We have also briefly mentioned the MPEG-4 Facial Animation standard in section 2-3. The idea of using FFPS was adapted from the MPEG-4 specification in which facial feature points are used to track the actions of facial animation. Similarly we also borrowed the MPEG-4 idea of using parameter combination to extend the expression range. However, the number of facial feature points we have used is about 10 times the number of feature control points defined in MPEG-4. This is because of our need to implement highly realistic facial animation in comparison to the need in MPEG-4 which is more focused on video coding, transmission and compression.

There are, of course, vast opportunities for further research. The logistics of working with art historians have to be worked out using a real case study, both for quality and authenticity. On the technical front too there are many problems/extensions that need to be addressed. We only list a few below.

- Preprocessing of point cloud data captured directly from 3D scanner, such as noise removal, should be developed.
- Automating the identification of FFPs in 3D scans is an important and interesting problem.
- Data from recordings of live performances should be used. We will also try to see the extent to which techniques such as optical flow [47] can be used to derive animation transformations directly from video.
- So far, we have only experimented with face animation. Problems in whole body animation also will be investigated.

References

[1] IEEE Multimedia, Special Issue on Virtual Heritage", IEEE Multimedia, 7(2), 2000 pp 20-74

[2] Glen Fraser, Scott S. Fisher, "Preserving Heritage Sites", ACM SIGGRAPH Vol.32 No.4 November 1998

[3] DeLeon, V. J. 1999. vrnd Notre Dame Cathedral: A Globally Accessible Multi-User Real-Time Virtual Reconstruction. http://www.vrndproject.com/

[4] Forte M., Siliotti A. Renfrew C., "Virtual Archaeology: Re-Creating Ancient Worlds", Harry N Abrams; ISBN: 0810939436, 1997.

[5] F. Bernardini, I. Martin, J. Mittleman, H. Rushmeier, G. Taubin. "Building a Digital Model of Michelangelo's Florentine Pieta." IEEE CGA, 22(1), 2002, pp. 59-67.

[6] Marc Levoy, Kari Pulli, etc. " The Digital Michelangelo Project: 3D Scanning of Large Statues ", Proc. SIGGRAPH 2000 URL: http://graphics.stanford.edu/projects/mich/

[7] Taylor J. Beraldin, J. A. Godin, G. Baribeau R.,Cournoyer, L. Blais, F. El-Hakim, S. F. Picard M. Rioux, M., and Domey J., "Culture as a Driving Force for Research and Technology Development - A Decade's Experience of Canada's NRC 3D Technology", EVA 2002 Conference Proceedings, pp 4.1 - 4.13.

[8] N. Magnenat-Thalmann, L. Moccozet, "Virtual Humans on Stage", in Virtual Worlds: Synthetic Universes, Digital Life and Complexity, JeanClaude Heudin (Ed.), New England Complex Systems Institute Series on Complexity, 1998, pp.95126.

[9] http://herbergercollege.asu.edu/college/news/newsreleases/2002/isa_loops_110402.html

[10] http://www.artistswithaids.org/artforms/dance/

[11] http://www-2.cs.cmu.edu/~virtualized-reality/

[12] Thalmann, N. M. , Thalmann D. (Eds.), "Synthetic Actors in Computer Generated 3D Films", 1987, Springer Verlag.

[13] Jessica Tan Gudnason, Gong Li, "Chinese Opera", Abbeville Press, Inc. (June, 2001) ISBN: 0789207095

[14] UNESCO PRESS, Paris, May 18 (No.20001-71), BUREAU OF PUBLIC INFORMATION (BPI), URL: http://www.unesco.org/bpi/intangible_heritage/index.htm

[15] People's Daily, "China Moves to Protect Intangible Heritage", Oct 29, '02, http://english.peopledaily.com.cn/

[16] People's Daily, "Peking Opera Classics Videotaped", September 6, 2002

[17] "China to Set up Database of Intangible Cultural Heritage", Xinhua News Agency December 9, 2002

[18] Parke F. I. M, Waters K., "Computer facial Animation", 1996, ISBN 1-56881-014-8, A. K. Peters Ltd.

[19] Jun-yong Noh, Ulrich Neumann, "A Survey of Facial Modeling and Animation Techniques", USC Technical Report 99-705, 1998 "

[20] Szymon Rusinkiewicz, Marc Levoy, Stanford University, "QSplat: A Multiresolution Point Rendering System for Large Meshes", SIGGRAPH, 2000.

[21] Hanspeter Pfister, Mattias Zwicker, Jeroen van Baar, Markus Gross, "Surfels: Surface Elements as Rendering Primitives ", Siggraph 2000

[22] F. I. Park. "A Parameteric Model from Human Faces". Ph.D thesis, University of Utah, Salt Lake City, UT, Dec. 1974 UTEC-CSc-75-047

[23] Fernando Pereira and Touradj Ebrahimi, "The MPEG-4 Book", pp. 389-410, IMSC Press Multimedia Series/Andrew Tescher, Series Editor, ISBN 0-13-061621-4, 2002

[24]S.M.Platt and N. I. Badler. "Animating facial expressions". Computer Graphics, 15(3):245-252,1981

[25] K. Waters, T. M. Levergood, Decface: An Automatic Lip-Synchronization Algorithm for Synthetic

Faces, 1993, DEC. Cambridge Research Laboratory Technical Report Series

[26] W. T. Reeves. "Simple and complex facial animation: Case studies", In State of the Art in Facial Animation, SIGGRAPH '90 Course Notes #26, pages 88-106. ACM, New York, August 1990.

[27]D. Terzopoulos and K. Waters. "Techniques for realistic facial modeling and animation", J. of Visualization and Computer Animation, 1(4):73-80, March 1990

[28]C. T. Waite. "The Facial Action Control Editor, FACE: A parametric facial expression editor for computer generated animation", Master's thesis, Massachusetts Institute of Technology, Media Arts and Sciences, Cambridge, MA, Feb. 1989

[29] E. Catmull, J. Clark, Recursively generated b-spline surfaces on arbitrary topological meshes,

Computer Aided Design, 1978, vol. 10(6), pp. 350-355

[30] J. Y. Noh and U. Neumann, "A Survey of Facial Modelling and Animation techniques", USC Technical Report 99-705

[31] N. Magnenat-Thalmann, N. E. Primeau, and D. Thalmann, “Abstract muscle actions prodedures for human face animation”, Visual Computer 3(5):290-297, 1988

[32] T. W. Sederberg and S. R. Parry. ”Free-form deformation of solid geometry models”, Computer Graphics (SIGGRAPHPH ‘86) 20(4)151-160, 1986

[33] S. Coquillart. “Extended free-form deformation: A sculpturing tool for 3D geometric modeling”, Computer Graphics, 24(4)187-196, 1990

[34] P. Kalra, A. Mangili, N. Magnenant-Thalmann, and D. Thalmann, “Simulation of facial muscle actions based on rational free form deformations”, In Proc. Eurographics 92, page 59-69, Cambridge, 1992

[35] D. Terzopoulos and K. Waters, “Analysis and synthesis of facial image sequences using physical and anatomical models”, IEEE Transactions on Pattern Analysis and Machine intelligence, 15(6)569-579, 1993

[36] P. Kalra, A. Mangili, N. Magnenat-Thalmann, D. Thalmann (1991), SMILE: A multi-layered Facial Animation System, Proc. IFIP WG 5. 10, Tokyo, Japan (Ed Kunii TL) pp. 189–198

[37] I. A. Essa, S. Basu, T. Darrell, A. Pentland, Modeling, Tracking and Interactive Animation of Faces and Heads using Input from Video, Proceedings of Computer Animation June 1996 Conference, Geneva, Switzerland, IEEE Computer Society Press

[38] F. Ulgen, A step Toward universal facial animation via volume morphing, 6th IEEE International Workshop on Robot and Human communication, 1997, pp. 358-363

[39] F. Pighin, J. Auslander, D. Lischinski, D. H. Salesin, R. Szeliski, Realistic Facial Animation Using Image-Based 3D Morphing, 1997, Technical report UW-CSE-97-01-03

[40] Ellen Wolff, “A Festival of Peers”, Video Systems, May 1, 2003

[41] Reprint of Lawrence S.Chen, Jörn Ostermann, "Animated Talking Head with Personalized 3D Head Model", Proceedings of 1997 Workshop of Multimedia Signal Processing pp: 274-279, June 23-25, 1997, Princeton , USA.

[42] Kalaih A. and Varshney A., “Modelling and Rendering Points with Local geometry”, IEEE Trans. On Visualization and Computer Graphics, 2002, pp 101-129.

[43] L. Yin, A fast feature detection algorithm for human face contour based on local maximum curvature tracking, Technique Report, ICG, Department of Computing Science, City U of HK, 1995

[44] Y. C. Lee, D. Terzopoulos, K. Waters. Realistic face modeling for animation. Siggraph proceedings, 1995, pp. 55-62

[45] P. Ekman and W. V. Friesen, "Manual for the Facial Action Coding System", Consulting Psychologists Press, Inc., Palo Alto, CA, 1978

[46] Haralick R. M., Shapiro L. G., "Computer and Robot Vision", AD. Wesley, 1993.

[47] Eiswert P., Girod B., "Model-based estimation of facial expression parameters from image sequences", Proc. IEEE International Conf. on Image Processing, ICIP'97, pp 418-422.

[48] Levoy M. and Garcia-Molina H., " Creating Digital Archives of 3D Artworks", White Paper submitted to NSF's Digital Libraries Initiative, http://graphics.stanford.edu/projects/dli/white-paper/dli.html

[49] Seo H., Thalmann N. M., "LoD Management on Animating Face models", IEEE VR 2000, pp. 161-168.

[50] Ekman P., Friesen W. V., "Manual for the Facial Action Coding System", Consulting Psychologists Press, 1978.

[51] H. Zhou, S. P. Mudur, "Application of 3D Facial Animation Techniques for Chinese Opera", accepted as a full paper for publication in the proceedings of the international conference VSMM 2003.

[52] The triangle mesh models used in our research to capture the point- sampled representations for simulating 3D scanned data are downloaded from http://www.facegen.com/.

Appendices

Appendix A Facial Action Coding System Tables

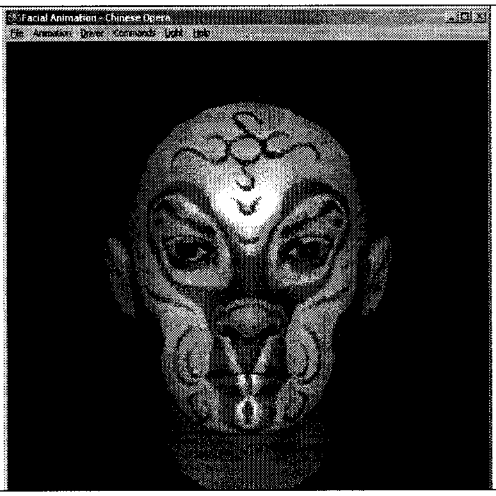
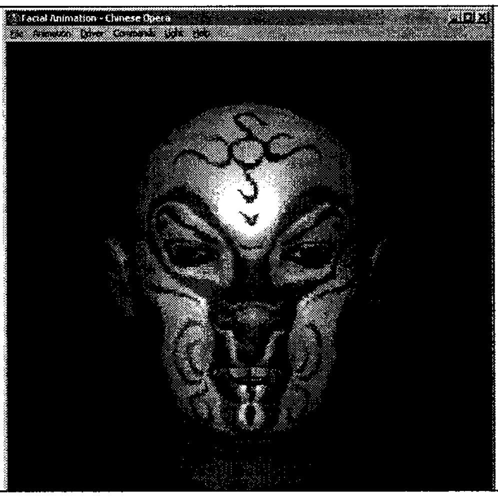
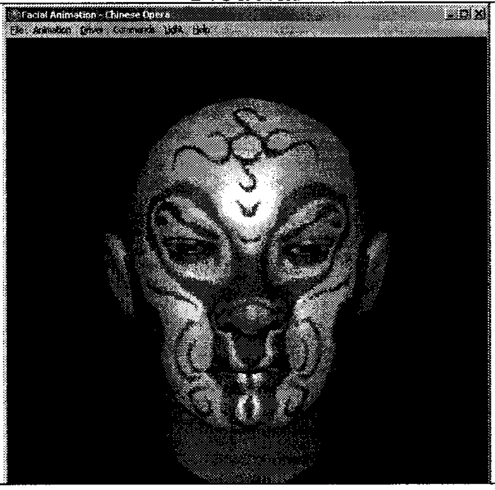
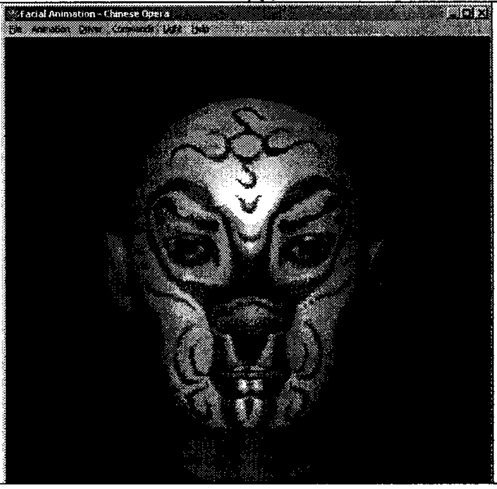
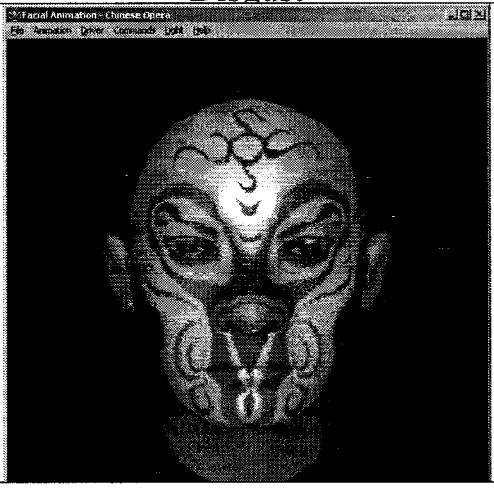
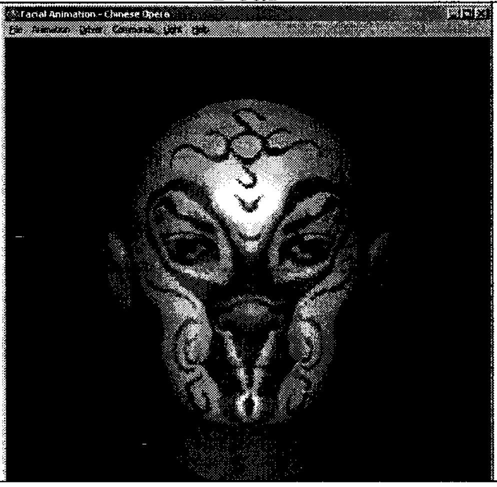
	Action Units	Anger	Disgust	Fear	Sad	Smile (close Mouth)	Smile (open Mouth)	Surprise
1	Forehead	57	57	31	57	57	57	57
2	Left eyebrow	41	41	37	41	41	41	41
3	Right eyebrow	34	34	34	34	34	34	34
4	Left Upper eyelid	59	59	60	59	52	52	59
5	Right Upper eyelid	62	62	62	62	53	53	62
6	Left eye ball							45
7	Right eye ball							45
8	Left Lower eyelid	46	47	47	46	48	48	46
9	Right Lower eyelid	46	46	46	46	46	46	46
10	Nose Bridge	54	54	54	54	54	54	54
11	Left nose nostril	67	67	36	57	57	67	65
12	Right nose nostril	65	65	39	56	56	65	62
13	Left cheek	84	84	66	78	78	78	73
14	Right cheek	85	85	79	74	85	85	82
15	Left ear							
16	Right ear							
17	Upper lip	122	122	122	64	122	122	122
18	Lower lip	111	111	111	71	111	111	111
19	Upper teeth							
20	Lower teeth						20	
21	Tongue							
22	Chin	150	150	149	123	149	150	150
23	Neck	50	50	50	36	48	48	50

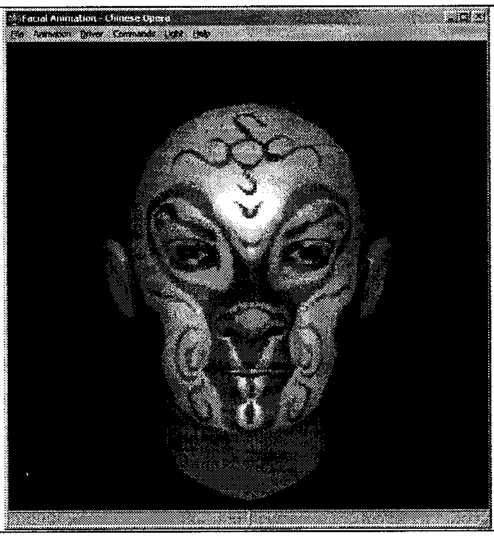
* Table content are affected FFR numbers of each action in different expressions.

	Action Units	Blink Left Eye	Blink Right Eye	Left Brow Down	Right Brow Down	Left Brow In	Right Brow In	Left Brow Up	Right Brow Up
1	Forehead			56	54	50	54	54	54
2	Left eyebrow			40	40	24	40	40	40
3	Right eyebrow			34	31	34	31	34	31
4	Left Upper eyelid	42		50	50	15	50	50	50
5	Right Upper eyelid		44	49	49	49	49	49	49
6	Left eye ball								
7	Right eye ball								
8	Left Lower eyelid	20		29	29	7	29	29	29
9	Right Lower eyelid		18	19	19	19	19	19	19
10	Nose Bridge			50	50	24	50	50	50
11	Left nose nostril			57	57	2	57	57	57
12	Right nose nostril			52	52		52	52	52
13	Left cheek			39	39	24	39	39	39
14	Right cheek			45	45	30	45	45	45
15	Left ear								
16	Right ear								
17	Upper lip			19	19	7	19	19	19
18	Lower lip			22	22	20	22	22	22
19	Upper teeth								
20	Lower teeth								
21	Tongue								
22	Chin			87	87	52	87	87	87
23	Neck			30	36	30	36	36	32

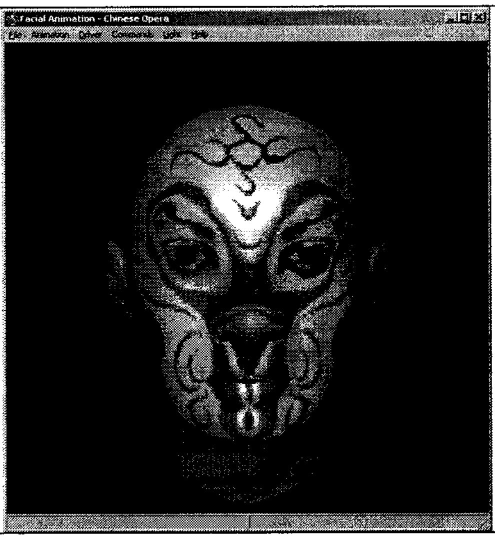
	Action Units	Left Eye Squint	Right Eye Squint	Look Down	Look Left	Look Right	Look Up		
1	Forehead								
2	Left eyebrow	30					4		
3	Right eyebrow		18				5		
4	Left Upper eyelid	54		42			55		
5	Right Upper eyelid		57	46			54		
6	Left eye ball			68	59	59	70		
7	Right eye ball			68	65	70	70		
8	Left Lower eyelid	40		29			32		
9	Right Lower eyelid		38	22			31		
10	Nose Bridge								
11	Left nose nostril								
12	Right nose nostril								
13	Left cheek	45							
14	Right cheek		48						
15	Left ear								
16	Right ear								
17	Upper lip								
18	Lower lip	11	11						
19	Upper teeth								
20	Lower teeth								
21	Tongue								
22	Chin								
23	Neck								

Appendix B Pictures of Basic Facial Expressions

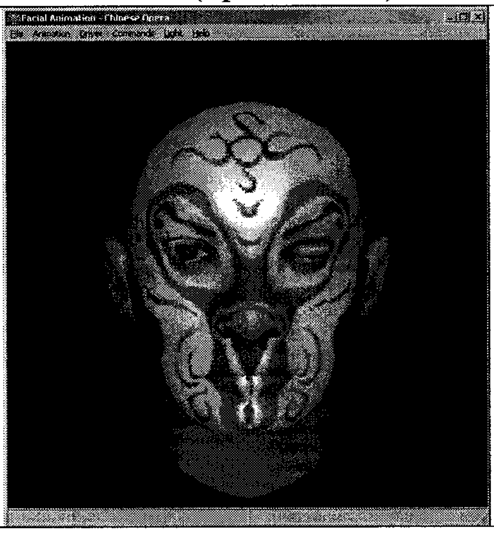
	
Neutral	Anger
	
Disgust	Fear
	
Sad	Smile (close Mouth)


Facial Animation - Chinese Opera
File Animation Driver Commands Light Help

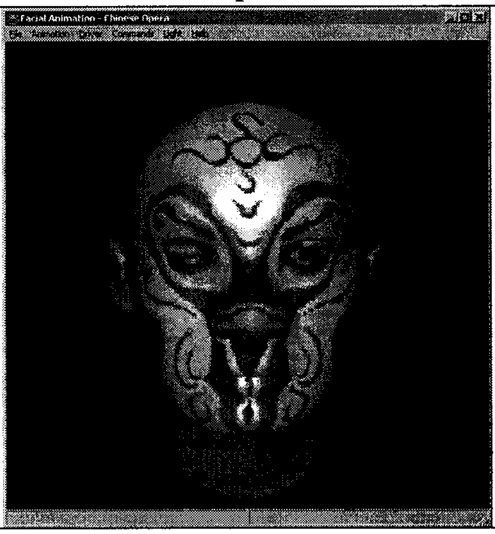
Smile (open Mouth)


Facial Animation - Chinese Opera
File Animation Driver Commands Light Help

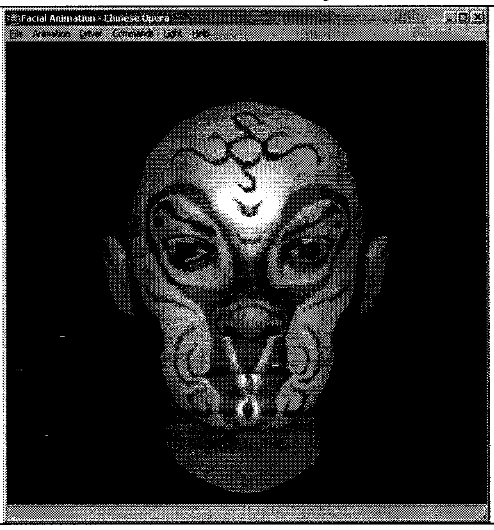
Surprise


Facial Animation - Chinese Opera
File Animation Driver Commands Light Help

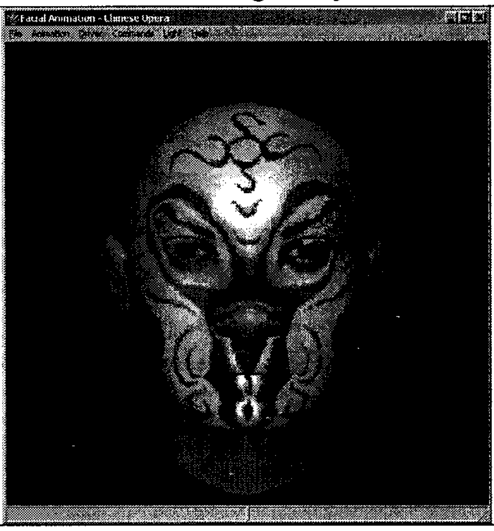
Blink Left Eye


Facial Animation - Chinese Opera
File Animation Driver Commands Light Help

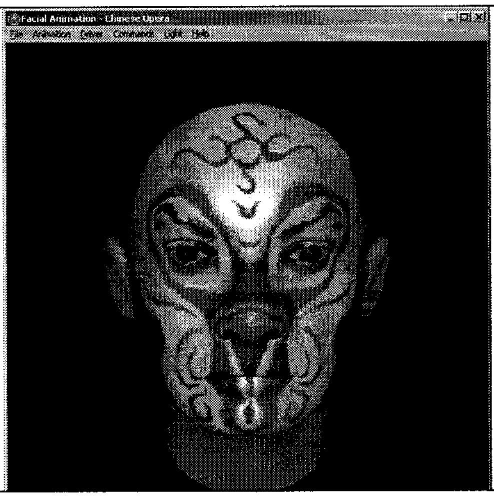
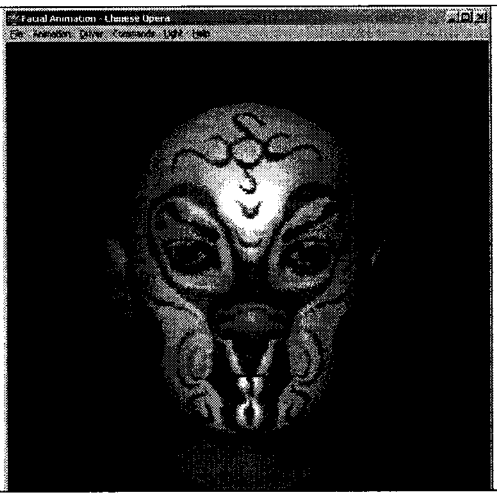
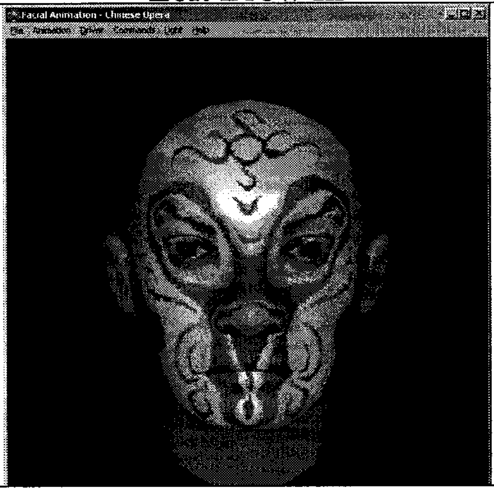
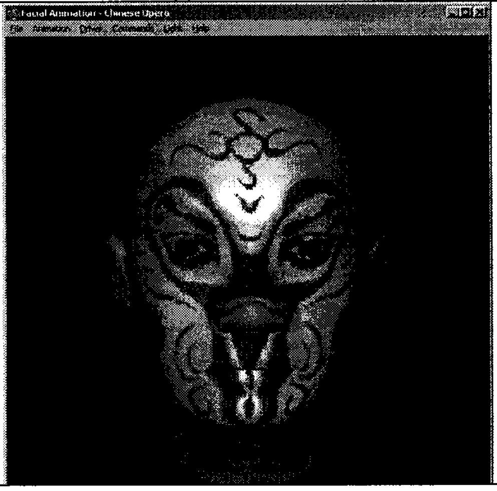
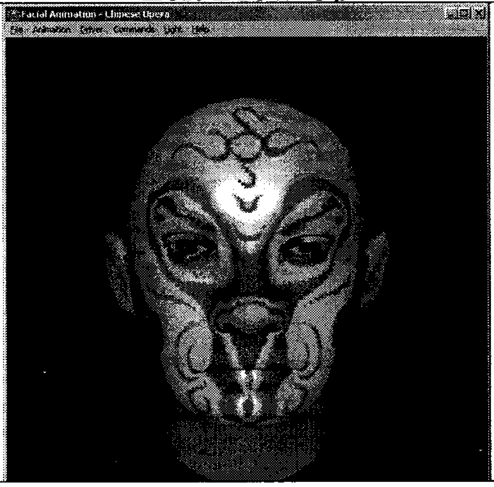
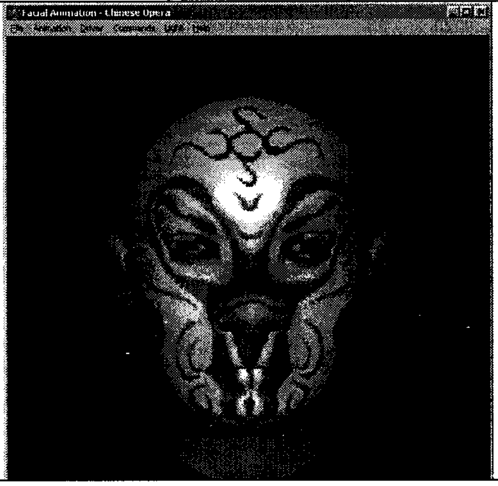
Blink Right Eye

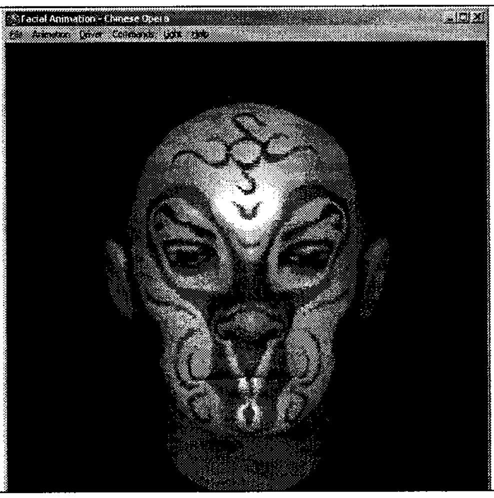
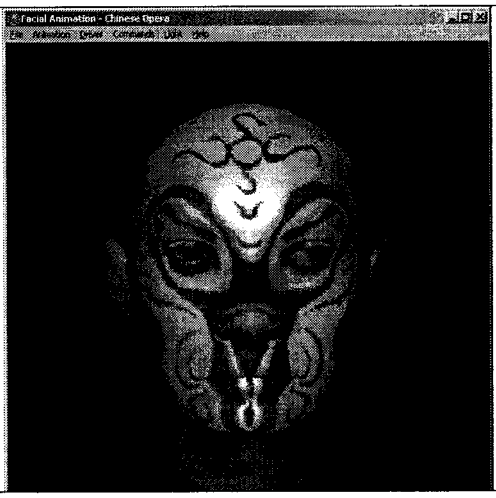
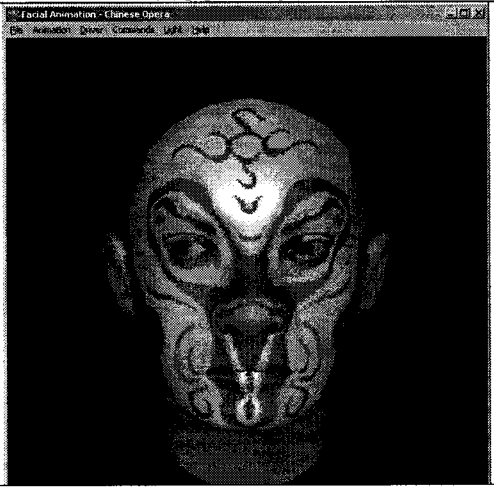
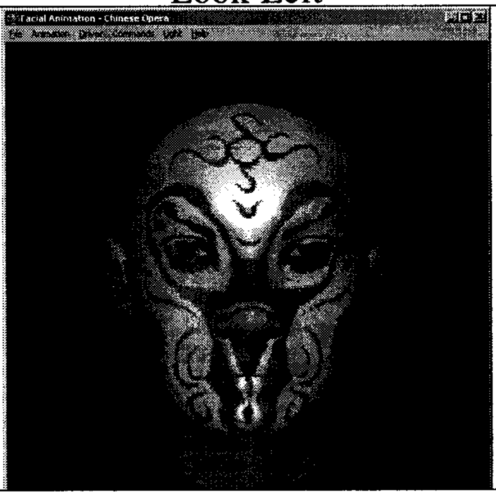
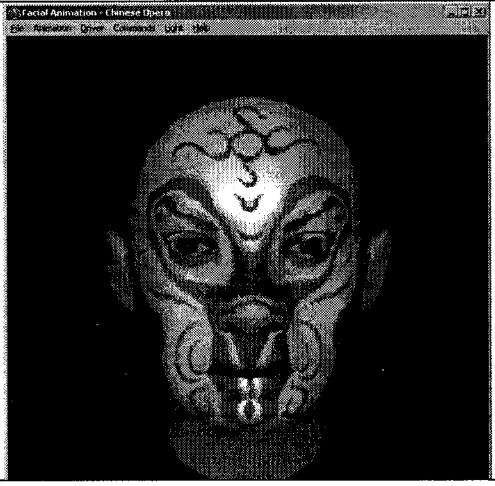
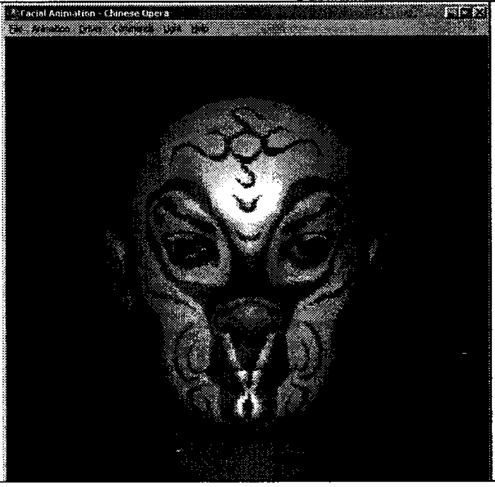

Facial Animation - Chinese Opera
File Animation Driver Commands Light Help

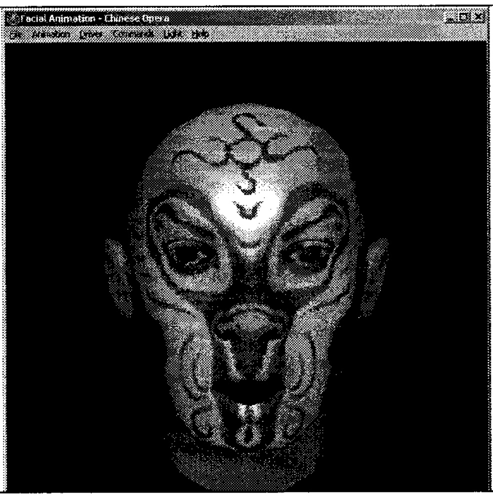
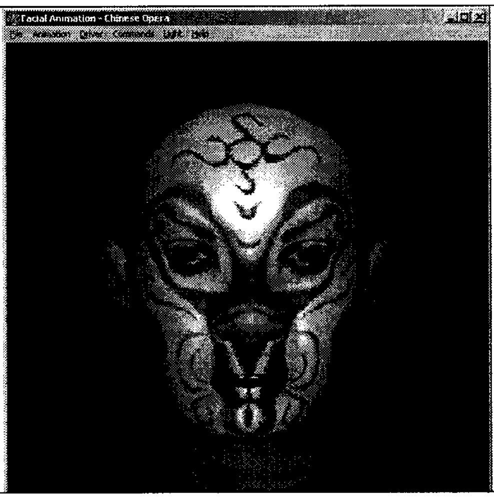
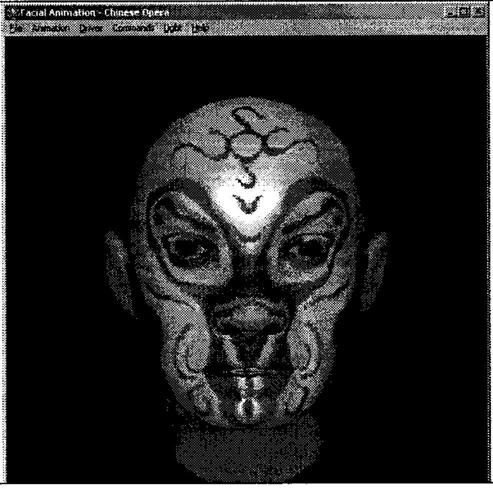
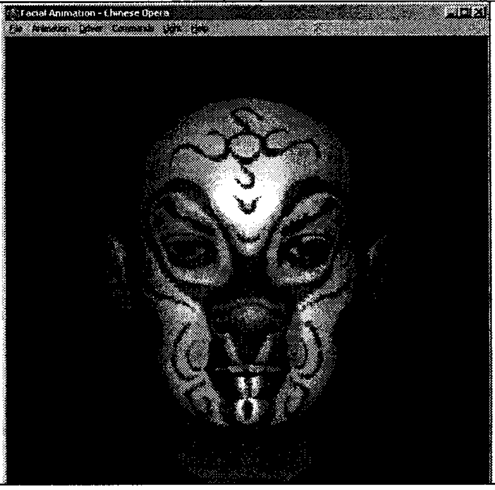
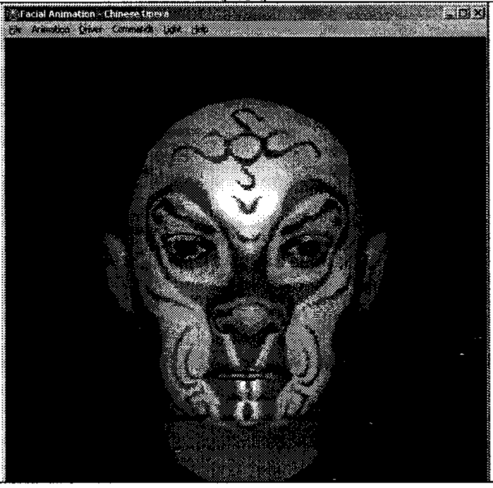
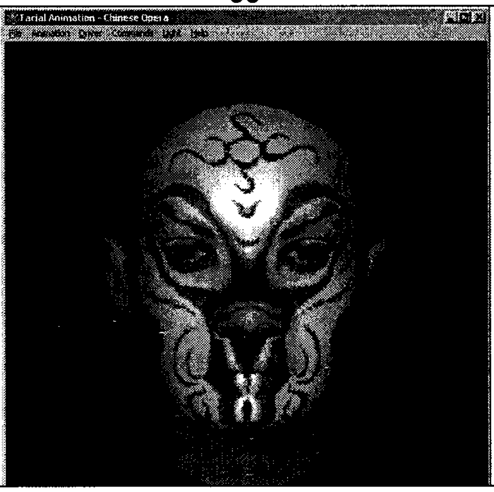
Left Brow Down

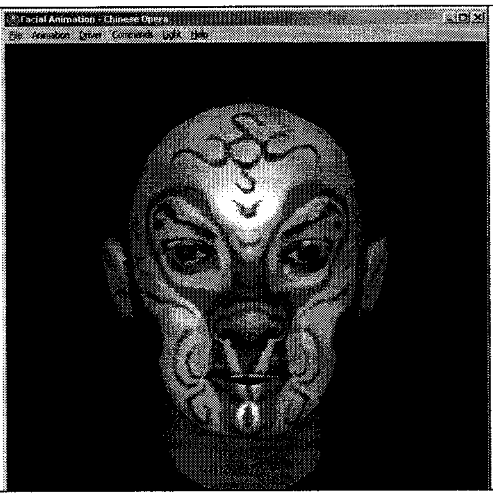
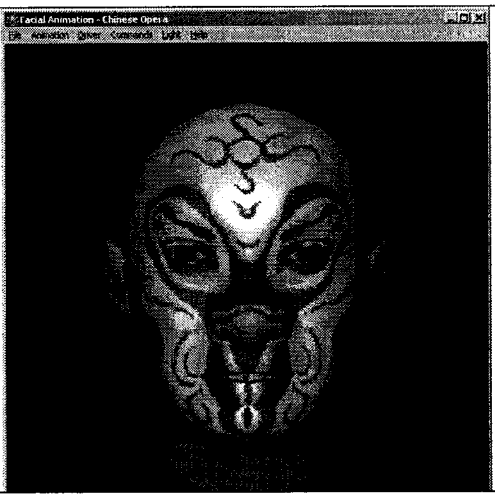
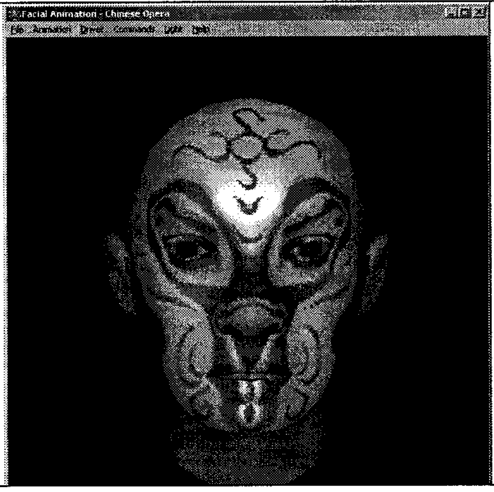
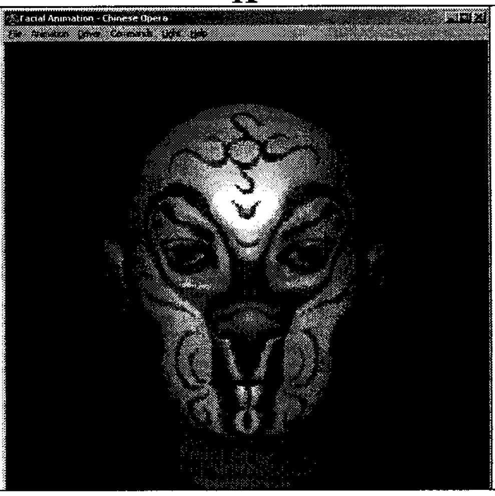
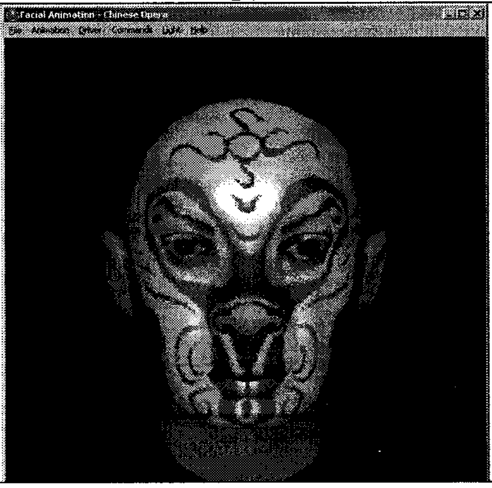
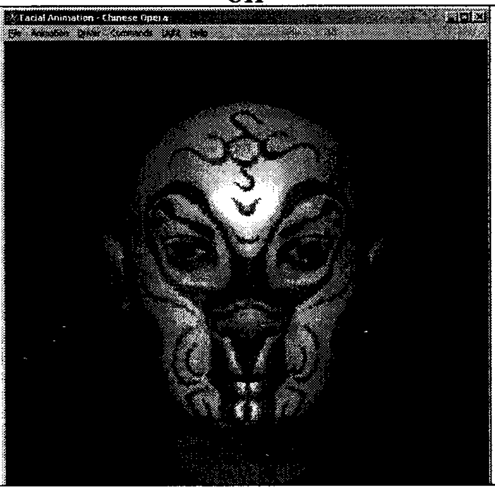

Facial Animation - Chinese Opera
File Animation Driver Commands Light Help

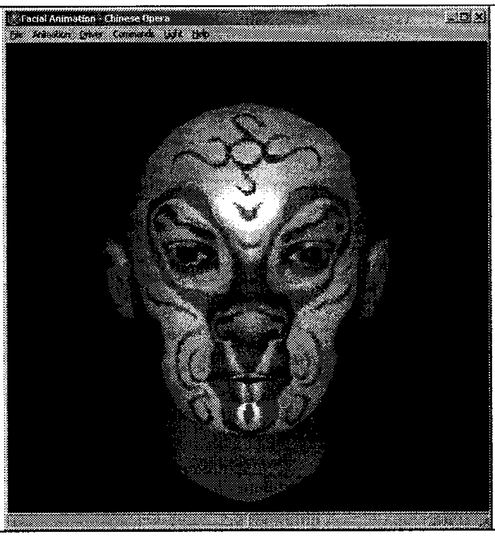
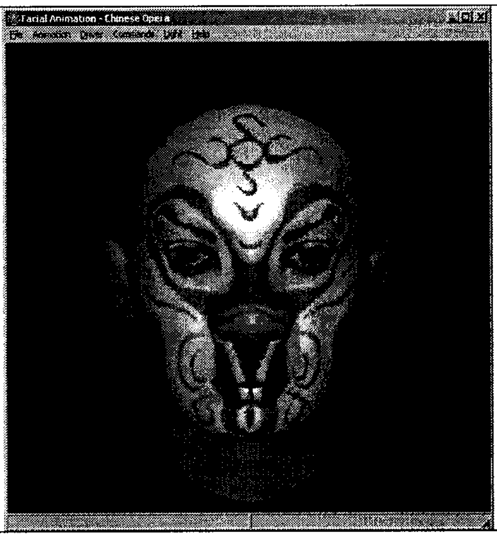
Right Brow Down

 Facial Animation - Chinese Opera File Animation Driver Commands Light Help	 Facial Animation - Chinese Opera File Animation Driver Commands Light Help
Left Brow In	Right Brow In
 Facial Animation - Chinese Opera File Animation Driver Commands Light Help	 Facial Animation - Chinese Opera File Animation Driver Commands Light Help
Left Brow Up	Right Brow Up
 Facial Animation - Chinese Opera File Animation Driver Commands Light Help	 Facial Animation - Chinese Opera File Animation Driver Commands Light Help
Left Eye Squint	Right Eye Squint

 Facial Animation - Chinese Opera File Animation Driver Commands Light Help	 Facial Animation - Chinese Opera File Animation Driver Commands Light Help
Look Down	Look Left
 Facial Animation - Chinese Opera File Animation Driver Commands Light Help	 Facial Animation - Chinese Opera File Animation Driver Commands Light Help
Look Right	Look Up
 Facial Animation - Chinese Opera File Animation Driver Commands Light Help	 Facial Animation - Chinese Opera File Animation Driver Commands Light Help
Aah	B, M, P

	
Big aah	ch, J, sh
	
D, S, T	ee
	
eh	F, V

 Facial Animation - Chinese Opera File Animation Driver Commands Light Help	 Facial Animation - Chinese Opera File Animation Driver Commands Light Help
ei	K
 Facial Animation - Chinese Opera File Animation Driver Commands Light Help	 Facial Animation - Chinese Opera File Animation Driver Commands Light Help
N	oh
 Facial Animatinn - Chinese Opera File Animation Driver Commands Light Help	 Facial Animation - Chinese Opera File Animation Driver Commands Light Help
ooh, O	R

Facial Animation - Chinese Opera File Animation Driver Commands Light Help	Facial Animation - Chinese Opera File Animation Driver Commands Light Help
Th	W